Copyright
by
Ruizhe Zhang
2023

The Dissertation Committee for Ruizhe Zhang
certifies that this is the approved version of the following dissertation:

Quantum Meets Optimization and Machine Learning

Committee:

Dana Moshkovitz, Supervisor

Scott Aaronson

Nai-Hui Chia

Georgios-Alexandros Dimakis

Nick Hunter-Jones

Eric Price

Quantum Meets Optimization and Machine Learning

by

Ruizhe Zhang

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 2023

Acknowledgments

First, I would like to express my deepest gratitude to my Ph.D. advisor, Dana Moshkovitz, for all the guidance and support over the last five years. Dana is a fantastic advisor who's been so kind and helpful with her invaluable advice and tailored advising style to suit her students' needs. She allowed me to explore diverse research interests freely. Her enthusiasm, persistence, and vision in the pursuit of challenging and fundamental problems in theoretical computer science have greatly inspired me. Dana encouraged me to continue my academic journey, and she has always been an excellent role model for me in being an exceptional researcher as well as a supportive, open-minded advisor.

I also want to thank Scott Aaronson, who opened the doors for me to the world of quantum computing through his fascinating and inspiring lectures. He has been so generous with his brilliant ideas and insightful observations as a mentor and collaborator. We had a lot of fun in our research discussions and weekly quantum group meetings. And his book, Quantum Computing Since Democritus, made a great impression on me during my undergraduate studies, and further influenced my own research trajectory.

I wish to express my gratefulness to Nai-Hui Chia, Alexandros G. Dimakis, Nick Hunter-Jones, Eric Price, together with Dana and Scott, to serve as my thesis committee.

I've been so fortunate to be able to work with so many talented and amazing coauthors over the last five years, including Scott Aaronson, Josh Alman, Nai-Hui Chia, Andrew M. Childs, Sitan Chen, Chi-Ning Chou, Yichuan Deng, Daniel Stilck França, Jason Gaitonde, Yeqi Gao, Max Hopkins, Baihe Huang, Shunhua Jiang, Ziyu Jiang, Peter D. Johnson, Tali Kaufman, Tongyang Li, Jiehao Liang, Yingbin Liang, Han-Hsuan Lin, Jiahui Liu, Jin-Peng Liu, Qipeng Liu, Shachar Lovett, Andrew C.

Potter, Lianke Qin, Zhao Song, Baocheng Sun, Runzhou Tao, Chunhao Wang, Guoming Wang, Zhangyang (Atlas) Wang, Omri Weinstein, Hongru Yang, Shuo Yang, Xin Yang, Mark Zhandry, Hengjie Zhang, Jiayu Zhang, Lichen Zhang, Xinzhi Zhang, Yuxuan Zhang, Shuchen Zhu, Danyang Zhuo.

I wish to express my gratitude to my supervisor at Fudan University, Haibin Kan, who guided me to the world of TCS and spurred my desire to conduct research in this area. Further, I was fortunate to spend an enriching summer as an undergrad research intern at the Shanghai University of Finance and Economics, working under the amazing guidance of Richard Peng and Jin-Yi Cai.

I would like to thank Peter D. Johnson and Guoming Wang (now at ORCA Computing), my mentors for two research internships at Zapata Computing, Inc. Collaborating with them on the early fault-tolerant quantum algorithms was a highly enjoyable experience. I am also grateful to Zapata's generous sponsorship of my travel to QIP 2021 and APS March Meeting 2022.

I would like to thank the computer science department and the theory group at UT Austin. I'm very grateful to have been a part of this community. I had a lot of fun with the people here. I wish to thank all my office mates over the past five years. Apart from the ones already mentioned above, this includes Joshua Cook, Sabee Grewal, Shivam Gupta, Kuan-Yi Ho, Shih-Han Hung, William Hoza, Vishnu Iyer, Michael Jaber, Akhil Jalan, Siddhartha Jain, Niels Kornerup, William Kretschmer, Vinayak Kumar, Lin Lin Lee, Fu Li, Jiawei Li, Daniel Liang, Vignesh Manoharan, Geoffrey Mon, Justin Oh, Patrick Rall, Ronak Ramachandran, Adrian Trejo Nunez, Zhiyang Xun, Justin Yirka, Joey Zhou.

I am especially thankful to Zhao Song for his substantial help in my research. I've had the privilege of knowing this guy since the final year of his Ph.D. at UT Austin, and he is, without a doubt, one of the most hard-working and productive researchers in TCS that I've encountered. Zhao has shared invaluable advice on research methodologies with me, introduced me to the fields of optimization and machine learning theory, and has since become one of my most frequent collaborators.

Outside of research, playing badminton has been a huge part of my life and a constant source of joy at UT Austin. I would like to thank all my badminton mates. A very partial list includes Si Chen, William Kretschmer, Simon Lautenbach, Jian He, Yuefei Huang, Qi Lei, Tianyu Liang, Yuxiang Lin, Jinsong Liu, Ziqi Liu, Hengfa Lu, Qinlan Luo, Dilin Wang, Yingchen Wang, Yixian Wang, Feng Yuan, Hongming Zhang, Jiyang Zhang, Weihan Zhang, Yi Zhang, Zhiting Zhu.

Lastly, I wish to extend my thanks to all the loved ones who have stood by me throughout this journey:

To Peach the cat, your presence has provided an immeasurable amount of joy.

To Ziqi Liu, my beloved fiancée, your endless love has been my anchor in this journey. I am deeply grateful for your companionship, understanding, patience, and encouragement. I am truly blessed to have you in my life.

The final thank is reserved for my parents, Gang Zhang and Hong Ma, as well as my late grandmother, Shaojun Fan. I am forever indebted to your unconditional support throughout my life and the sacrifices you have made in raising me and shaping me into who I am today. I'm sorry for not having been able to spend more time with you in the past five years. To my grandmother, your love continues to inspire me and live on in every step I take. Thank you all for everything.

Abstract

Quantum Meets Optimization and Machine Learning

Ruizhe Zhang, PhD
The University of Texas at Austin, 2023

SUPERVISOR: Dana Moshkovitz

With the advent of the quantum era, what role the quantum computer will play in optimization and machine learning becomes a natural and salient question. The development of novel quantum computing techniques is essential to showcase the quantum advantage in these fields. At the same time, new findings in classical optimization and machine learning algorithms also have the potential to stimulate quantum computing research.

In the dissertation, we explore the fascinating connections between quantum computing, optimization, and machine learning, paving the way for transformative advances in all three fields. First, on the quantum side, we present efficient quantum algorithms for fundamental problems in sampling, optimization, and quantum physics. Our results highlight the practical advantages of quantum computing in these fields. In addition, we introduce new approaches to quantum complexity theory for characterizing the quantum hardness of optimization and machine learning problems. Second, on the optimization side, we improve the efficiency of the state-of-the-art classical algorithms for solving semi-definite programming (SDP), Fourier sensing, dynamic distance estimation, etc. Our classical results are closely intertwined with quantum computing. Some of them serve as stepping stones to new quantum algorithms, while others are motivated by quantum applications or inspired by quantum techniques. Third, on the machine learning side, we develop fast classical and quantum algorithms for training over-parameterized neural networks with provable guarantees of convergence and generalization. Furthermore, we contribute to the security aspect of machine learning by theoretically investigating some potential approaches to (classically) protect private data in collaborative machine learning and to (quantumly) protect the copyright of machine learning models. Fourth, we investigate the concentration and discrepancy properties of hyperbolic polynomials and higher-order random walks, which could potentially be applied to quantum computing, optimization, and machine learning.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Quantum computing, optimization, and machine learning are three exciting areas where the research community has achieved great success in the few years. In quantum computing, several research teams have built increasingly powerful quantum computers. The landmark achievement of “quantum supremacy" was demonstrated by Google [AAB$^+$19], USTC [ZWD$^+$20a, WBC$^+$21b], and Xanadau [MLA$^+$22]. Simultaneously, in the field of optimization, groundbreaking algorithms for solving problems like linear programming [CLS19], shortest paths [BNWN22], and network flows [CKL$^+$22] have been proposed. Furthermore, to overcome the challenges of “big data”, the landscape of optimization has been significantly extended with numerous new techniques being introduced, including distributed optimization [BPC$^+$11], stochastic optimization [KB15], online optimization [GKR13], and private optimization [FKT20], among others. The third domain, machine learning, through the rise of deep neural networks, has witnessed breakthroughs across a myriad of tasks from natural language processing [BMR$^+$20] to playing Go [SAH$^+$20] and Dota 2 [OB$^+$19].

Some people may perceive quantum computing, optimization, and machine learning as individually isolated islands, but in fact, they are highly intertwined. A primary objective of quantum computing is to solve problems that classical algorithms struggle with. Indeed, many optimization problems, such as job scheduling, portfolio optimization, and integrated circuit layout, pose excellent opportunities to leverage the advantages of quantum computing. While it is unlikely that quantum computers will find optimal solutions to NP-hard problems, we still hope they can find approximate solutions faster than classical computers or even produce heuristically better approximations. And this hope has been largely fueled by methods based on machine learning, such as the variational quantum algorithms [PMS$^+$14, FGG14a] and quantum neural networks [ASZ$^+$21]. Furthermore, a series of techniques known as quantum linear algebra [HHL09, CGJ19, GSLW19, LKAS$^+$21] has been developed to exponentially speed up a range of linear algebra operations, such as solving linear systems, matrix multiplications, etc. As a result, they lay the groundwork for numerous quantum optimization and machine learning algorithms that are much faster than their classical counterparts.

Concurrently, optimization and machine learning have proved vital in addressing fundamental problems in quantum computing and quantum information. For instance, semi-definite programming has emerged as an indispensable tool for characterizing quantum correlations and probabilities [BZ06, Mir23]. Additionally, online learning has been effectively applied to learn quantum states [ACH$^+$18, CHL$^+$22]. Furthermore, the principles of deep learning theory have been adapted to predict complex quantum systems [HKT$^+$22, CHC$^+$22].

However, there remain several important questions underlying the intricate interplay among quantum computing, optimization, and machine learning:

- **Question 1:** *To what extent can a quantum computer assist in an optimization or machine learning problem?* On the algorithmic side, even for problems that are well-understood in the classical context, it often proves challenging to directly apply existing quantum techniques for their resolution. Due to the sharp differences between quantum and classical computations, there exist some quantum-unique issues that have never been studied in the classical literature. On the complexity side, understanding the limitations of quantum computing with regard to problems is a daunting task. Notably, most optimization and machine learning problems are more complicated than those typically studied in quantum complexity theory, and people care more about the average-case complexity rather than their worst-case complexity, thus demanding novel approaches and techniques.

- **Question 2:** *How kind of quantum resources are required to implement these quantum algorithms?* The capabilities of current quantum devices are still limited, and the quantum community widely believes that it may take decades to construct a large-scale, fully fault-tolerant quantum computer. Therefore, the usefulness of a quantum algorithm for optimization or machine learning essentially depends on the quantum resources required (i.e., circuit depth, number of ancilla qubits, noise tolerance, etc.). For instance, many quantum machine learning algorithms that use quantum linear algebra will require Quantum Random Access Memory (QRAM) to load/store classical data, which is quite challenging for real-world implementation (e.g., [JR23]). In contrast, the variational quantum algorithms can be implemented on Noisy Intermediate-Scale Quantum (NISQ) devices [Pre18], which could be achieved in the near future. However, the performance of these algorithms is either constrained [MBS+18, SFGP21] or lacks provable guarantees.

- **Question 3:** *What are the most efficient classical algorithms for optimization problems?* Many fundamental optimization problems, such as semi-definite programming, have broad applications in quantum computing and machine learning, but we currently do not fully understand them classically. As a result, these classical optimization procedures often become the limiting factor in their respective applications. On the other hand, classical algorithms serve as essential benchmarks for optimization problems to gauge the effectiveness of quantum algorithms. If existing classical algorithms are suboptimal or simply some naïve approaches, it obscures the extent to which quantum computers truly help in resolving these problems.

- **Question 4:** *Is there any provable approach to improve the efficiency of deep learning?* The success of AIs can be partially attributed to the larger deep learning model and bigger training dataset. For instance, some of the current Large Language Models like GPT-3 [BMR+20] and PaLM [CND+22] have more than 100 billion parameters. The training of such gigantic models, as well as their use for inference, requires substantial computational resources. Therefore, developing more efficient training and evaluation algorithms for deep neural networks become an important question for both academic and industrial fields. There exist some training algorithms, such as SLIDE [CMJF$^+$20], Reformer [KKL20], and MONGOOSE [CLP$^+$21], that can save the running time empirically. However, they lack provable guarantees of efficiency and accuracy. Thus, A gap exists in theoretically understanding fast neural network training methods.

- **Question 5:** *How to address security concerns emerging from machine learning?* The rapid advancements of AI and machine learning increase the importance of ensuring security in several aspects. For example, when private data (e.g., medical images or financial information) is used in training a deep model, we are facing the challenge of protecting data privacy during training and inference. Also, the developments of generative AIs [BMR$^+$20, RDN$^+$22] pose potential intellectual property issues. Furthermore, the evolving role of quantum computing in machine learning has both positive and negative impacts on ML security. On the one hand, it may introduce extra power to attack existing security protocols, thereby demanding the development of post-quantum secure schemes. Conversely, it provides opportunities to design new cryptographic objects that are unachievable in the classical world, such as quantum money [AC12, Zha19] and quantum copy-protection [Aar09, CMP20]. Thus, how to leverage quantum advantages to assist in resolving machine learning security issues remains an open and salient question.

1.1 Organization

In this thesis, we develop theoretical insights into these questions and make progress toward resolving them. Specifically,

1. In Section 1.2, we introduce our contributions to quantum computing, focusing on answering Question 1 and Question 2. For the first question, we present quantum algorithms for optimization and machine learning with quantum speed-ups

over the currently best classical algorithms. And we generalize several techniques in classical complexity theory to quantum and develop new approaches to studying the quantum hardness of learning and other natural problems. For the second question, we propose resource-efficient quantum algorithms for Hamiltonian ground state estimations with provable guarantees and for solving network flow problems. This is the subject of Chapters 2 to 8, based on the following seven works:

- A. M. Childs, T. Li, J. P. Liu, C. Wang, R. Zhang. Quantum Algorithms for Sampling Log-Concave Distributions and Estimating Normalizing Constants. *Advances in Neural Information Processing Systems 35 (NeurIPS 2022).*
- T. Li, R. Zhang. Quantum Speedups of Optimizing Approximately Convex Functions With Applications to Logarithmic Regret Stochastic Convex Bandits. *Advances in Neural Information Processing Systems 35 (NeurIPS 2022).*
- G. Wang, D. S. França, R. Zhang, S. Zhu, P. D. Johnson. Quantum Algorithm for Ground State Energy Estimation Using Circuit Depth With Exponentially Improved Dependence on Precision. *arXiv:2209.06811.*
- R. Zhang, G. Wang, P. D. Johnson. Computing Ground State Properties With Early Fault-Tolerant Quantum Computers. *Quantum, 6:761, 2022.*
- Y. Zhang, R. Zhang, A. C. Potter. QED Driven QAOA for Network-Flow Optimization. *Quantum, 5:510, 2021.*
- S. Aaronson, N. H. Chia, H. H. Lin, C. Wang, R. Zhang. On the Quantum Complexity of Closest Pair and Related Problems. *Proceedings of the 35th Computational Complexity Conference (CCC 2020).*
- N. H. Chia, C. N. Chou, J. Zhang, R. Zhang. Quantum Meets the Minimum Circuit Size Problem. *Proceedings of the 13th Innovations in Theoretical Computer Science Conference (ITCS 2022).*

2. In Section 1.3, we introduce our contributions to classical optimization, specifically addressing Question 3. We improve the state-of-the-art classical algorithms for solving SDPs with high accuracy. And our classical key technique also serves as a stepping stone to a new quantum SDP algorithm. And we improve the efficiencies of Fourier set-query and Fourier interpolation algorithms, which have strong connections to early fault-tolerant quantum algorithms. In addition, we design a dynamic distance oracle for general symmetric norms, which has wide applications in machine learning. This is the subject of Chapters 9 to 13, based on the following five works:

    - B. Huang, S. Jiang, Z. Song, R. Tao, R. Zhang. Solving SDP Faster: A Robust IPM Framework and Efficient Implementation. *Proceedings of the 63rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2022).*
    - B. Huang, S. Jiang, Z. Song, R. Tao, R. Zhang. A Faster Quantum Algorithm for Semidefinite Programming via Robust IPM Framework. *arXiv:2207.11154.*
    - Z. Song, B. Sun, O. Weinstein, R. Zhang. Sparse Fourier Transform Over Lattices: A Unified Approach to Signal Reconstruction. *arXiv:2205.00658.*
    - Z. Song, B. Sun, O. Weinstein, R. Zhang. Quartic Samples Suffice for Fourier Interpolation. *Proceedings of the 64th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2023).*
    - Y. Deng, Z. Song, O. Weinstein, R. Zhang. Fast Distance Oracles for Any Symmetric Norm. *Advances in Neural Information Processing Systems 35 (NeurIPS 2022).*

3. In Section 1.4, we introduce our contributions to machine learning, specifically addressing Question 4 and Question 5. First, we present classical and quantum algorithms for training wide two-layer and multi-layer neural networks. These

algorithms not only improve the time complexity of traditional training methods but also offer provable optimization guarantees. Second, we study machine learning security from aspects: protecting private training data and protecting copyrights of ML models. We identify a new mathematical problem underlying the security of a recently proposed private training scheme and theoretically analyze the time complexity and sample complexity of attacking this scheme. In addition, we propose a quantum approach to copy-protect a large family of programs that is post-quantum secure relative to a classical oracle. This is the subject of Chapter 14 to 18, based on the following six works:

- Z. Song, S. Yang, R. Zhang. Does Preprocessing Help Training Over-Parameterized Neural Networks? *Advances in Neural Information Processing Systems 34 (NeurIPS 2021).*
- J. Alman, J. Liang, Z. Song, R. Zhang, D. Zhuo. Bypass Exponential Time Preprocessing: Fast Neural Network Training via Weight-Data Correlation Preprocessing. *arXiv:2211.14227.*
- Z. Song, L. Zhang, R. Zhang. Training Multi-Layer Over-Parametrized Neural Network in Subquadratic Time. *arXiv:2112.07628.*
- B. Huang, Z. Song, R. Tao, R. Zhang, D. Zhuo. InstaHide's Sample Complexity When Mixing Two Private Images. *arXiv:2011.11877.*
- S. Chen, Z. Song, R. Tao, R. Zhang. Symmetric Sparse Boolean Matrix Factorization and Applications. *Proceedings of the 13th Innovations in Theoretical Computer Science Conference (ITCS 2022).*
- S. Aaronson, J. Liu, Q. Liu, M. Zhandry, R. Zhang. New Approaches for Quantum Copy-Protection. *Proceedings of 41st Annual International Cryptology Conference (CRYPTO 2021).*

4. In Section 1.5, we introduce our contributions to concentration and discrepancy theory, which complement the first three parts and lay a mathematical ground-

work for future research in quantum computing, optimization, and machine learning. We prove several Chernoff-type concentration inequalities in the setting of the hyperbolic polynomial, which is one of the most important objects in the field of *the geometry of polynomials*. We also extend several Kadison-Singer-type discrepancy results to hyperbolic polynomials. Furthermore, we study the concentration properties of the higher-order random walks on the expanding posets, which is a sub-class of high-dimensional expanders that generalizes the simplicial complexes. This is the subject of Chapter 19 to 21, based on the following three works:

- Z. Song, R. Zhang. Hyperbolic Concentration, Anti-concentration, and Discrepancy. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (RANDOM 2022).*
- R. Zhang, X. Zhang. Hyperbolic Extension of Kadison-Singer Type Results. *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023).*
- J. Gaitonde, M. Hopkins, T. Kaufman, S. Lovett, R. Zhang. Eigenstripping, Spectral Decay, and Edge-Expansion on Posets. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (RANDOM 2022).*

1.2 Our Contributions in Quantum Computing

Our research on quantum computing consists of two parts: 1) designing new quantum algorithms for solving optimization and machine learning problems, and 2) proposing new approaches to characterizing the limitations of quantum computing. More specifically, in Section 1.2.1, we highlight our quantum algorithms for sampling (and related problems), Hamiltonian's ground-state estimation, and network flow problems. In Section 1.2.2, we briefly talk about the quantum fine-grained complexity and quantum meta-complexity.

1.2.1 Quantum algorithms for optimization and machine Learning

In the past few years, a large number of quantum algorithms have been proposed to solve problems in many different areas, including linear algebra [HHL09, LC17, GSLW19, LC19, MRTC21], optimization [CCLW20, AGGW20, AG19, KP20b, KPS21], machine learning [RML14, KP17, AM20, KP20a, WZL+22], quantum chemistry [TMZE+18, MMS+19, Cam21], scientific computing [ALL+21, TAWL21, LKK+21], etc. In this dissertation, we further extend the line of quantum algorithm research. We first leverage quantum computing to solve sampling problems and achieve quantum speedups compared to current classical algorithms. Furthermore, we also consider the resource-limited quantum computing scenario, which closely resembles practical constraints where a small-scale, noisy quantum device might be built soon, while a fully fault-tolerant, large-scale quantum computer may not be feasible within a few decades. We propose small quantum circuit depth approaches to achieve high-accuracy estimation on the ground-state energy and properties. And we also develop a QAOA-based algorithm for solving network flow problems.

Log-concave sampling and normalizing constant estimation Log-concave sampling is an important problem in machine learning, physics, statistics, etc., where you are given query access to a d-dimensional convex function f, and the goal is to sample from a distribution with density $\propto e^{-f(x)}$. Some examples of log-concave distributions include a high-dimensional Gaussian distribution and the uniform distribution over a convex body.

More formally, we assume that $f : \mathbb{R}^d \to \mathbb{R}$ is L-smooth and μ-strongly convex, i.e., $L, \mu > 0$ are such that $\mu I \preceq \nabla^2 f(x) \preceq LI$. And we denote by $\kappa := L/\mu$ the condition number of f, which is an important parameter in the sample complexity. The corresponding log-concave distribution has probability density $\rho : \mathbb{R}^d \to \mathbb{R}$ as follows:

$$\rho(x) := Z^{-1} \cdot e^{-f(x)}, \quad \text{where} \quad Z := \int_{\mathbb{R}^d} e^{-f(x)} dx.$$

Given an $\epsilon \in (0,1)$, the goal of *log-concave sampling* is to output a random variable with distribution $\widetilde{\rho}$ such that the total variation (TV) distance $\|\rho - \widetilde{\rho}\| \leq \epsilon$, using as few queries to the evaluation and gradient oracles of f as possible. A fundamental tool in the classical log-concave sampling area is the *Langevin diffusion*, which is defined by the solution $(X_t)_{t>0}$ of the following stochastic differential equation (SDE):

$$dX_t = \underbrace{-\nabla f(X_t)dt}_{\text{Gradient flow}} + \underbrace{\sqrt{2}dB_t}_{\text{Brownian motion}} . \tag{1.1}$$

And it is well-known that Langevin diffusion will converge to the target log-concave distribution $\rho(x)$. However, a straightforward discretization of Langevin diffusion will introduce a *bias* to the stationary distribution [DMM19]. To resolve this issue, the Metropolis-adjusted Langevin algorithm (MALA) was developed, which applies the Metropolis-Hastings accept-reject mechanism in each iteration. It not only corrects the bias but also mixes very fast: [LST20] proves that MALA can achieve ϵ-TV distance using $\widetilde{O}_\epsilon(\kappa d)$ classical queries.

In this dissertation, we proposed a quantum algorithm to speed up MALA. Suppose we have access to the *quantum evaluation oracle* and *quantum gradient oracle* of f defined as follows:

$$\begin{aligned} \mathcal{O}_f|x\rangle|y\rangle &= |x\rangle|y + f(x)\rangle \quad \forall x \in \mathbb{R}^d, y \in \mathbb{R}, \\ \mathcal{O}_{\nabla f}|x\rangle|y\rangle &= |x\rangle|z + \nabla f(x)\rangle \quad \forall x \in \mathbb{R}^d, z \in \mathbb{R}^d. \end{aligned}$$

And we focus on preparing the quantum state (so-called a "qsample") that encodes the target log-concave distribution[1]:

$$|\rho\rangle = \int_{\mathbb{R}^d} \sqrt{\rho(x)}|x\rangle dx.$$

Our quantum algorithm can prepare a state that is ϵ-close to $|\rho\rangle$ using $\widetilde{O}_\epsilon(\sqrt{\kappa}d)$ quantum queries, achieving a quadratic quantum speedup in terms of κ compared

[1]If we measure the state $|\rho\rangle$ in the standard basis, the measurement outcome will be distributed according to the distribution ρ.

to the classical result [LST20]. Moreover, we also consider the *warm-start* scenario, where we get access to some initial distribution that is not too far from the target distribution.[2] Classically, the currently best sample complexity is $\widetilde{O}_\epsilon(\kappa\sqrt{d})$ due to [WSC22]. Our quantum algorithm only uses $\widetilde{O}_\epsilon(\kappa^{1/2}d^{1/4})$ quantum queries.

A closely related problem is the *normalization constant estimation*, whose goal is to output a value $\widetilde{Z}$ such that with probability at least 2/3, $(1-\epsilon)Z \le \widetilde{Z} \le (1+\epsilon)Z$. This problem is also known as the partition function estimation in physics, where the distribution is called the Gibbs distribution. For discrete systems, partition function estimation has been studied in the quantum setting [WCNA09, HW20]. However, for continuous systems, we are the first (to the best of our knowledge) to develop quantum algorithms for estimating normalizing constants. In this dissertation, we combine our quantum MALA with simulated annealing and non-destructive mean estimation to achieve an ϵ-relative error using $\widetilde{O}(\kappa^{1/2}d^{3/2}\epsilon^{-1})$ queries, while classical approach based on MALA uses $\widetilde{O}(\kappa d^2\epsilon^{-2})$ queries [GLL20]. To further reduce the dependence on d, we also propose quantum versions of underdamped Langevin dynamics (ULD) and the randomized middle-point method for ULD (ULD-RMM). Together with the quantum-accelerated multi-level Monte-Carlo, we achieve $\widetilde{O}(\kappa^2 d^{3/2}\epsilon^{-1})$ and $\widetilde{O}((\kappa^{7/6}d^{7/6} + \kappa d^{4/3})\epsilon^{-1})$, respectively. Compared to the classical results [GLL20], our quantum algorithms (based on ULD and ULD-RMM) quadratically speed up the ϵ-dependence and do not require the quantum gradient oracle (i.e., they can be viewed as zeroth-order sampling methods). In addition, we also prove a quantum query lower bound $1/\epsilon^{1-o(1)}$ by reducing the normalizing constant estimation problem to the *Hamming weight problem* [NW99], which demonstrates that our quantum algorithms have a nearly-optimal error dependence. See Chapter 2 for more details.

Sampling beyond the log-concave regime In practice, many problems are not convex, and it is very important to understand non-convex optimization from both

[2]In practice, such "warm" initial distributions can be prepared using other (possibly heuristic) sampling methods.

empirical and theoretical perspectives. It is the same for the sampling problem. Consider a sampling task for the distribution $\rho(x) \propto \exp(-f(x))$. When f is non-convex, this problem is classically hard, and the Langevin diffusion (Eq. (1.1)) will take an exponential time to converge in the worst-case. Recently, there are a lot of classical works focusing on non-log-concave sampling for some special families of distributions (e.g., those satisfying some functional inequality, such as log-Sobolev inequality or Poincaré inequality [VW19, Wib19, LE20, CEL+21, MCC+19, AC23, FYC23]; or with certain tail-growth conditions [DM17, CCAY+18, LWME19, MMS20, EH21, HBE22]).

Then, it is natural to ask: *is there any quantum advantage for sampling in the non-log-concave regime?* In general, this question might be difficult to answer since many classical non-log-concave sampling algorithms are heuristic or based on some *irreversible* process. In this dissertation, we consider a slightly non-log-concave regime: approximately log-concave sampling, where the distribution has a kernel function $F : \mathbb{R}^d \to \mathbb{R}$ satisfying

$$\sup_x |F(x) - f(x)| \leq \epsilon/d,$$

for some unknown convex function $f : \mathbb{R}^d \to \mathbb{R}$. By a standard optimization to sampling reduction, we can also ask to find an ϵ-minimizer for the function $F(x)$, which is denoted as the *approximately convex optimization* problem. This problem is motivated by the robustness of optimization algorithms, where $F(x)$ can be viewed as a noisy observation of the convex function $f(x)$. An algorithm for solving the approximately convex optimization will imply a convex optimization algorithm robust to arbitrary point-wise noise. Classically, the state-of-the-art algorithm for this problem is by [BLNR15], which takes $\widetilde{O}(d^{4.5})$ queries to the evaluation oracle of F.[3] In this dissertation, we propose a quantum algorithm that takes $\widetilde{O}(d^3)$ queries. The key

[3]Unlike the log-concave sampling, we do not get access to the gradient oracle here since F could be non-smooth due to the noise.

technique of our result is a quantum speed-up of the *hit-and-run walk* with respect to an approximately log-concave distribution.

A closely related problem is called the *stochastic convex optimization* problem. In this problem, the objective function $F : \mathbb{R}^n \to \mathbb{R}$ can be expressed as:

$$F(x) = f(x) + \xi_x \quad \forall x \in \mathbb{R}^d,$$

where f is an unknown convex function, and ξ_x is an independent sub-Gaussian random variable. The goal is to minimize f given query access to F. This problem has a lot of applications in optimization with private data [BLNR15], stochastic programming [DKS14], online learning [RSS12], etc. The current best classical algorithm (due to [BLNR15]) uses $\widetilde{O}(n^{7.5}/\epsilon^2)$ queries. To embed this problem in the quantum setting, we considered the quantum stochastic oracle $\mathcal{O}_f^{\mathsf{stoc}}$:

$$\mathcal{O}_f^{\mathsf{stoc}}|x, y\rangle = |x\rangle \int_{\mathbb{R}} \sqrt{g_x(z)}|y + f(x) + z\rangle dz \quad \forall x \in \mathbb{R}^n, y \in \mathbb{R}, \tag{1.2}$$

where g_x is the density function of the sub-Gaussian variable ξ_x. By combining our quantum hit-and-run walk sampler with a quantum sub-Gaussian mean estimator (due to [Ham21]), we obtain a quantum algorithm for the stochastic convex optimization with $\widetilde{O}(n^{5.5}/\epsilon)$ queries to $\mathcal{O}_f^{\mathsf{stoc}}$.

As an application, we also study the quantum version of the *zeroth-order stochastic convex bandit* problem, which is a widely studied bandit model (see, e.g., [HL16, Lat20, LG21]). An informal definition of the problem is as follows. Let $f : \mathbb{R}^d \to [0, 1]$ be a convex function. An online learner and environment interact alternatively over T rounds. In each round $t \in [T]$, the learner makes a query to the quantum stochastic evaluation oracle (Eq. (1.2)) and returns a point x_t as the current guess. The learner aims to minimize the regret

$$\mathcal{R}_T := \mathbb{E}\left[\sum_{t=1}^{T} \left(f(x_t) - \min_x f(x)\right)\right],$$

and the expectation is taken over all randomness. Based on our quantum stochastic convex optimization algorithm, we develop a quantum algorithm that achieves

$O(\text{poly}(d)\log^2(T))$ regret in expectation. For comparison, the classical regret lower bound is $\Omega(\sqrt{T})$ by [DHK09]. The exponential quantum advantage comes from the quadratically improved error dependence in the cost of the stochastic convex optimizer by our quantum algorithm. See Chapter 3 for more details.

Our results on quantum algorithms for the sampling-related problems are summarized in Table 1.1. We note that these algorithms require fully fault-tolerant quantum computers. In the following of this sub-section, we consider more quantum resource-efficient algorithms.

Early fault-tolerant ground-state energy and properties estimation A significant effort in applied quantum computing has been devoted to the problem of ground-state energy estimation (GSEE) for molecules and materials. A typical approach based on quantum singular value transformation (QSVT) can achieve nearly optimal time complexity [LT20a] but also requires very deep quantum circuits. On the other hand, we can also use some noisy intermediate-scale quantum (NISQ)-friendly approaches (e.g., the variational quantum eigensolver (VQE)) to approximate the ground state and estimate its energy by repeated measurements. However, some recent works [SFGP21, DPMRF23] show that VQE-based methods are not robust enough and have some limitations. Thus, we are looking for a new approach to *reliably* solve the GSEE problem using *small* quantum circuits.

This research direction was first studied in [LT22], where they proposed the *early fault-tolerant* regime to characterize the quantum resources needed for such algorithms. Roughly speaking, an early fault-tolerant algorithm could run some quantum circuits multiple times together with some classical pre-processing and post-processing procedures. Moreover, it is desired to satisfy the following properties:

1. The maximal quantum circuit depth is small.

2. The number of ancilla qubits is $O(1)$.

Problem	Method	Complexity
Log-concave sampling	MALA [LST20]	κd
	MALA with warm start [WSC22]	$\kappa d^{1/2}$
	Q-MALA (Thm 2.6)	$\kappa^{1/2} d$
	Q-MALA with warm start (Thm 2.36)	$\kappa^{1/2} d^{1/4}$
Normalization constant estimation	Annealing with MALA [GLL20]	$\kappa d^2 \epsilon^{-2}$
	Multilevel ULD [GLL20]	$\kappa^2 d^{3/2} \epsilon^{-2}$
	Multilevel ULD-RMM [GLL20]	$(\kappa^{7/6} d^{7/6} + \kappa d^{4/3}) \epsilon^{-2}$
	Q-annealing with Q-MALA (Thm 2.7)	$\kappa^{1/2} d^{3/2} \epsilon^{-1}$
	Multilevel Q-ULD (Thm 2.8)	$\kappa^2 d^{3/2} \epsilon^{-1}$
	Multilevel Q-ULD-RMM (Thm 2.8)	$(\kappa^{7/6} d^{7/6} + \kappa d^{4/3}) \epsilon^{-1}$
	Quantum lower bound (Thm 2.9)	$\epsilon^{-1+o(1)}$
Approximately convex optimization	Classical [BLNR15]	$d^{4.5}$
	Quantum (Thm 3.1)	d^3
Stochastic convex optimization	Classical [BLNR15]	$d^{7.5} \epsilon^{-2}$
	Quantum (Cor 3.2)	$d^{5.5} \epsilon^{-1}$
Stochastic convex bandit	Classical [DHK09, LG21]	$\Theta_d(\sqrt{T})$ (regret)
	Quantum (Thm 3.3)	$O_d(\log^2(T))$ (regret)

Table 1.1: Summary of classical and quantum algorithms for sampling and related problems. We omit all polylog$(\kappa, d, \epsilon^{-1})$ factors in the complexities. The quantum improvements are highlighted in blue color.

3. The gate set is simple, i.e., no multi-qubit controlled gate.

In this dissertation, we propose an early fault-tolerant quantum algorithm for GSEE that exponentially reduces the error dependence in the maximal circuit depth compared to the previous methods [LT22, WBC21a, DLT22]. More specifically, let H be a Hamiltonian (a giant complex matrix) with eigenvalues $E_0 \leq E_1 \leq \cdots \leq E_N$ such that the energy gap $E_1 - E_0 \geq \Delta$. Suppose we can prepare an initial state ρ with a non-trivial overlap with the ground state $|E_0\rangle$, i.e., $\langle E_0|\rho|E_0\rangle \geq \eta$. We further assume that we are given black-box access to the controlled *Hamiltonian evolution* c-$\exp(-2\pi \mathbf{i} H t)$, which is a commonly-used model. And the goal is to estimate the ground state energy E_0 (the minimum eigenvalue) within additive error ϵ. As is typical, the circuit depth and quantum runtime of the algorithm are measured in terms of the *Hamiltonian evolution time* t. The maximal evolution time of our algorithm scales as $\Delta^{-1} \cdot \operatorname{poly}\log(\Delta/\epsilon)$, while existing methods require maximal evolution time at least ϵ^{-1}. Our algorithm will be of great advantage to solving the GSEE for a large family of industrially-relevant Hamiltonians in quantum chemistry with $\Delta \gg \epsilon$. One caveat of our algorithm is that the total cost, in terms of the Hamiltonian evolution time, increases. However, from an implementation perspective, reducing the maximal evolution time is the foremost object since a lower circuit depth decreases the number of required physical qubits and minimizes the need for error correction infrastructure. Nevertheless, we also develop another algorithm with a tunable circuit depth between Δ^{-1} and ϵ^{-1} such that there exists a *smooth tradeoff* between the maximal evolution time and the total evolution time. In particular, if we set the circuit depth to be about ϵ^{-1}, the total evolution time of our algorithm has the same scaling as [LT22]. Additionally, relative to recent resource estimates of GSEE for the industrially-relevant molecules of ethylene-carbonate and PF_6^- [KLP+22], the estimated gate count and circuit depth is reduced by a factor of 43 and 78, respectively.

The key technique of our algorithms comes from *classical signal processing.*

For an initial state ρ, its spectral density measure is defined as:

$$p(x) := \sum_{j=0}^{N} p_j \delta(x - E_j),$$

where $p_j = \langle E_j|\rho|E_j\rangle$ is the overlap between ρ and the j-th eigenstate $|E_j\rangle$. On the other hand, by Fourier transform, we know that $p(x)$ is the discrete frequency spectrum of the following signal:

$$\hat{p}(t) = \sum_{j=0}^{N} p_j e^{-2\pi \mathbf{i} E_j t}.$$

By running the standard *Hadamard test* circuits with the unitary being the Hamiltonian evolution operator $\exp(2\pi\mathbf{i}Ht)$ on the initial state ρ, we obtain an unbiased estimator for $\hat{p}(t)$. Then, the GSEE problem is reduced to find the minimum frequency of a signal with *gaped, discrete* frequencies, given noisy observations in the time duration $[0, T]$. And correspondingly, minimizing the circuit depth is equivalent to minimizing the observing length T. Our main strategy is to apply a Gaussian derivative filter:

$$g_\sigma(x) = -\frac{1}{\sqrt{2\pi}\sigma^3} x e^{-\frac{1}{2}x^2/\sigma^2},$$

which will significantly suppress the magnitudes of higher frequencies and “highlight” the minimum frequency. In this way, the minimum frequency (i.e., the ground-state energy) can be accurately estimated by a grid search on the filtered signal. See Chapter 4 for more details.

For more complicated tasks in quantum chemistry, e.g., computing electron transport in materials or electric dipoles of molecules, we need to go beyond the ground state energy and estimate additional properties of the ground state (i.e., the expectation value of any observable). More specifically, the setting of the ground-state property estimation (GSPE) is similar to GSEE, but we are also given an observable O as a block-encoded unitary. And the goal is to estimate $\langle E_0|O|E_0\rangle$ within additive error ϵ. We use similar ideas to develop an early fault-tolerant algorithm for GSPE with maximal Hamiltonian evolution time depending *logarithmically* on the ϵ. Technically, the quantum circuit we use is a modification of the standard Hadamard test circuit, which inserts the (unitary) observable in between two parameterized Hamiltonian evolution operators. From the signal processing perspective, we "lift" the 1-D signal in GSEE to the following 2-D signal in order to deal with the observables that are non-commute with the Hamiltonian:

$$q(x,y) = \sum_{j,k=0}^{N} q_{j,k}\delta(x-E_j)\delta(y-E_k), \quad \text{and} \quad \widehat{q}(s,t) = \sum_{j,k=0}^{N} q_{j,k}e^{-2\pi\mathbf{i}(E_j s + E_k t)},$$

where $q_{j,k} := p_j p_k \langle E_j|O|E_k\rangle$. And the GSPE problem is reduced to estimating the coefficient of the minimum frequency of a signal. Then, we leverage the polynomial filter function proposed by [LT22] to recover the coefficient, which is of a higher degree but more accurate compared to the Gaussian derivative filter.

Our early fault-tolerant GSPE algorithm can be applied to solve a wide range of scientific problems. One example is to compute the charge density of a molecule, which gives many useful properties like electric dipole moments of a molecule [RGM+21]. Another example is to estimate the linear system properties (i.e., $x^\top Mx$ for any matrix M and $x = A^{-1}b$), where we can transform the linear system to a Hamiltonian such that the ground-state corresponds to the solution. Compared to the existing methods for these problems, our GSPE approach can significantly reduce the required quantum circuit depths. See Chapter 5 for more details.

QAOA for network flow problems Quantum approximate optimization algorithm (QAOA) is another kind of resource-efficient quantum algorithm with potential applications to solve practical combinatorial optimization problems (e.g., resource allocation, job scheduling, and portfolio optimization). It is a quantum/classical hybrid algorithm that uses a parameterized quantum circuit to prepare a quantum state that encodes the approximate solution and uses some classical optimization algorithms (e.g., Adam [KB15], L-BFGS-B [BLNZ95], COBYLA [Pow07]) to optimize the parameters.

In this dissertation, we presented a general framework for modifying QAOA to solve constrained network flow problems (more specifically, the *edge-disjoint path problem*). The key observation is an analogy between flow constraints and Gauss's law for electromagnetism. We designed lattice quantum electrodynamics (QED) inspired mixing operators in QAOA that resulted in an exponential reduction in the size of the search space. We also showed through numerical simulations that our algorithm can yield higher quality approximate solutions compared to the original QAOA routine. We believe our new QAOA approach will be useful in solving large-scale optimization problems on graphs in the near future. See Chapter 6 for more details.

1.2.2 New approaches to quantum complexity theory

Quantum complexity aims to characterize the advantages and limitations of quantum computers. However, for many practical and natural problems, their quantum complexities remain mysterious. On the other hand, classical complexity theory has been more thoroughly studied, and several approaches have been developed to capture the hardness of discrete optimization and learning problems. In this part, we will introduce two new approaches that "lift" classical techniques to help us understand quantum complexity theory.

Quantum fine-grained complexity Previous studies of quantum computing focused on distinguishing problems in BQP from those that require exponential time. However, for problems with real-world applications (e.g., those in optimization and machine learning), only knowing they can be solved in quantum polynomial-time is not enough, and what we need are *really fast* quantum algorithms. This motivates the following question: *Is there a "finer" theory to characterize the "exact" quantum complexity of natural problems?*

In this dissertation, we propose the quantum fine-grained complexity framework. As an example, we revisited a fundamental problem in computational geometry—the closest pair (CP) problem: given a set of n points in a d-dimensional space, find a pair with the smallest distance. For CP in $O(1)$-dimension, we gave an $\widetilde{O}(n^{2/3})$-time quantum algorithm, which is optimal due to the quantum lower bound for the element distinctness problem [AS04] and beats the classical $O(n \log n)$-time algorithm in the textbook (e.g., [CLRS09]). Our algorithm is based on the quantum walk on the subsets of points (i.e., a Johnson graph). For time efficiency, we develop novel *history-independent* quantum data structures to dynamically update the points and maintain the closest pair. Here, the term "history-independent" indicates that the memory layout of the data structure solely depends on the stored data, specifically the maintained points, and does not consider the sequence of previous operations.

For CP in $\omega(\log n)$-dimension, the straightforward Grover's algorithm takes $\widetilde{O}(n)$-time, and we prove that it is indeed nearly optimal. Similar to the classical fine-grained complexity, our quantum lower bound assumes the quantum strong exponential-time hypothesis (QSETH), which roughly conjectures that k-SAT cannot be solved by a $2^{(1/2-o(1))n}$-time quantum algorithm when k is large enough. The QSETH is based on an observation that our current non-trivial quantum speedups for k-SAT only hold for $k = 3$ or some small constants. For larger k, we currently can only have Grover speedup. Our quantum fine-grained lower bound for CP follows from the following chains of reductions:

$$k\text{-}\mathsf{SAT} \leq \mathsf{OV} \leq \mathsf{BCP} \leq \mathsf{CP},$$

where OV denotes the *orthogonal vector problem*, and BCP denotes the *bichromatic closest pair problem*. Since the quantum fine-grained lower bounds are sublinear in the input length, we also propose the quantum fine-grained reduction to reduce the reduction time via quantum oracles. Additionally, we also apply the quantum fine-grained complexity framework to study the complexity of OV, BCP, and approximate-BCP in different dimension regimes. See Chapter 7 for more details.

Quantum meta-complexity Meta-complexity is the complexity of problems about complexity. One important problem is the minimum circuit size problem (MCSP), which asks to determine the circuit complexity of an input Boolean function. While MCSP has been studied as early as the 1950s in the Russian cybernetics program, its complexity remains mysterious.

In this dissertation, we initiated the study of quantum meta-complexity. We first defined the quantum versions of MCSP for Boolean functions (MQCSP), quantum states (SMCSP), and unitaries (UMCSP). For example, MQCSP is roughly defined as follows: The input is the truth table of a Boolean function $f : \{0,1\}^n \to \{0,1\}$ and the size parameter s. The goal is to distinguish the following two cases:

- **Yes:** there exists a quantum circuit $\mathcal{C}$ of size at most s such that

$$\|(\langle f(x)| \otimes I)\mathcal{C}|x, \mathbf{0}\rangle\|^2 \geq 2/3 \quad \forall x \in \{0,1\}^n.$$

- **No:** for all quantum circuit $\mathcal{C}$ of size at most s,

$$\|(\langle f(x)| \otimes I)\mathcal{C}|x, \mathbf{0}\rangle\|^2 \leq 1/3 \quad \exists x \in \{0,1\}^n.$$

We remark that MQCSP is defined as a *promise problem* since quantum computation is inherently random and erroneous. We investigate the basic complexity-theoretic properties of these minimum quantum circuit size problems and show that these problems are not trivially in NP but in QCMA (or have QCMA protocols). Next, we explore the relations between the three quantum MCSPs and their variants. We discover that some reductions that are not known for classical MCSP exist for quantum MCSPs for unitaries and states, e.g., search-to-decision reduction and self-reduction, which mainly follow from the fact that quantum computation is reversible. Furthermore, we show that MQCSP is closely related to other important quantum problems in circuit lower bound, cryptography, learning theory, fine-grained complexity, as well as tomography, and quantum gravity. For example,

- *Circuit lower bound:* an efficient quantum algorithm for MQCSP implies that BPE and $\mathsf{BQP}^{\mathsf{QCMA}}$ do not have polynomial-sized quantum circuits.

- *Cryptography:* algorithms for MQCSP will break one-way function (OWF), and algorithms for SMCSP will break pseudorandom state (PRS).
- *Learning theory:* PAC-learning for quantum circuits is "equivalent" to an efficient randomized algorithm for MQCSP, and a quantum learning algorithm is "equivalent" to an efficient quantum MQCSP algorithm.
- *Fine-grained complexity:* MQCSP for partial Boolean functions cannot be solved by an $N^{o(\log\log N)}$-time quantum algorithm unless QETH[4] fails.
- *Quantum gravity:* estimating the volume of a wormhole can be reduced to solve an SMCSP instance (assuming the Quantum Extended Church-Turing Thesis).

Given the fundamental differences between classical and quantum circuits, our results require meticulous attention and unveil distinctive properties and phenomena specific to the quantum setting. Our findings hold significant potential for future studies of quantum meta-complexity. See Chapter 8 for more details.

1.3 Our Contributions in Optimization

Optimization is one of the most important playgrounds for demonstrating quantum advantages, and a long line of research focuses on quantum optimization algorithms (as we discussed in Section 1.2). Nevertheless, the study of classical optimization algorithms is still of great importance. On the one hand, many classical optimization algorithms are served as sub-routines in some quantum algorithms. For example, semi-definite programming (SDP) was used in the shadow tomography [Aar18]; sparse recovery was used in quantum benchmarking [HYF21] and phase estimation [YZT23]. On the other hand, developing more efficient classical algorithms

[4]QETH is similar to QSETH. It conjectures that $\mathsf{3\text{-}SAT}$ cannot be solved by a $2^{o(n)}$-time *quantum* algorithm.

could establish better baselines to justify the advantage of quantum optimization algorithms.

This section mainly introduces our contributions in classical optimization algorithms, which are also closely connected to quantum computing. In Section 1.3.1, we discuss our state-of-the-art SDP solver based on a robust interior-point method (IPM) framework, which is also the stepping stone to a high-accuracy quantum SDP solver. In Section 1.3.2, we introduce our efficient algorithms to reconstruct continuous, Fourier-sparse signals. This problem has strong motivation from early fault-tolerant quantum algorithms. In Section 1.3.3, we present an efficient dynamic distance estimation data structure for any general symmetric norm.

1.3.1 Semi-definite programming

Semi-definite programming (SDP) is of great interest both in theory and in practice, where many important problems can be modeled or approximated by SDPs. An SDP instance has the following form:

$$\max_{X\in\mathbb{R}^{n\times n}} \langle C, X\rangle \text{ subject to } \langle A_i, X\rangle = b_i, \quad \forall i\in[m],\ X\succeq 0,$$

where $\langle A, B\rangle := \sum_{i,j} A_{i,j}B_{i,j}$ is the matrix inner product, and $C, A_1, \ldots, A_m \in \mathbb{R}^{n\times n}$ are symmetric matrices. Under strong duality, the above primal formulation is equivalent to the following dual formulation:

$$\min_{y\in\mathbb{R}^m} b^\top y \quad \text{subject to } S = \sum_{i=1}^m y_i A_i - C, \qquad S\succeq 0.$$

Inspired by the best linear programming solver due to [CLS19], which runs in the current matrix multiplication time; ideally, we want to solve a general SDP instance with high accuracy in matrix multiplication time, i.e., $(mn^2+m^\omega+n^\omega)\log(1/\epsilon)$, where m is the number of constraints, n is the dimension of the matrices, ϵ is the accuracy parameter, and $\omega\approx 2.373$ is the fast matrix multiplication exponent [AW21]. Unfortunately, the state-of-the-art result due to [JKL$^+$20] is still $\sqrt{n}$-times slower.

In this dissertation, we improve the running time of [JKL+20]'s SDP solver to:

$$O^*\left((\sqrt{n}(m^2+n^4)+m^\omega+n^{2\omega})\cdot\log(1/\epsilon)\right),$$

where O^* hides $(mn)^{o(1)}$ terms. In particular, for the *tall dense* SDP instances (i.e., $m=\Omega(n^2)$), our SDP solver runs in the current matrix multiplication time m^ω. And we remark that there are many applications for tall dense SDPs (e.g., the sparsest cut [ARV09], the minimum uncut [ACMM05], the metric embedding [LLR95], etc.), and it is one of the two predominate cases in [JKL+20]. Moreover, the tall, dense case was a milestone for solving LP in matrix multiplication time [LS15].

Technically, [JKL+20]'s algorithm is based on the interior-point method (IPM), which constructs a path (so-called "central path") connecting the initial solution to the optimal solution. More formally, the central path is defined by the solution of the following unconstrained optimization problem parameterized by $\eta>0$:

$$\min_{y\in\mathbb{R}^m} f_\eta(y) \quad \text{where} \quad f_\eta(y):=\eta\cdot\langle b,y\rangle+\phi(y),$$

where $\phi(y)$ is a barrier function such that $\phi(y)\to\infty$ as y approaching to the boundary of the feasible region $\mathcal{K}=\{y\in\mathbb{R}^m \mid S=\sum_{i=1}^m y_iA_i-C\succeq 0\}$. And we can see that as $\eta\to\infty$, the central path converges to the optimal solution of the SDP instance. Hence, IPM starts with an initial feasible solution y for a small η, and increases η in each iteration. To keep y in the proximity of the central path, it takes a Newton step $\delta_y=-H(y)^{-1}g(y)$ in each iteration, where $g(y)=\nabla f_\eta(y)$ and $H(y)=\nabla^2 f_\eta(y)$ are the gradient and Hessian, respectively. Therefore, the total runtime of an IPM algorithm can be expressed as:

$$\mathcal{T}_{\mathrm{IPM}}=\#\text{iterations}\times\text{cost-per-iteration}.$$

The number of iterations depends on the *self-concordance* parameter of the barrier function $\phi(y)$. For a θ-self-concordant barrier function, the IPM takes $\sqrt{\theta}\log(\theta/\epsilon)$ iterations to obtain an ϵ-optimal solution. There are three choices of barrier functions for solving SDPs: logarithmic barrier, volumetric barrier, and hybrid barrier. Their definitions and self-concordance parameters are summarized in Table 1.2. Thus, for the log-barrier $\phi_{\log}(y)$ used in [JKL+20], it requires $\sim \sqrt{n}$ iterations to converge.

Ref.	**Barrier functions**	**Self-concordance**
[NN92, JKL+20]	$\phi_{\log}(y) = -\log\det(S)$	n
[Vai89a, NN94]	$\phi_{\mathrm{vol}}(y) = \frac{1}{2}\log\det(\nabla^2\phi_{\log}(y))$	$m\sqrt{n}$
[Ans00]	$\phi_{\mathrm{hyd}}(y) = c_1\phi_{\log}(y) + c_2\phi_{\mathrm{vol}}(y)$	$\sqrt{mn}$

Table 1.2: Barrier functions for solving SDPs. For the hybrid barrier, $c_1 = 225\sqrt{n/m}(n-1)/(m-1)$, and $c_2 = 225\sqrt{n/m}$.

The cost-per-iteration is dominated by the cost for computing the Hessian inverse $H(y)^{-1}$, which takes m^{ω}-time *per iteration*. Therefore, the total running time is at least $\sqrt{n} \cdot m^{\omega}$-time if we implement the IPM iterations exactly.

The high-level idea of our algorithm is to develop some numerical data structures to dynamically maintain the gradient $g(y)$, Hessian $H(y)$, and Newton's step δ_y in a fast but inexact way. More specifically, to speed up the computation of $H(y)^{-1}$, we apply a low-rank approximation for the *change* of the slack matrix S in each iteration, and use the Woodbury identity and the low-rank update for the Kronecker product to very efficiently update $H(y)$ and its inverse. Then, we propose a general amortization technique that can automatically determine the truncated rank of S in each iteration to optimize the total running time. However, the approximation errors in each IPM iteration could make the algorithm converge to some in-optimal solution. To rule out this bad case, we develop a *robust IPM framework* that guarantees the convergence of the IPM iterations even if there exist some errors in the gradient, Hessian, and Newton's step in each iteration. Based on our framework, we obtain a more efficient high-accuracy SDP solver. Additionally, we also apply the robust IPM framework to improve the hybrid barrier-based SDP solver [Ans00]. See Chapter 9 for more details.

On the other hand, there has been a long line of research on quantum SDP solvers [BS17, AGGW17, BKL+19, KP20b, KPS21, ANTZ21], but they only achieved low accuracy (or feasibility). On the one hand, some of these works quantized the first-order SDP solvers, whose running times intrinsically depend on $1/\epsilon$. For those quantum second-order methods, the bottleneck is that the gradient, Hessian, and Newton step computed in each iteration need to be transferred between quantum and classical memory, which incurs a $\mathrm{poly}(1/\epsilon)$ factor to the running time due to the cost of the high-accuracy quantum state tomography.

In this dissertation, we overcome the quantum bottleneck and show a high-accuracy quantum SDP solver with running time:

$$(mn^{1.5} + n^3) \cdot \mathrm{poly}(\kappa, \log(mn/\epsilon)),$$

where κ includes the condition numbers of the intermediate matrices. We note that our quantum SDP solver can achieve quantum speedups and high accuracy on a large family of well-conditioned SDP instances.

The main insight of our result is that, despite the requirement for state tomography, our robust IPM framework enables us to achieve quantum-classical data transfer in IPM iterations with only $O(1)$-accurate tomography. This approach circumvents the $\mathrm{poly}(1/\epsilon)$ barrier present in previous quantum SDP solvers. Then, by carefully implementing the IPM SDP algorithm using several techniques in quantum linear algebra, we obtain the above-stated total time complexity. See Chapter 10 for more details.

1.3.2 Sparse Fourier signal reconstruction

Fast Fourier transform (FFT) is one of the most important algorithms of the 20th century. However, many real-world signals are (almost) sparse, which motivates the study of Sparse Fourier transform (SFT). For discrete-time signals, SFT has been well-studied, with time/sample complexity nearly linear in the signal's sparsity [HIKP12a, HIKP12b, IKP14, NSW19]. But for continuous-time Fourier-sparse signals, i.e.,

$$x(t) = \sum_{j=1}^{k} v_j \exp(2\pi \mathbf{i} f_j t) \quad \forall t \in [0, T],$$

SFT is currently far from being well-studied. In addition to its essential real-world applications for continuous audio or radio signals, as we discussed in Section 1.2.1, Continuous SFT is closely related to the early fault-tolerant quantum algorithms for estimating a Hamiltonian's ground-state energy/properties. More specifically, given an initial quantum state as input, the energy information of the Hamiltonian is encoded in a continuous-time signal $x(t) = \sum_{j=1}^{N} p_j \exp(-2\pi \mathbf{i} E_j t)$, where E_j's are the eigenenergies of the Hamiltonian and p_j's are the overlaps between the initial state and the eigenstates. By running the Hadamard-test quantum circuit (with parameter t), we get a noisy observation of the signal at time t[5]. Thus, estimating the ground-state energy corresponds to locating the minimum frequency, and estimating the ground-state properties corresponds to reconstructing the coefficient.

Prior research shows that if we want to recover the ground-truth "tones" (i.e., $\{(v_j, f_j)\}$) from noisy observations, the frequency gap $\min_{i \neq j \in [k]} |f_i - f_j|$ must be large [Moi15, PS15]. This phenomenon is known as the "super-resolution". For a continuous-time Fourier-sparse signal with gapped frequencies, an interesting question is: can we reconstruct the coefficients corresponding to a given set of frequencies? This problem is known as the *Fourier set-query* problem, and we expect an algorithm's cost only proportional to the size of the frequency set. [Kap17] studied this problem for discrete signals. In this dissertation, we develop a unified framework for the Fourier set-query problem, which consists of the following four steps:

- **Step 1:** Proving energy bound for the signal family, which controls the sample complexity of the algorithm.

[5]More precisely, each observation can be written as $x(t) + z_t$, where $\mathbb{E}[z_t] = 0$ and $|z_t| \leq 1$ is independent random noise.

- **Step 2:** Obliviously sketching the signal, which draws random samples to shrink the continuous universe to a discrete set.
- **Step 3:** Distilling the sketch via some carefully designed sub-sampling distribution, which effectively minimizes the required number of samples.
- **Step 4:** Applying a weighted linear regression on the sketch to recover the signal.

Then, we instantiate this framework for continuous and discrete Fourier set-query in one-dimension and high-dimension, which are briefly summarized in Table 1.3. Compared to [Kap17]'s discrete Fourier set-query algorithm, we achieve the same sample complexity. And more importantly, our time complexity depends linearly on d, while [Kap17] has an exponential dependence on d due to the curse of dimensionality. See Chapter 11 for more details.

Signals	**Sample**	**Time**
1-D cont.	k	$k^{\omega+1}$
	$\epsilon^{-1}k$	$\epsilon^{-1}k^{\omega}$
d-D cont.	k	$k^{O(d)}$
	$\epsilon^{-1}k^{O(d)}$	$\epsilon^{-1}k^{O(d)}$
d-D disc.	$\epsilon^{-1}k$	$\epsilon^{-1}(k^{\omega+1}+k^{\omega-1}d)$

Table 1.3: Summary of our Fourier set-query results. For continuous signals, we propose a sample-efficient algorithm and a high-accuracy algorithm. ϵ is the error parameter. k is the output-sparsity of the algorithms. For discrete signals, $k = |L|$, the size of the queried frequency set. And for continuous signals, $k \propto |L|$ and also depends on the frequency gap.

Another research problem is the SFT without the frequency gap assumption. In this setting, we can only reconstruct a signal that "looks like" the ground-truth signal, but the tones could be very different, which is called the *Fourier interpolation* problem. More formally, given noisy access to the ground truth k-Fourier-sparse signal $x^*(t)$ in limited time duration $t \in [0, T]$, the goal is to reconstruct a $\widetilde{k}$-Fourier-sparse signal $y(t)$ such that

$$\|y - x^*\|_T^2 := \frac{1}{T}\int_0^T |y(t) - x^*(t)|^2 dt \le c \cdot \left(\|g\|_T^2 + \delta\|x^*\|_T^2\right),$$

where $g(t)$ is the noise in the observations, c is the approximation ratio, and $\widetilde{k}$ is the output sparsity. while there is a polynomial-time algorithm for Fourier interpolation (due to [CKPS16]), it is not very efficient (with at least k^{51} sample complexity, k^{62} running time, $\widetilde{k} \ge k^{10}$ output sparsity, and $c \ge 2000$ approximation ratio.), let alone practical. An open question is: can we improve the efficiency and accuracy of the Fourier interpolation algorithm?

In this dissertation, we push forward the research on this problem. We observe that the Fourier interpolation problem can be divided into two tasks: frequency estimation and signal estimation. During frequency estimation, we identify a set of $\widetilde{k}$ candidate frequencies such that there exists a signal $y^*(t)$ supported on these frequencies and well-approximates the ground-truth signal $x^*(t)$. Subsequently, in the signal estimation phase, we reconstruct the coefficients of the signal $y^*(t)$.We propose another algorithm that improves the sample complexity to $\widetilde{O}(k^4)$, time complexity to $\widetilde{O}(k^{4\omega})$, and output sparsity to $\widetilde{O}(k^4)$, where $\omega \approx 2.372$ is the fast matrix multiplication exponent. It is worth noting that the current state-of-the-art sample complexity of Fourier interpolation is $\sim k^4$, but was only known to be achieved by an *exponential-time* algorithm [CP19a]. In contrast, our algorithm uses the same number of samples but runs in polynomial time, presenting a significant advancement toward an efficient Fourier Interpolation solution. Technically, at the core of our algorithm lies a new spectral analysis tool called the *Signal Equivalent Method*, which leverages the inherent structure of Fourier signals to establish energy properties that are nearly optimal. The utilization of this method becomes instrumental in achieving both efficient and accurate frequency estimation. Complementing this technique, we introduce a new criterion for frequency recovery, called the *high signal-to-noise ratio (SNR) band* condition, which effectively prunes unnecessary frequencies from the observed signal while preserving accuracy. By combining these components, we devise a cheap algorithm capable of estimating "significant" frequencies within a narrow range. Finally, we incorporate a signal estimation procedure to complete the Fourier interpolation task. See Chapter 12 for more details.

1.3.3 Dynamic distance estimation

Estimating and detecting similarities in datasets is a basic subroutine in most industry-scale optimization and machine-learning applications. It motivates the notion of distance oracles (DO) [Pel00, GPPR04, WP11], which asks for a small-space data structure such that after preprocessing a dataset of n points $\{x_1, \dots, x_n\}$ in $\mathbb{R}^d$, in each query, given any point $q \in \mathbb{R}^d$ and a subset $S \subset [n]$, it can very efficiently estimate all distances $\|q - x_i\|_l$ for all $i \in S$ (faster than the trivial $O(d|S|)$-time). By considering the inner-product distance $\langle x_i, q \rangle$, DO can be viewed as generalizing matrix-vector product $X \cdot q$, and we aim for this computation to be accomplished in significantly less than nd time.

For ℓ_1-norm and ℓ_2-norm, the standard dimension-reduction (sketching) [JL84, AC09, LDFU13] provides very efficient solutions for DO. For ℓ_p norms, the DO problem is well-understood [BYJKS04], where the standard tool for constructing the data structure is via randomized *linear sketching* [SS02, BYJKS04, Ind06, KNPW11, CN20]. Nevertheless, in numerous real-world scenarios, these metrics may not adequately capture the underlying similarities between data points. Extensive research has shown that employing more intricate metrics can substantially enhance prediction accuracy and data compression [DKJ+07]. Over the past decade, many efforts have been dedicated to extending optimization problems beyond the common metrics [ANN+17, ANN+18, LSV18, SWY+19, SWZ19]. These endeavors motivate us to study DO in more general distance metrics.

In this dissertation, we propose a dynamic DO for any general symmetric norm, which is invariant under permutations on coordinates. Some examples of symmetric norms include Orlicz norms, top-k norms, max-mixture and sum-mixture of ℓ_p norms, small-support norms, and the box-norm, which can be very useful in modeling complicated real-world problems. More specifically, our data structure supports the following operations:

- *Initialization:* it takes $\widetilde{O}\,(n \cdot (d + \mathrm{mmc}(l))^2 \cdot \mathrm{poly}(1/\epsilon))$-time and space to preprocess the dataset $\{x_i\}_{i\in[n]} \subseteq \mathbb{R}^d$, where ϵ is the error parameter and $\mathrm{mmc}(l)$ is a complexity measure (concentration modulus) of the symmetric norm $\|\cdot\|_l$.
- *Query:* for any query point $q \in \mathbb{R}^d$ and subset $S \subseteq [n]$, it outputs a set of estimates $\{\mathrm{dst}\}_{i\in S}$ in $\widetilde{O}(d + |S| \cdot \mathrm{mmc}(l)^2)$-time, such that

$$(1-\epsilon)\|q - x_i\|_l \le \mathrm{dst}_i \le (1+\epsilon)\|q - x_i\|_l \quad \forall i \in S.$$

- *Update:* it takes $\widetilde{O}(d \cdot \mathrm{poly}(1/\epsilon))$-time to set $x_i \leftarrow z$.

And we note that when $l = \ell_p$, the running time of our result matches the state-of-art DOs. Our algorithm follows a “sketch-and-decode” approach. During the preprocessing phase, each data point $x_i \in \mathbb{R}^d$ is compressed into a linear sketch $\Phi x_i \in \mathbb{R}^{d'}$ with $d' \ll d$ while $\|\Phi x_i\|_l \approx \|x_i\|_l$. For each query, it first computes the sketch of the query point, i.e., Φq, and then decodes the distance $\|q - x_i\|_l$ from the difference $\Phi q - \Phi x_i$. To achieve this, We employ a sketching method based on the *layer approximation* [IW05, BBC+17]. Notably, we introduce several novel techniques, including shared randomness, locate-and-verify decoding, etc., to adapt this sketching method to the DO setting with highly efficient time and space complexities. See Chapter 13 for more details.

1.4 Our Contributions in Machine Learning

Machine learning has achieved incredible success over the past few years in many different areas. Even in quantum computing, machine learning has played an important role in learning quantum states [ACH+18, CHL+22] and predicting predict complex quantum systems [HKT+22, CHC+22]. Moreover, there has been a large amount of work on quantum algorithms for machine learning (e.g., [RML14, AdW17, KLP19, BBF+20, AHKZ20]). However, we currently do not fully understand why the large models will work so well, and several classical and quantum machine learning algorithms still lack provable guarantees.

Our research focuses primarily on the theoretical aspects of machine learning and aims to address fundamental challenges, such as efficient training, data privacy, and intellectual property protection. First, how to train large neural networks more efficiently has become a crucial concern in the era of giant AI models. In Section 1.4.1, we present several classical and quantum training algorithms with provable efficiency and convergence guarantees. Second, the success of AI heavily relies on large quantities and high-quality data, which also raises concerns about privacy. In Section 1.4.2, we discuss private data protection in a collaborative training experiment. Third, how to ensure the secure distribution of pre-trained models to clients while preserving ownership and intellectual property rights is another important question for AI companies. In Section 1.4.3, we introduce a new theoretical approach to copyright protection of ML models (and other types of programs) with the help of quantum computers.

1.4.1 Over-parameterized neural network training

Designing a fast and provable training method for neural networks stands as a fundamental and formidable challenge. Currently, most deep learning models are optimized using gradient descent or its variants. The overall training time can be divided into two components: the number of iterations and the cost-per-iteration. Several practical studies [CMJF+20, LXJ+20, CLP+21, DMZS21] have explored the use of nearest-neighbor search data structures to reduce cost-per-iteration. Empirically, these approaches have demonstrated significant advantages over straightforward training methods across various deep neural network architectures. However, none of these approaches provide a provable guarantee. This raises a crucial and natural question: *Is it possible to enhance the cost-per-iteration of neural network training algorithm theoretically?* Addressing this question and establishing a theoretical foundation for fast neural network training remains an open and promising area of research.

In this dissertation, we make some progress toward resolving this research question. We first theoretically improve the training algorithms for the *two-layer* neural network with the ReLU activation function. Let n denote the number of training data points in d-dimension, and m denote the number of neurons[6]. Then, the neural network can be expressed as:

$$f(x) = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} a_i \cdot \sigma(\langle w_i, x \rangle),$$

where $\sigma(z) = \max\{z, 0\}$ is the ReLU function, $a_i \in \{\pm 1\}$ is the output layer's weight, and $w_i \in \mathbb{R}^n$ is the weight of the i-th neuron. In each gradient descent (GD) iteration, the bottleneck is to compute the neural network predictions $f(x_i)$ for all n training data points, and each requires computing m inner-products in d-dimension. Hence, the cost-per-iteration in training neural networks (including both forward and backward computation) faces a natural barrier of $\Omega(mnd)$. We manage to overcome this barrier and present a classical and a quantum training algorithm with $O(m^{\alpha}nd)$ cost-per-iteration, where $0 < \alpha < 1$ is a universal constant. Furthermore, we can prove a linear convergence rate for our algorithms.

To bypass the $\Omega(mnd)$ barrier, we observe that the bottleneck of each iteration is the identify those neurons with non-zero outputs (or so-called "activated neurons"). It motivates us to explore the possibility of leveraging *quantum computing* to speed up this step. More specifically, Grover's search algorithm [Gro96] is one of the most

[6]We assumed the over-parameterization regime where the number of neurons is much larger than the number of training data points, i.e., $m \gg n$. It is a very common regime in deep learning theory [AZLS19a, AZLS19b, DZPS19], where the convergence and generalization of the gradient descent-based training algorithms have provable guarantees.

famous quantum algorithms and can find one specific element out of n elements in $\widetilde{O}(\sqrt{n})$-time. Moreover, if there are k such elements, Grover's algorithm can find all of them in $\widetilde{O}(\sqrt{nk})$-time. Therefore, a natural idea is to use Grover's algorithm to find the set of activated neurons, denoted as $\mathcal{S}_{\mathrm{fire}}$. Then, the forward computation can be written as follows:

$$f(x) = \frac{1}{\sqrt{m}} \sum_{i \in \mathcal{S}_{\mathrm{fire}}} a_i \cdot \sigma(\langle w_i, x \rangle).$$

Hence, the cost of the forward (and backward) computation only depends on $|\mathcal{S}_{\mathrm{fire}}|$, instead of m. When the number of activated neurons is small (i.e., sublinear in m), Grover's algorithm can offer a nearly quadratic speedup, and the training algorithm will have a $o(mnd)$ cost-per-iteration.

However, by the standard randomized initialization for the neural network, each w_i is sampled from a Gaussian distribution. As a result, about half of the neurons are activated, and we still need to take $\Omega(m)$-time to find all of them. Therefore, *sparsity* is the key to speed-up. In this dissertation, we design a shifted-ReLU-sparsifier by adding a small threshold in the ReLU activation function. That is, $\sigma_b(z) := \max\{z - b, 0\}$ for some small $b > 0$. Each training iteration is guaranteed to have $o(m)$ number of activated neurons. Furthermore, we developed a shifted neural tangent kernel (NTK) to provide a convergence analysis rigorously. In this way, we obtained a hybrid quantum-classical algorithm with a truly sublinear cost-per-iteration. Moreover, we observe that only the quantum part of our training algorithm is using Grover's algorithm to find $\mathcal{S}_{\mathrm{fire}}$. It can be formulated as a computational geometry problem: we are given n points in d-dimension. In each query, the input is a half-space, and the goal is to find all the points contained in this half-space. This problem has been studied by [AEM92]. They proposed *half-space reporting (HRS) data structures* such that after preprocessing the dataset, each query can be answered with sublinear costs.[7] Therefore, we can use the HRS data structures to replace Grover's

[7]More precisely, the costs are $o(n) + O(k)$, where k is the output size.

algorithm. By preprocessing the training data points or the initial weights, we obtain *purely classical* training algorithms with sublinear cost-per-iteration.

However, our sublinear-cost training algorithms are far from being practical since the HRS data structures are extremely complicated with very large preprocessing costs. In this dissertation, we also propose much-simplified tree-based data structures for preprocessing in the two-layer training algorithm so that the d-dependence in the preprocessing time is exponentially improved, with a slightly worse (but still sublinear) cost-per-iteration. In addition, we proved a strong exponential-time hypothesis (SETH)-based lower bound for the task of identifying the activated neurons, which indicates that one could not improve much on the running time of our data structures. See Chapter 14 for more details.

On the other hand, we theoretically improve the efficiency of training a multi-layer wide neural network, which is of the following form:

$$f(W, x) = a^\top \sigma(W_L(\sigma(\cdots \sigma(W_1 x)))),$$

where $W_1, \ldots, W_L$ are m-by-m matrices. Naively, one has to pay $\Omega(m^2)$ time to read the weight matrix and evaluate the neural network function in both forward and backward computation. We present a classical algorithm that only pays $O(m^2)$ in the preprocessing phase, and each iteration takes *truly sub-quadratic* time, i.e., $m^{1+\beta}$ for some universal constant $\beta \in (0, 1)$. Technically, we apply the shifted ReLU-based sparsifier to the multi-layer setting to reduce the number of activated neurons in each iteration. And we analyze the convergence behavior of a general Gram-based optimization framework, which provides the convergence guarantee for our training algorithm. Moreover, we develop a novel algorithm to efficiently and accurately solve tensor-based regression, which is a pivotal part of our training algorithm. See Chapter 15 for more details.

1.4.2 Classical protection for private training data

The concept of collaboratively training neural networks using sensitive data is highly attractive and applicable to numerous AI domains, ranging from healthcare to virtual assistants. One commonly employed strategy is to incorporate differential privacy [DMNS06, CMS11, ACG+16] into the training process by adding noise to gradients. This approach aims to prevent the trained model from containing sufficient information to reveal individual training data. Nevertheless, utilizing differential privacy in the training process often results in a notable reduction in test accuracy. Another strategy is to employ cryptographic tools like fully holomorphic encryption (FHE) or multiparty computation (MPC) to safeguard sensitive training data. However, the downside of these cryptographic protocols is that they often result in a substantial decrease in efficiency. Preserving data confidentiality, training efficiency, and prediction accuracy while training neural networks has emerged as a critical and common research objective in both academic and industrial domains [SS15, KMY+16, MMR+17, RTD+18, PAH+18].

InstaHide is a private distributed learning scheme proposed by [HSLA20b] that provides privacy without slowing down training or reducing accuracy. The key idea is to train the model on a dataset where (1) each synthetic image is a mix of k_{priv} private images and k_{pub} public images, and (2) each pixel is independently sign-flipped after the mixing. It shows promising prediction accuracy on several image data sets. To evaluate the security aspect of the *InstaHide* scheme in real-world deployment scenarios, the authors present a challenge [HSLA20a]. The challenge entails $n_{\mathsf{priv}} = 100$ private images, with the public images sourced from the ImageNet [DDS+09]. The challenge dataset comprises $m = 5000$ synthetic images, where each image is a combination of $k_{\mathsf{priv}} = 2$ private images and $k_{\mathsf{pub}} = 4$ public images, and the sign of each pixel is randomly flipped. The objective is to recover a private image using the set of sample images. Unfortunately, this challenge was broken by heuristic-based practical attacks [CDG+20, LXW+21], which are capable of reconstructing images *visually similar* to the private images in the *InstaHide* challenge data set. However, the key drawback of these attacks is that they lack provable guarantees, and it is unclear whether they still work when we use *InstaHide* to protect large numbers of private images (i.e., large n_{priv}). This raises a salient question: *Is InstaHide scheme secure in any provable sense?*

InstaHide's sample complexity [CLSZ21] is the first theoretical work that formulates the *InstaHide* attack problem as a recovery problem. They proposed an algorithm to recover a private image assuming each private and public image is a random Gaussian image (i.e., each pixel is an i.i.d. sample from $\mathbb{N}(0,1)$). The algorithm shows that $O(n_{\mathsf{priv}}^{k_{\mathsf{priv}}-2/(k_{\mathsf{priv}}+1)})$ sample images are sufficient to recover one private image.

In this dissertation, we proposed an algorithm that recovers *all* private images using only $\Omega(n_{\mathsf{priv}} \log(n_{\mathsf{priv}}))$ samples when mixing two private images (i.e., $k_{\mathsf{priv}} = 2$ as in the *InstaHide* challenge setting), improving [CLSZ21]'s algorithm which needs $O(n_{\mathsf{priv}}^{4/3})$ samples. Furthermore, we summarized the existing attacks into a unifying framework and revealed the vulnerability of the existing methods to recover all private images by proving a computational hardness of approximation result: ℓ_2-regression with hidden signs cannot be $(1+\epsilon)$-approximated in $2^{o(n)}$-time, assuming the exponential-time hypothesis (ETH). Our results demonstrate that *InstaHide* is not information-theoretically secure but computationally secure in the worst case, even in the simplest setup of mixing two private images. See Chapter 16 for more details.

Efficient InstaHide attacking algorithm with provable guarantee We observe that a key step in the existing *InstaHide* attacks can be formulated as the batched-k-vector sum ($\mathsf{BkV\text{-}Sum}$) problem: let $X = \{x_1, \ldots, x_r\}$ be a hidden set of d-dimensional vectors. We are given vectors $y_1, \ldots, y_m$ with the promise that for each $j \in [m]$, there is a set $S_j \subset [r]$ of size k for which $y_j = \sum_{i \in S_j} x_i$. The goal is to recover sets $S_1, \dots, S_m$. In the context of *InstaHide*, $r = n_{\mathsf{priv}}$ represents the number of private images, $k = k_{\mathsf{priv}}$ denotes the number of private images used for mixing, and the set S_j provides information about which private images were used to generate the j-th synthetic image. [CLSZ21, CDG+20] demonstrated that when x_i are Gaussian vectors or real images, it is possible to construct a *similarity oracle*. It takes any y_i and y_j as inputs, and returns $|S_i \cap S_j|$. In the presence of such an oracle, BkV-Sum can be reduced to the *sparse symmetric Boolean matrix factorization* problem (SSBMF). It is a variant of the well-studied *non-negative matrix factorization* problem [AGKM12, Moi13, RSW16, SWZ17, SWZ19]. In SSBMF, we are given a symmetric non-negative matrix $M \in \mathbb{R}_{\geq 0}^{m \times m}$, and the goal is to find an m-by-r Boolean matrix W with k-row-sparse such that $\|M - WW^\top\|_0$ is minimized. This problem is intriguing not only due to its connection to *InstaHide* attacks but also because it is closely related to other significant problems in combinatorics and theoretical computer science, such as hypergraph reconstruction, clique decomposition, and more. Thus, exploring this problem offers valuable insights into various research areas.

In this dissertation, we conduct systematic research on SSBMF. We first show that it is NP-hard in the worst case. And we present a quasi-polynomial-time algorithm for approximating SSBMF, based on the dense 2-CSP solver by [MM15]. Then, we study the SSBMF in the average case. More specifically, we assume that each row of the ground-truth W is randomly sampled from the set of k-sparse Boolean vectors, and the input matrix $M = WW^\top$. By a bootstrapping technique and Jennrich's tensor decomposition algorithm [HAR70, LRA93], we show that W can be reconstructed up to a permutation of its rows with high probability. In the analysis of our algorithm, we establish a novel result in random matrix theory concerning the probability of a rectangular k-sparse Boolean random matrix being singular. Furthermore, we leverage our average-case algorithm for SSBMF to obtain the first provable attack on InstaHide with a polynomial runtime in all parameters, including m, d, k_{priv}, and n_{priv}. More specifically, we show that when the sets S_j are uniformly random k-subsets and $m \geq \widetilde{\Omega}(rk)$, given a similarity oracle, there exists a $O(m^{\omega+1} + mrd)$-time algorithm to approximately recover the magnitudes of the "heavy" coordinates of every vector in the original data set $X = \{x_1, \dots, x_r\}$. Remarkably, this recovery can be achieved even if we only have access to the entrywise absolute values of the y_i's. In the context of *InstaHide*, this algorithm can recover the "significant pixels" in each private image, which are sufficient in many real-world scenarios. See Chapter 17 for more details.

1.4.3 Quantum protection for copyrights of machine learning models

Training large-scale ML models, such as GPT-4 [Ope23], necessitates vast amounts of training data and GPU resources. Consequently, many AI companies have adopted the business strategy of selling pre-trained models. However, a significant concern arises regarding the vulnerability of these sold models or programs to unauthorized copying and sharing. One conventional solution is to incorporate watermarking techniques into machine learning models [ZGJ+18, LNE+21, FAESS22], enabling the detection of pirated models. Unfortunately, this approach can adversely affect training efficiency and may not withstand transfer learning or fine-tuning, as minor modifications to the model could remove the watermark. Furthermore, achieving a provable security guarantee in this context is exceptionally difficult. Another common approach is to offer ML services through cloud platforms exclusively. However, reliance on cloud services can be inconvenient due to the risk of leaking private data during network communication and potential server outages. Therefore, developing a reliable and efficient approach to protect machine learning intellectual property has become a critical demand.

In this dissertation, we investigate a new approach to protect the copyrights of ML models using quantum computers. Provably secure copy-protection in the classical world is theoretically impossible since we cannot prevent adversaries from copying the program codes. In the quantum realm, we use the no-cloning theorem to propose a scheme that provides provable protection against piracy for any *unlearnable program*, resolving a long-standing open question in [AC12]. Notably, compared to the prior quantum copy-protection scheme [Aar09], which is relative to a quantum oracle, our protocol only relies on a classical oracle, making it more practical. By instantiating the classical oracle with post-quantum candidate obfuscation, we will obtain a heuristic construction of copy protection. Moreover, we establish that any program with a watermarking can be *copy-detected*, i.e., piracy may not be prevented but can be detected. Our quantum copy-protection results provide a new approach to theoretically copy-protecting a pre-trained ML model with the assistance of quantum computers, with the security guarantee that adversaries can only pirate it by training the model from scratch. See Chapter 18 for more details.

1.5 Our Contributions in Concentration and Discrepancy

Concentration inequalities play an important role in various fields of computer science and mathematics by demonstrating that random quantities are likely to be close to their means. The most popular concentration inequalities might be the Chernoff bound [Che52], which provides an exponential tail-bound for the sum of independent and identically distributed (*i.i.d.*) scalar random variables. One application of Chernoff bounds is the probabilistic method, which is a powerful technique for proving the existence of desired objects through random constructions (see [AS16]). However, it does not always yield optimal solutions. One such example is the discrepancy minimization problem, where we are given n subsets of $[n]$, and the goal is to find a bi-coloring of the elements that minimizes the maximum discrepancy or imbalance among the subsets. By Chernoff bound, we can easily show that a random coloring has discrepancy $O(\sqrt{n \log n})$. However, Spencer's famous result of "six standard deviations" [Spe85] states that there exists a bi-coloring with the discrepancy at most $6\sqrt{n}$. Discrepancy theory has important connections to other fields, such as theoretical computer science and optimization, and has found applications in both theory and practice [Cha00b].

More recently, many efforts have been devoted to extending these findings to the matrix settings. For instance, researchers have generalized scalar Chernoff bounds to matrix Chernoff bounds in studies such as [AW02, RV07, Tro12, MJC$^+$14, Tro15, Tro18]. Additionally, works like [Mek14, MSS15b, Coh16a, Brä18, KLS20, HRS21, DJR21, BJM22] have investigated matrix discrepancy in various settings.

We further extend the line of research on concentration and discrepancy. In Section 1.5.1, we introduce the hyperbolic polynomial, which is a natural generalization of matrix and has important applications in TCS and mathematics. We generalize several Chernoff-type and Spencer-type results to the hyperbolic setting. In Section 1.5.2, we introduce the expanding posets (eposets), which are closely related to the recent breakthrough of c3-LTC, NLTS conjecture, etc. We show the relationship between the concentration properties of the higher-order random walk and the underlying structure of the poset. Although these results may not have immediate applications in quantum computing, optimization, or machine learning, we believe that they hold promise for future studies and can contribute to advancing our understanding of these fields.

1.5.1 Hyperbolic-extensions of concentration and discrepancy

We say a real, multivariate homogeneous polynomial $h(x) \in \mathbb{R}[x_1, \dots, x_m]$ to be *hyperbolic* in direction $e \in \mathbb{R}^m$ if its univariate restriction $h(te - x) \in \mathbb{R}[t]$ has only real roots for any x. Hyperbolic polynomials were initially explored in the field of partial differential equations [Går51, Hor83, Kry95]. Güler [Gül97] was the first to discover its application in optimization, specifically hyperbolic programming (HP), which is a generalization of LP and SDP. Later, substantial efforts have been dedicated to developing efficient HP solvers (e.g., [RS14, Ren16, Ren19a]) and investigating the relationship between HP and SDP (e.g., [HV07, LPR05, Brä14, Sau18, RRSW19]). Moreover, hyperbolic polynomials serve as a powerful tool in resolving significant problems in mathematics and TCS. These include the Van der Waerden/Schrijver-Valiant conjecture [Gur06, Gur07], Kadison-Singer problem [MSS15b, Brä18], and the construction of bipartite Ramanujan graphs [MSS18].

An intriguing property of the hyperbolic polynomial is that it induces a norm known as the *hyperbolic norm*, which is a natural generalization of the matrix spectral norm. Specifically, the matrix spectral norm $\|X\|$ is defined as the maximum absolute value among the eigenvalues of X. Alternatively, $\|X\|$ corresponds to the largest root (in absolute value) of the polynomial $\det(tI - X) \in \mathbb{R}[t]$. Notably, the determinant polynomial $\det(X)$ for a real symmetric matrix X is hyperbolic in the direction I.[8] Hence, it motivates us to extend the notions of eigenvalue and spectral norm to all hyperbolic polynomials. Let $h(x) \in \mathbb{R}[x_1, \dots, x_m]$ be a hyperbolic polynomial in direction e. For any vector x, its *hyperbolic eigenvalues* are the roots of $h(te - x) = 0$, denoted as $\lambda_1(x) \geq \cdots \geq \lambda_d(x)$. And its hyperbolic norm is $\|x\|_h := \max_{i\in[n]}\{|\lambda_i(x)|\}$. In addition to the determinant polynomial, another example of the hyperbolic polynomial is $h(x) = \prod_{i=1}^m x_i$ with the hyperbolic direction $e = (1, \dots, 1)$. It is easy to see that the hyperbolic norm induced by this polynomial corresponds to the ℓ_∞-norm, i.e., $\|x\|_h = \|x\|_\infty$. Our research mainly lies in exploring the concentration and discrepancy properties with respect to the hyperbolic norm.

Hyperbolic concentration As we discussed earlier, concentration inequalities for random matrices have been studied for decades. The matrix Chernoff bound [Tro15] shows that for d-by-d symmetric matrices $X_1, \dots, X_n$ and a uniformly random Boolean vector $r \in \{\pm 1\}^n$,

$$\Pr_{r\sim\{\pm1\}^n}\left[\Big\|\sum_{i=1}^n r_i X_i\Big\| > t\right] \leq 2d \cdot \exp\Big(-\frac{t^2}{2\left\|\sum_{i=1}^n X_i^2\right\|}\Big) \quad \forall t > 0.$$

In this dissertation, we prove several Chernoff-type results for the hyperbolic norm. First, we show a nearly optimal hyperbolic Chernoff bound for Rademacher sums. Specifically, let h be an m-variate, degree-d hyperbolic polynomial in direction

[8] We vectorize n-by-n symmetric matrices as vectors in $\mathbb{R}^{n(n+1)/2}$.

e. Let $x_1, \ldots, x_n \in \mathbb{R}^m$ such that $\sum_{i=1}^n \|x_i\|_h^2 \le \sigma^2$. Then, we have

$$\Pr_{r\sim\{\pm 1\}^n}\left[\Big\|\sum_{i=1}^n r_i x_i\Big\|_h > t\right] \le 2\exp\Big(-\frac{ct^2}{\sigma^2\log(s+1)}\Big) \quad \forall t > 0.$$

By choosing different hyperbolic polynomials, this result can recover Hoeffding's inequality [Hoe63], the dimension-free vector-valued Bernstein inequality [Min17], and the matrix Chernoff bound [Tro15][9].

Second, similar to the matrix Chernoff bound for positive semi-definite (PSD) matrices ([Tro15]), we also prove a hyperbolic Chernoff bound for random vectors in the hyperbolic cone Γ_+^h, which is defined to be the set of vectors with all hyperbolic eigenvalues being non-negative. Suppose $\mathsf{x}_1, \ldots, \mathsf{x}_n$ are n independent random vectors over Λ_+ such that $\lambda_{\max}(\mathsf{x}_i) \le R$ for all $i \in [n]$ and $\sum_{i=1}^n \mathbb{E}[\lambda_{\max}(\mathsf{x}_i)] = \mu_{\max}$. Then, we have

$$\Pr\left[\lambda_{\max}\Big(\sum_{i=1}^n \mathsf{x}_i\Big) \ge (1+\delta)\mu_{\max}\right] \le \; d\cdot\Big(\frac{(1+\delta)^{1+\delta}}{e^\delta}\Big)^{-\mu_{\max}/R} \quad \forall \delta \ge 0.$$

In addition, we have a similar result for the minimum hyperbolic eigenvalue.

Third, we delve into the anti-concentration with respect to the hyperbolic norm. Anti-concentration provides an opposite perspective of concentration inequalities by focusing on how random variables avoid clustering or becoming overly concentrated in specific regions. A notable example is the Littlewood-Offord theorem [LO43], which states that the probability of a random hypercube vertex laying on the boundary of any d-dimensional half-plane $\mathbf{1}_{\langle a,x\rangle \le \theta}$ with $|a_i| \ge 1$ is at most $O(\frac{1}{\sqrt{d}})$. Recently, this theorem was extended from half-plane to polytope [OST19] and positive spectrahedron [AY22]. We establish a similar result in the hyperbolic setting, demonstrating that the hyperbolic spectral norm of a Rademacher sum of vectors in the hyperbolic cone cannot concentrate within a small interval. It has some potential applications in constructing pseudorandom generators that fool hyperbolic cones. See Chapter 19 for more details.

[9]In fact, our result only implies the matrix Chernoff bound for constant-dimensional or constant-rank matrices.

Hyperbolic discrepancy The discrepancy minimization has been extensively studied across various settings. In the context of a set system over $[n]$, Spencer's theorem [Spe85] establishes an upper bound of $O(\sqrt{n})$ on the discrepancy. Similary, the minimum discrepancy of n matrices $X_1, \dots, X_n \in \mathbb{R}^{d\times d}$ can be defined as

$$\min_{s\in\{\pm 1\}^n} \|\sum_{i=1}^n s_i X_i\|.$$

Using the matrix Chernoff bound, we can derive an upper bound of $O(\sqrt{n\log d})$ on the matrix discrepancy, and it is conjectured that the $\log d$ factor can be shaved [Mek14]. When the matrices are rank-one, this problem is strongly connected to the Kadison-Singer (KS) problem [KS59] in functional analysis, which was resolved by the seminal work of Marcus, Spielman, and Srivastava [MSS15b]. Specifically, for any independent random vectors $u_1, \cdots, u_n \in \mathbb{C}^m$ in isotropic positions[10] in expectation, there is a positive probability that $\|\sum_{i=1}^n u_i u_i^*\| \le 1 + O(\max_{i\in[n]} \|u_i\|)$. From a discrepancy theory perspective, this result implies that: for any isotropic $u_1, \dots, u_n \in \mathbb{C}^m$ with $\|u_i u_i^*\| \le \epsilon$, it holds that

$$\min_{s\in\{\pm 1\}^n} \Big\|\sum_{i=1}^n s_i u_i u_i^*\Big\| \le O(\sqrt{\epsilon}),$$

which corresponds to the matrix Spencer conjecture with rank-one isotropic matrices. Several generalizations of the Kadison-Singer-type results have been established. For instance, [KLS20] relaxed the condition on the norm of each rank-one matrix to the norm of the sum of squared matrices, i.e., $\|\sum_{i=1}^n (u_i u_i^*)^2\| \le \sigma^2$, and showed a discrepancy bound of 4σ. [AO15] relaxed the independent sampling of u_i's to sampling from any strongly Rayleigh distribution. [Coh16a, Brä18] proved the Kadison-Singer theorem for high-rank matrices. In particular, [Brä18] obtained this result by proving a hyperbolic version of the Kadison-Singer theorem.

Our contribution to the hyperbolic discrepancy theory consists of two parts.

[10]We say vectors $v_1, \dots, v_n$ are in isotropic positions if $\sum_{i=1}^n v_i v_i^* = I$.

First, in the hyperbolic rank-one setting[11], we generalize [KLS20] and [AO15]'s results to the hyperbolic setting:

- *Four hyperbolic deviations suffice for the hyperbolic KS:* Let $h \in \mathbb{R}[x_1, \ldots, x_m]$ denote a hyperbolic polynomial in direction $e \in \mathbb{R}^m$. Let $v_1, \ldots, v_n \in \Gamma_+^h$ be n vectors in the hyperbolicity cone with $\sigma := \| \sum_{i=1}^n \mathrm{tr}_h[v_i] v_i \|_h$, where the hyperbolic trace $\mathrm{tr}_h[x] := \sum_{i=1}^d \lambda_i(x)$. Then it holds that
$$\min_{s \in \{\pm 1\}^n} \Big\| \sum_{i=1}^n s_i v_i \Big\|_h \leq 4\sigma.$$

- *Hyperbolic KS with a strongly Rayleigh distribution:* Let $h \in \mathbb{R}[x_1, \ldots, x_m]$ denote a hyperbolic polynomial in direction $e \in \mathbb{R}^m$. Let $v_1, \cdots, v_n \in \Gamma_+^h$ such that $\sum_{i=1}^n v_i = e$, and $\|v_i\|_h \leq \epsilon_2$ for all $i \in [n]$. Let μ be a homogeneous strongly Rayleigh probability distribution[12] on $[n]$ such that the marginal probability of each element is at most ϵ_1. Then, it holds that there exists $S \subseteq [n]$ in the support of μ, such that
$$\Big\| \sum_{i \in S} v_i \Big\|_h \leq 4(\epsilon_1 + \epsilon_2) + 2(\epsilon_1 + \epsilon_2)^2.$$

In addition, we present an algorithm that runs sub-exponentially in m to construct the solutions in both cases.

Second, we have some preliminary results about the hyperbolic discrepancy for general vectors. Note that [Brä18]'s result requires that the vectors are in the hyperbolic cone Γ_+^h and in an isotropic position. We relax this condition to hyperbolic cone vectors in a sub-isotropic position and slightly improve their hyperbolic discrepancy bound. Additionally, We put forward the hyperbolic Spencer conjecture, which states that for any $x_1, \ldots, x_n \in \mathbb{R}^m$ with $\|x_i\|_h \leq 1$, there exists an $s \in \{\pm 1\}^n$

[11]The hyperbolic rank of a vector x is the number of its non-zero hyperbolic eigenvalues.

[12]A distribution μ over the subsets of $[n]$ is *strongly Rayleigh* if its generating polynomial $g_\mu(z) := \sum_{S \subseteq [n]} \mu(S) z^S \in \mathbb{R}[z_1, \ldots, z_n]$ is a *real-stable polynomial*, i.e., no root in the upper-half of $\mathbb{C}^n$.

such that $\| \sum_{i=1}^n s_i x_i \|_h \leq O(\sqrt{n})$. This conjecture is a natural generalization of the matrix Spencer conjecture. And we prove a special case when the hyperbolic rank of each vector is constant. We hope our results will be helpful in fully resolving the matrix Spencer conjecture in the future. See Chapter 20 for more details.

1.5.2 Higher-order random walk on expanding posets

High-dimensional expanders (HDXs) and random walks on them have garnered significant attention in mathematics, theoretical computer science, and quantum computing. They exhibit strong connections to various areas such as approximating constraint satisfaction problems (CSPs) [AJT19, ALG20], approximate sampling and counting [ALGV19], agreement testing [DD19], constructing c3-locally testable codes (LTCs) [DEL$^+$22] and good quantum low-density parity-check (LDPC) codes [PK21, LH22, LZ22], resolving no low energy state conjecture (NLTS) [ABN23], etc. While most studies on HDXs have focused on hypergraphs or simplicial complexes, the understanding of non-simplicial objects (e.g., the Grassmanian complex) remains unclear. In this thesis, we delve into a generalized notion of high-dimensional expansion on partially ordered sets (posets) called expanding posets, as introduced by [DDFH18], and quantify the advantage of different poset architectures, highlighting how structural *regularity* controls the spectral decay and edge-expansion of corresponding random walks.

We specifically focus on the d-dimensional graded poset, which is a set X equipped with a partial order "$<$". Each element $x \in X$ has a rank $r(x) \in [d]$ that respects the partial order and partitions X into levels $X(0) \cup \ldots \cup X(d)$. For instance, in hypergraph, the rank of an element is the size of the subset, while in the Grassmann poset, where elements represent subspaces, the rank corresponds to the subspace dimension. Random walks on a graded poset are called *higher-order random walks*, which can be defined by composing the following up-operator and down-operator:

$$U_i f(x) = \mathop{\mathbb{E}}_{y \lessdot x}[f(y)], \quad D_i f(y) = \mathop{\mathbb{E}}_{x \gtrdot y}[f(x)] \quad \forall f : X(i) \to \mathbb{R},$$

where $y \lessdot x$ denotes $y < x$ and $r(y) = r(x) - 1$. For example, the upper walk $D_{i+1}U_i$ can be described as: starting from $x \in X(i)$, first moving up to $y \in X(i+1)$ with $x < y$, and then moving down to $x' \in X(i)$ with $x' < y$. One motivation for the higher-order random walks is they determine the expansion of the underlying poset. Inspired by the notion of spectral expansion of a graph, [DDFH18] defines the (δ, γ)-expansion of a poset for $\delta \in [0,1]^d$ and $\gamma \in [0,1)$ as:

$$\|D_{i+1}U_i - (1-\delta_i)I - \delta_i U_{i-1}D_i\| \le \gamma \quad \forall 1 \le i < d.$$

And we call an expanding graded poset an *eposet*.

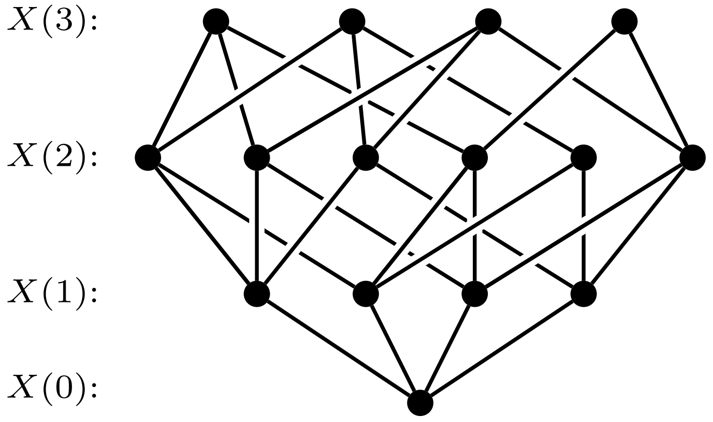


Figure 1.1: An example of a graded poset. The relation "$\lessdot$" corresponds to the edges in the figure.

Our research is centered around the concentration behavior of the higher-order walks on eposets, shedding light on their spectral properties. We demonstrate that the eigenvalues of the walk operators concentrate within strips around a small number of approximate eigenvalues, controlled by the regularity of the poset. More specifically, we establish that the i-th approximate eigenvalue of the walk operator can be expressed as $\lambda_i = \frac{R(k-1,i)}{R(k,i)}$, where k is the dimension of the random walk and $R(j,i)$ denotes the total number of rank-i elements less than any fixed rank-j element. Our findings indicate an exponential decay of eigenvalues in the poset architectures like the Grassmann, whereas architectures like hypergraphs exhibit only linear decay. This distinction is crucial in applications to the hardness of approximation and agreement testing, such as the recent proof of the 2-2 Games Conjecture [SMS18]. In contrast to the previous study on eposets [KT21a], which focuses only on the second eigenvalue, our result goes beyond and captures the full spectrum. It allows us to obtain a more comprehensive understanding of the concentration behavior of higher-order random walks on eposets.

As a consequence, we show a tight variance-based characterization of edge-expansion on eposets, which is an important combinatorial property used in algorithms and hardness of unique games [BBK+20, BHKL20, SMS18]. We prove that the expansion of any i-link[13] of an eposet is almost exactly $1-\lambda_i(M)$, where M is the higher-order random walk operator. And conversely, any set with expansion less than $1-\lambda_i(M)$ has a *high variance* across i-links. While a similar result was presented in a previous work [BHKL20], their proof heavily relies on the simplicial structure. Furthermore, we conduct a thorough analysis of the Grassmann, showing that our results are tight for a natural set of sparsifications of the Grassmann graphs (i.e., the q-eposets). See Chapter 21 for more details.

[13]For a rank-i element x, its k-dimensional link consists of all the rank-k elements $y > x$.

Part I

Quantum Computing

Chapter 2: Quantum Speedups of Log-concave Sampling

2.1 Introduction

Sampling from a given distribution is a fundamental computational problem. For example, in statistics, samples can determine confidence intervals or explore posterior distributions. In machine learning, samples are used for regression and to train supervised learning models. In optimization, samples from well-chosen distributions can produce points near local or even global optima.

Sampling can be nontrivial even when the distribution is known. Indeed, efficient sampling is often a challenging computational problem, and bottlenecks the running time in many applications. Many efforts have been made to develop fast sampling methods. Among those, one of the most successful tools is Markov Chain Monte Carlo (MCMC), which uses a Markov chain that converges to the desired distribution to (approximately) sample from it.

Here we focus on the fundamental task of *log-concave sampling*, i.e., sampling from a distribution proportional to e^{-f} where $f\colon \mathbb{R}^d \to \mathbb{R}$ is a convex function. This covers many practical applications such as multivariate Gaussian distributions and exponential distributions. Provable performance guarantees for log-concave sampling have been widely studied [DCWY18]. A closely related problem is estimating the normalizing constants of log-concave distributions, which also has many applications [GLL20].

Quantum computing has been applied to speed up many classical algorithms based on Markov processes, so it is natural to investigate quantum algorithms for log-concave sampling. If we can prepare a quantum state whose amplitudes are the square roots of the corresponding probabilities, then measurement yields a random sample from the desired distribution. In this approach, the number of required qubits is only poly-logarithmic in the size of the sample space. Unfortunately, such a quantum state probably cannot be efficiently prepared in general, since this would imply $\mathsf{SZK} \subseteq \mathsf{BQP}$ [ATS03]. Nevertheless, in some cases, quantum algorithms can achieve polynomial speedup over classical algorithms. Examples include uniform sampling on a 2D lattice [Ric07], estimating partition functions [WA08, WCNA09, Mon15, HW20, AHN$^+$21], and estimating volumes of convex bodies [CCH$^+$19]. However, despite the importance of sampling log-concave distributions and estimating normalizing constants, we are not aware of any previous quantum speedups for general instances of these problems.

Formulation In this chapter, we consider a d-dimensional convex function $f\colon \mathbb{R}^d \to \mathbb{R}$ which is L-smooth and μ-strongly convex, i.e., $\mu, L > 0$ and for any $x, y \in \mathbb{R}^d$, $x \neq y$,

$$\frac{f(y) - f(x) - \langle \nabla f(x), y - x\rangle}{\|x - y\|_2^2/2} \in [\mu, L]. \tag{2.1}$$

We denote by $\kappa := L/\mu$ the condition number of f. The corresponding log-concave distribution has probability density $\rho_f\colon \mathbb{R}^d \to \mathbb{R}$ with

$$\rho_f(x) := \frac{e^{-f(x)}}{Z_f}, \tag{2.2}$$

where the normalizing constant is

$$Z_f := \int_{x \in \mathbb{R}^d} e^{-f(x)}\,\mathrm{d}x. \tag{2.3}$$

When there is no ambiguity, we abbreviate ρ_f and Z_f as ρ and Z, respectively. Given an $\epsilon \in (0, 1)$,

- the goal of *log-concave sampling* is to output a random variable with distribution $\widetilde{\rho}$ such that $\|\widetilde{\rho} - \rho\| \leq \epsilon$, and
- the goal of *normalizing constant estimation* is to output a value $\widetilde{Z}$ such that with probability at least $2/3$, $(1 - \epsilon)Z \leq \widetilde{Z} \leq (1 + \epsilon)Z$.

Here $\|\cdot\|$ is a certain norm. We consider the general setting where the function f is specified by an oracle. In particular, we consider the quantum evaluation oracle O_f, a standard model in the quantum computing literature [CCH+19, AGGW20, CCLW20, ZLL21]. The evaluation oracle acts as

$$O_f|x, y\rangle = |x, f(x) + y\rangle \qquad \forall x \in \mathbb{R}^d, y \in \mathbb{R}. \tag{2.4}$$

(Quantum computing notations are briefly explained in Section 2.2.) We also consider the quantum gradient oracle $O_{\nabla f}$ with

$$O_{\nabla f}|x, z\rangle = |x, \nabla f(x) + z\rangle \qquad \forall x, z \in \mathbb{R}^d. \tag{2.5}$$

In other words, we allow superpositions of queries to both function evaluations and gradients. The essence of quantum speedup is the ability to compute with carefully designed superpositions.

Contributions Our main results are quantum algorithms that speed up log-concave sampling and normalizing constant estimation.

Theorem 2.1 (Main log-concave sampling result)**.** *Let ρ denote the log-concave distribution (2.2). There exist quantum algorithms that output a random variable distributed according to $\widetilde{\rho}$ such that*

- *$W_2(\widetilde{\rho}, \rho) \leq \epsilon$ where W_2 is the Wasserstein 2-norm (2.8), using $\widetilde{O}(\kappa^{7/6}d^{1/6}\epsilon^{-1/3} + \kappa d^{1/3}\epsilon^{-2/3})$ queries to the quantum evaluation oracle (2.4); or*
- *$\|\widetilde{\rho} - \rho\|_{TV} \leq \epsilon$ where $\|\cdot\|_{TV}$ is the total variation distance (2.7), using $\widetilde{O}\big(\kappa^{1/2}d\big)$ queries to the quantum gradient oracle (2.5), or $\widetilde{O}\big(\kappa^{1/2}d^{1/4}\big)$ queries when the initial distribution is warm (formally defined in Section 2.9.2.1).*

In the above results, the query complexity $\widetilde{O}(\kappa^{7/6}d^{1/6}\epsilon^{-1/3} + \kappa d^{1/3}\epsilon^{-2/3})$ is achieved by our quantum ULD-RMM algorithm. Although the quantum query complexity is the same as the best known classical result [SL19], we emphasize that our quantum algorithm uses a zeroth-order oracle while [SL19] uses a first-order oracle. The query complexity $\widetilde{O}(\kappa^{1/2}d)$ is achieved by our quantum MALA algorithm that uses a first-order oracle (as in classical algorithms). This is a quadratic speedup in κ compared with the best known classical algorithm [LST20]. With a warm start, our quantum speedup is even more significant: we achieve quadratic speedups in κ and d as compared with the best known classical algorithm with a warm start [WSC22].

Theorem 2.2 (Main normalizing constant estimation result)**.** *There exist quantum algorithms that estimate the normalizing constant by $\widetilde{Z}$ within multiplicative error ϵ with probability at least $3/4$,*

- *using $\widetilde{O}(\kappa^{7/6}d^{7/6}\epsilon^{-1} + \kappa d^{4/3}\epsilon^{-1})$ queries to the quantum evaluation oracle (2.4); or*
- *using $\widetilde{O}(\kappa^{1/2}d^{3/2}\epsilon^{-1})$ queries to the quantum gradient oracle (2.5).*

Furthermore, this task has quantum query complexity at least $\Omega(\epsilon^{-1+o(1)})$ (Theorem 2.9).

Our query complexity $\widetilde{O}(\kappa^{7/6}d^{7/6}\epsilon^{-1} + \kappa d^{4/3}\epsilon^{-1})$ for normalizing constant estimation achieves a quadratic speedup in precision compared with the best known classical algorithm [GLL20]. More remarkably, our quantum ULD-RMM algorithm again uses a zeroth-order oracle while the slower best known classical algorithm uses a first-order oracle [GLL20]. Our quantum algorithm working with a first-order oracle achieves polynomial speedups in all parameters compared with the best known classical algorithm [GLL20]. Moreover, the precision-dependence of our quantum algorithms is nearly optimal, which is quadratically better than the classical lower bound in $1/\epsilon$ [GLL20].

To the best of our knowledge, these are the first quantum algorithms with quantum speedup for the fundamental problems of log-concave sampling and estimating normalizing constants. We explore multiple classical techniques including the underdamped Langevin diffusion (ULD) method [DM17, DK19, DMM19, VW19], the randomized midpoint method for underdamped Langevin diffusion (ULD-RMM) [SL19, RSBG19], and the Metropolis adjusted Langevin algorithm (MALA) [DCWY18, CDWY20, LST20, CLA+21, WSC22, LST21], and achieve quantum speedups. Our main contributions are as follows.

- *Log-concave sampling.* For this problem, our quantum algorithms based on ULD and ULD-RMM have the same query complexity as the best known classical algorithms, but our quantum algorithms only use a zeroth-order (evaluation) oracle, while the classical algorithms use the first-order (gradient) oracle. For MALA, this improvement on the order of oracles is nontrivial, but we can use the quantum gradient oracle in our quantum MALA algorithm to achieve a quadratic speedup in the condition number κ. Furthermore, given a warm-start distribution, our quantum algorithm achieves a quadratic speedup in all parameters.
- *Normalizing constant estimation.* For this problem, our quantum algorithms provide larger speedups. In particular, our quantum algorithms based on ULD and ULD-RMM achieve quadratic speedup in the multiplicative precision ϵ (while using a zeroth-order oracle) compared with the corresponding best-known classical algorithms (using a first-order oracle). Our quantum algorithm based on MALA achieves polynomial speedups in all parameters. Furthermore, we prove that our quantum algorithm is nearly optimal in terms of ϵ.

We summarize our results and compare them to previous classical algorithms in Table 2.1 and Table 2.2. See Section 2.6 for more detailed comparisons to related classical and quantum work.

Techniques In this chapter, we develop a systematic approach for studying the complexity of quantum walk mixing and show that for any reversible classical Markov chain, we can obtain quadratic speedup for the mixing time as long as the initial distribution is warm. In particular, we apply the quantum walk and quantum annealing in the context of Langevin dynamics and achieve polynomial quantum speedups.

The technical ingredients of our quantum algorithms are highlighted below.

Table 2.1: Summary of the query complexities of classical and quantum algorithms for sampling a d-dimensional log-concave distribution. Here $\kappa = L/\mu$ in (2.1) and ϵ is the error in the designated norm.

Method	Oracle	Complexity	Norm
ULD [CCBJ18]	gradient	$\widetilde{O}(\kappa^2 d^{1/2}\epsilon^{-1})$	W_2
ULD-RMM [SL19]	gradient	$\widetilde{O}(\kappa^{7/6} d^{1/6}\epsilon^{-1/3} + \kappa d^{1/3}\epsilon^{-2/3})$	W_2
MALA [LST20]	gradient	$\widetilde{O}(\kappa d)$	TV
MALA with warm start [WSC22]	gradient	$\widetilde{O}(\kappa d^{1/2})$	TV
Quantum Inexact ULD (Theorem 2.30)	evaluation	$\widetilde{O}(\kappa^2 d^{1/2}\epsilon^{-1})$	W_2
Quantum Inexact ULD-RMM (Theorem 2.31)	evaluation	$\widetilde{O}(\kappa^{7/6} d^{1/6}\epsilon^{-1/3} + \kappa d^{1/3}\epsilon^{-2/3})$	W_2
Quantum MALA (Theorem 2.38)	gradient	$\widetilde{O}(\kappa^{1/2} d)$	TV
Quantum MALA (warm start) (Theorem 2.36)	gradient	$\widetilde{O}(\kappa^{1/2} d^{1/4})$	TV

- *Quantum simulated annealing (Lemma 2.5).* Our quantum algorithm for estimating normalizing constants combines the quantum simulated annealing framework of [WA08] and the quantum mean estimation algorithm of [Mon15]. For each type of Langevin dynamics (which are random walks), we build a corresponding quantum walk. Crucially, the spectral gap of the random walk is quadratically amplified in the phase gap of the corresponding quantum walk. This allows us to use a Grover-like procedure to produce the stationary distribution state given a sufficiently good initial state. In the simulated annealing framework, this initial state is the stationary distribution state of the previous Markov chain.

- *Effective spectral gap (Lemma 2.21).* We show how to leverage a "warm" initial distribution to achieve a quantum speedup for sampling. Classically, a warm start leads to faster mixing even if the spectral gap is small. Quantumly, we generalize the notion of "effective spectral gap" [Rei09, LMR+11, CCH+19] to our more general sampling problem. We show that with a bounded warmness parameter, quantum algorithms can achieve a quadratic speedup in the mixing time. By viewing the sampling problem as a simulated annealing process with only one Markov chain, we prove a quadratic speedup for quantum MALA by analyzing the effective spectral

Table 2.2: Summary of the query complexities of classical and quantum algorithms for estimating the normalizing constant of a d-dimensional log-concave distribution. Here $\kappa = L/\mu$ in (2.1) and ϵ is the multiplicative error.

Method	Oracle	Complexity
Multilevel ULD [GLL20]	gradient	$\widetilde{O}\big(\kappa^2 d^{3/2}\epsilon^{-2}\big)$
Multilevel ULD-RMM [GLL20]	gradient	$\widetilde{O}\big(\kappa^{7/6} d^{7/6}\epsilon^{-2} + \kappa d^{4/3}\epsilon^{-2}\big)$
MALA [GLL20]	gradient	$\widetilde{O}\big(\kappa d^2\epsilon^{-2}\max\{1, \frac{\kappa}{d}\}\big)$
Multilevel Quantum Inexact ULD (Theorem 2.47)	evaluation	$\widetilde{O}\big(\kappa^2 d^{3/2}\epsilon^{-1}\big)$
Multilevel Quantum Inexact ULD-RMM (Theorem 2.48)	evaluation	$\widetilde{O}\big(\kappa^{7/6} d^{7/6}\epsilon^{-1} + \kappa d^{4/3}\epsilon^{-1}\big)$
Quantum annealing with Quantum MALA (Theorem 2.43)	gradient	$\widetilde{O}\big(\kappa^{1/2} d^{3/2}\epsilon^{-1}\big)$

gap.

- *Quantum gradient estimation (Lemma 2.27).* We adapt Jordan's quantum gradient algorithm [Jor05] to the ULD and ULD-RMM algorithms and give rigorous proofs to bound the sampling error due to gradient estimation errors.

Open questions Our work raises several natural questions for future investigation:

- Can we achieve quantum speedup in d and κ for unadjusted Langevin algorithms such as ULD and ULD-RMM? The main difficulty is that ULD and ULD-RMM are irreversible, while most available quantum walk techniques only apply to reversible Markov chains. New techniques might be required to resolve this question.

- Can we achieve further quantum speedup for estimating normalizing constants with a warm start distribution? This might require a more refined version of quantum mean estimation.

- Can we give quantum algorithms for estimating normalizing constants with query complexity sublinear in d? Such a result would give a provable quantum-classical separation due to the $\Omega(d^{1-o(1)}/\epsilon^{2-o(1)})$ classical lower bound proved in [GLL20].

2.2 Preliminaries

We defer the basics of quantum computing to Appendix B. In this subsection, we introduce a standard "quantum approach" to black-box access some functions—quantum oracle and quantum query access. We also introduce the application of quantum computing in sampling problems. Then, we provide some important notations/definitions in classical literature that are used in this chapter.

Quantum query access and quantum sampling Quantum access to a function, referred to as a *quantum oracle*, must be reversible and allow access to different values of the function in *superposition* (i.e., for linear combinations of computational basis states). For example, consider the unitary evaluation oracle O_f defined in (2.4). Given a probability distribution $\{p_i\}_{i=1}^{n}$ and a set of points $\{x_i\}_{i=1}^{n}$, we have

$$O_f \sum_{i=1}^{n} \sqrt{p_i}|x_i\rangle|0\rangle = \sum_{i=1}^{n} \sqrt{p_i}|x_i\rangle|f(x_i)\rangle. \tag{2.6}$$

Then a measurement would give $f(x_i)$ with probability p_i. However, a quantum oracle can not only simulate random sampling, but can enable uniquely quantum behavior through interference. Examples include amplitude amplification—the main idea behind Grover's search algorithm [Gro96] and the amplitude estimation procedure used in this chapter—and many other quantum algorithms relying on coherent quantum access to a function. Similar arguments apply to the quantum gradient oracle (2.5). If a classical oracle can be computed by an explicit classical circuit, then the corresponding quantum oracle can be implemented by a quantum circuit of approximately the same size. Therefore, these quantum oracles provide a useful framework for understanding the quantum complexity of log-concave sampling and normalizing constant estimation.

To sample from a distribution π, it suffices to prepare the state $|\pi\rangle := \sum_x \sqrt{\pi_x}|x\rangle$ and then measure it. For a Markov chain specified by a transition matrix P with stationary distribution π, one can construct a corresponding *quantum walk* operator

$W(P)$. Intuitively, quantum walks can be viewed as applying a sequence of quantum unitaries on a quantum state encoding the initial distribution to rotate it to the subspace of stationary distribution $|\pi\rangle$. The number of rotations needed (i.e., the angle between the initial distribution and stationary distribution) depends on the spectral gap of P, and a quantum algorithm can achieve a quadratic speedup via *quantum phase estimation* and *amplification* algorithms. More background on quantum walk is given in Section 2.9.2.2.

Classical definitions Throughout this chapter, the big-O notations $O(\cdot)$, $o(\cdot)$, $\Omega(\cdot)$, and $\Theta(\cdot)$ follow common definitions. The $\tilde{O}$ notation omits poly-logarithmic terms, i.e., $\tilde{O}(f) := O(f\operatorname{poly}(\log f))$. We say a function f is *L-Lipschitz continuous* at x if $|f(x) - f(y)| \leq L\|x - y\|$ for all y sufficiently near x. The total variation distance (TV-distance) between two functions $f, g\colon \mathbb{R}^d \to \mathbb{R}$ is defined as

$$\|f - g\|_{\mathrm{TV}} := \frac{1}{2}\int_{\mathbb{R}^d} |f(x) - g(x)|\,\mathrm{d}x. \tag{2.7}$$

Let $\mathcal{B}(\mathbb{R}^d)$ denote the Borel σ-field of $\mathbb{R}^d$. Given probability measures μ and ν on $(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d))$, a *transference plan* ζ between μ and ν is defined as a probability measure on $(\mathbb{R}^d \times \mathbb{R}^d, \mathcal{B}(\mathbb{R}^d) \times \mathcal{B}(\mathbb{R}^d))$ such that for any $A \subseteq \mathbb{R}^d$, $\zeta(A \times \mathbb{R}^d) = \mu(A)$ and $\zeta(\mathbb{R}^d \times A) = \nu(A)$. We let $\Gamma(\mu, \nu)$ denote the set of all transference plans. We let

$$W_2(\mu, \nu) := \left(\inf_{\zeta \in \Gamma(\mu,\nu)} \int_{\mathbb{R}^d \times \mathbb{R}^d} \|x - y\|_2^2 \,\mathrm{d}\zeta(x, y)\right)^{\frac{1}{2}} \tag{2.8}$$

denote the Wasserstein 2-norm between μ and ν.

2.3 Quantum Algorithm for Log-Concave Sampling

In this section, we describe several quantum algorithms for sampling log-concave distributions.

Quantum inexact ULD and ULD-RMM We first show that the gradient oracle in the classical ULD and ULD-RMM algorithms can be efficiently simulated by the quantum evaluation oracle via quantum gradient estimation.

Suppose we are given access to the evaluation oracle (2.4) for $f(x)$. Then by Jordan's algorithm [Jor05] (see Lemma 2.27 for details), there is a quantum algorithm that can compute $\nabla f(x)$ with a polynomially small ℓ_1-error by querying the evaluation oracle $O(1)$ times. Using this, we can prove the following theorem (see Section 2.9.1 for details).

Theorem 2.3 (Informal version of Theorem 2.30 and Theorem 2.31)**.** *Let $\rho \propto e^{-f}$ be a d-dimensional log-concave distribution with f satisfying (2.1). Given a quantum evaluation oracle for f,*

- *the quantum inexact ULD algorithm uses $\widetilde{O}(\kappa^2 d^{1/2}\epsilon^{-1})$ queries, and*
- *the quantum inexact ULD-RMM algorithm uses $\widetilde{O}(\kappa^{7/6}d^{1/6}\epsilon^{-1/3}+\kappa d^{1/3}\epsilon^{-2/3})$ queries,*

to quantumly sample from a distribution that is ϵ-close to ρ in W_2-distance.

We note that the query complexities of our quantum algorithms using a *zeroth-order* oracle match the state-of-the-art classical ULD [CCBJ18] and ULD-RMM [SL19] complexities with a *first-order* oracle. The main technical difficulty of applying the quantum gradient algorithm is that it produces a *stochastic gradient oracle* in which the output of the quantum algorithm $\mathbf{g}$ satisfies $\|\mathbb{E}[\mathbf{g}] - \nabla f(x)\|_1 \leq d^{-\Omega(1)}$. In particular, the randomness of the gradient computation is "entangled" with the randomness of the Markov chain. We use the classical analysis of ULD and ULD-RMM processes [RSBG19] to prove that the stochastic gradient will not significantly slow down the mixing of ULD processes, and that the error caused by the quantum gradient algorithm can be controlled.

Quantum MALA We next propose two quantum algorithms with lower query complexity than classical MALA, one with a Gaussian initial distribution and another with a warm-start distribution. The main technical tool we use is a quantum walk in continuous space.

The classical MALA (i.e., Metropolized HMC) starts from a Gaussian distribution $\mathcal{N}(0, L^{-1}I_d)$ and performs a leapfrog step in each iteration. It is well-known that the initial Gaussian state

$$|\rho_0\rangle = \int_{\mathbb{R}^d} \left(\frac{L}{2\pi}\right)^{d/4} e^{-\frac{L}{4}\|z-x^\star\|_2^2} |z\rangle \, \mathrm{d}z \tag{2.9}$$

can be efficiently prepared. We show that the quantum walk update operator

$$U := \int_{\mathbb{R}^d} \mathrm{d}x \int_{\mathbb{R}^d} \mathrm{d}y \, \sqrt{p_{x\to y}} |x\rangle\langle x| \otimes |y\rangle\langle 0| \tag{2.10}$$

can be efficiently implemented, where $p_{x\to y} := p(x, y)$ is the transition density from x to y, and the density p satisfies $\int_{\mathbb{R}^d} p(x, y) \, \mathrm{d}y = 1$ for any $x \in \mathbb{R}^d$.

Lemma 2.4 (Informal version of Lemma 2.35)**.** *The continuous-space quantum walk operator corresponding to the MALA Markov chain can be implemented with $O(1)$ gradient and evaluation queries.*

In general, it is difficult to quantumly speed up the mixing time of a classical Markov chain, which is upper bounded by $O(\delta^{-1} \log(\rho_{\min}^{-1}))$, where δ is the spectral gap. However, [WA08] shows that a quadratic speedup is possible when following a sequence of *slowly-varying* Markov chains. More specifically, let $\rho_0, \ldots, \rho_r$ be the stationary distributions of the *reversible* Markov chains $\mathcal{M}_0, \ldots, \mathcal{M}_r$ and let $|\rho_0\rangle, \ldots, |\rho_r\rangle$ be the corresponding quantum states. Suppose $|\langle\rho_i|\rho_{i+1}\rangle| \geq p$ for all $i \in \{0, \ldots, r-1\}$, and suppose the spectral gaps of $\mathcal{M}_0, \ldots, \mathcal{M}_r$ are lower-bounded by δ. Then we can prepare a quantum state $|\widetilde{\rho_r}\rangle$ that is ϵ-close to $|\rho_r\rangle$ using $\tilde{O}\big(\delta^{-1/2} r p^{-1}\big)$ quantum walk steps. To fulfill the slowly-varying condition, we consider an annealing process that goes from $\rho_0 = \mathcal{N}(0, L^{-1}I_d)$ to the target distribution $\rho_{M+1} = \rho$ in $M = \widetilde{O}(\sqrt{d})$ stages. At the ith stage, the stationary distribution is $\rho_i \propto e^{-f_i}$ with $f_i := f + \frac{1}{2}\sigma_i^{-2}\|x\|^2$. By properly choosing $\sigma_1 \leq \cdots \leq \sigma_M$, we prove that this sequence of Markov chains is slowly varying.

Lemma 2.5 (Informal version of Lemma 2.15)**.** *If we take* $\sigma_1^2 = \frac{\epsilon}{2dL}$ *and* $\sigma_{i+1}^2 = (1+\frac{1}{\sqrt{d}})\sigma_i^2$*, then for* $0 \le i \le M$*, we have* $|\langle \rho_i | \rho_{i+1}\rangle| \ge \Omega(1)$.

Combining Lemma 2.4, Lemma 2.5, and the effective spectral gap of MALA (Lemma 2.21), we have:

Theorem 2.6 (Informal version of Theorem 2.38)**.** *Let* $\rho \propto e^{-f}$ *be a* d*-dimensional log-concave distribution with* f *satisfying* (2.1)*. There is a quantum algorithm (Algorithm 1) that prepares a state* $|\widetilde{\rho}\rangle$ *with* $\||\widetilde{\rho}\rangle - |\rho\rangle\| \le \epsilon$ *using* $\widetilde{O}(\kappa^{1/2}d)$ *gradient and evaluation oracle queries.*

Algorithm 1 Quantum MALA for Log-concave Sampling (Informal)

```
Input: Evaluation oracle $\mathcal{O}_f$, gradient oracle $\mathcal{O}_{\nabla f}$
Output: Quantum state $|\widetilde{\rho}\rangle$ close to the stationary distribution state $\int_{\mathbb{R}^d} e^{-f(x)/2}\,\mathrm{d}|x\rangle$
1: Compute the cooling schedule parameters $\sigma_1, \ldots, \sigma_M$
2: Prepare the state $|\rho_0\rangle \propto \int_{\mathbb{R}^d} e^{-\frac{1}{4}\|x\|^2/\sigma_1^2}\,\mathrm{d}|x\rangle$
3: for $i \leftarrow 1, \ldots, M$ do
4:     Construct $\mathcal{O}_{f_i}$ and $\mathcal{O}_{\nabla f_i}$ where $f_i(x) = f(x) + \frac{1}{2}\|x\|^2/\sigma_i^2$
5:     Construct quantum walk update unitary $U$ with $\mathcal{O}_{f_i}$ and $\mathcal{O}_{\nabla f_i}$
6:     Implement the quantum walk operator and the approximate reflection $\widetilde{R}_i$
7:     Prepare $|\rho_i\rangle$ by performing $\frac{\pi}{3}$-amplitude amplification with $\widetilde{R}_i$ on $|\rho_{i-1}\rangle|0\rangle$
8: end for
9: return $|\rho_M\rangle$
```

For the classical MALA with a Gaussian initial distribution, it was shown by [LST21] that the mixing time is at least $\widetilde{\Omega}(\kappa d)$. Theorem 2.6 quadratically reduces the κ dependence.

Note that Algorithm 1 uses a first-order oracle, instead of the zeroth-order oracle used in the quantum ULD algorithms. The technical barrier to applying the quantum gradient algorithm (Lemma 2.27) in the quantum MALA is to analyze the classical MALA with a stochastic gradient oracle. We currently do not know whether the "entangled randomness" dramatically increases the mixing time.

More technical details and proofs are provided in Section 2.9.

2.4 Quantum Algorithm for Estimating Normalizing Constants

In this section, we apply our quantum log-concave sampling algorithms to the normalizing constant estimation problem. A very natural approach to this problem is via MCMC, which constructs a multi-stage annealing process and uses a sampler at each stage to solve a mean estimation problem. We show how to quantumly speed up these annealing processes and improve the query complexity of estimating normalizing constants.

Quantum speedup for the standard annealing process We first consider the standard annealing process for log-concave distributions, as already applied in the previous section. Recall that we pick parameters $\sigma_1 < \sigma_2 < \cdots < \sigma_M$ and construct a sequence of Markov chains with stationary distributions $\rho_i \propto e^{-f_i}$, where $f_i = f + \frac{1}{2\sigma_i^2}\|x\|^2$. Then, at the ith stage, we estimate the expectation

$$\mathbb{E}_{\rho_i}[g_i] \quad \text{where} \quad g_i = \exp\left(\frac{1}{2}(\sigma_i^{-2} - \sigma_{i+1}^{-2})\|x\|^2\right). \tag{2.11}$$

If we can estimate each expectation with relative error at most $O(\epsilon/M)$, then the product of these M quantities estimates the normalizing constant $Z = \int_{\mathbb{R}^d} e^{-f(x)}\,\mathrm{d}x$ with relative error at most ϵ.

For the mean estimation problem, [Mon15] showed that when the relative variance $\frac{\mathbf{Var}_{\rho_i}[g_i]}{\mathbb{E}_{\rho_i}[g_i]^2}$ is constant, there is a quantum algorithm for estimating the expectation $\mathbb{E}_{\rho_i}[g_i]$ within relative error at most ϵ using $\widetilde{O}(1/\epsilon)$ quantum samples from the distribution ρ_i. Our annealing schedule satisfies the bounded relative variance condition. Therefore, by the quantum mean estimation algorithm, we improve the sampling complexity of the standard annealing process from $\widetilde{O}(M^2\epsilon^{-2})$ to $\widetilde{O}(M\epsilon^{-1})$.

To further improve the query complexity, we consider using the quantum MALAs developed in the previous section to generate samples. Observe that Algorithm 1 outputs a quantum state corresponding to some distribution that is close to ρ_i, instead of an individual sample. If we can estimate the expectation without destroying the quantum state, then we can reuse the state and evolve it for the $(i+1)$st Markov chain. Fortunately, we can use non-destructive mean estimation to estimate the expectation and restore the initial states. A detailed error analysis of this algorithm can be found in [CCH+19, HW20]. We first prepare $\widetilde{O}(M\epsilon^{-1})$ copies of initial states corresponding to the Gaussian distribution $\mathcal{N}(0, L^{-1}I_d)$. Then, for each stage, we apply the non-destructive mean estimation algorithm to estimate the expectation $\mathbb{E}_{\rho_i}[g_i]$ and then run quantum MALA to evolve the states $|\rho_i\rangle$ to $|\rho_{i+1}\rangle$. This gives our first quantum algorithm for estimating normalizing constants.

Theorem 2.7 (Informal version of Theorem 2.43)**.** *Let Z be the normalizing constant in (2.3). There is a quantum algorithm (Algorithm 2) that outputs an estimate $\widetilde{Z}$ with relative error at most ϵ using $\widetilde{O}\big(d^{3/2}\kappa^{1/2}\epsilon^{-1}\big)$ queries to the quantum gradient and evaluation oracles.*

Algorithm 2 Quantum MALA for Estimating Normalizing Constant (Informal)

```
Input: Evaluation oracle $\mathcal{O}_f$, gradient oracle $\mathcal{O}_{\nabla f}$
Output: Estimate $\widetilde{Z}$ of $Z$ with relative error at most $\epsilon$
 1: $M \leftarrow \widetilde{O}(\sqrt{d})$, $K \leftarrow \widetilde{O}(\epsilon^{-1})$
 2: Compute the cooling schedule parameters $\sigma_1, \ldots, \sigma_M$
 3: for $j \leftarrow 1, \ldots, K$ do
 4:     Prepare the state $|\rho_{1,j}\rangle \propto \int_{\mathbb{R}^d} e^{-\frac{1}{4}\|x\|^2/\sigma_1^2}|x\rangle \mathrm{d}x$
 5: end for
 6: $\widetilde{Z} \leftarrow (2\pi\sigma_1^2)^{d/2}$
 7: for $i \leftarrow 1, \ldots, M$ do
 8:     $\widetilde{g}_i \leftarrow$ Non-destructive mean estimation for $g_i$ using $\{|\rho_{i,0}\rangle, \ldots, |\rho_{i,K}\rangle\}$
 9:     $\widetilde{Z} \leftarrow \widetilde{Z}\widetilde{g}_i$
10:     for $j \leftarrow 1, \ldots, K$ do
11:         $|\rho_{i+1,j}\rangle \leftarrow$ QuantumMALA$(\mathcal{O}_{f_{i+1}}, \mathcal{O}_{\nabla, f_{i+1}}, |\rho_{i,j}\rangle)$        ▷ Algorithm 1
12:     end for
13: end for
14: return $\widetilde{Z}$
```

Quantum speedup for MLMC Now we consider using multilevel Monte Carlo (MLMC) as the annealing process and show how to achieve quantum speedup. MLMC was originally developed by [Hei01] for parametric integration; then [Gil08] applied MLMC to simulate stochastic differential equations (SDEs). The idea of MLMC is natural: we choose a different number of samples at each stage based on the cost and variance of that stage.

To estimate normalizing constants, a variant of MLMC was proposed in [GLL20]. Unlike the standard MLMC for bounding the mean-squared error, they upper bound the bias and the variance separately, and the analysis is technically difficult. The first quantum algorithm based on MLMC was subsequently developed by [ALL+21] based on the quantum mean estimation algorithm. Roughly speaking, the quantum algorithm can quadratically reduce the ϵ-dependence of the sample complexity compared with classical MLMC.

In this chapter, we apply the quantum accelerated MLMC (QA-MLMC) scheme [ALL+21] to simulate underdamped Langevin dynamics as the SDE. One challenge in using QA-MLMC is that g_i in our setting is not Lipschitz. Fortunately, as suggested by [GLL20], this issue can be resolved by truncating large x and replacing g_i by $h_i := \min\bigl\{g_i, \exp\bigl(\frac{(r_i^+)^2}{\sigma_i^2(1+\alpha^{-1})}\bigr)\bigr\}$, with the choice

$$\alpha = \widetilde{O}\left(\frac{1}{\sqrt{d}\log(1/\epsilon)}\right) \qquad r_i^+ = \mathbb{E}_{\rho_{i+1}}\|x\| + \Theta(\sigma_i\sqrt{(1+\alpha)\log(1/\epsilon)}) \tag{2.12}$$

to ensure $\frac{h_i}{\mathbb{E}_{\rho_i} g_i}$ is $O(\sigma_i^{-1})$ Lipschitz. Furthermore, $\bigl|\mathbb{E}_{\rho_i}(h_i - g_i)\bigr| < \epsilon$ by Lemmas C.7 and C.8 in [GLL20]. For simplicity, we regard g_i as a Lipschitz continuous function in our main results. We present QA-MLMC in Algorithm 3, where the sampling algorithm A can be chosen to be quantum inexact ULD/ULD-RMM or quantum MALA.

This QA-MLMC framework reduces the ϵ-dependence of the sampling complexity for estimating normalizing constants from ϵ^{-2} to ϵ^{-1} in both the ULD and ULD-RMM cases, as compared with the state-of-the-art classical results [GLL20].

Algorithm 3 QA-MLMC (Informal)

```
Input: Evaluation oracle $\mathcal{O}_f$, function $g$, error $\epsilon$, a quantum sampler $\mathrm{A}(x_0, f, \eta)$ for $\rho$
Output: An estimate of $\widetilde{R} = \mathbb{E}_\rho h$
 1: $K \leftarrow \widetilde{O}(\epsilon^{-1})$
 2: Compute the initial point $x_0$ and the step size $\eta_0$
 3: Compute the number of samples $N_1, \ldots, N_K$
 4: for $j \leftarrow 1, \ldots, K$ do
 5:     $\eta_j \leftarrow \eta/2^{j-1}$
 6:     for $i \leftarrow 1, \ldots, N_j$ do
 7:         Sample $X_i^{\eta_j}$ by $\mathrm{A}(f, x_0, \eta_j)$, and sample $X_i^{\eta_j/2}$ by $\mathrm{A}(f, x_0, \eta_j/2)$
 8:         $\widetilde{G}_i^- \leftarrow \textsc{QMeanEst}(\{g(X_i^{\eta_j})\}_{i\in[N_j]})$     ▷ Quantum Mean Estimation
 9:         $\widetilde{G}_i^+ \leftarrow \textsc{QMeanEst}(\{g(X_i^{\eta_j/2})\}_{i\in[N_j]})$
10:     end for
11: end for
12: return $\widetilde{R} = \widetilde{G}_0 + \sum_{j=0}^{K}(\widetilde{G}_i^- - \widetilde{G}_i^+)$
```

Using the quantum inexact ULD and ULD-RMM algorithms (Theorem 2.3) to generate samples, we obtain our second quantum algorithm for estimating normalizing constants (see Section 2.10 for proofs).

Theorem 2.8 (Informal version of Theorem 2.47 and Theorem 2.48)**.** *Let Z be the normalizing constant in (2.3). There exist quantum algorithms for estimating Z with relative error at most ϵ using*

- *quantum inexact ULD with $\widetilde{O}(d^{3/2}\kappa^2\epsilon^{-1})$ queries to the evaluation oracle, and*
- *quantum inexact ULD-RMM with $\widetilde{O}((d^{7/6}\kappa^{7/6} + d^{4/3}\kappa)\epsilon^{-1})$ queries to the evaluation oracle.*

2.5 Quantum Lower Bound

Finally, we lower bound the quantum query complexity of normalizing constant estimation.

Theorem 2.9. *For any fixed positive integer k, given query access (2.4) to a function*

$f\colon \mathbb{R}^k \to \mathbb{R}$ that is 1.5-smooth and 0.5-strongly convex, the quantum query complexity of estimating the partition function $Z = \int_{\mathbb{R}^k} e^{-f(x)}\,\mathrm{d}x$ within multiplicative error ϵ with probability at least $2/3$ is $\Omega(\epsilon^{-\frac{1}{1+4/k}})$.

The proof of our quantum lower bound is inspired by the construction in Section 5 of [GLL20]. They consider a log-concave function whose value is negligible outside a hypercube centered at 0. The interior of the hypercube is decomposed into cells of two types. The function takes different values on each type, and the normalizing constant estimation problem reduces to determining the number of cells of each type. Quantumly, we follow the same construction and reduce the cell counting problem to the *Hamming weight problem*: given an n-bit Boolean string and two integers $\ell < \ell'$, decide whether the Hamming weight (i.e., the number of ones) of this string is ℓ_1 or ℓ_2. This problem has a known quantum query lower bound [NW99], which implies the quantum hardness of estimating the normalizing constant. The full proof of Theorem 2.9 appears in Section 2.11.

2.6 Related Work

2.6.1 Classical MCMC methods

Our quantum algorithms are inspired by a major class of classical MCMC algorithms based on *Langevin dynamics*. There has been extensive work on non-asymptotic error bounds for the mixing times of Langevin-type algorithms for sampling [DCWY18, SL19, CDWY20, LST20, WSC22]. One commonly used type of algorithm is based on the mixing time of Langevin dynamics, including the underdamped Langevin diffusion process described by the stochastic differential equations

$$\mathrm{d}v_t = -\gamma v_t\,\mathrm{d}t - u\nabla f(x_t)\,\mathrm{d}t + \sqrt{2\gamma u}\,\mathrm{d}W_t \tag{2.13}$$

$$\mathrm{d}x_t = v_t\,\mathrm{d}t \tag{2.14}$$

with parameters γ, u, where $W_t \sim \mathcal{N}(0,t)$ is a standard Wiener process. The coefficients of (2.13) are Lipschitz continuous since f is L-smooth; and the overdamped

Langevin diffusion process

$$\mathrm{d}x_t = -u\nabla f(x_t)\,\mathrm{d}t + \sqrt{2u}\,\mathrm{d}W_t \tag{2.15}$$

is obtained by taking $\gamma \to \infty$ and $t \to t/\gamma$.

It can be shown that taking $\gamma = 2$ and $u = 1/L$, the stationary distribution of the underdamped Langevin diffusion (2.13) is proportional to $e^{-(f(x)+L\|v\|^2/2)}$, and the marginal distribution of x is proportional to $e^{-f(x)}$. When $\gamma \to \infty$, the stationary distribution of the overdamped version (2.15) is proportional to $e^{-f(x)}$. The numerical discretization of (2.15) is used in unadjusted Langevin algorithms, while sampling algorithms based on the discretization of (2.13) can have a better dependence on d and ϵ.

We now introduce a few common classical sampling algorithms: the underdamped Langevin diffusion (ULD) method; the randomized midpoint method for underdamped Langevin diffusion (ULD-RMM), with the best known dependence on d; and the Metropolis adjusted Langevin algorithm (MALA), with the best known dependence on κ and ϵ. To simulate the random process in discrete time, ULD can be viewed as the first-order forward Euler discretization of the continuous process (2.13). In particular, ULD takes $\widetilde{O}\big(\kappa^2\sqrt{d}/\epsilon\big)$ steps to approximate the stationary distribution $e^{-f(x)}$ within ϵ in the Wasserstein 2-norm [CCBJ18], where κ is the condition number of f, and d is the dimension. ULD-RMM approximates the integral of the random process (2.13) by randomly choosing the midpoint in the integral, which reduces the bias in the accumulation of the integration. As a more accurate approximation, ULD-RMM converges in the Wasserstein 2-distance ϵ with $\widetilde{O}\Big(\frac{\kappa^{7/6}d^{1/6}}{\epsilon^{1/3}} + \frac{\kappa d^{1/3}}{\epsilon^{2/3}}\Big)$ steps [SL19], a polynomial reduction in κ, d, ϵ over ULD. As an alternative approach, MALA also constructs the Euler discretization of (2.13), and then applies the Metropolis-Hastings acceptance/rejection step to ensure convergence to the correct stationary distribution. It was first shown that MALA converges in the total variation distance ϵ with $\widetilde{O}\big(\kappa d \max\{1, \kappa/d\} \log(\kappa d/\epsilon)\big)$ steps for Gaussian initial distributions [DCWY18, CDWY20]. Later, this result was improved to $\widetilde{O}\big(\kappa d \log(\kappa d/\epsilon)\big)$ based on an improved non-asymptotic analysis of the mixing time [LST20]. For warm-start distributions, the complexity of MALA can be further reduced to $\widetilde{O}\big(\kappa d^{1/2} \log(\kappa d/\epsilon)\big)$ [WSC22]. Compared to ULD and ULD-RMM, this exponentially improves the dependence on ϵ, and polynomially improves the dependence on κ, while it suffers from a worse dependence on d. We introduce the algorithms and complexities of ULD and ULD-RMM in Section 2.7.1, and introduce these results of MALA in Section 2.9.2.1.

For the task of estimating the normalizing constant (2.3), the state-of-the-art classical results are given by [GLL20]. That work applies the classical sampling algorithms described above with an annealing strategy. The normalizing constant is estimated by a sequence of telescoping sums, each of which can be approximated by a Monte Carlo method that samples from a log-concave distribution. We introduce this annealing procedure in Section 2.7.2. Reference [GLL20] employed the mixing time of MALA for Gaussian initial distributions developed by [DCWY18, CDWY20] with the annealing procedure, achieving the overall complexity $\widetilde{O}\Big(\frac{\kappa d^2}{\epsilon^2} \max\{1, \kappa/d\}\Big)$ for estimating the normalizing constant. They also combined ULD and ULD-RMM with the annealing and the multilevel Monte Carlo (MLMC) method to achieve complexities of $\widetilde{O}\Big(\frac{\kappa^2 d^{3/2}}{\epsilon^2}\Big)$ and $\widetilde{O}\Big(\frac{\kappa^{7/6} d^{7/6}}{\epsilon^2} + \frac{\kappa d^{4/3}}{\epsilon^2}\Big)$, respectively. Here MLMC, introduced in Section 2.10.2, is utilized to resolve the worse dependence on ϵ in ULD and ULD-RMM, resulting in the same $\widetilde{O}(1/\epsilon^2)$ scaling of the error compared to the annealing with MALA. Annealing with MLMC and ULD/ULD-RMM also has a better dependence on d over annealing with MALA, while they suffer from a worse dependence on κ.

2.6.2 Quantum methods for sampling and partition function estimation

Previous literature developed alternative approaches to generating quantum states corresponding to classical probability distributions on a quantum computer, sometimes referred to as quantum sampling (or qsampling) from a distribution. References [Zal98], [GR02], and [KM01] propose direct state generation approaches using controlled rotations. However, this approach is limited to the regime in which the distribution is efficiently integrable. As an alternative, [ATS03] develops an adiabatic approach to qsampling. They apply adiabatic evolution techniques to qsample the stationary distributions of a sequence of slowly varying Markov chains, a technique referred to as quantum simulated annealing (QSA) in subsequent literature [SBB07, SBBK08, WA08, YAG12, HW20]. The time complexity of Aharanov and Ta-Shma's approach is $O(1/\delta)$ as a function of the spectral gap δ, comparable to the running time of analogous classical sampling methods. Reference [WA08] adopted Szegedy's quantum walks [Sze04] and amplitude amplification [BHMT02] to improve the time complexity of this qsampling procedure to $O(1/\sqrt{\delta})$, achieving a quadratic speedup in the spectral gap. As a generalization, [TOV+11] proposes a quantum Metropolis sampling method that extends qsampling to quantum Hamiltonians, with time complexity $O(1/\delta)$. Reference [YAG12] combines quantum Metropolis sampling with QSA to achieve time complexity $O(1/\sqrt{\delta})$. Another alternative approach is quantum rejection sampling [ORR13, LYC14, WG15], which provides a method for transforming an initial superposition of desired and undesired states into the desired state using amplitude amplification. Reference [WG15] employs semi-classical Bayesian updating to achieve time complexity $O(1/\sqrt{\epsilon})$ as a function of the approximation error ϵ. The quantum rejection sampling approach is generally less efficient than the QSA approach, as the latter can achieve $O(\log 1/\epsilon)$ by choosing proper slowly varying MCs that mix rapidly.

Previous quantum computing literature on partition function estimation mainly focused on discrete systems with

$$Z(\beta) = \sum_{x\in\Omega} e^{-\beta H(x)}, \tag{2.16}$$

where β is an inverse temperature and H is a Hamiltonian function of x over a finite state space Ω. The space Ω is usually assumed to be a simple discrete set, such as $\{0,1\}^n$, and H is assumed to be a sum of local terms. For instance, [Mon15] considers H taking integer values $\{0, 1, \dots, n\}$, and [HW20] assumes $0 \le H(x) \le n$ for all x.

To estimate $Z = Z(\infty)$ in (2.16), [Mon15] considers a classical Chebyshev cooling schedule $0 = \beta_0 < \beta_1 < \ldots \beta_l = \infty$ for Z with the length $l = O(\sqrt{\log|\Omega|}\log\log|\Omega|)$ [ŠVV09]. Reference [Mon15] applies fast qsampling algorithms to estimate Z with $\widetilde{O}(l^2/\sqrt{\delta}\epsilon) = \widetilde{O}(\log|\Omega|/\sqrt{\delta}\epsilon)$ quantum walk steps to sample from Gibbs distributions $\pi_i(x) = \frac{1}{Z(\beta_i)}e^{-\beta_i H(x)}$, whereas a corresponding classical algorithm takes $\widetilde{O}(l^2/\delta\epsilon^2) = \widetilde{O}(\log|\Omega|/\delta\epsilon^2)$ random walk steps. Here ϵ denotes the relative error for estimating Z, and δ denotes the spectral gap of the Markov chains with stationary distributions $\pi_i(x)$. Reference [Mon15] also points out that this quantum algorithm relies on classical Markov chain Monte Carlo for computing the Chebyshev cooling schedule, introducing an overhead of $\widetilde{O}(\log|\Omega|/\delta)$ [ŠVV09]. Hence, the overall cost is $\widetilde{O}(\log|\Omega|/\sqrt{\delta}(\epsilon + \sqrt{\delta}))$, a quadratic reduction with respect to ϵ over classical algorithms. Reference [HW20] develops a fully quantized version of the Chebyshev cooling schedule that only requires additional cost $\widetilde{O}(\log|\Omega|/\sqrt{\delta})$. This results in overall cost $\widetilde{O}(\log|\Omega|/\sqrt{\delta}\epsilon)$, a quadratic speedup in terms of δ over [Mon15] and classical algorithms. Reference [AHN$^+$21] constructs a shorter Chebyshev cooling schedule by using a paired-product estimator with length $l = O(\sqrt{\log|\Omega|})$, eliminating the $l = O(\log\log|\Omega|)$ factors in the previous schedule [ŠVV09]. Reference [AHN$^+$21] develops a fully quantized version of this shorter schedule, almost matching the same overall cost $\widetilde{O}(\log|\Omega|/\sqrt{\delta}(\epsilon + \sqrt{\delta}))$ of [HW20].

Estimating the partition function of a discrete system corresponds to a discrete counting problem, with applications such as counting colorings or matchings of a graph and estimating statistics of Ising models, while estimating partition functions of continuous systems is relevant to the volume estimation problem.

2.7 Tools from Classical MCMC Algorithms

2.7.1 ULD and ULD-RMM

We now describe underdamped Langevin diffusion (ULD) and the randomized midpoint method for underdamped Langevin diffusion (ULD-RMM), as introduced in [GLL20] with Lipschitz continuous constants. We consider the underdamped Langevin diffusion with parameters γ, u:

$$\mathrm{d}v_t = -\gamma v_t \,\mathrm{d}t - \nabla f(x_t)\,\mathrm{d}t + \sqrt{2\gamma u}\,\mathrm{d}W_t, \tag{2.17}$$

$$\mathrm{d}x_t = v_t \,\mathrm{d}t, \tag{2.18}$$

The discrete dynamics of ULD can be described by

$$\mathrm{d}v_t^h = -\gamma v_t^h \,\mathrm{d}t - u\nabla f(x^h_{\lfloor t/h \rfloor h})\,\mathrm{d}t + \sqrt{2\gamma u}\,\mathrm{d}W_t, \tag{2.19}$$

$$\mathrm{d}x_t^h = v_t^h \,\mathrm{d}t. \tag{2.20}$$

According to [GLL20], taking $\gamma = 2$ and $u = 1/L$, the explicit discrete-time update of ULD is integrated as

$$v^h_{t+h} = e^{-2h} v^h_t + \frac{1}{2L}(1 - e^{-2h})\nabla f(x^h_t) + \frac{2}{\sqrt{L}} W^h_{1,t}, \tag{2.21}$$

$$x^h_{t+h} = x^h_t + \frac{1}{2}(1 - e^{-2h}) v^h_t + \frac{1}{2L}[h - (1 - e^{-2h})]\nabla f(x^h_t) + \frac{1}{\sqrt{L}} W^h_{2,t}, \tag{2.22}$$

where

$$W^h_{1,t} = \int_0^h e^{2(s-h)} \mathrm{d}B_{t+s}, \tag{2.23}$$

$$W^h_{2,t} = \int_0^h (1 - e^{2(s-h)}) \mathrm{d}B_{t+s}. \tag{2.24}$$

$W^h_{1,t}$ and $W^h_{2,t}$ can be obtained by sampling the d-dimensional standard Brownian motion B_t.

The ULD algorithm is presented in Algorithm 4. The convergence of ULD has been established by Theorem 1 of [CCBJ18], which was restated by Theorem C.3 of [GLL20] as follows.

Lemma 2.10 (Theorem 1 of [CCBJ18]). *Assume the target distribution ρ is strongly log-concave with L-smooth and μ-strongly convex negative log-density. Let ρ_n be the distribution of the underdamped Langevin diffusion with the initial point x_0 satisfying*

Algorithm 4 Underdamped Langevin Dynamics (ULD)

```
1: procedure ULD($f$, $h$, $T$, $x_0$)    ▷ $f$ is the function, $h$ is the step size, $T$ is the time, and $x_0$ is sampled from the initial distribution $\rho_0$
2:     $x_0^h \leftarrow x_0$
3:     for $t = 0, h, \ldots, \lfloor T \rfloor$ do
4:         Draw $W_{1,t}^h = \int_0^h e^{2(s-h)} \mathrm{d}B_{t+s}$, $W_{2,t}^h = \int_0^h (1 - e^{2(s-h)}) \mathrm{d}B_{t+s}$
5:         $v_{t+h}^h \leftarrow e^{-2h} v_t^h + \frac{1}{2L}(1 - e^{-2h}) \nabla f(x_t^h) + \frac{2}{\sqrt{L}} W_{1,t}^h$
6:         $x_{t+h}^h \leftarrow x_t^h + \frac{1}{2}(1 - e^{-2h}) v_t^h + \frac{1}{2L}[h - (1 - e^{-2h})] \nabla f(x_t^h) + \frac{1}{\sqrt{L}} W_{2,t}^h$
7:     end for
8:     return $x_h^h, x_{2h}^h, \ldots, x_{\lfloor T \rfloor + 1}^h$
9: end procedure
```

$\|x_0 - x^*\| \leq D$, step size $h \leq \frac{\epsilon}{104\kappa}\sqrt{\frac{1}{d/\mu + D^2}}$, and time $T \geq \frac{\kappa}{2} \log\Big(\frac{24\sqrt{d/\mu + D^2}}{\epsilon}\Big)$. Then ULD achieves

$$\mathbb{E}\Big(\|\widehat{X}_n - X_T\|^2\Big) \leq \widetilde{O}\Big(\frac{d^2\kappa^2h^2}{\mu}\Big), \tag{2.25}$$

$$W_2(\rho_n, \rho) \leq \epsilon, \tag{2.26}$$

using

$$\frac{T}{h} = \widetilde{\Theta}\Big(\frac{\kappa^2\sqrt{d}}{\epsilon}\Big) \tag{2.27}$$

queries to ∇f.

According to [GLL20], the explicit discrete-time update of ULD-RMM is integrated as

$$v_{t+h}^h = e^{-2h} v_t^h + \frac{h}{L} e^{-2(1-\alpha)h} \nabla f(y_t^h) + \frac{2}{\sqrt{L}} W_{1,t}^h, \tag{2.28}$$

$$x_{t+h}^h = x_t^h + \frac{1}{2}(1 - e^{-2h}) v_t^h + \frac{h}{2L}(1 - e^{-2(1-\alpha)h}) \nabla f(y_t^h) + \frac{1}{\sqrt{L}} W_{2,t}^h, \tag{2.29}$$

$$y_{t+h}^h = x_t^h + \frac{1}{2}(1 - e^{-2\alpha h}) v_t^h + \frac{1}{2L}[\alpha h - (1 - e^{-2\alpha h})] \nabla f(x_t^h) + \frac{1}{\sqrt{L}} W_{3,t}^h, \tag{2.30}$$

where

$$W_{1,t}^h = \int_0^h e^{2(s-h)} \mathrm{d}B_{t+s}, \tag{2.31}$$

$$W_{2,t}^h = \int_0^h (1 - e^{2(s-h)}) \mathrm{d}B_{t+s}, \tag{2.32}$$

$$W_{3,t}^h = \int_0^{\alpha h} (1 - e^{2(s-h)}) \mathrm{d}B_{t+s}. \tag{2.33}$$

$W_{1,t}^h$, $W_{2,t}^h$, and $W_{3,t}^h$ can be obtained by sampling the d-dimensional standard Brownian motion B_t.

The ULD-RMM algorithm is presented in Algorithm 5. The convergence of ULD-RMM has been established by Theorem 3 of [SL19], which was restated by Theorem C.5 of [GLL20] as follows.

Algorithm 5 Underdamped Langevin Dynamics with Randomized Midpoint Method (ULD-RMM)

```
1: procedure ULD-RMM($f$, $h$, $T$, $x_0$)  ▷ $f$ is the function, $h$ is the step size, $T$ is the time, and $x_0$ is sampled from the initial distribution $\rho_0$
2:     $x_0^h \leftarrow x_0$, $y_0^h \leftarrow x_0$
3:     for $t = 0, h, \ldots, \lfloor T \rfloor$ do
4:         Draw $W_{1,t}^h = \int_0^h e^{2(s-h)} \mathrm{d}B_{t+s}$, $W_{2,t}^h = \int_0^h (1 - e^{2(s-h)}) \mathrm{d}B_{t+s}$, $W_{3,t}^h = \int_0^{\alpha h} (1 - e^{2(s-h)}) \mathrm{d}B_{t+s}$
5:         $v_{t+h}^h \leftarrow e^{-2h} v_t^h + \frac{h}{L} e^{-2(1-\alpha)h} \nabla f(y_t^h) + \frac{2}{\sqrt{L}} W_{1,t}^h$
6:         $x_{t+h}^h \leftarrow x_t^h + \frac{1}{2}(1 - e^{-2h}) v_t^h + \frac{h}{2L}(1 - e^{-2(1-\alpha)h}) \nabla f(y_t^h) + \frac{1}{\sqrt{L}} W_{2,t}^h$
7:         $y_{t+h}^h \leftarrow x_t^h + \frac{1}{2}(1 - e^{-2\alpha h}) v_t^h + \frac{1}{2L}[\alpha h - (1 - e^{-2\alpha h})] \nabla f(x_t^h) + \frac{1}{\sqrt{L}} W_{3,t}^h$
8:     end for
9:     return $x_h^h, x_{2h}^h, \ldots, x_{\lfloor T \rfloor + 1}^h$
10: end procedure
```

Lemma 2.11 (Theorem 3 of [SL19]). *Assume the target distribution ρ is strongly log-concave with L-smooth and μ-strongly convex negative log-density. Let ρ_n be the distribution of the randomized midpoint method for underdamped Langevin diffusion with the initial point x_0, step size $h \le \min\left\{ \frac{\epsilon^{1/3} \mu^{1/6}}{\kappa^{1/6} d^{1/6} \log^{1/6}\left(\frac{\sqrt{d/\mu}}{\epsilon}\right)}, \frac{\epsilon^{2/3} \mu^{1/3}}{d^{1/3} \log^{1/3}\left(\frac{\sqrt{d/\mu}}{\epsilon}\right)} \right\}$, and* *time* $T \geq 2\kappa \log\Big(\frac{20d/\mu}{\epsilon^2}\Big)$. *Then ULD-RMM achieves*

$$\mathbb{E}\Big(\|\widehat{X}_n - X_T\|^2\Big) \leq \widetilde{O}\Big(\frac{d\kappa h^6}{\mu} + \frac{dh^3}{\mu}\Big), \tag{2.34}$$

$$W_2(\rho_n, \rho) \leq \epsilon, \tag{2.35}$$

using

$$\frac{2T}{h} = \widetilde{\Theta}\Big(\frac{\kappa^{7/6} d^{1/6}}{\epsilon^{1/3}} + \frac{\kappa d^{1/3}}{\epsilon^{2/3}}\Big) \tag{2.36}$$

queries to ∇f.

2.7.2 Annealing for estimating the normalizing constant

Having described the sampling procedure for a log-concave function, we now move to the problem of estimating the normalizing constant

$$Z = \int_{x \in \mathbb{R}^d} e^{-f(x)} \mathrm{d}x. \tag{2.37}$$

We consider a sequence of auxiliary distributions, given by

$$f_i(x) = \frac{1}{2} \frac{\|x\|^2}{\sigma_i^2} + f(x) \tag{2.38}$$

for $i \in [M]$, where $\sigma_1 \leq \sigma_2 \leq \cdots \leq \sigma_M$. We define $\sigma_{M+1} = \infty$ and $f_{M+1} = f$ for convenience. We consider the sequence of distributions

$$\rho_i(\mathrm{d}x) = Z_i^{-1} e^{-f_i(x)} \mathrm{d}x, \tag{2.39}$$

where Z_i is the normalizing constant

$$Z_i = \int_{x \in \mathbb{R}^d} e^{-f_i(x)} \mathrm{d}x. \tag{2.40}$$

Then Z is estimated by the telescoping product

$$Z = Z_{M+1} = Z_1 \prod_{i=1}^{M} \frac{Z_{i+1}}{Z_i}. \tag{2.41}$$

In (2.41), we first approximate Z_1 by the normalizing constant of the Gaussian distribution with variance σ_1^2, which is bounded by the following lemma.

Lemma 2.12 (Lemma 3.1 of [GLL20]). *Letting $\sigma_1^2 = \frac{\epsilon}{2dL}$, we have*

$$\Big(1 - \frac{\epsilon}{2}\Big) \int_{x\in\mathbb{R}^d} e^{-\frac{1}{2}\frac{\|x\|^2}{\sigma_1^2}} \mathrm{d}x \le Z_1 \le \int_{x\in\mathbb{R}^d} e^{-\frac{1}{2}\frac{\|x\|^2}{\sigma_1^2}} \mathrm{d}x. \tag{2.42}$$

We then approximate $\frac{Z_{i+1}}{Z_i}$ by sampling the distribution ρ_i, with

$$\frac{Z_{i+1}}{Z_i} = \mathbb{E}_{\rho_i}(g_i), \tag{2.43}$$

where

$$g_i = \exp\Big(\frac{1}{2}\big(\frac{1}{\sigma_i^2} - \frac{1}{\sigma_{i+1}^2}\big)\|x\|^2\Big). \tag{2.44}$$

If $X_i^{(1)}, X_i^{(2)}, \dots, X_i^{(K)}$ are i.i.d. samples generated according to the distribution ρ_i, then

$$\frac{Z_{i+1}}{Z_i} \approx \frac{1}{K}\sum_{k=1}^{K} g_i(X_i^{(k)}). \tag{2.45}$$

For the sequence of σ_i^2 with the annealing schedule $\frac{\sigma_{i+1}^2}{\sigma_i^2} = 1 + \alpha$, we aim to bound the relative variance of $\frac{Z_{i+1}}{Z_i}$. First, for $\frac{Z_{M+1}}{Z_M}$, we have the following lemma.

Lemma 2.13 (Lemma 3.2 of [GLL20]). *For $\sigma_M^2 \ge \frac{2}{\mu}$, we have*

$$\frac{\mathbb{E}_{\rho_M}(g_M^2)}{\mathbb{E}_{\rho_M}(g_M)^2} \le \exp\big(\frac{4d}{\mu\sigma_M^4}\big). \tag{2.46}$$

When $\sigma_M^2 \ge \frac{2\sqrt{d}}{\mu}$, and assuming $\mu < 1$, we have $\frac{\mathbb{E}_{\rho_M}(g_M^2)}{\mathbb{E}_{\rho_M}(g_M)^2} \le e$.

Second, for $\frac{Z_{i+1}}{Z_i}$ with $i \in [M-1]$, the relative variance of can be bounded by the following lemma.

Lemma 2.14 ([Lemma 3.3 of [GLL20]). *Let ρ be a log-concave distribution. For $\alpha \le \frac{1}{2}$, we have*

$$\frac{\mathbb{E}_{\rho_i}(g_i^2)}{\mathbb{E}_{\rho_i}(g_i)^2} = \frac{\mathbb{E}_\rho\Big[\exp(-\frac{1+\alpha}{2}\frac{\|x\|^2}{\sigma^2})\Big]\cdot\mathbb{E}_\rho\Big[\exp(-\frac{1-\alpha}{2}\frac{\|x\|^2}{\sigma^2})\Big]}{\mathbb{E}_\rho\Big[\exp(-\frac{1}{2}\frac{\|x\|^2}{\sigma^2})\Big]^2} \le \exp(4\alpha^2 d). \tag{2.47}$$

Therefore, if we choose the annealing schedule

$$\frac{\sigma_{i+1}^2}{\sigma_i^2} = 1 + \frac{1}{\sqrt{d}}, \tag{2.48}$$

then $\frac{\mathbb{E}_{\rho_i}(g_i^2)}{\mathbb{E}_{\rho_i}(g_i)^2} \le e^4$.

The estimate of the normalizing constant (2.3) relies on the above annealing framework and the sampling algorithms for the log-concave distribution ρ_i including ULD, ULD-RMM, and MALA. In the following sections, we discuss the quantum speedup for (2.3) using MALA and annealing and using multilevel ULD/ULD-RMM and annealing.

2.7.3 Annealing Markov chains are slowly varying

The goal of this subsection is to prove the following lemma.

Lemma 2.15 (Slowly varying MCs)**.** *Let* $f_0(x) = \frac{\|x\|^2}{2\sigma_1^2}$ *and let* $\mathrm{d}\pi_0 = (2\pi\sigma_1^2)^{d/2} \cdot e^{-f_0(x)}\mathrm{d}x$ *be the Gaussian distribution. For* $i \in \{1, \dots, M\}$*, let* $f_i(x) = f(x) + \frac{\|x\|^2}{2\sigma_i^2}$ *and let* $\mathrm{d}\pi_i = Z_i^{-1}e^{-f_i(x)}\mathrm{d}x$ *be its stationary distribution. Let* $f_{M+1}(x) = f(x)$ *and let* $\mathrm{d}\pi_{M+1}$ *be the target log-concave distribution. Define the qsample state*

$$|\pi_i\rangle = \int_\Omega \mathrm{d}x \sqrt{\pi_i(x)}|x\rangle \quad \forall\, 0 \le i \le M+1. \tag{2.49}$$

Then, for $0 \le i \le M$*, we have*

$$|\langle \pi_i | \pi_{i+1}\rangle| \ge \Omega(1). \tag{2.50}$$

Proof. First, we consider the case when $i = 0$. Note that $|\langle \pi_0|\pi_1\rangle|$ can be written as

$$|\langle \pi_0|\pi_1\rangle| = \int_\Omega \mathrm{d}x \cdot (2\pi\sigma_1^2)^{-d/4} e^{-\frac{1}{2}f_0(x)} \cdot Z_1^{-1/2} e^{-\frac{1}{2}f_1(x)} \tag{2.51}$$

$$= \frac{\int_\Omega e^{-\frac{1}{2}f(x) - \frac{\|x\|^2}{2\sigma_1^2}}\mathrm{d}x}{(2\pi\sigma_1^2)^{d/4} \cdot \sqrt{Z_1}}. \tag{2.52}$$

Since $0 \le f(x) \le \frac{1}{2}L\|x\|^2$, the numerator can be lower bounded by

$$\int_\Omega e^{-\frac{1}{2}f(x)-\frac{\|x\|^2}{2\sigma_1^2}}\mathrm{d}x \ge \int_\Omega e^{-\frac{1}{2}(L/2+\sigma_1^{-2})\|x\|^2}\mathrm{d}x = \left(2\pi(L/2+\sigma_1^{-2})^{-1}\right)^{d/2} \tag{2.53}$$

and the denominator can be upper bounded by

$$(2\pi\sigma_1^2)^{d/4}\cdot\sqrt{\int_\Omega e^{-f(x)-\frac{1}{2}\|x\|^2/\sigma_1^2}\mathrm{d}x} \le (2\pi\sigma_1^2)^{d/4}\cdot\sqrt{\int_\Omega e^{-\frac{1}{2}\|x\|^2/\sigma_1^2}\mathrm{d}x} = (2\pi\sigma_1^2)^{d/2}. \tag{2.54}$$

Therefore

$$|\langle\pi_0|\pi_1\rangle| \ge \frac{\left(2\pi(L/2+\sigma_1^{-2})^{-1}\right)^{d/2}}{(2\pi\sigma_1^2)^{d/2}} = (1+\sigma_1^2L/2)^{-d/2} \ge e^{-\sigma_1^2 dL/4}. \tag{2.55}$$

By our choice of $\sigma_1^2 = \frac{\epsilon}{2dL}$, we have $|\langle\pi_0|\pi_1\rangle| \ge e^{-\epsilon/8} = \Omega(1)$.

Now consider the case where $1 \le i \le M-1$. The inner product between $|\pi_i\rangle$ and $|\pi_{i+1}\rangle$ can be written as

$$|\langle\pi_i|\pi_{i+1}\rangle| = \int_\Omega \mathrm{d}x \cdot Z_i^{-1/2}e^{-\frac{1}{2}f_i(x)}\cdot Z_{i+1}^{-1/2}e^{-\frac{1}{2}f_{i+1}(x)} \tag{2.56}$$

$$= \frac{\int_\Omega e^{-f(x)-\frac{\|x\|^2}{4}(\sigma_i^{-2}+\sigma_{i+1}^{-2})}\mathrm{d}x}{\sqrt{Z_iZ_{i+1}}}. \tag{2.57}$$

Let $\sigma^2 = \sigma_{i+1}^2$ and $\sigma^2/(1+\alpha) = \sigma_i^2$. Also, let ρ be the log-concave distribution $\rho(\mathrm{d}x) = Z^{-1}e^{-f(x)}\mathrm{d}x$. Then we have

$$\int_\Omega e^{-f(x)-\frac{\|x\|^2}{4}(\sigma_i^{-2}+\sigma_{i+1}^{-2})}\mathrm{d}x = Z\cdot\mathbb{E}_\rho\left[e^{-\frac{1+\alpha/2}{2\sigma^2}\|x\|^2}\right]. \tag{2.58}$$

Similarly,

$$Z_i = Z\cdot\mathbb{E}_\rho\left[e^{-\frac{1+\alpha}{2\sigma^2}\|x\|^2}\right] \quad\text{and}\quad Z_{i+1} = Z\cdot\mathbb{E}_\rho\left[e^{-\frac{1}{2\sigma^2}\|x\|^2}\right]. \tag{2.59}$$

Hence,

$$|\langle\pi_i|\pi_{i+1}\rangle| = \frac{\mathbb{E}_\rho\left[e^{-\frac{1+\alpha/2}{2\sigma^2}\|x\|^2}\right]}{\mathbb{E}_\rho\left[e^{-\frac{1+\alpha}{2\sigma^2}\|x\|^2}\right]^{1/2}\cdot\mathbb{E}_\rho\left[e^{-\frac{1}{2\sigma^2}\|x\|^2}\right]^{1/2}}. \tag{2.60}$$

Let $\alpha' := \frac{\alpha}{\alpha+2}$ and $\sigma'^2 := \frac{\sigma^2}{1+\alpha/2}$. Then

$$|\langle \pi_i | \pi_{i+1} \rangle| = \frac{\mathbb{E}_\rho\left[e^{-\frac{1}{2\sigma'^2}\|x\|^2}\right]}{\mathbb{E}_\rho\left[e^{-\frac{1+\alpha'}{2\sigma'^2}\|x\|^2}\right]^{1/2} \cdot \mathbb{E}_\rho\left[e^{-\frac{1-\alpha'}{2\sigma'^2}\|x\|^2}\right]^{1/2}} \tag{2.61}$$

$$\geq e^{-2\alpha'^2 d}, \tag{2.62}$$

where the last step follows from Lemma 2.14. Since we choose $\alpha = d^{-1/2}$, we have $\alpha' = \frac{1}{1+2\sqrt{d}} = O(d^{-1/2})$, which implies that $e^{-2\alpha^2 d} = \Omega(1)$.

Finally, we consider the case where $i = M$. The inner product can be written as

$$|\langle \pi_M | \pi_{M+1} \rangle| = \frac{\int_\Omega \mathrm{d}x \cdot e^{-\frac{1}{2}f(x) - \frac{1}{4}\|x\|^2/\sigma_M^2} \cdot e^{-\frac{1}{2}f(x)}}{\sqrt{Z_M}\sqrt{Z}} \tag{2.63}$$

$$= \frac{\int_\Omega \mathrm{d}x \cdot e^{-f(x) - \frac{1}{4}\|x\|^2/\sigma_M^2}}{\sqrt{Z_M}\sqrt{Z}}. \tag{2.64}$$

Let ρ' be a log-concave distribution with density proportional to $e^{-f(x)-\frac{1}{4}\|x\|^2/\sigma_M^2}$. Then

$$\frac{\int_\Omega \mathrm{d}x \cdot e^{-f(x) - \frac{1}{4}\|x\|^2/\sigma_M^2}}{\sqrt{Z_M}\sqrt{Z}} = \mathbb{E}_{\rho'}\left[e^{-\frac{1}{4}\|x\|^2/\sigma_M^2}\right]^{-1/2} \cdot \mathbb{E}_{\rho'}\left[e^{\frac{1}{4}\|x\|^2/\sigma_M^2}\right]^{-1/2} \tag{2.65}$$

$$\geq e^{-\frac{d}{2\mu\sigma_M^4}}, \tag{2.66}$$

where the second step follows from the proof of Lemma 2.13 in [GLL20]. Since we take $\sigma_M^2 = \Theta(\frac{\sqrt{d}}{\mu})$, we find that $|\langle \pi_M | \pi_{M+1} \rangle| \geq e^{-\Theta(1)} = \Omega(1)$.

Combining the three cases, the proof is complete. □

2.8 Basic Facts about Quantum Walk

In this section, we first define the quantum walk operators and introduce some spectral properties. Then, we show how to efficiently implement a quantum walk.

2.8.1 Definitions and spectral properties of quantum walk

Let P be the transition operator of the classical Markov chain over the space K such that

$$\int_K P(x,y)\mathrm{d}y = 1 \quad \forall x \in K.$$

We define the following states, which capture key properties of the quantum walk:

$$|\psi_x\rangle := \int_K \sqrt{P(x,y)}|y\rangle \mathrm{d}y \quad \forall x \in K.$$

Definition 2.1 (Quantum walk operators)**.** The quantum walk uses the following three operators:

- $U := \int_K |x\rangle|\psi_x\rangle\langle x|\langle 0|\mathrm{d}x$ for any $x \in K$.
- $\Pi := \int_K |x\rangle|\psi_x\rangle\langle x|\langle\psi_x|\mathrm{d}x$ is the projection to the subspace $\mathrm{span}\{|x\rangle|\psi_x\rangle\}_{x\in K}$.
- $S := \int_K\int_K |y\rangle|x\rangle\langle x|\langle y|\mathrm{d}x\mathrm{d}y$ is to swap the two quantum registers.

Then, the quantum walk operator W is defined by:

$$W := S(2\Pi - I).$$

Definition 2.2 (Alternative definition of quantum walk operator, [WA08])**.** Define the quantum walk operator

$$W' := U^\dagger S U R_{\mathcal{A}} U^\dagger S U R_{\mathcal{A}},$$

where $R_{\mathcal{A}}$ denotes the reflection about the subspace $\mathcal{A} := \{|x\rangle|0\rangle \mid x \in K\}$ for random walk space K, S is the swap operator, and U is the following operator:

$$U|x\rangle|0\rangle = \int_{y\in K} \sqrt{P(x,y)}|x\rangle|y\rangle\mathrm{d}y,$$

for P being the transition operator of the Markov chain.

Fact 2.16 (Equivalence of the definitions, [CCH+19])**.** *Let W be defined as in Definition 2.1 and let W' be defined as in Definition 2.2. Then, W' and W have the same set of eigenvalues.*

Fact 2.17 ([CCH+19])**.** *Let D be the discriminant operator of P defined as $D(x, y) := \sqrt{P(x, y)P(y, x)}$. Then, P and D have the same set of eigenvalues.*

Fact 2.18 ([CCH+19])**.** *Let $\{\lambda_j\}$ be the eigenvalues of D. Then, the eigenvalues of W are*

$$\left\{\pm 1, \lambda_j + \sqrt{1 - \lambda_j^2} i\right\}.$$

If we have a sequence of slowly varying log-concave distributions $\rho_0, \ldots, \rho_r$, we can quantumly sample from ρ_r via MALA with a quadratic speedup.

Theorem 2.19 (Quantum speedup for slowly varying Markov chains [WA08])**.** *Let $M_0, \ldots, M_r$ be classical reversible Markov chains with stationary distributions $\rho_0, \ldots, \rho_r$ such that each chain has spectral gap at least δ^{-1}. Assume that $|\langle \rho_i | \rho_{i+1} \rangle| \geq p$ for some $p > 0$ and all $i \in \{0, \ldots, r-1\}$, and that we can prepare the state $|\rho_0\rangle$. Then, for any $\epsilon > 0$, there is a quantum algorithm which produces a quantum state $|\widetilde{\rho}_r\rangle$ such that $\| |\widetilde{\rho}_r\rangle - |\rho_r\rangle |0^a\rangle \| \leq \epsilon$, for some integer a. The algorithm uses*

$$\tilde{O}\left(\delta^{-1/2} \cdot \frac{r}{p}\right) \tag{2.67}$$

applications of the quantum walk operators W_i' corresponding to the chains M_i for $i \in [r]$.

However, we cannot directly apply Theorem 2.19 to speed up many "useful" Markov chains in the continuous space (e.g., the hit-and-run walk [LV07], MALA [LST20], etc.) since their spectral gaps are very difficult to bound directly. Classically, several techniques have been developed to overcome this issue. One approach is to only consider good initial distributions, i.e., warm-starts. Another approach is to bound more restricted quantities that discard some small and problematic sets when computing the spectral gap, e.g., s-conductance. It is an interesting question to see whether these techniques can be adapted to a quantum setting. [CCLW20] used the idea of the effective spectral gap of a Markov chain. The intuition is that suppose we know the mixing time $t = t_{\text{mix}}(\epsilon, \pi_0)$ for some initial distribution π_0, then we can show that $|\pi_0\rangle$ has very small overlap with the eigenspace of W' with the corresponding eigenvalues of P that are very close to 1. More formally,

Lemma 2.20 (Effective spectral gap for ℓ_2-warm start, [CLW19])**.** *Let $M = (\Omega, p)$ be an ergodic reversible Markov chain with a transition operator P and unique stationary state with a corresponding density ρ. Let $\{(\lambda_i, f_i)\}$ be the set of eigenvalues and eigenfunctions of P, and $|\psi_i\rangle$ be the eigenvectors of the corresponding quantum walk operator W. Let ρ_0 be a probability density that is a warm start for ρ and mixes up to TV-distance ϵ in t steps of M. Furthermore, assume that $\|\rho/\rho_0\| = \int_\Omega \frac{\rho(x)}{\rho_0(x)}\rho(x)\mathrm{d}x \leq \gamma$.*

Let $|\phi_{\rho_0}\rangle$ be the resulting state of applying the quantum walk update operator U to the state $|\rho_0\rangle$:

$$|\phi_{\rho_0}\rangle = \int_\Omega \sqrt{\rho_0(x)} \int_\Omega \sqrt{P(x,y)}|x\rangle|y\rangle \mathrm{d}x\mathrm{d}y.$$

Then, we have $|\langle\phi_{\rho_0}|\psi_i\rangle| = O(\gamma^{1/4}\epsilon^{3/4} + \sqrt{\epsilon})$ for all i with $1 > \lambda_i \geq 1 - O(1/t)$.

Note that Lemma 2.20 requires the initial distribution to satisfy an L_2-norm condition. We can relax this requirement in the following lemma, which only relies on the standard warmness of the initial distribution.

Lemma 2.21 (Effective spectral gap for warm start)**.** *Let $M = (\Omega, p)$ be an ergodic reversible Markov chain with a transition operator P and unique stationary state with a corresponding density ρ. Let $\{(\lambda_i, f_i)\}$ be the set of eigenvalues and eigenfunctions of P, and let $|u_i\rangle$ be the eigenvectors of the corresponding quantum walk operator W. Let ρ_0 be a probability density that is a warm start for ρ and mixes up to TV-distance*

ϵ in t steps of M. Furthermore, assume that ρ_0 is a β-warm start of ρ. Let $|\phi_{\rho_0}\rangle$ be the state obtained by applying the quantum walk update operator U to the state $|\rho_0\rangle$:

$$|\phi_{\rho_0}\rangle = \int_\Omega \sqrt{\rho_0(x)} \int_\Omega \sqrt{p_{x\to y}} |x\rangle|y\rangle \,\mathrm{d}x\,\mathrm{d}y. \tag{2.68}$$

Then $|\langle\phi_{\rho_0}|u_i\rangle| = O(\beta\sqrt{\epsilon})$ for all i with $1 > |\lambda_i| \geq 1 - O(1/t)$.

Furthermore, for MALA with ρ_0 being $\mathcal{N}(0, L^{-1}I)$, $|\langle\phi_{\rho_0}|u_i\rangle| = O(\sqrt{\epsilon})$.

Remark 2.1. Since $|v_i\rangle = U^\dagger|u_i\rangle$ is the corresponding eigenvector of W', and $|\phi_{\rho_0}\rangle = U|\rho_0, 0\rangle$, Lemma 2.21 implies that $|\langle\rho_0, 0|v_i\rangle| \leq \beta\sqrt{\epsilon}$ for any i with $|\lambda_i| \in (1 - O(1/t), 1)$. In other words, effectively, the spectral gap is $\Omega(1/t)$.

Proof. Let $S := \left\{x \in \mathbb{R}^d : \frac{\rho(x)}{\rho_0(x)} \geq \frac{1}{\epsilon}\right\}$. Since $\mathbb{E}_{\rho_0}\left[\frac{\rho(x)}{\rho_0(x)}\right] = 1$, by Markov's inequality, we have

$$\int_S \rho_0(x)\mathrm{d}x = \Pr_{\rho_0}[x \in S] \leq \epsilon. \tag{2.69}$$

Then we define a quantum state $|\rho_1\rangle$ such that $\langle\rho_1|x\rangle = \langle\rho_0|x\rangle$ for $x \notin S$, and $\langle\rho_1|x\rangle = 0$ for $x \in S$. Furthermore, let $|\phi_{\rho_1}\rangle := U|\rho_1\rangle$.

We have

$$\||\phi_{\rho_0}\rangle - |\phi_{\rho_1}\rangle\| = \left\|\int_S \sqrt{\rho_0(x)} T|x\rangle \mathrm{d}x\right\| = \left|\int_S \rho_0(x)\mathrm{d}x\right|^{1/2} \leq \sqrt{\epsilon}, \tag{2.70}$$

where T is the isometry

$$T := \int_\Omega \int_\Omega \sqrt{p_{x\to y}} |x, y\rangle\langle x| \,\mathrm{d}x\,\mathrm{d}y.$$

Moreover, by Eqs. (4.35) and (4.36) in [CCH+19], if $1 > \lambda_i \geq 1 - O(1/t)$, we have

$$|\langle\phi_{\rho_1}|u_i\rangle| \leq 2|\langle\rho_1|v_i\rangle| = 2\left|\int_{\overline{S}} \sqrt{\frac{\rho(x)}{\rho_0(x)}} \frac{\rho_0(x) f_i(x)}{\rho(x)} \mathrm{d}x\right| \leq \frac{2\langle\rho_0, f_i\rangle_\rho}{\epsilon^{1/2}} = O(\beta\sqrt{\epsilon}), \tag{2.71}$$

where the third step follows from $\frac{\rho(x)}{\rho_0(x)} \leq \frac{1}{\epsilon}$ for $x \notin S$ and the Cauchy-Schwarz inequality, and the last step follows from the claim that $\langle\rho_0, f_i\rangle_\rho = O(\beta\epsilon)$.

Combining these observations, we find

$$|\langle \phi_{\rho_0} | u_i \rangle| \leq |\langle \phi_{\rho_0} - \phi_{\rho_1} | u_i \rangle| + |\langle \phi_{\rho_1} | u_i \rangle| \leq \sqrt{\epsilon} + O(\beta\sqrt{\epsilon}) = O(\beta\sqrt{\epsilon}) \tag{2.72}$$

when $1 > \lambda_i \geq 1 - O(1/t)$, which gives the desired result.

Now, it remains to prove the claim that $\langle \rho_0, f_i \rangle_\rho = O(\beta\epsilon)$ for $1 > \lambda_i > 1 - O(1/t)$. Suppose ρ_0 can be decomposed in the eigenbasis of P as $\rho_0 = \rho + \sum_{i=2}^{\infty} \langle \rho_0, f_i \rangle_\rho f_i$. Then $P^t \rho_0 = \rho + \sum_{i=2}^{\infty} \lambda_i^t \langle \rho_0, f_i \rangle_\rho f_i$, where λ_i is the eigenvalue of f_i. Since $\|P^t \rho_0 - \rho\|_1 \leq \epsilon$, by Fact 2.22, we have $\|P^t \rho_0 - \rho\|_\rho \leq \beta\epsilon$. Hence, by the orthogonality of f_i, we have $\lambda_i^t \langle a, f_i \rangle_\rho \leq \beta\epsilon$. Therefore, when $1 > \lambda_i > 1 - O(1/t)$, we have $\lambda_i^t = \Omega(1)$, which implies that $\langle \rho_0, f_i \rangle_\rho = O(\beta\epsilon)$.

For MALA, by Fact 2.22, $\|P^{O(t)} \rho_0 - \rho\|_\rho \leq O(\epsilon)$, which implies that $\langle \rho_0, f_i \rangle_\rho = O(\epsilon)$ by the same argument. Therefore, we get that for MALA with Gaussian initial distribution, $|\langle \phi_{\rho_0} | u_i \rangle| \leq O(\sqrt{\epsilon})$ for i with $1 > |\lambda_i| > 1 - O(1/t)$. $\square$

Fact 2.22. *Let ρ_0 be a β-warm start of a Markov chain with transition operator P and stationary distribution ρ. If $\|P^t \rho_0 - \rho\|_1 \leq \epsilon$ for some $t > 0$, then $\|P^t \rho_0 - \rho\|_\rho \leq \beta \cdot \epsilon$. For MALA with $\rho_0 = \mathcal{N}(0, L^{-1} I)$, $\|P^{O(t)} \rho_0 - \rho\|_\rho \leq \epsilon$.*

Proof. We expand $\|P^t \rho_0 - \rho\|_\rho$ in terms of its definition, giving

$$\|P^t \rho_0 - \rho\|_\rho = \int_{\mathbb{R}^d} \frac{(P^t \rho_0 - \rho)^2(x)}{\rho(x)} \mathrm{d}x = \int_{\mathbb{R}^d} \frac{(P^t \rho_0)^2(x)}{\rho(x)} \mathrm{d}x - 1 = \chi^2(P^t \rho_0, \rho), \tag{2.73}$$

where the second step follows since $P^t \rho_0$ and ρ are distributions, and the last step follows from the definition of χ^2-distance. Let $\rho_1 := P^t \rho_0$. By Lemma 27 in [CLA+21], we know that ρ_1 is also β-warm. Furthermore, by Lemma 28 in [CLA+21], the χ-square distance can be upper bounded by

$$\chi^2(P^t \rho_0, \rho) \leq \beta \cdot \|P^t \rho_0 - \rho\|_1 \leq \beta\epsilon \tag{2.74}$$

as claimed. For MALA, as shown in [CDWY20], MALA with a Gaussian start also mixes in about $O(t)$ steps in $\sqrt{\chi^2}$ metric. Therefore, we have $\chi^2(P^{O(t)} \rho_0, \rho) \leq \epsilon$. $\square$

2.8.2 Efficient implementation of quantum walk

The goal of this section is to give a user-friendly quantum walk implementation cost analysis (Theorem 2.25).

Lemma 2.23 (Approximate reflector, [CCH+19, Corollary 4.1])**.** *Let W be a unitary operator with a unique leading eigenvector $|\psi_0\rangle$ with eigenvalue 1. Denote the remaining eigenvectors by $|\psi_j\rangle$ with corresponding eigenvaluese $e^{2\pi i\xi_j}$ for $j \geq 1$. For any $\Delta \in (0,1]$ and $\epsilon < 1/2$, define $a := \log(1/\Delta)$ and $c := \log(1/\sqrt{\epsilon})$. Let R be the reflector such that $R = \alpha|\psi_0\rangle\langle\psi_0| + (I - |\psi_0\rangle\langle\psi_0|)$.*

For any constant $\alpha \in \mathbb{C}$, there exists a quantum circuit $\widetilde{R}$ that uses $a \cdot c$ ancilla qubits and invokes the controlled-W gate $2^{a+1}c$ times such that

- $\widetilde{R}|\psi_0\rangle|0\rangle^{\otimes ac} = R|\psi_0\rangle|0\rangle^{\otimes ac}$.
- $\left\|\widetilde{R}|\psi_j\rangle|0\rangle^{\otimes ac} - R|\psi_j\rangle|0\rangle^{\otimes ac}\right\|_2 \leq \sqrt{\epsilon}$ *for $j \geq 1$ with $\xi_j \geq \Delta$.*

Lemma 2.24 ($\pi/3$-amplitude amplification, [WA08, Lemma 1])**.** *Let $|\psi\rangle, |\phi\rangle$ be two quantum states with $|\langle\psi|\phi\rangle| \geq p$ for some $p \in (0,1]$. Let $\omega = e^{i\pi/3}$. Define $R_\psi := \omega|\psi\rangle\langle\psi| + (I - |\psi\rangle\langle\psi|)$, and $R_\phi := \omega|\phi\rangle\langle\phi| + (I - |\phi\rangle\langle\phi|)$. Then, for $m \geq 1$, there exists a sequence of unitaries:*

$$V_0 = I, \quad V_{j+1} = V_j R_\psi V_j^\dagger R_\phi V_j \quad \forall j \in [m],$$

such that

$$|\langle\psi|V_m|\phi\rangle|^2 \geq 1 - (1-p)^{3^m}.$$

Furthermore, the unitaries R_ϕ, R_ψ and their inverses are used at most 3^m times in V_m.

Theorem 2.25 (Quantum walk implementation cost)**.** *Let M_0, M_1 be two ergodic reversible Markov chains with stationary distributions π_0, π_1, respectively. Suppose π_0 is β_0-warm with respect to M_1 and mixes up to total variation distance ϵ in $t_0(\epsilon)$*

steps. Similarly, suppose π_1 is β_1-warm with respect to M_0 and mixes in $t_1(\epsilon)$ steps. Let $\beta := \max\{\beta_0, \beta_1\}$. Moreover, we assume that $|\langle \pi_0 | \pi_1 \rangle| \geq p$.

Given $|\pi_0\rangle$, we can obtain a state $|\widetilde{\pi}_1\rangle$ such that $\| |\widetilde{\pi}_1\rangle - |\pi_1\rangle \|_2 \leq \epsilon$ using

$$O\Big(\sqrt{t_0(\epsilon) + t_1(\epsilon)} \cdot p^{-1} \log(\beta/p) \log^2(1/(p\epsilon))\Big)$$

calls to the controlled walk operators controlled-W_0', controlled-W_1'.

Proof. By assumption, we know that π_0 mixes in $t_0' = t_0(\epsilon_1/\beta_0^2)$ steps in M_1 to achieve total variation distance ϵ_1/β_0^2, where ϵ_1 is a parameter to be chosen later. Similarly, π_1 mixes in $t_1' = t_1(\epsilon_1/\beta_0^2)$ steps in M_0 to achieve total variation distance ϵ_1/β_1^2.

We start from $|\pi_0\rangle$. By Lemma 2.21, we have $|\pi_0\rangle = |\pi_{0,\mathsf{good}}\rangle + |e_0\rangle$, where $|\pi_{0,\mathsf{good}}\rangle$ lies in the subspace spanned by the eigenvectors $|\psi_j\rangle$ of W_1' with corresponding eigenvalue λ_j of P_1 such that $\lambda_j = 0$ or $\lambda_j \leq 1 - \Omega(1/t_0')$. Let $e^{2\pi i \xi_j}$ be the eigenvalue of $|\psi_j\rangle$ of W_1'. By Fact 2.16 and Fact 2.18, we get that $\xi_j = 0$ or $\xi_j \geq \Omega({t_0'}^{-1/2})$. By Lemma 2.21, we also have $\| |e_0\rangle \| \leq \epsilon_1$.

Then, by Lemma 2.23 with $\Delta = \Omega({t_0'}^{-1/2})$ and $\epsilon = \epsilon_1^2$, we can implement $\widetilde{R}_1$ such that $\| R_1 |\phi\rangle - \widetilde{R}_1 |\phi\rangle \|_2 \leq 2\epsilon_1$ using $O(\sqrt{t_0'} \log(1/\epsilon_1))$ calls to controlled-W_1', where $|\phi\rangle$ is any state that occurs during $\pi/3$-amplitude amplification (Lemma 2.24) for $|\pi_0\rangle$ towards $|\pi_1\rangle$.

In the same way, we can start from $|\pi_1\rangle$ and show that $\widetilde{R}_0$ can be implemented using $O(\sqrt{t_1'} \log(1/\epsilon_1))$ calls to controlled-W_0' such that $\| R_0 |\phi'\rangle - \widetilde{R}_0 |\phi'\rangle \| \leq 2\epsilon_1$, where $|\phi'\rangle$ is any state that occurs during $\pi/3$-amplitude amplification for $|\pi_1\rangle$ towards $|\pi_0\rangle$.

Suppose we can implement R_0 and R_1 perfectly. Then, we can prepare a state $|\widetilde{\pi}_1\rangle$ such that $|\langle \widetilde{\pi}_1 | \pi_1 \rangle| \geq 1 - (1-p)^{3^m}$ using 3^m calls to R_0, R_1 and their inverses, by applying $\pi/3$-amplitude amplification (Lemma 2.24) to $|\pi_i\rangle$. Thus, by taking $m = O(p^{-1} \log(1/\epsilon_2))$ where ϵ_2 is a parameter to be chosen later, we have $\| |\pi_1\rangle - |\widetilde{\pi}_1\rangle \|_2 \leq \epsilon_2$. However, since each call to $\widetilde{R}_0$ or $\widetilde{R}_1$ causes an error of ϵ_1, the total error will be

$$O(\epsilon_2 + \epsilon_1 \cdot p^{-1} \log(1/\epsilon_2)) = \epsilon,$$

where we take $\epsilon_1 := O(p\epsilon \log^{-1}(1/\epsilon))$ and $\epsilon_2 := \epsilon_1^2$.

Therefore, the total number of calls to controlled-W_0', controlled-W_1' is

$$O\Big((\sqrt{t_0'} + \sqrt{t_1'}) \cdot p^{-1} \log^2(1/p\epsilon)\Big),$$

where $t_i' = t_i(\epsilon_1/\beta_i^2) = O(t_i(\epsilon) \cdot \log(\beta_i/p))$.

The theorem is then proved. $\square$

The following corollary is an immediate consequence of Theorem 2.25, and it also gives Theorem 3.5.

Corollary 2.26 (Quantum walk implementation cost (ℓ_2-warm starts))**.** *Let M_0, M_1 be two ergodic reversible Markov chains with stationary distributions π_0, π_1, respectively. Suppose π_0 mixes towards π_1 in M_1 up to total variation distance ϵ in $t_0(\epsilon)$ steps. Similarly, suppose π_1 mixes towards π_0 in M_0 in $t_1(\epsilon)$ steps. Suppose $\|\pi_0/\pi_1\| = O(1)$ and $\|\pi_1/\pi_0\| = O(1)$. Moreover, we assume that $|\langle \pi_0|\pi_1\rangle| = \Omega(1)$.*

Given $|\pi_0\rangle$, we can obtain a state $|\widetilde{\pi}_1\rangle$ such that $\||\widetilde{\pi}_1\rangle - |\pi_1\rangle\|_2 \le \epsilon$ using

$$O\Big(\sqrt{t_0(\epsilon) + t_1(\epsilon)} \log^2(1/\epsilon)\Big)$$

calls to the controlled walk operators controlled-W_0', controlled-W_1'.

2.9 Quantum Algorithm for Log-Concave Sampling: Details

In this section, we provide several quantum algorithms for sampling log-concave distributions. In Section 2.9.1, we show that the classical underdamped Langevin diffusion (ULD) and the randomized midpoint method for underdamped Langevin diffusion (ULD-RMM) can be improved by replacing the first-order oracle by the zeroth-order quantum oracle, while achieving the same efficiency and accuracy guarantees. In Section 2.9.2, we show that the Metropolis adjusted Langevin algorithm (MALA) can be quantumly sped up in terms of query complexity, for both Gaussian initial distributions and warm-start distributions.

2.9.1 Quantum inexact ULD and ULD-RMM

In the quantum setting, we can estimate $\nabla f(x)$ by using Jordan's algorithm with queries to the quantum zeroth-order evaluation oracle (2.4). The following lemma provides an ℓ_1-error guarantee.

Lemma 2.27 (Lemma 2.3 in [CCLW20]). *Let f be a convex, L_0-Lipschitz continuous function that is specified by an evaluation oracle with error at most ϵ. Suppose f is L-smooth in $B_\infty(x, 2\sqrt{\epsilon/L})$. Let*

$$\widetilde{g} = \textsc{SmoothQuantumGradient}(f, \epsilon, L_0, L, x). \tag{2.75}$$

Then for any $i \in [d]$, we have $|\widetilde{g}_i| \leq L_0$ and $\mathbb{E}|\widetilde{g}_i - \nabla f(x)_i| \leq 3000\sqrt{d\epsilon L}$; hence

$$\mathbb{E}\|\widetilde{g} - \nabla f(x)\|_1 \leq 3000 d^{1.5}\sqrt{\epsilon L}. \tag{2.76}$$

If L_0, $1/L$, and $1/\epsilon$ are $\mathrm{poly}(d)$, *the* SmoothQuantumGradient *algorithm uses $O(1)$ queries to the quantum evaluation oracle and $\widetilde{O}(d)$ gates.*

We then introduce inexact ULD and ULD-RMM by using a stochastic zeroth-order oracle as follows.

Lemma 2.28 (Theorem 2.2 of [RSBG19]). *Let ρ_n be the distribution of the underdamped Langevin diffusion with the initial point x_0 satisfying $\|x_0 - x^*\| \leq D$, step size $h \leq \frac{\epsilon}{104\kappa}\sqrt{\frac{1}{d/\mu + D^2}}$, and time $T \geq \frac{\kappa}{2}\log\Bigl(\frac{24\sqrt{d/\mu + D^2}}{\epsilon}\Bigr)$. Assume there is a stochastic zeroth-order oracle that provides an unbiased evaluation of $\nabla f(x)$ with bounded variance $\mathbb{E}\|\widetilde{g} - \nabla f(x)\|^2 \leq \sigma^2$. Then inexact ULD achieves $W_2(\rho_n, \rho) \leq \epsilon$ using*

$$\frac{T}{h} = \widetilde{\Theta}\Bigl(\frac{\kappa^2\sqrt{d}}{\epsilon}\Bigr) \tag{2.77}$$

iterations and

$$b = \frac{d^{1.5}\max\{1, \sigma^2\}}{\epsilon} \tag{2.78}$$

queries to the zeroth-order oracle per iteration. The total number of calls is $\frac{bT}{h}$.

Lemma 2.29 (Theorem 2.3 of [RSBG19]). *Let* ρ_n *be the distribution of the randomized midpoint method for underdamped Langevin diffusion with the initial point* x_0*, step size* $h \le \min\Big\{\frac{\epsilon^{1/3}\mu^{1/6}}{\kappa^{1/6}d^{1/6}\log^{1/6}\big(\frac{\sqrt{d/\mu}}{\epsilon}\big)}, \frac{\epsilon^{2/3}\mu^{1/3}}{d^{1/3}\log^{1/3}\big(\frac{\sqrt{d/\mu}}{\epsilon}\big)}\Big\}$*, and time* $T \ge 2\kappa\log\big(\frac{20d/\mu}{\epsilon^2}\big)$*. Assume there is a stochastic zeroth-order oracle that provides an unbiased evaluation of* $\nabla f(x)$ *with bounded variance* $\mathbb{E}\|\widetilde{g} - \nabla f(x)\|^2 \le \sigma^2$*. Then inexact ULD-RMM achieves* $W_2(\rho_n, \rho) \le \epsilon$ *using*

$$\frac{2T}{h} = \widetilde{\Theta}\Big(\frac{\kappa^{7/6}d^{1/6}}{\epsilon^{1/3}} + \frac{\kappa d^{1/3}}{\epsilon^{2/3}}\Big) \tag{2.79}$$

iterations and

$$b = \frac{d\kappa}{h^3} \tag{2.80}$$

queries to the zeroth-order oracle per iteration. The total number of calls is $\frac{bT}{h}$.

As a quantum counterpart, we are able to reduce the number of queries from $O(b)$ to $O(1)$ for each iteration in Lemma 2.28 and Lemma 2.29 based on Lemma 2.27. Here we are able to choose $\epsilon = O(\frac{\sigma^2}{d^3L})$ to preserve the condition

$$\mathbb{E}\|\widetilde{g} - \nabla f(x)\|^2 \le \mathbb{E}\|\widetilde{g} - \nabla f(x)\|_1^2 \le \sigma^2 \tag{2.81}$$

used in Lemma 2.28 and Lemma 2.29 with $O(1)$ additional quantum queries. The total number of calls is $O(\frac{T}{h})$ in Lemma 2.28 and Lemma 2.29, the same scaling as in Lemma 2.10 and Lemma 2.11. The query complexities of ULD and ULD-RMM are as follows.

Theorem 2.30. *Assume the target distribution* ρ *is strongly log-concave with* L*-smooth and* μ*-strongly convex negative log-density. Let* ρ_n *be the distribution of the* *underdamped Langevin diffusion with the initial point x_0 satisfying $\|x_0 - x^*\| \leq D$, step size $h \leq \frac{\epsilon}{104\kappa}\sqrt{\frac{1}{d/\mu + D^2}}$, and time $T \geq \frac{\kappa}{2}\log\Big(\frac{24\sqrt{d/\mu + D^2}}{\epsilon}\Big)$. Then quantum inexact ULD (Algorithm 6) achieves*

$$\mathbb{E}\Big(\|\widehat{X}_n - X_T\|^2\Big) \leq \widetilde{O}\Big(\frac{d^2\kappa^2h^2}{\mu}\Big), \tag{2.82}$$

$$W_2(\rho_n, \rho) \leq \epsilon, \tag{2.83}$$

using

$$\frac{T}{h} = \widetilde{\Theta}\Big(\frac{\kappa^2\sqrt{d}}{\epsilon}\Big) \tag{2.84}$$

queries to the quantum evaluation oracle.

Proof. By Lemma 2.28, we know that the number of iterations of ULD with an inexact gradient oracle is $\widetilde{O}(\kappa^2\sqrt{d}/\epsilon)$, as long as the oracle satisfies $\mathbb{E}\|\widetilde{g} - \nabla f(x)\|^2 \leq \sigma^2$. By Lemma 2.27, this condition can be achieved by the quantum gradient algorithm such that each gradient computation takes $O(1)$ queries to the quantum evaluation oracle. Therefore, the total number of queries is $\widetilde{O}(\kappa^2\sqrt{d}/\epsilon)$ for the quantum inexact ULD. $\square$

Theorem 2.31. *Assume the target distribution ρ is strongly log-concave with L-smooth and μ-strongly convex negative log-density. Let ρ_n be the distribution of the randomized midpoint method for underdamped Langevin diffusion with initial point x_0, step size $h \leq \min\Big\{\frac{\epsilon^{1/3}\mu^{1/6}}{\kappa^{1/6}d^{1/6}\log^{1/6}\big(\frac{\sqrt{d/\mu}}{\epsilon}\big)}, \frac{\epsilon^{2/3}\mu^{1/3}}{d^{1/3}\log^{1/3}\big(\frac{\sqrt{d/\mu}}{\epsilon}\big)}\Big\}$, and time $T \geq 2\kappa\log\Big(\frac{20d/\mu}{\epsilon^2}\Big)$. Then quantum inexact ULD-RMM (Algorithm 7) achieves*

$$\mathbb{E}\Big(\|\widehat{X}_n - X_T\|^2\Big) \leq \widetilde{O}\Big(\frac{d\kappa h^6}{\mu} + \frac{dh^3}{\mu}\Big), \tag{2.85}$$

$$W_2(\rho_n, \rho) \leq \epsilon, \tag{2.86}$$

using

$$\frac{2T}{h} = \widetilde{\Theta}\Big(\frac{\kappa^{7/6}d^{1/6}}{\epsilon^{1/3}} + \frac{\kappa d^{1/3}}{\epsilon^{2/3}}\Big) \tag{2.87}$$

queries to the quantum evaluation oracle.

The proof is almost the same as Theorem 2.30, so we omit it here.

Algorithm 6 Quantum Inexact Underdamped Langevin Dynamics (Quantum IULD)

```
1: procedure QuantumIULD($f$, $h$, $T$, $x_0$, $\epsilon$, $L$, $\beta$)    ▷ Function $f$, step size $h$, time $T$, and a sample $x_0$ from a starting distribution $\rho_0$, evaluation error $\epsilon$, Lipschitz constant $L$, smoothness parameter $\beta$
2:     $x_0^h \leftarrow x_0$
3:     $\widetilde{g}(x_0) \leftarrow$ SmoothQuantumGradient$(f, \epsilon, L, \beta, x_0)$
4:     for $t = 0, h, \ldots, \lfloor T \rfloor$ do
5:         Draw $W_{1,t}^h = \int_0^h e^{2(s-h)} \mathrm{d}B_{t+s}$, $W_{2,t}^h = \int_0^h (1 - e^{2(s-h)}) \mathrm{d}B_{t+s}$
6:         $v_{t+h}^h \leftarrow e^{-2h} v_t^h + \frac{1}{2L}(1 - e^{-2h}) \widetilde{g}(x_t^h) + \frac{2}{\sqrt{L}} W_{1,t}^h$
7:         $x_{t+h}^h \leftarrow x_t^h + \frac{1}{2}(1 - e^{-2h}) v_t^h + \frac{1}{2L}[h - (1 - e^{-2h})] \widetilde{g}(x_t^h) + \frac{1}{\sqrt{L}} W_{2,t}^h$
8:         $\widetilde{g}(x_{t+h}^h) \leftarrow$ SmoothQuantumGradient$(f, \epsilon, L, \beta, x_{t+h}^h)$
9:     end for
10:    return $x_h^h, x_{2h}^h, \ldots, x_{\lfloor T \rfloor + 1}^h$
11: end procedure
```

2.9.2 Quantum MALA

In Section 2.9.2.1, we introduce several classical results on the mixing of MALA. Then, in Section 2.9.2.2, we describe how to implement a quantum walk for MALA. Then, in Section 2.9.2.3 and Section 2.9.2.4, we show quantum MALA with a warm start distribution and a Gaussian initial distribution, respectively.

2.9.2.1 Mixing time and spectral gap of MALA

The Metropolis adjusted Langevin algorithm (MALA) is a key method for sampling log-concave distributions. Classically, the state-of-the-art mixing time bound of MALA was proven by [LST20]. They show that MALA is equivalent to the Metropolized Hamiltonian Monte Carlo method (HMC) and use the blocking conductance analysis of [KLM06] to upper bound the mixing time.

Define $\mathcal{H}(x, v) := f(x) + \frac{1}{2}\|v\|_2^2$. Let $\mathrm{d}\pi^\star$ denote the target distribution, i.e., $\mathrm{d}\pi^\star(x)/\mathrm{d}x \propto \exp(-f(x))$. Then the Markov chain defined by Algorithm 8 has the following property.

Lemma 2.32 ([LST20]). *The Markov chain of Algorithm 8 is reversible, and its*

Algorithm 7 Quantum Inexact Underdamped Langevin Dynamics with Randomized Midpoint Method (Quantum IULD-RMM)

```
1: procedure QuantumIULD-RMM($f$, $h$, $T$, $x_0$) ▷ Function $f$, step size $h$, time $T$, and a sample $x_0$ from a starting distribution $\rho_0$
2:     $x_0^h \leftarrow x_0$, $y_0^h \leftarrow x_0$
3:     $\widetilde{g}(x_0^h) \leftarrow$ SmoothQuantumGradient$(f, \epsilon, L, \beta, x_0^h)$
4:     $\widetilde{g}(y_0^h) \leftarrow$ SmoothQuantumGradient$(f, \epsilon, L, \beta, y_0^h)$
5:     for $t = 0, h, \ldots, \lfloor T \rfloor$ do
6:         Draw $W_{1,t}^h = \int_0^h e^{2(s-h)} \mathrm{d}B_{t+s}$, $W_{2,t}^h = \int_0^h (1 - e^{2(s-h)}) \mathrm{d}B_{t+s}$, $W_{3,t}^h = \int_0^{\alpha h} (1 - e^{2(s-h)}) \mathrm{d}B_{t+s}$
7:         $v_{t+h}^h \leftarrow e^{-2h} v_t^h + \frac{h}{L} e^{-2(1-\alpha)h} \widetilde{g}(y_t^h) + \frac{2}{\sqrt{L}} W_{1,t}^h$
8:         $x_{t+h}^h \leftarrow x_t^h + \frac{1}{2}(1 - e^{-2h}) v_t^h + \frac{h}{2L}(1 - e^{-2(1-\alpha)h}) \widetilde{g}(y_t^h) + \frac{1}{\sqrt{L}} W_{2,t}^h$
9:         $y_{t+h}^h \leftarrow x_t^h + \frac{1}{2}(1 - e^{-2\alpha h}) v_t^h + \frac{1}{2L}[\alpha h - (1 - e^{-2\alpha h})] \widetilde{g}(x_t^h) + \frac{1}{\sqrt{L}} W_{3,t}^h$
10:        $\widetilde{g}(x_{t+h}^h) \leftarrow$ SmoothQuantumGradient$(f, \epsilon, L, \beta, x_{t+h}^h)$
11:        $\widetilde{g}(y_{t+h}^h) \leftarrow$ SmoothQuantumGradient$(f, \epsilon, L, \beta, y_{t+h}^h)$
12:    end for
13:    return $x_h^h, x_{2h}^h, \ldots, x_{\lfloor T \rfloor + 1}^h$
14: end procedure
```

stationary distribution is $\mathrm{d}\pi^\star$.

The main result of [LST20] is the following theorem on the mixing time of Algorithm 8.

Theorem 2.33 (Mixing of Hamiltonian Monte Carlo, Theorem 4.7 of [LST20])**.** *There is an algorithm initialized from a point drawn from* $\mathcal{N}(x^\star, L^{-1} I_d)$ *that iterates Algorithm 8*

$$O(\kappa d \log(\kappa/\epsilon) \log(d \log(\kappa/\epsilon)) \log(1/\epsilon)) \tag{2.88}$$

times and produces a point from a distribution ρ *such that* $\|\rho - \pi^\star\|_{TV} \le \epsilon$.

The algorithm in the above theorem defines a new Markov chain where in each step, we draw an integer j uniformly from 0 to $O(\kappa d \log(\kappa/\epsilon) \log(d \log(\kappa/\epsilon)))$ and run Algorithm 8 for j iterations. One step of this Markov chain gives a distribution with

Algorithm 8 Metropolis adjusted Langevin algorithm (MALA)

```
1: procedure MALA($x_0$, $h$)                 ▷ Initial point $x_0 \in \mathbb{R}^d$, step size $h$
2:     for $k \geq 0$ do
3:         Draw $z_{k+1} \sim \mathcal{N}(0, I_d)$
4:         $z_{k+1} \leftarrow x_k - h\nabla f(x_k) + \sqrt{2h} z_{k+1}$
5:         Compute
               $\alpha_k := \min\left\{1, \frac{\exp(-f(z_{k+1}) - \|x_k - z_{k+1} + h\nabla f(z_{k+1})\|_2^2/(4h))}{\exp(-f(x_k) - \|x_k - z_{k+1} + h\nabla f(x_k)\|_2^2/(4h))}\right\}$
6:         Draw $u \sim \mathcal{U}([0,1])$
7:         if $u \leq \alpha_k$ then
8:             $x_{k+1} \leftarrow z_{k+1}$
9:         else
10:            $x_{k+1} \leftarrow x_k$
11:        end if
12:    end for
13:    return $\{x_k\}_{k\geq 0}$
14: end procedure
```

TV-distance from $\pi^\star$ at most $(2e)^{-1}$ [LST20]. Hence, if we run for $\log(1/\epsilon)$ steps, we get ϵ TV-distance.

Furthermore, we can show that MALA converges faster under a certain warm start condition [DCWY18, WSC22, CLA+21]. We say the initial distribution ρ_0 is *β-warm* if there is a constant β independent of κ, d such that

$$\sup_{S \in \mathcal{B}(\mathbb{R}^d)} \frac{\rho_n(S)}{\rho(S)} \leq \beta. \tag{2.89}$$

The warmness of the Gaussian $\rho_0 = \mathcal{N}(x^*, \frac{1}{L}I)$ satisfies $\beta \leq \kappa^{d/2}$ [DCWY18, WSC22].

Given a β-warm initial distribution, MALA has the following improved convergence.

Lemma 2.34 (Theorem 1 of [WSC22]). *Assume the target distribution ρ is strongly log-concave with L-smooth and μ-strongly convex negative log-density. Let ρ_n be the distribution of the $\frac{1}{2}$-lazy version of MALA with β-warm initial distribution ρ_0*

and step size $h = c_0(Ld\log^2(\max\{\kappa, d, \frac{\beta}{\epsilon}, c_2\}))^{-1}$*. There exist universal constants* $c_0, c_1, c_2 > 0$*, such that MALA achieves*

$$d_{TV}(\rho_n, \rho) \leq \epsilon \tag{2.90}$$

after

$$n \geq c_1\kappa\sqrt{d}\log^3(\max\{\kappa, d, \frac{\beta}{\epsilon}, c_2\}) \tag{2.91}$$

steps.

2.9.2.2 Quantum walks for MALA

The goal of this section is to show quantum speedup for the Metropolis adjusted Langevin algorithm (MALA) using the continuous-space quantum walks defined by [CCH+19], which generalize the discrete-time quantum walk of [Sze04] to continuous space.

Section 2.8 contains a detailed introduction of the quantum walk. To implement a quantum version of Algorithm 8, we prepare the initial state $|\pi_0\rangle$ and implement the quantum walk operator W.

Initial state. For the initial state $|\rho_0\rangle$, by Theorem 2.33, it suffices to take $\rho_0 = \mathcal{N}(x^\star, L^{-1}I_d)$, where $x^\star$ is the minimum point of $f(x)$. Suppose we already have $x^\star$. Appendix A.3 of [CCH+19] shows that the state

$$\int_{\mathbb{R}^d} \left(\frac{L}{2\pi}\right)^{d/4} e^{-\frac{L}{4}\|z\|_2^2}|z\rangle \mathrm{d}z \tag{2.92}$$

can be efficiently prepared by applying a Box-Muller transformation to the state corresponding to the uniform distribution (i.e., an equal superposition of points). Then, for the ith register, we apply the shift operation U_{shift} with $U_{\text{shift}}|x_i\rangle = |x_i + x_i^\star\rangle$. The resulting state is

$$|\rho_0\rangle = \int_{\mathbb{R}^d} \left(\frac{L}{2\pi}\right)^{d/4} e^{-\frac{L}{4}\|z-x^\star\|_2^2}|z\rangle \mathrm{d}z. \tag{2.93}$$

Quantum walk operator. The quantum walk operator $W(P)$ can be implemented[1] using the quantum walk update unitary U that maps each point $|x\rangle$ to the superposition $\int_{\mathbb{R}^d} \mathrm{d}y \sqrt{p_{x\to y}}|y\rangle$. We show how to efficiently implement U.

We first prepare a standard Gaussian state

$$|\xi\rangle := \int_{\mathbb{R}^d} \left(\frac{1}{2\pi}\right)^{d/4} e^{-\frac{1}{4}\|z\|_2^2}|z\rangle \mathrm{d}z. \tag{2.94}$$

Then, we apply an affine transformation to change the distribution: for any $z \in \mathbb{R}^d$,

$$|x\rangle|z\rangle \mapsto |x\rangle\Big|x - h\nabla f(x) + \sqrt{2h}z\Big\rangle.$$

This step will query the gradient oracle $\mathcal{O}_{\nabla f}$. Then, we apply a controlled rotation to the third register based on the following accept/reject probability:

$$\alpha(x,z) := \min\left\{1, \frac{\exp(-f(z) - \|x - z + h\nabla f(z)\|_2^2/(4h))}{\exp(-f(x) - \|x - z + h\nabla f(x)\|_2^2/(4h))}\right\} \quad \forall x, z \in \mathbb{R}^d.$$

That is, based on the values x and z in the first and second registers, we have

$$|x\rangle|z\rangle|0\rangle \mapsto |x\rangle|z\rangle\Big(\sqrt{\alpha(x,z)}|0\rangle + \sqrt{1-\alpha(x,z)}|1\rangle\Big).$$

It can be implemented by querying the evaluation oracle $\mathcal{O}_f$ and the gradient oracle $\mathcal{O}_{\nabla f}$. After that, we post-select the third register to be zero. Note that the cost of the post-selection depends on the acceptance probability of the classical MALA. It was shown in [DCWY18, Section 4.4] that by choosing a proper step size, the acceptance probability will be a constant. Therefore, the cost of the post-selection is $O(1)$.

Overall, the resulting state will be the desired quantum walk state:

$$|x\rangle|\psi_x\rangle := |x\rangle \int_{\mathbb{R}^d} \sqrt{p_{x\to y}}|y\rangle dy.$$

It is equivalent to the classical MALA with one acceptance step. And this process will query the evaluation oracle and gradient oracle for $O(1)$ times. We note that an alternative implementation of the quantum walk operator for the Metropolis-Hastings type Markov chains is shown in [LHP+20].

[1]As shown in [WA08], $W(P) = U^\dagger SURU^\dagger SUR$, where S is the swap gate and R is a reflection operator with respect to the state space $\mathrm{span}\{|x\rangle|0\rangle : x \in \mathbb{R}^d\}$.

Algorithm 9 Quantum Update Unitary

```
1: procedure QuantumUpdate($|\rho\rangle|0\rangle$)    ▷ $|\rho\rangle = \int_{\mathbb{R}^d} \mathrm{d}x\sqrt{\rho_x}|x\rangle$
2:     Prepare $|\rho\rangle|\xi\rangle$ where $|\xi\rangle$ is a $d$-dimensional Gaussian state
3:     Apply an affine transformation on the second register:    ▷ Query $\mathcal{O}_{\nabla f}$
           $$|x\rangle|z\rangle \mapsto |x\rangle\Big|x - h\nabla f(x) + \sqrt{2h}z\Big\rangle$$
4:     Apply a controlled rotation:    ▷ Query $\mathcal{O}_f$ and $\mathcal{O}_\nabla f$
           $$|x\rangle|z\rangle|0\rangle \mapsto |x\rangle|z\rangle\Big(\sqrt{\alpha(x,z)}|0\rangle + \sqrt{1-\alpha(x,z)}|1\rangle\Big),$$
   where $\alpha(x,z) := \min\left\{1, \frac{\exp(-f(z)-\|x-z+h\nabla f(z)\|_2^2/(4h))}{\exp(-f(x)-\|x-z+h\nabla f(x)\|_2^2/(4h))}\right\}$
5:     Post-select the third register being zero and drop the register
6:     return the final state $|\phi\rangle$    ▷ $|\phi\rangle = \int_{\mathbb{R}^d} \mathrm{d}x\sqrt{\rho_x}|x\rangle \int_{\mathbb{R}^d} \mathrm{d}y\sqrt{p_{x\to y}}|y\rangle$
7: end procedure
```

Algorithm 10 Quantum MALA

```
1: procedure QuantumMALA($\mathcal{O}_f$, $\mathcal{O}_{\nabla f}$, $|\rho_0\rangle$)    ▷ Evaluation oracle $\mathcal{O}_f$, gradient
   oracle $\mathcal{O}_{\nabla f}$, initial state $|\rho_0\rangle$
2:     Construct quantum walk update unitary $U$ from QuantumUpdate with $\mathcal{O}_f$
   and $\mathcal{O}_{\nabla f}$    ▷ Algorithm 9
3:     Implement the quantum walk operator $W(P)$
4:     Perform $\frac{\pi}{3}$-amplitude amplification with $W(P)$ on the state $|\rho_0\rangle|0\rangle$
5:     return the resulting state $|\widetilde{\rho}\rangle$    ▷ $|\widetilde{\rho}\rangle \approx \int_{\mathbb{R}^d} e^{-f(x)}\mathrm{d}|x\rangle$, the stationary
   distribution state
6: end procedure
```

Lemma 2.35 (Continuous-space quantum walk implementation)**.** *The Markov chain of Algorithm 8 can be implemented as a continuous-space quantum walk where the quantum walk unitary for one step can be implemented with $O(1)$ queries to the gradient oracle and the evaluation oracle, and $O(d)$ quantum gates.*

2.9.2.3 Quantum MALA with a warm start

We first show that quantum MALA can achieve quadratic speedup in query complexity when the initial distribution is a warm start (Theorem 2.36).

Theorem 2.36 (Quantum MALA with warm start)**.** *Let $|\rho_0\rangle$ be a β-warm start with respect to the log-concave distribution $\rho \propto e^{-f}$. Let $t(\epsilon) = \tilde{O}(\kappa\sqrt{d}\log^3(\beta/\epsilon))$ be the mixing time of classical MALA with initial distribution ρ_0 as shown in Lemma 2.34, i.e., $\|P^{t(\epsilon)}\rho_0 - \rho\|_{\mathrm{TV}} \le \epsilon$. Then there is a quantum algorithm that prepares a state $|\widetilde{\rho}\rangle$ that is ϵ-close to $|\rho\rangle$ using*

$$\widetilde{O}(\kappa^{1/2}d^{1/4}\beta^{1/2}) \tag{2.95}$$

queries to the evaluation oracle $\mathcal{O}_f$ and gradient oracle $\mathcal{O}_{\nabla f}$.

Proof. Let $|\rho_0\rangle$ be the initial state corresponding to a distribution ρ_0 that is β-warm. Then we know that $t = t(\epsilon/\beta)$ steps of the classical MALA random walk suffice to achieve $\|P^t\rho_0 - \rho\|_1 \le \epsilon/\beta$.

By Lemma 2.21, we have $|\rho_0\rangle = |\rho_0'\rangle + |e\rangle$, where $|\rho_0'\rangle$ lies in the space of eigenvectors $|v_i\rangle$ of W such that $\lambda_i = 1$ or $\lambda_i \le 1 - \widetilde{\Omega}(t^{-1})$, and $\||e\rangle\| \le \epsilon_1$.

Hence, by Corollary 4.1 in [CCLW20], the approximate reflection in the quantum walk $\widetilde{R}$ can be implemented using $\widetilde{O}(t^{1/2})$ calls to the controlled-W operator. Furthermore, β-warmness also implise that $|\langle\rho_0|\rho\rangle| \ge \beta^{-1/2}$. Thus, the approximated stationary state $|\widetilde{\rho}\rangle$ can be prepared via $O(\beta^{1/2}\log(1/\epsilon_2))$ recursive levels of $\frac{\pi}{3}$-amplitude amplification such that $\||\rho\rangle - |\widetilde{\rho}\rangle\| \le \epsilon_2$.

By choosing $\epsilon_1 = \epsilon/(2\log(2/\epsilon))$ and $\epsilon_2 = \epsilon/2$, we achieve a final approximation error of $O(\epsilon_1 \log(\epsilon_2) + \epsilon_2) \leq \epsilon$. By Lemma 2.35, each controlled-W operator takes a constant number of queries to $\mathcal{O}_f$ and $\mathcal{O}_{\nabla f}$. Therefore, by plugging-in the classical mixing time of MALA, we obtain the desired query complexity. $\square$

Algorithm 11 Quantum MALA with Warm Start

```
Input: Evaluation oracle $\mathcal{O}_f$, gradient oracle $\mathcal{O}_{\nabla f}$, smoothness parameter $L$, convexity parameter $\mu$, warm-start state $|\rho_0\rangle$
Output: Quantum state $|\widetilde{\rho}\rangle$ close to the stationary distribution state $\int_{\mathbb{R}^d} e^{-f(x)} \mathrm{d}|x\rangle$
1: $|\widetilde{\rho}\rangle \leftarrow$ QuantumMALA($\mathcal{O}_{f_i}, \mathcal{O}_{\nabla f_i}, |\rho_0\rangle$)    ▷ Algorithm 10
2: return $|\widetilde{\rho}\rangle$
```

2.9.2.4 Quantum MALA with the Gaussian-start

We cannot directly apply Theorem 2.36 for a Gaussian initial distribution because the overlap between $|\rho_0\rangle$ and $|\rho\rangle$ is exponentially small. Instead, we use the idea of simulated annealing and construct a sequence of slowly-varying Markov-chains (as in [WA08]). We have the following result, which looks like Theorem 2.19. But our result uses the effective spectral gap of MALA (by Lemma 2.21).

Corollary 2.37 (Quantum speedup for slowly varying MALAs)**.** *Let $\rho_0, \ldots, \rho_r$ be a sequence of log-concave distributions such that $|\langle \rho_i | \rho_{i+1} \rangle| \geq p$ for some $p > 0$ and for all $i \in \{0, \ldots, r-1\}$. Suppose we can prepare the initial state $|\rho_0\rangle$. Then, for any $\epsilon > 0$, there is a quantum procedure to produce a state $|\widetilde{\rho}_r\rangle$ such that $\||\widetilde{\rho}_r\rangle - |\rho_r\rangle\| \leq \epsilon$ using*

$$\widetilde{O}(\kappa^{1/2} d^{1/2} \cdot (r/p)) \tag{2.96}$$

applications of the quantum walk operators W_i corresponding to the MALA procedure for ρ_i for $i \in [r]$.

Proof. For two consecutive distributions, quantum MALA can evolve $|\rho_i\rangle$ to $|\rho_{i+1}\rangle$ using $\tilde{O}(\sqrt{\kappa d} \cdot p^{-1})$ quereis, which follows from Theorem 2.36 and Theorem 2.33

(which holds for all initial distributions with warmness $\beta \le \kappa^{d/2}$). Then, the corollary immediately follows. $\square$

Algorithm 12 Quantum MALA for Log-concave Sampling

```
Input: Evaluation oracle $\mathcal{O}_f$, gradient oracle $\mathcal{O}_{\nabla f}$, smoothness parameter $L$, convexity parameter $\mu$
Output: Quantum state $|\widetilde{\rho}\rangle$ close to the stationary distribution state $\int_{\mathbb{R}^d} e^{-f(x)} \mathrm{d}|x\rangle$
1: Compute the cooling schedule parameters $\sigma_1, \ldots, \sigma_M$
2: Prepare the state $|\rho_0\rangle \propto \int_{\mathbb{R}^d} e^{-\frac{1}{4}\|x\|^2/\sigma_1^2} \mathrm{d}|x\rangle$
3: for $i \leftarrow 1, \ldots, M$ do
4:     Construct $\mathcal{O}_{f_i}$ and $\mathcal{O}_{\nabla f_i}$ where $f_i(x) = f(x) + \frac{1}{2}\|x\|^2/\sigma_i^2$
5:     $|\rho_i\rangle \leftarrow$ QuantumMALA($\mathcal{O}_{f_i}, \mathcal{O}_{\nabla f_i}, |\rho_{i-1}\rangle$)        ▷ Algorithm 10
6: end for
7: return $|\rho_M\rangle$
```

Theorem 2.38 (Quantum MALA for log-concave sampling)**.** *Assume the target distribution $\rho \propto e^{-f}$ is strongly log-concave with $f\colon \mathbb{R}^d \to \mathbb{R}_+$ being L-smooth and μ-strongly convex. Let $|\rho\rangle$ be the quantum state corresponding to the distribution ρ. Then, for any $\epsilon > 0$, there is a quantum algorithm (Algorithm 12) that prepares a state $|\widetilde{\rho}\rangle$ such that $\||\widetilde{\rho}\rangle - |\rho\rangle\| \le \epsilon$ using $\widetilde{O}(\kappa^{1/2} d)$ queries to the evaluation oracle $\mathcal{O}_f$ and gradient oracle $\mathcal{O}_{\nabla f}$.*

Proof. By Lemma 2.15, we know that the cooling schedule $\sigma_1, \ldots, \sigma_M$ gives a sequence of slowly-varying Markov chains with overlap $p = \Omega(1)$. We also know that the length of the schedule is $M = \tilde{O}(d^{1/2})$.

Hence, by Corollary 2.37 with $r = \tilde{O}(d^{1/2})$ and $p = \Omega(1)$, $|\widetilde{\rho}\rangle$ can be prepared by $\tilde{O}(\kappa^{1/2} d)$ quantum walk steps. By Lemma 2.35, each step queries $\mathcal{O}_f$ and $\mathcal{O}_{\nabla f}$ twice. The result follows. $\square$

2.10 Quantum Algorithm for Estimating Normalizing Constants: Details

We now come back to the problem of estimating the normalizing constant.

2.10.1 Quantum MALA and annealing

In this section, we first describe a quantum speedup for the annealing process via a quantum-accelerated Monte Carlo method, which quadratically improves the ϵ-dependence of the sampling complexity of the classical algorithm. Then we further reduce the κ- and d-dependence of the query complexity using the quantum MALA procedure developed in Section 2.9.2.

2.10.1.1 Quantum speedup for the standard annealing process

Reference [Mon15] developed a quantum-accelerated Monte Carlo method for mean estimation with B-bounded relative variance.[2] We state the result as follows.

Lemma 2.39 (Theorem 6 of [Mon15])**.** *Assume there is an algorithm $\mathcal{A}$ such that $v(\mathcal{A}) \geq 0$ and $\frac{\mathrm{Var}(v(\mathcal{A}))}{\mathbb{E}[v(\mathcal{A})]^2} \leq B$ for some $B \geq 1$, and an accuracy $\epsilon < 27B/4$. Then there is a quantum algorithm which outputs an estimate $\widetilde{\mu}$ such that*

$$\Pr\Big[|\widetilde{\mu} - \mathbb{E}[v(\mathcal{A})]| \geq \epsilon\mathbb{E}[v(\mathcal{A})]\Big] \leq \frac{1}{4}, \tag{2.97}$$

with

$$O\Big(\frac{B}{\epsilon}\log^{3/2}\Big(\frac{B}{\epsilon}\Big)\log\log\Big(\frac{B}{\epsilon}\Big)\Big) \tag{2.98}$$

queries to $\mathcal{A}$.

Lemma 2.40 (Lemma 1 of [Mon15])**.** *Let $\mathcal{A}$ be a (classical or quantum) algorithm which aims to estimate some quantity μ, and whose output $\tilde{\mu}$ satisfies $|\mu - \tilde{\mu}| \leq \epsilon$ except with probability γ, for some fixed $\gamma < 1/2$. Then, for any $\delta > 0$, it suffices to repeat $\mathcal{A}$ $O(\log 1/\delta)$ times and take the median to obtain an estimate which is accurate to within ϵ with probability at least $1 - \delta$.*

[2]Reference [AHN$^+$21] improves the scaling from $\widetilde{O}(B/\epsilon)$ to $\widetilde{O}(\sqrt{B}/\epsilon)$. Such a result also follows from the quantum Chebyshev inequality of [HM19]. Since $B = O(1)$ in our case, we apply the original algorithm of [Mon15].

This result provides a way of estimating the telescoping product (2.41). The following theorem and its proof closely follows Theorem 8 of [Mon15], while the definitions of the partition function and the cooling schedule are different.

Theorem 2.41 (Quantum speedup of annealing)**.** *Let Z be the normalizing constant in (2.3). Consider a sequence of values g_i as in (2.44), with $\frac{\mathbb{E}_{\rho_i}(g_i^2)}{\mathbb{E}_{\rho_i}(g_i)^2} = O(1)$. Further assume that we have the ability to sample ρ_i for $i \in [M]$. Then there is a quantum algorithm which outputs an estimate $\widetilde{Z}$, such that*

$$\Pr[(1-\epsilon)Z \leq \widetilde{Z} \leq (1+\epsilon)Z] \geq \frac{3}{4}, \tag{2.99}$$

using

$$O\Big(\frac{M^2}{\epsilon}\log^{3/2}\Big(\frac{M}{\epsilon}\Big)\log\log\Big(\frac{M}{\epsilon}\Big)\Big) = \widetilde{O}\Big(\frac{M^2}{\epsilon}\Big) \tag{2.100}$$

samples in total.

Proof. For $i \in [M]$, we estimate $\mathbb{E}_{\rho_i}(g_i)$ with output $\widetilde{g}_i$ up to additive error $(\epsilon/2M)\mathbb{E}_{\rho_i}(g_i)$ with failure probability $1/4M$. We output as a final estimate

$$\widetilde{Z} = \widetilde{Z}_1 \prod_{i=1}^{M} \widetilde{g}_i, \tag{2.101}$$

where $\widetilde{Z}_1$ is the normalizing constant of the Gaussian distribution with variance σ_1^2 as in Lemma 2.12. Assuming that all the estimates are indeed accurate, we have

$$1-\epsilon \leq (1-\frac{\epsilon}{2})\Big(1-\frac{\epsilon}{2M}\Big)^M \leq \frac{\widetilde{Z}}{Z} \leq \Big(1+\frac{\epsilon}{2M}\Big)^M \leq e^{\epsilon/2} \leq 1+\epsilon. \tag{2.102}$$

Thus $|\widetilde{Z} - Z| \leq \epsilon Z$ with probability at least

$$\Big(1-\frac{1}{4M}\Big)^M \geq 1 - \frac{1}{4} = \frac{3}{4}. \tag{2.103}$$

Based on Lemma 2.13 and Lemma 2.14, $\frac{\mathbb{E}_{\rho_i}(g_i^2)}{\mathbb{E}_{\rho_i}(g_i)^2} = O(1)$, so $\frac{\mathrm{Var}_{\rho_i}(g_i)}{\mathbb{E}_{\rho_i}(g_i)^2} = O(1)$. By Lemma 2.39, each requires

$$O\Big(\frac{M}{\epsilon}\log^{3/2}\Big(\frac{M}{\epsilon}\Big)\log\log\Big(\frac{M}{\epsilon}\Big)\Big) \tag{2.104}$$

samples from ρ_i, and the total number of samples is

$$O\Big(\frac{M^2}{\epsilon}\log^{3/2}\Big(\frac{M}{\epsilon}\Big)\log\log\Big(\frac{M}{\epsilon}\Big)\Big). \tag{2.105}$$

This completes the proof. □

2.10.1.2 Quantum MALA for estimating the normalizing constant

We now describe how to combine quantum annealing with quantum MALA to reduce the query complexity of estimating the normalizing constants.

We begin with the following lemma on non-destructive mean estimation.

Lemma 2.42 (Non-destructive mean estimation with quantum MALA)**.** *For $\epsilon < 1$, given $\widetilde{O}(\epsilon^{-1})$ copies of a state $|\widetilde{\rho}_{i-1}\rangle$ such that $\||\widetilde{\rho}_{i-1}\rangle - |\rho_{i-1}\rangle\| \leq \epsilon$, there exists a quantum procedure that outputs $\widetilde{g}_i$ such that*

$$|\widetilde{g}_i - \mathbb{E}_{\rho_i}[g_i]| \leq \epsilon \cdot \mathbb{E}_{\rho_i}[g_i] \tag{2.106}$$

with success probability $1 - o(1)$ using

$$\widetilde{O}(\kappa^{1/2}d^{1/2}\epsilon^{-1}) \tag{2.107}$$

steps of the quantum walk operator corresponding to the MALA with stationary distribution ρ_i, where δ is the spectral gap of the Markov chain. The quantum procedure also returns $\widetilde{O}(\epsilon^{-1})$ copies of the state $|\widetilde{\rho}_i\rangle$ such that $\||\widetilde{\rho}_i\rangle - |\rho_i\rangle\| \leq \epsilon$.

Proof sketch. This lemma is a variant of Lemma 4.4 in [CCLW20]. Notice that Corollary 2.37 implies that we can prepare $|\widetilde{\rho}_i\rangle$ from $|\widetilde{\rho}_{i-1}\rangle$ using $\widetilde{O}(\kappa^{1/2}d^{1/2}p^{-1})$ quantum walk steps, where $p \leq |\langle\rho_i|\rho_{i-1}\rangle|$. By Lemma 2.15, we have $p = \Omega(1)$. The lemma follows by properly choosing the parameters in Lemma 4.4 in [CCLW20]. □

Theorem 2.43 (Quantum speedup using MALA, annealing, and quantum walk)**.** *Let Z be the normalizing constant in (2.3). Assume we are given the access to query* *the quantum gradient oracle (2.5). Then there is a quantum algorithm which outputs an estimate $\widetilde{Z}$, such that*

$$\Pr[(1-\epsilon)Z \leq \widetilde{Z} \leq (1+\epsilon)Z] \geq \frac{3}{4}, \tag{2.108}$$

using

$$\widetilde{O}\big(d^{3/2}\kappa^{1/2}\epsilon^{-1}\big) \tag{2.109}$$

queries to the quantum gradient oracle in total.

Proof. The number of annealing stages is $M = \widetilde{O}(\sqrt{d})$. At the ith stage, we estimate $\mathbb{E}_{\rho_i}[g_i]$ with relative error ϵ/M. Hence, we can apply Lemma 2.42 M times, where each application takes

$$\widetilde{O}(\kappa^{1/2}d^{1/2}(\epsilon/M)^{-1}) = \widetilde{O}(\kappa^{1/2}d\epsilon^{-1}) \tag{2.110}$$

MALA quantum walk steps. This process takes

$$M \cdot \widetilde{O}(\kappa^{1/2}d\epsilon^{-1}) = \widetilde{O}(\kappa^{1/2}d^{3/2}\epsilon^{-1}) \tag{2.111}$$

steps in total.

By Lemma 2.35, each step of the quantum walk operator can be implemented by querying the gradient oracle and the evaluation oracle $O(1)$ times. Therefore, we can estimate Z with relative error ϵ using $\widetilde{O}(\kappa^{1/2}d^{3/2}\epsilon^{-1})$ queries to the gradient and evaluation oracles. □

2.10.2 Quantum multilevel Langevin algorithms

We now consider an alternative approach for estimating the normalizing constant, by replacing MALA by a multilevel Langevin approach. More concretely, for each sample we perform the underdamped Langevin diffusion (ULD) or the randomized midpoint method for underdamped Langevin diffusion (ULD-RMM) that has an improved dependence on the dimension, and apply the multilevel Monte Carlo (MLMC) to preserve the dependence on the accuracy.

Multilevel Monte Carlo methods have attracted extensive attention in stochastic simulations and financial models [Gil08, Gil15]. This approach was originally developed by [Hei01] for parametric integration, and used to simulate SDEs in [Gil08]. Considering a general random variable P, MLMC gives a sequence of estimators $P_0, P_l, \ldots, P_L$ for approximating P with increasing accuracy and cost, and uses the telescoping sum of $\mathbb{E}[P_l - P_{l-1}]$ to estimate $\mathbb{E}[P]$. For $P_l - P_{l-1}$ with smaller variance but larger cost, MLMC performs fewer samples to reach a given error tolerance, reducing the overall complexity. MLMC has been widely discussed and improved under many settings, and has been used in various applications [Gil15].

To estimate normalizing constants, a variant of MLMC has been proposed by [GLL20, Lemmas C.1 and C.2]. Unlike standard MLMC for bounding the mean-squared error, this approach upper bounds the bias and the variance separately, making the analysis more technically involved. The first quantum algorithm based on MLMC was developed by [ALL+21, Theorem 2]. They upper bound the additive error with high probability (as is common for quantum algorithms). They also observe that the mean-squared error can control both the bias and the variance [ALL+21, Section 2.2] and that the mean-squared error is almost equivalent to the additive error with high probability [ALL+21, Appendix A]. Considering this, we still use the additive error scenario for estimating normalizing constants, both for convenience and for compatibility with the quantum annealing speedup of Theorem 2.41.

We first introduce the general quantum speedup of MLMC as described in [ALL+21], and then apply these results to our problem.

2.10.2.1 Quantum-accelerated multilevel Monte Carlo method

We begin by describing the following general result on quantum-accelerated multilevel Monte Carlo (QA-MLMC).

Lemma 2.44 (Theorem 2 of [ALL⁺21])**.** *Let P denote a random variable, and let P_l (for $l \in \{0, 1, \ldots, L\}$) denote a sequence of random variables such that P_l approximates P at level l. Also define $P_{-1} = 0$. Let C_l be the cost of sampling from P_l, and let V_l be the variance of $P_l - P_{l-1}$. If there exist positive constants $\alpha, \beta = 2\hat{\beta}, \gamma$ such that $\alpha \geq \min(\hat{\beta}, \gamma)$ and*

- $|\mathbb{E}[P_l - P]| = O(2^{-\alpha l})$,
- $V_l = O(2^{-\beta l}) = O(2^{-2\hat{\beta} l})$, *and*
- $C_l = O(2^{\gamma l})$,

then for any $\epsilon < 1/e$ there is a quantum algorithm that estimates $\mathbb{E}[P]$ up to additive error ϵ with probability at least 0.99, and with cost

$$\begin{cases} O\Big(\epsilon^{-1}(\log 1/\epsilon)^{3/2}(\log\log 1/\epsilon)^2\Big), & \hat{\beta} > \gamma, \\ O\Big(\epsilon^{-1}(\log 1/\epsilon)^{7/2}(\log\log 1/\epsilon)^2\Big), & \hat{\beta} = \gamma, \\ O\Big(\epsilon^{-1-(\gamma-\hat{\beta})/\alpha}(\log 1/\epsilon)^{3/2}(\log\log 1/\epsilon)^2\Big), & \hat{\beta} < \gamma. \end{cases} \tag{2.112}$$

We apply this result to the payoff model of general stochastic differential equations (SDEs) as discussed in [ALL⁺21]. Consider an SDE

$$\mathrm{d}X_t = \mu(X_t, t)\,\mathrm{d}t + \sigma(X_t, t)\,\mathrm{d}W_t \tag{2.113}$$

for $t \in [0, T]$, where we assume μ and σ are Lipschitz continuous. Given an initial condition X_0 and an evolution time $T > 0$, we aim to compute

$$\mathbb{E}[\mathcal{P}(X_T)], \tag{2.114}$$

where $\mathcal{P}(X)$ is the so-called payoff function as a functional of X. In Lemma 2.44, we denote $\mathcal{P}(X_T)$ as P, and the goal is to estimate $\mathbb{E}[P]$.

We also consider a numerical scheme that produces $\widehat{X}_k$ with time step size $h = T/n$. We say the scheme is of strong order r if for any $m \in \{1, 2\}$, there exists a constant C_m such that

$$\mathbb{E}\Big(\|\widehat{X}_n - X_T\|^m\Big) \leq C_m h^{rm}. \tag{2.115}$$

Note that it suffices to verify this condition for $m = 2$ since $(\mathbb{E}\|\widehat{X}_n - X_T\|)^2 \leq \mathbb{E}\|\widehat{X}_n - X_T\|^2$. We further assume the coefficients of the scheme are Lipschitz continuous. For the discretization $n_l = 2^l$ with step size $h = T/2^l$, we let P_l denote $\mathcal{P}(\widehat{X}_{n_l})$, an estimator of P.

Finally, we assume $\mathcal{P}(X)$ is L_P-Lipschitz continuous. Thus, we have satisfied the three assumptions of Proposition 2 of [ALL+21], which estimates the rates of $|\mathbb{E}[P_l - P]|, V_l, C_l$ in Lemma 2.44. We state a simpler version as follows.

Lemma 2.45 (Proposition 2 of [ALL+21])**.** *Given an SDE and a scheme of strong order r with Lipschitz continuous constants, and given a Lipschitz continuous payoff function $\mathcal{P}$, we have $\alpha = r$, $\beta = 2r$, and $\gamma = 1$.*

Note that while we relax the definition of a scheme of strong order r, our definition (2.115) is sufficient to prove Lemma 2.45. More concretely, in the proof of Proposition 2 of [ALL+21], we have the following simplified inequalities:

$$|\mathbb{E}[P_l - P]| \leq \mathbb{E}|\mathcal{P}(\widehat{X}_n) - \mathcal{P}(X_T)| \leq L_P\mathbb{E}\|\widehat{X}_n - X_T\| \leq L_P C_1 h^r = O(2^{-rl}), \quad (2.116)$$

$$V_l \leq \mathbb{E}|\mathcal{P}(\widehat{X}_n) - \mathcal{P}(X_T)|^2 \leq L_P\mathbb{E}\|\widehat{X}_n - X_T\|^2 \leq L_P C_2 h^{2r} = O(2^{-2rl}), \quad (2.117)$$

$$C_l = O(n_l) = O(2^l), \quad (2.118)$$

and therefore $\alpha = r$, $\beta = 2r$, and $\gamma = 1$.

Finally, we can characterize the performance of QA-MLMC as follows.

Lemma 2.46 (Theorem 3 of [ALL+21])**.** *Given an SDE and a scheme of strong order r with Lipschitz continuous constants, and given a Lipschitz continuous payoff function $\mathcal{P}$, QA-MLMC estimates $\mathbb{E}[\mathcal{P}]$ up to additive error ϵ with probability at least 0.99 with cost*

$$O\Big(\epsilon^{-1}(\log 1/\epsilon)^{3/2}(\log\log 1/\epsilon)^2\Big), \quad r > 1, \quad (2.119)$$

$$O\Big(\epsilon^{-1}(\log 1/\epsilon)^{7/2}(\log\log 1/\epsilon)^2\Big), \quad r = 1, \quad (2.120)$$

$$O\Big(\epsilon^{-1/r}(\log 1/\epsilon)^{3/2}(\log\log 1/\epsilon)^2\Big), \quad r < 1. \quad (2.121)$$

Note that we can amplify the success probability to $1-\delta$ for arbitrarily small $\delta > 0$ using the powering lemma (Lemma 2.40).

2.10.2.2 Quantum-accelerated multilevel Langevin method

We have described ULD and ULD-RMM in Algorithm 4 and Algorithm 5, respectively. We now apply these schemes to simulate the underdamped Langevin dynamics as the SDE. According to (2.115), ULD and ULD-RMM are schemes of order 1 and 1.5, respectively.

Let the payoff function $\mathcal{P}$ be g_i as defined in (2.44). Our goal is to estimate the mean of $\mathcal{P}(\widehat{X}_{n_l}) = g_i(\widehat{X}_{n_l})$ using several samples $\widehat{X}_{n_l}$ produced by ULD or ULD-RMM. If g is assumed to be L_g-Lipschitz as in Lemma C.2 of [GLL20], we have a Lipschitz continuous payoff function $\mathcal{P}$ with $L_P = L_g$. Although $g_i = \exp\bigl(\frac{\|x\|^2}{\sigma_i^2(1+\alpha^{-1})}\bigr)$ is not Lipschitz, according to Section 4.3 of [GLL20], we can truncate at large x and replace g_i by $h_i := \min\bigl\{g_i, \exp\bigl(\frac{(r_i^+)^2}{\sigma_i^2(1+\alpha^{-1})}\bigr)\bigr\}$ with

$$\alpha = \widetilde{O}\left(\frac{1}{\sqrt{d}\log(1/\epsilon)}\right), \tag{2.122}$$

$$r_i^+ = \mathbb{E}_{\rho_{i+1}}\|x\| + \Theta(\sigma_i\sqrt{(1+\alpha)\log(1/\epsilon)}), \tag{2.123}$$

to ensure $\frac{h_i}{\mathbb{E}_{\rho_i} g_i}$ is $O(\sigma_i^{-1})$ Lipschitz and $\bigl|\mathbb{E}_{\rho_i}(h_i - g_i)\bigr| < \epsilon$ by Lemmas C.7 and C.8 of [GLL20]. For simplicity, as in Section 4.2 of [GLL20], we regard g_i as a Lipschitz continuous payoff function in our main results.

Thus, using Lemma 2.46 to estimate $\frac{Z_{i+1}}{Z_i} = \mathbb{E}_{\rho_i}(g_i)$, QA-MLMC using either ULD or ULD-RMM can reduce the ϵ-dependence of the number of steps to $\widetilde{O}(\epsilon^{-1})$. Then each step of ULD or ULD-RMM uses the value of $\nabla f(x)$ about $O(1)$ times as shown in Algorithm 4 and Algorithm 5.

Having described the implementations of quantum inexact ULD and ULD-RMM, we now state the quantum speedup for estimating normalizing constants using multilevel ULD and annealing, or multilevel ULD-RMM and annealing, as follows.

Theorem 2.47 (Quantum speedup using multilevel ULD and annealing)**.** *Let Z be the normalizing constant in (2.3). Assume we are given access to the quantum gradient oracle (2.5). Then there is a quantum algorithm which outputs an estimate $\widetilde{Z}$ such that*

$$\Pr[(1-\epsilon)Z \le \widetilde{Z} \le (1+\epsilon)Z] \ge \frac{3}{4} \tag{2.124}$$

using

$$\widetilde{O}\Big(\frac{d^{3/2}\kappa^2}{\epsilon}\Big) \tag{2.125}$$

queries to the quantum gradient oracle.

Proof. As in Theorem 2.41 and Theorem 2.43, for $i \in [M]$ with M stages, we estimate $\mathbb{E}_{\rho_i}(g_i)$ with output $\widetilde{g}_i$ up to additive error $(\epsilon/2M)\mathbb{E}_{\rho_i}(g_i)$ with failure probability $1/4M$, which ensures $|\widetilde{Z} - Z| \le \epsilon Z$ with probability at least $\frac{3}{4}$.

According to Lemma 2.46, each sample of ρ_i using multilevel ULD uses $\widetilde{O}(\frac{M\kappa^2\sqrt{d}}{\epsilon})$ queries to the quantum evaluation oracle (2.4) used in Algorithm 6 or Algorithm 7. For $M = \widetilde{O}(\sqrt{d})$ stages, the query complexity of estimating the normalizing constant is $\widetilde{O}(\frac{M^2\kappa^2\sqrt{d}}{\epsilon}) = \widetilde{O}(\frac{d^{3/2}\kappa^2}{\epsilon})$. □

Theorem 2.48 (Quantum speedup using multilevel ULD-RMM and annealing)**.** *Let Z be the normalizing constant in (2.3). Assume we are given the access to the quantum gradient oracle (2.5). Then there is a quantum algorithm which outputs an estimate $\widetilde{Z}$ such that*

$$\Pr[(1-\epsilon)Z \le \widetilde{Z} \le (1+\epsilon)Z] \ge \frac{3}{4} \tag{2.126}$$

using

$$\widetilde{O}\Big(\frac{d^{7/6}\kappa^{7/6} + d^{4/3}\kappa}{\epsilon}\Big) \tag{2.127}$$

queries to the quantum gradient oracle.

Proof. As above, for $i \in [M]$ with M stages, we estimate $\mathbb{E}_{\rho_i}(g_i)$ with output $\widetilde{g}_i$ up to additive error $(\epsilon/2M)\mathbb{E}_{\rho_i}(g_i)$ with failure probability $1/4M$.

According to Lemma 2.46 and Lemma 2.11, each sample of ρ_i using multilevel ULD uses $\widetilde{O}(\frac{M(\kappa^{7/6}d^{1/6}+\kappa d^{1/3})}{\epsilon})$ queries to the quantum gradient oracle (2.5) or the quantum evaluation oracle (2.4) (with additional $\widetilde{O}(1)$ cost). For $M = \widetilde{O}(\sqrt{d})$ stages, the query complexity of estimating the normalizing constant is $\widetilde{O}(\frac{M^2(\kappa^{7/6}d^{1/6}+\kappa d^{1/3})}{\epsilon}) = \widetilde{O}(\frac{d^{7/6}\kappa^{7/6}+d^{4/3}\kappa}{\epsilon})$. □

2.11 Proof of the Quantum Lower Bound

To prove Theorem 2.9, we use the following quantum query lower bound on the Hamming weight problem.

Proposition 2.49 (Theorem 1.3 of [NW99])**.** *For $x = (x_1, \ldots, x_n) \in \{0,1\}^n$, let $\|x\|_1 = \sum_{i=1}^n x_i$ be the Hamming weight of x. Furthermore, let ℓ, ℓ' be integers such that $0 \le \ell < \ell' \le n$. Define the partial boolean function $f_{\ell,\ell'}$ on $\{0,1\}^n$ as*

$$f_{\ell,\ell'}(x) = \begin{cases} 0 & \text{if } \|x\|_1 = \ell \\ 1 & \text{if } \|x\|_1 = \ell'. \end{cases} \tag{2.128}$$

Let $m \in \{\ell, \ell'\}$ be such that $|\frac{n}{2} - m|$ is maximized, and let $\Delta = \ell' - \ell$. Then given the quantum query oracle

$$O_x|i\rangle|b\rangle = |i, b \oplus x_i\rangle \quad \forall i \in [n], b \in \{0,1\}, \tag{2.129}$$

the quantum query complexity of computing the function $f_{\ell,\ell'}$ is $\Theta(\sqrt{n/\Delta} + \sqrt{m(n-m)}/\Delta)$.

Now we prove Theorem 2.9 using a construction motivated by Section 5 of [GLL20].

Proof. We start from a basic function $f_0(x) = \frac{\|x\|^2}{2}$. The partition function of f_0 is

$$\int_{\mathbb{R}^k} e^{-f_0(x)} \mathrm{d}x = (2\pi)^{k/2}. \tag{2.130}$$

We then construct n cells in $\mathbb{R}^d$. Without loss of generality we assume that $n^{1/k}$ is an integer, and let $l := 1/(\sqrt{k}n^{1/k})$. We partition $[-1/\sqrt{k}, 1/\sqrt{k}]$ into $n^{1/k}$ intervals, each having length $2l$. Let I_i denote the i^{th} interval, where $i \in [n^{1/k}]$. Each cell will thus be represented as a k-tuple $(i_1, \ldots, i_k) \in \{1, 2, \ldots, n^{1/k}\}^k$ corresponding to $I_{i_1} \times \cdots \times I_{i_k} \subset \mathbb{R}^k$.

Next, each cell $\tau = (i_1, \ldots, i_k)$ with center denoted v_τ is assigned one of two types (as detailed below), and we let

$$f(x) = \begin{cases} f_0(x) & \text{if cell } \tau \text{ is of type 1} \\ f_0(x) + c_\tau q(\frac{1}{l}(x - v_\tau)) & \text{if cell } \tau \text{ is of type 2.} \end{cases} \tag{2.131}$$

The function q and the normalizing constant c_τ are carefully chosen, following Lemma D.1 in [GLL20], such that

- $f(x)$ is 1.5-smooth and 0.5-strongly convex; and
- the partition function $Z_f = \int_{\mathbb{R}^k} e^{-f(x)} \mathrm{d}x = (2\pi)^{k/2} - C \cdot \frac{n_2}{n}$, where n_2 is the number of type-2 cells, and C is at least $\Omega(l^2)$.

With these properties, we consider two functions as follows. We choose δ such that $\epsilon = \Theta(\delta^{1+4/k})$. One of the functions has a $1/2+\delta$ fraction of its cells of type 1 (and a $1/2-\delta$ fraction of type 2). The other function has a $1/2-\delta$ fraction of its cells of type 1 (and a $1/2+\delta$ fraction of type 2). Note that one query to the quantum evaluation oracle (2.4) can be implemented using one quantum query to the binary information indicating the type of the corresponding cell. In addition, by Proposition 2.49 with $\ell = (1-\delta)n/2$ and $\ell' = (1+\delta)n/2$, it takes $\Omega(1/\delta)$ quantum queries to distinguish whether there are $(1+\delta)n/2$ or $(1-\delta)n/2$ cells of type 1. Since $C = \Omega(l^2)$, the partition functions of the two functions differ by a multiplicative factor of at least $1 + \Omega(l^2\delta)$, where $l = \Theta(n^{-1/k}) = \Theta(\delta^{2/k})$, and hence $l^2\delta = \Theta(\delta^{1+4/k}) = \Theta(\epsilon)$. The quantum query complexity is therefore $\Omega(1/\delta) = \Omega(\epsilon^{-\frac{1}{1+4/k}})$ as claimed. □

Chapter 3: Quantum Speedups of Approximately Convex Optimization

3.1 Introduction

Optimization theory is a central research topic in computer science, mathematics, operations research, etc. Currently, many efficient algorithms for optimizing convex functions have been proposed (see for instance [BV04]), but much less is known for nonconvex optimization. In this chapter, we investigate polynomial-time algorithms for optimizing approximately convex functions. On the one hand, such algorithms enjoy robustness and cover many natural scenarios including stochastic convex optimization, empirical risk minimization, etc. On the other hand, approximately convex optimization paves the way of understanding nonconvex optimization in the general case.

Specifically, let $\mathrm{K} \subseteq \mathbb{R}^n$ be a convex set. We call $F\colon \mathbb{R}^n \to \mathbb{R}$ as an *approximately convex function* over K if there is a convex function $f\colon \mathrm{K} \to \mathbb{R}$ such that

$$\sup_{x\in\mathrm{K}} |F(x) - f(x)| \leq \epsilon/n. \tag{3.1}$$

Throughout the chapter, we assume that f is L-Lipschitz with respect to ℓ_∞ norm, i.e., $|f(x)-f(y)| \leq L\|x-y\|_\infty$ for any $x, y \in \mathrm{K}$. We assume that $\mathcal{B}_2(0,1) \subseteq \mathrm{K} \subseteq \mathcal{B}_2(0,R)$, i.e., the convex body K contains the unit ball centered at 0 and is contained by a ball of radius R centered at 0. Unless otherwise mentioned, we aim at algorithms with $\mathrm{poly}(\log R)$ dependence in R.

It is standard to assume the *zeroth-order oracle* of F, which returns the function value $F(x)$ given an input x. As far as we know, the state-of-the-art algorithm for finding an $x^* \in \mathrm{K}$ such that $F(x^*) - \min_{x\in\mathrm{K}} F(x) \leq \epsilon$ with high probability was proposed by Belloni et al. [BLNR15], which takes $\tilde{O}(n^{4.5})$[1] queries to the zeroth-order

[1]The $\tilde{O}$ and $\tilde{\Omega}$ notation omits poly-logarithmic terms, i.e., $\tilde{O}(g) = O(g\,\mathrm{poly}(\log g))$ and $\tilde{\Omega}(g) =$

oracle. On the other hand, Ref. [RL16] proved that if the approximation error in (3.1) is at least $\tilde{\Omega}(\max\{\epsilon/n, \epsilon^2/\sqrt{n}\})$, there exists a function F which no algorithm can find a point $x^* \in \mathrm{K}$ such that $F(x^*) - \min_{x\in\mathrm{K}} F(x) \leq \epsilon$ using $\mathrm{poly}(n, 1/\epsilon)$ queries to F. In other words, the ϵ/n term in (3.1) is fundamental, and it has been the standard assumption of studying approximately convex optimization in [BLNR15, RL16].

Based on approximate convex functions, a closely related scenario is stochastic convex functions, where we have a function $F\colon \mathbb{R}^n \to \mathbb{R}$ such that $F(x) = f(x) + \epsilon_x$. Here $f\colon \mathbb{R}^n \to \mathbb{R}$ is a convex function and ϵ_x is a sub-Gaussian random variable with parameter σ, i.e., $\mathbb{E}[\exp(\lambda\epsilon_x)] \leq \exp(\sigma^2\lambda^2/2)$ for any λ. This implies that

$$\Pr[|\epsilon_x| \geq \sigma t] \leq 2\exp(-t^2/2) \quad \forall t \geq 0. \tag{3.2}$$

Using this fact, Belloni et al. [BLNR15] gave an algorithm for stochastic convex optimization with $\tilde{O}(n^{7.5}/\epsilon^2)$ queries to a zeroth-order oracle of F.

Contributions. In this chapter, we conduct a systemic study of quantum algorithms for the optimization of approximately convex functions, with applications to zeroth-order stochastic convex bandits. Quantum computing is a rapidly advancing technology, the capability of quantum computers is dramatically increasing and recently reached “quantum supremacy" by Google [AAB+19] and USTC [ZWD+20b]. In optimization theory, quantum advantages have been proven for semidefinite programs [AGGW17, BKL+17, BS17, AG19], general convex optimization [AGGW20, CCLW20], the escaping from saddle point problem in optimization [ZLL21], etc. Nevertheless, as far as we know, quantum algorithms for approximately convex optimization and stochastic convex optimization are widely open. In this chapter, we consider these problems using the quantum zeroth-order evaluation oracle O_F, a standard model used in previous quantum computing literature [AGGW20, CCH+19,

$\Omega(g\,\mathrm{poly}(\log g))$. Unless otherwise mentioned, both notations also omit $\mathrm{poly}(\log 1/\epsilon)$ terms.

CCLW20, ZLL21]:

$$O_F|x,y\rangle = |x, F(x)+y\rangle \quad \forall x \in \mathbb{R}^n, y \in \mathbb{R}. \tag{3.3}$$

Here $|\cdot\rangle$ is the Dirac notation, and preliminaries of quantum computing will be covered in Section 3.2. Intuitively, Eq. (3.3) can take inputs with form $\sum_{i=1}^m \mathbf{c}_i|\mathbf{x}_i\rangle \otimes |0\rangle$ where $\mathbf{x}_i \in \mathbb{R}^n$ $\forall i \in [m]$ and $\sum_{i=1}^m |\mathbf{c}_i|^2 = 1$, and if we measure the outcome quantum state, we get $F(\mathbf{x}_i)$ with probability $|\mathbf{c}_i|^2$. In particular, the quantum zeroth-order oracle allows the ability to query different locations in *superposition*, which is stronger than the classical counterpart (i.e., $m = 1$). Nevertheless, if the classical zeroth-order oracle can be implemented by explicit arithmetic circuits, the quantum oracle in (3.3) can be implemented by quantum circuits of the same size up to logarithmic factors. As for stochastic convex functions, similar to (3.3), we assume the following oracle:

$$O_f|x,y\rangle = |x\rangle \int_{\xi\in\mathbb{R}} \sqrt{g_x(\xi)}|f(x)+y+\xi\rangle \,\mathrm{d}\xi \quad \forall x \in \mathbb{R}^n, y \in \mathbb{R}, \tag{3.4}$$

where for any $x \in \mathbb{R}^n$, $g_x(\xi)$ follows a sub-Gaussian distribution as in (3.2).

Our first result is a quantum algorithm for optimizing approximately convex functions:

Theorem 3.1. *With probability at least* 0.9*, we can find an* $x^* \in \mathrm{K}$ *such that*

$$F(x^*) - \min_{x\in\mathrm{K}} F(x) \le \epsilon \tag{3.5}$$

using $\tilde{O}(n^3)$ *queries to the quantum evaluation oracle (3.3).*

We remark that the succss probability can be easily boosted up to $1-\delta$ for any $\delta \in (0,1)$, by paying an extra $\log(1/\delta)$ factor in the quantum query complexity. Compared to the best-known classical result by Belloni et al. [BLNR15] with query complexity $\tilde{O}(n^{4.5})$, we achieve a polynomial quantum speedup in terms of n. Technically, Belloni et al.'s algorithm is based on the simulated annealing process and using the Hit-and-Run walk to generate samples in each stage. In this chapter, we give a *user-friendly* version of the quantum walk framework (Theorem 3.5) that can be applied very easily to obtain quantum speedup for the mixing of classical Markov chains. We then analyze the warmness and the overlap between adjacent Hit-and-Run walks in the simulated annealing process, showing that our quantum walk framework is applicable to each classical Hit-and-Run sampler. By implementing the random walk quantumly, we improve the query complexity of the sampling procedure from $\widetilde{O}(n^3)$ to $\widetilde{O}(n^{1.5})$ (Theorem 3.8). We also design a non-destructive rounding procedure (Lemma 3.9) in each stage of the simulated annealing that simulates the classical rounding procedure in [BLNR15] but will not destroy the quantum states of the quantum walks. Combining them together gives the $\widetilde{O}(n^3)$-query quantum algorithm for minimizing an approximately convex function. Our result can also be applied to give an $\widetilde{O}(n^3)$-query quantum algorithm for optimizing approximately convex functions with decreasing fluctuations.

See Section 3.3 for more details and the proof of Theorem 3.1.

Furthermore, to estimate the mean of a random variable up to multiplicative error ϵ, classical algorithms need to take $1/\epsilon^2$ samples due to concentration inequalities such as Chernoff's bound, whereas quantum algorithms can take roughly $1/\epsilon$ queries (see Proposition 3.10). As a result, we obtain polynomial quantum speedup in both n and $1/\epsilon$ for stochastic convex optimization:

Corollary 3.2. *With probability at least* 0.8*, we can find an* $x^* \in \mathrm{K}$ *such that*

$$f(x^*) - \min_{x \in \mathrm{K}} f(x) \leq \epsilon \tag{3.6}$$

using $\tilde{O}(n^5/\epsilon)$ *queries to the quantum stochastic evaluation oracle (3.4).*

We apply Corollary 3.2 to solve the zeroth-order stochastic convex bandit problem, which is a widely studied bandit model (see e.g., [HL16, Lat20, LG21]). The problem is defined as follows. Let $\mathrm{K} \subseteq \mathbb{R}^n$ be a convex body and $f\colon \mathrm{K} \to [0,1]$ be a convex function. Here $\mathcal{B}_2(0,1) \subseteq \mathrm{K} \subseteq \mathcal{B}_2(0,R)$. An online learner and environment interact alternatively over T rounds. In each round $t \in [T]$, the learner makes a query to the quantum stochastic evaluation oracle (3.4), and returns a value $x_t \in \mathrm{K}$ as the current guess. The learner aims to minimize the regret

$$\mathcal{R}_T := \mathbb{E}\left[\sum_{t=1}^{T}(f(x_t) - f^*)\right], \quad \text{where } f^* = \min_{x\in\mathrm{K}} f(x), \tag{3.7}$$

and the expectation is taken over all randomness. The classical state-of-the-art algorithm using a classical stochastic evaluation in each round achieves a regret bound of $\tilde{O}(n^{4.5}\sqrt{T})$ [LG21], and there is a classical lower bound $\Omega(n\sqrt{T})$ on the regret [DHK09]. Here we prove:

Theorem 3.3. *There is a quantum algorithm for which* $\mathcal{R}_T = \tilde{O}(n^5 \log(T)\log(TR))$.

This achieves poly$(\log T)$ regret for zeroth-order stochastic convex bandits, **an exponential quantum advantage** in terms of T compared to classical zeroth-order stochastic convex bandits. As far as we know, we give the first quantum algorithms with poly-logarithmic regret bound on online learning problems. An independent work [WZL+22] gave quantum algorithms with poly-logarithmic regret for multi-armed bandits and stochastic linear bandits, but these two types of bandits concern reward of discrete objects and linear functions, respectively, which are fundamentally different from the stochastic convex bandits we study.

To achieve this poly$(\log T)$ regret for zeroth-order stochastic convex bandits, we divide the T iterations into $\lfloor \log_2 T \rfloor$ intervals with doubling length $1, 2, 4, \ldots$. We use the quantum queries from a previous interval to run our quantum stochastic convex optimization algorithm and use the output as the guess for all iterations in the next interval. Since we can achieve linear dependence in $1/\epsilon$ in Corollary 3.2, it can be calculated that the total regret in each iteration is at most $\tilde{O}(n^5 \log(TR))$, leading to our claim. The proof details are given in Section 3.4.

Open questions. Our work raises several natural questions for future investigation:

- Can we further improve the dimension dependence of our approximate convex optimization algorithm? The current quantum speedup mainly leverages the quantum hit-and-run walk; it is of general interest to understand whether we can gain quantum advantage at other steps, or whether the convergence analysis of the hit-and-run walk per se can be improved.
- Can we improve the dimension dependence of the regret of our zeroth-order stochastic convex bandits? It is natural to check whether our n^5 term can be improved by quantizing the classical state-of-the-art [LG21] and other recent works.
- Can we give fast quantum algorithms for more general nonconvex optimization problems, or with poly-logarithmic regret for more general bandit problems?

3.2 Preliminaries

3.2.1 Quantum computing in continuous space

Some basic notations and properties of quantum computing are introduced in Appendix B. Here, we note that in general, the definition of quantum states can be extended to a continuous domain. For instance,

$$|v\rangle = \int_{\mathbb{R}^n} v_x |x\rangle \mathrm{d}x$$

represents a quantum state as long as $\int_{\mathbb{R}^n} |v_x|^2 \mathrm{d}x = 1$. To keep quantum states normalized in ℓ_2 norm, operations in quantum computing are *unitary transformations*. And we note that this continuous state notation is just for simplicity and has already been used in previous works (e.g., [CCH$^+$19]). To implement the algorithms in real-world quantum computers, some discretization techniques will be applied.

3.2.2 Classical and quantum walks

A classical Markov over the space Ω is a sequence of random variables $\{X_i\}_{i\in\mathbb{N}}$ such that for any $i > 0$ and $x_0, \dots, x_i \in \Omega$,

$$\Pr[X_i = x_i \mid X_0 = x_0, \cdots, X_{i-1} = x_{i-1}] = \Pr[X_i = x_i \mid X_{i-1} = x_{i-1}].$$

The Markov chain can be represented by its stochastic transition matrix P such that $\sum_{y\in\Omega} P(x, y) = 1$ for any $x \in \Omega$. A distribution π is *stationary* if it satisfies $\sum_{x\in\Omega} \pi(x)P(x, y) = \pi(y)$ for any $y \in \Omega$. A Markov chain is *reversible* if its stationary distribution satisfies the following detailed balance condition:

$$\pi(x)P(x, y) = \pi(y)P(y, x) \quad \forall x, y \in \Omega.$$

The *mixing time* of a Markov chain with initial distribution π_0 is the number of steps $t = t(\epsilon) \in \mathbb{N}$ such that the total variation distance between the time-t distribution and the stationary distribution is at most ϵ for any $\epsilon \in (0, 1)$, i.e.,

$$d_{\mathrm{TV}}(P^t\pi_0, \pi) \leq \epsilon.$$

In quantum, we can also define the discrete-time quantum walk [Sze04], which is a quantum-analogue of classical Markov chain. More precisely, we use a quantum state to represent a classical probability distribution:

$$\{\pi(x)\}_{x\in\Omega} \longleftrightarrow |\pi\rangle = \sum_{x\in\Omega} \sqrt{\pi(x)}|x\rangle.$$

In quantum walk, there are two operations:

- **Reflect:** An operator R that reflects the quantum states with respect to the subspace $\mathrm{span}\{|x\rangle|\psi_x\rangle\}_{x\in\Omega}$, where $|\psi_x\rangle = \sum_{y\in\Omega} \sqrt{P(x, y)}|y\rangle$.
- **Swap:** An operator S that swap the two quantum registers: $S|x\rangle|y\rangle = |y\rangle|x\rangle$.

Then, the quantum walk operator W is defined as: $W := S \circ R$. Intuitively, the first quantum register contains the current position of the random walk, and the second register contains the previous position. In each step of quantum walk, the R operator makes a superposition in the second register of the next step positions with amplitudes proportional to their transition probabilities. Then, the S operator swaps the two quantum registers, using the first register to store the new positions and the second register to store the old position. In this way, we complete one-step of the random walk coherently (in superposition).

The quantum advantage of the quantum walk comes from the spectrum of W. Note that the state of stationary distribution $\sum_{x\in\Omega}\sqrt{\pi(x)}|x\rangle|\psi_x\rangle$ is invariant under W. In other words, it is an eigenvector with eigenvalue 1 (eigenphase 0). On the other hand, the other eigenvectors of W has eigenphase at least $\sqrt{2\delta}$, where δ is the spectral gap[2] of the transition matrix P. Therefore, applying the *quantum phase estimation* algorithm using $O(1/\sqrt{\delta})$ calls to W can distinguish the state corresponding to the stationary distribution and other eigenstates. See [Sze04, WA08, Chi21] for more details.

3.2.3 Hit-and-Run walk

Hit-and-Run walk was introduced by R.L. Smith [Smi84], and has a long line of reserach [BBRR$^+$87, BRS93, CS93, ZSM$^+$93, Lov99, LV03, LV06, LV07, AYBGM17] for its mixing time and applications in sampling, optimization, and volume estimation. Intuitively, the Hit-and-Run walk is defined as follows:

1. Pick a uniformly distributed random line ℓ through the current point x.

2. Move to a random point y along the line chosen from the restricted distribution π_ℓ.

[2]The difference between the first and the second largest eigenvalues.

More specifically, let $f : \mathbb{R}^n \to \mathbb{R}$ be a logconcave distribution density. Then, the distribution induced by restricting f to the line ℓ is defined as follows:

$$\pi_\ell(S) := \frac{\int_S f(x)\mathrm{d}x}{\int_\ell f(x)\mathrm{d}x} \quad \forall S \subset \ell.$$

The following lemma show the transition probability of the Hit-and-Run walk.

Lemma 3.4 ([Lov99]). *Let f be the density of a logconcave distribution. If the current point of Hit-and-Run is u, then the density function of the distribution of the next point x is*

$$f_u(x) = \frac{2}{n\pi_n} \frac{f(x)}{\mu_f(u,x)\|x-u\|^{n-1}},$$

where $\pi_n = \frac{\pi^{n/2}}{\Gamma(1+n/2)}$ and $\mu_f(u,x)$ is the f-measure of the chord through u and x.

3.3 Quantum Algorithm for Optimizing Approximately Convex Functions

We give a polynomial quantum speedup for minimizing approximately convex functions using the quantum walk algorithm [Sze04, WA08]. More specifically, we consider the quantum walk in continuous space.[3] Let P denote the (column) stochastic transition density of a reversible Markov chain, i.e.,

$$\int_{\mathbb{R}^n} P(x,y)\mathrm{d}y = 1 \quad \forall x \in \mathbb{R}^n,$$

and let $\pi(x)$ denote the density of the stationary distribution. Then, we can implement a quantum walk unitary $W(P)$ such that its unique eigenvector with eigenvalue 1 (or equivalently eigenphase 0) is:

$$\int_{\mathbb{R}^n} \sqrt{\pi(x)}|x\rangle \otimes |\psi_x\rangle \mathrm{d}x,$$

[3]It can be naturally discretized as we do in simulating a Markov chain on classical digital computers.

where $|\psi_x\rangle := \int_{\mathbb{R}^n} \sqrt{P(x,y)}|y\rangle \mathrm{d}y$ mixes all the points that can be moved from x, with amplitudes proportional to the transition probabilities. This quantum state can be considered as a *coherent encoding* of the classical distribution π. The advantage of quantum walk comes from the fact that $W(P)$ has phase gap $\sqrt{\delta}$, where δ is the spectral gap of P. Therefore, by the quantum phase estimation algorithm [Kit95], quantum walk can achieve quadratic speedup in δ. In general, quantum walk algorithm can quadratically speedup the hitting time of classical Markov chain. For the mixing time, it requires some additional complicated constraints on the Markov chain and distributions (see e.g., [AAKV01, WCNA09, OBD18, CLW19]). Based on previous studies on quantum walk mixing (see Section 2.8), we summarize the following user-friendly quantum walk framework, which gives the quantum costs for speeding up the mixing of classical Markov chains based on three *purely classical properties.*

Theorem 3.5 (User-friendly quantum walk framework)**.** *Let M_0 be the initial Markov chain with stationary distribution π_0, M_1 be the target Markov chain with stationary distribution π_1. Suppose M_0, M_1 satisfy the following properties:*

- **Mixing time:** $d_{\mathrm{TV}}(P_1^{t_0} \cdot \pi_0, \pi_1) \leq \epsilon$ *and* $d_{\mathrm{TV}}(P_0^{t_1} \cdot \pi_1, \pi_0) \leq \epsilon$.
- **Warmness:** $\|\pi_0/\pi_1\| = O(1)$ *and* $\|\pi_1/\pi_0\| = O(1)$, *where* $\|\pi/\sigma\| := \int_{\mathbb{R}^n} \frac{\pi(x)}{\sigma(x)}\pi(x)\mathrm{d}x$.
- **Overlap:** $|\langle\pi_0|\pi_1\rangle| = \int_{\mathbb{R}^n} \sqrt{\pi_0(x)\pi_1(x)}\mathrm{d}x = \Omega(1)$.

Furthermore, suppose we have access to a unitary U that prepares the initial state $|\pi_0\rangle = \int_{\mathbb{R}^n} \sqrt{\pi_0(x)}|x\rangle \mathrm{d}x$. Then, we can obtain a state $|\widetilde{\pi}_1\rangle$ with $\||\widetilde{\pi}_1\rangle - |\pi_1\rangle\|_2 \leq \epsilon$ using

$$O\big(\sqrt{t_0+t_1}\log^2(1/\epsilon)\big)$$

calls to the quantum walk operators.

Next, we can use Theorem 3.5 to speed-up the best-known classical algorithm for optimizing approximately convex functions [BLNR15], which has the following three levels:

- **High level:** Perform a simulated annealing with K stages. At the i-th stage, the target distribution π_{g_i} has density $\propto g_i(x) = e^{-F(x)/T_i}$, where $T_i := (1 - 1/\sqrt{n})^i$.
- **Middle level:** Use N samples from π_{g_i} to construct a linear transformation Σ_i, rounding the distribution to near-isotropic position.
- **Low level:** Run the hit-and-run walk to evolve the distribution from $\pi_{g_{i-1}}$ to π_{g_i}.

Here, each step of the hit-and-run walk picks a uniformly random direction at the current point, and then walks on the 1-dimensional chord intersected by the direction and K with probability density proportional to the logconcave density. We formally state the hit-and-run walk in Algorithm 15.

We focus on speeding-up the Low level using Theorem 3.5. Hence, we need to show that π_{g_i} and $\pi_{g_{i+1}}$ satisfy the properties therein. First of all, it has been proved in [BLNR15] that $\|\pi_{g_i}/\pi_{g_{i+1}}\| = O(1)$ (Lemma 3.15). We prove the following lemma with proof deferred to Section 3.6.1.

Lemma 3.6 (Informal version of Lemma 3.18)**.** *Let π_{g_i} be a distribution with density proportional to $g_i(x) = \exp(-F(x)/T_i)$, where $F(x)$ is β-approximately convex. Then, for any $0 \leq i \leq K-1$,*

$$\|\pi_{g_{i+1}}/\pi_{g_i}\| \leq 8\exp(2\beta/T_{i+1}) \leq O(1).$$

Therefore, the warmness property is satisfied.

We also prove that the Markov chains in this annealing schedule are slowly evolving:

Lemma 3.7 (Informal version of Lemma 3.20)**.** *Let π_{g_i} be a distribution with density proportional to $g_i(x) = \exp(-F(x)/T_i)$, where $F(x)$ is β-approximately convex. Then, for any $0 \leq i \leq K-1$,*

$$|\langle \pi_i | \pi_{i+1} \rangle| \geq \exp(-(\beta/T_{i+1} + 1)/2) = \Omega(1).$$

Hence, the overlap property is also satisfied.

With the warmness and classical analysis of the hit-and-run walk (Theorem 3.13), we get that the classical mixing time from π_{g_i} to $\pi_{g_{i+1}}$ and vice versa can be bounded by $\widetilde{O}(n^3)$.

Moreover, we also show in Section 3.6.1 and Lemma 3.21 that each call to the quantum walk operator can be implemented by querying the evaluation oracle $O(1)$ times.

Thus, by Theorem 3.5, we get the following theorem:

Theorem 3.8 (Low level quantum speedup, informal version of Theorem 3.22)**.** *Let $\gamma \in (0, 1/e)$. Let $g_i(x) = \exp(-F(x)/T_i)$ be the density of π_{g_i} with $F(x)$ being $\beta/2$-approximately convex. Let $T_i = (1 - 1/\sqrt{n})^i$ for $0 \leq i \leq K$. Then, for each $0 \leq i \leq K - 1$, given a state $|\pi_{g_i}\rangle$, we can produce a state $|\hat{\sigma}_i^{(m)}\rangle$ such that*

$$\||\pi_{g_{i+1}}\rangle - |\hat{\sigma}_i^{(m)}\rangle\|_2 \leq O(\gamma),$$

using $m = \widetilde{O}(n^{1.5})$ calls for the evaluation oracle of F.

In the Middle level, we need to use N independent samples from π_{g_i} to construct a linear transformation Σ_i for rounding. However, we cannot directly measure the state $|\pi_{g_i}\rangle$, since it will destroy the quantum coherence. Instead, we propose a non-destructive approach to construct the linear transformation (the proof is deferred to Section 3.6.2):

Lemma 3.9 (Non-destructive rounding, informal version of Lemma 3.24)**.** *For each $i \in [K]$, the linear transformation Σ_i can be obtained using $\widetilde{O}(N)$ copies of the states $\left|\pi_{g_{i-1}}\right\rangle$, with query complexity $\widetilde{O}(N \cdot n^{1.5})$. Moreover, the states $\left|\pi_{g_{i-1}}\right\rangle$ will be recovered with high probability.*

Now, we can put all the components together and obtain a quantum algorithm for optimizing approximately convex function with $\widetilde{O}(n^{1.5})$ quantum query complexity

(Algorithm 13). We sketch the proof in below and the formal proof is given in Section 3.6.3:

Proof sketch of Theorem 3.1. Observe that at each annealing stage, the sample distribution is the same as the classical algorithm. Thus, by the classical analysis (Theorem 3.17) in [BLNR15], the same optimization guarantee still holds for the quantum algorithm. Thus, if we take $K = \sqrt{n}\log(n/\epsilon)$ and $N = \widetilde{O}(n)$, the output x_* of QSimAnnealing procedure satisfies:

$$F(x_*) - \min_x F(x) \leq O(\epsilon)$$

with high probability.

Then, consider the query complexity. There are K stages in the annealing process, where each stage maintains $\widetilde{O}(N)$ samples (quantum states). By Theorem 3.8, evolving each state takes $\widetilde{O}(n^{1.5})$ queries. Therefore, the total query complexity is

$$K \cdot N \cdot \widetilde{O}(n^{1.5}) = \widetilde{O}(n^3).$$

Here, we assume that the convex body $\mathcal{K}$ is known, e.g., $\mathcal{K} = \mathbb{R}^n$ or $\mathbb{S}^n$. However, even if $\mathcal{K}$ is unknown, we can call its membership oralce to run our algorithm. More specifically, in constructing the initial state $|\pi_0\rangle$, we need to query the membership oracle for $\widetilde{O}(1)$ times. And since we prepare N copies, this step takes $\widetilde{O}(N)$ queries in total. Then, in each step of the quantum walk, we need to query the membership oracle for $\widetilde{O}(1)$ times to determine the intersection point for the hit-and-run process. Thus, the number of queries to the membership oracle of $\mathcal{K}$ is the same as the the number of evaluation oracle queries. Hence, our algorithm will query the membership oracle for $\widetilde{O}(n^3)$ times in all.

As for the number of qubits to implement our algorithm, each state $|\pi_0\rangle$ uses $\widetilde{O}(n)$ qubits. Thus, $\widetilde{O}(n^2)$ qubits are used to store all states. And we need $O(n)$ ancilla qubits for the quantum walk unitaries. Therefore, our algorithm uses $\widetilde{O}(n^2)$ qubits in total. □

Algorithm 13 Quantum speedup for approximately convex optimization (Informal version)

```
1: procedure QSimAnnealing                                                  ▷ Theorem 3.1
2:     $N \leftarrow \widetilde{O}(n)$, $K \leftarrow \sqrt{n}\log(n/\epsilon)$
3:     Prepare $N$ (approximately) copies of $|\pi_0\rangle$, denoted as $|\widetilde{\pi}_0^{(1)}\rangle, \ldots, |\widetilde{\pi}_0^{(N)}\rangle$
4:     for $i \leftarrow 1, \ldots, K$ do
5:         Use $\{|\widetilde{\pi}_0^{(j)}\rangle\}_{j\in[N]}$ to nondestructively construct $\Sigma_i$          ▷ Lemma 3.9
6:         Apply quantum walk to evolve the states $|\widetilde{\pi}_{i-1}^{(j)}\rangle$ to $|\widetilde{\pi}_i^{(j)}\rangle$     ▷ Theorem 3.8
7:     end for
8:     $x_K^j \leftarrow$ measure the final state $|\widetilde{\pi}_K^{(j)}\rangle$ for $j \in [N]$
9:     return $\arg\min_{j\in[N]} F(x_K^j)$
10: end procedure
```

Optimization of approximately convex functions with decreasing fluctuations. Beyond the ℓ_∞-norm assumption for all $x \in \mathrm{K}$ in Eq. (3.1), it is also possible to give efficient algorithms for optimizing other types of approximately convex functions. Specifically, Belloni et al. [BLNR15, Section 7] studied approximately convex functions with decreasing fluctuations.

Suppose that the function f in (3.1) is 1-Lipschtiz and α-strongly convex with minimum at $x_{\min} \in \mathrm{K}$:

$$f(x) - f(x_{\min}) \geq \langle \nabla f(x_{\min}), x - x_{\min}\rangle + \tfrac{\alpha}{2}\|x - x_{\min}\|^2 \geq \tfrac{\alpha}{2}\|x - x_{\min}\|^2.$$

Define a measure of "non-convexity" of F w.r.t to f in an n-dimensional ball of radius r near $x_{\min}$:

$$\Delta(r) := \sup_{x\in\mathcal{B}_2(x_{\min},r)} |F(x) - f(x)|.$$

We can call Theorem 3.1 iteratively. Suppose that at the start of the t^{th} iteration we have a ball $\mathcal{B}_2(x_{t-1}, 2r_{t-1})$ satisfying

$$\mathcal{B}_2(x_{\min}, r_{t-1}) \subset \mathcal{B}_2(x_{t-1}, 2r_{t-1}) \subset \mathcal{B}_2(x_{\min}, 3r_{t-1}).$$

After executing Theorem 3.1 in this iteration with $\widetilde{O}(n^3)$ quantum queries, we reach a point x_t such that with high probability $f(x_t) - f(x_{\min}) \leq Cn\Delta(3r_{t-1}))$ for some global constant $C > 0$. Due to strong convexity, this gives a new radius r_t recursively:

$$\frac{\alpha}{2Cn} r_t^2 := \frac{\alpha}{2Cn} \|x_t - x_{\min}\|^2 \leq \Delta(3r_{t-1}).$$

When $\Delta(r) = cr^p$ for some $c > 0, p \in (0, 2)$, the iteration stops when $r^* = (2 \cdot 3^p cCn/\alpha)^{1/(2-p)}$, and when $\Delta(r) = c\log(1 + dr)$ for some $c, d > 0$, the iteration stops at r^* satisfying

$$(2cCn/\alpha)\log(1 + 3dr^*) = (r^*)^2.$$

The total number of quantum queries is still $\widetilde{O}(n^3)$.

3.4 Quantum Algorithm for Zeroth-Order Stochastic Convex Bandits

We first prove Corollary 3.2 using quantum mean estimation with a Gaussian tail:

Proposition 3.10 (Adapted from [Ham21, Theorem 4.2])**.** *Suppose that X is a random variable on a probability space (Ω, p) with mean μ and variance σ^2. Suppose we have a unitary oracle U satisfying $U|0\rangle = \int_{x\in\Omega} \sqrt{p(x)}|x\rangle \mathrm{d}x$. Then, for any $\Delta \in (0, 1)$ and $\tau \in \mathbb{N}$ such that $\tau \geq \log(1/\Delta)$, there is a quantum algorithm that outputs a mean estimate $\tilde{\mu}$ such that*

$$\Pr[|\tilde{\mu} - \mu| > \frac{\sigma \log(1/\Delta)}{\tau}] \leq \Delta,$$

using $O(\tau \log^{3/2}(\tau) \log\log(\tau))$ queries to U.

Proof of Corollary 3.2. We follow Section 6 of [BLNR15] while use Theorem 3.1 and Proposition 3.10. Specifically, for a parameter $0 < \alpha < 1$, we let $\mathcal{N}_\alpha$ be a box grid of K with side length α. In other words, $\mathcal{N}_\alpha$ is α-net of K in ℓ_∞ norm. Since $\mathrm{K} \subseteq \mathcal{B}_2(0, R)$, $|\mathcal{N}_\alpha| \leq (R/\alpha)^n$.

Note that for the sub-Gaussian random variable ϵ_x in (3.2), it has variance at most $4\sigma^2$ because

$$\mathbb{E}[|\epsilon_x|^2] = \int_0^\infty \Pr[|\epsilon_x| > \sqrt{s}]\mathrm{d}s \leq 2\int_0^\infty e^{-\frac{s}{2\sigma^2}}\mathrm{d}s = 4\sigma^2.$$

Upon a query $x' \in \mathrm{K}$, we define an oracle $O_f^{\tau,\alpha}$ which returns $f(x) + \tilde{\epsilon}_x$ for $x \in \mathcal{N}_\alpha$ which is closest to x', and the $\tilde{\epsilon}_x$ here is obtained by applying Proposition 3.10 with the unitary oracle O_f in Eq. (3.4) to estimate the function value $f(x)$. Specifically, with $\Delta = \exp(-t^2)$ where t is a parameter determined later, we have

$$\Pr[|\tilde{\epsilon}_x| > \frac{\sigma t^2}{\tau}] \leq \exp(-t^2). \tag{3.8}$$

We note that in our algorithm based on the hit-and-run walk, with probability 1 we do not revisit the same point. As a result, $O_f^{\tau,\alpha}$ is no more powerful than O_f since the learner only obtains information on $\mathcal{N}_\alpha$, and in the rest of the proof we assume $O_f^{\tau,\alpha}$ as the oracle we use. We take

$$\alpha = \epsilon/2nL, \quad t = \sqrt{n\ln(R/\alpha) + \ln 10}.$$

Note that the value of t promises that $\exp(-t^2)(R/\alpha)^n \leq 0.1$. In other words, with probability at least 0.9, we promise that

$$\max_{x\in\mathcal{N}_\alpha} |\tilde{\epsilon}_x| \leq \frac{\sigma t^2}{\tau} = \frac{\sigma(n\ln(R/\alpha) + \ln 10)}{\tau}. \tag{3.9}$$

Finally, we take τ such that the RHS of (3.9) equals to $\epsilon/2n$, which is equivalent to

$$\tau = \frac{2n\sigma(n\ln(R/\alpha) + \ln 10)}{\epsilon} = \tilde{O}(n^2/\epsilon).$$

This will finally promise that

$$\sup_{x\in\mathrm{K}} |F(x) - f(x)| \leq \max_{x\in\mathcal{N}_\alpha} |\tilde{\epsilon}_x| + \alpha L \leq \frac{\epsilon}{2n} + \frac{\epsilon}{2n} = \frac{\epsilon}{n}, \tag{3.10}$$

meeting the condition of Theorem 3.1. Consequently, with probability at least $0.9 \cdot 0.9 > 0.8$, we can find an $x^* \in \mathrm{K}$ such that $f(x^*) - \min_{x\in\mathrm{K}} f(x) \leq \epsilon$ using $\tilde{O}(n^3) \cdot \tau = \tilde{O}(n^5/\epsilon)$ queries to the quantum stochastic evaluation oracle (3.4). □

Algorithm 14 Quantum zeroth-order stochastic convex bandits

```
1: procedure QBandits($T$)
2:     $m \leftarrow \lfloor \log_2 T \rfloor$, $K \leftarrow \lfloor \log_2(TR) \rfloor$
3:     $x_1 \leftarrow 0$
4:     for $i \leftarrow 1, 2, \ldots, m+1$ do
5:         $t \leftarrow 2^{i-1}$
6:         if $t \leq m$ then
7:             $L \leftarrow 2^{i-1}$                          ▷ $T_i := \{2^{i-1}, 2^{i-1}+1, \ldots, 2^i - 1\}$
8:         else
9:             $L \leftarrow T - 2^m + 1$                     ▷ $T_{m+1} := \{2^m, 2^m+1, \ldots, T\}$
10:        end if
11:        for $j \leftarrow 1, 2, \ldots, K$ do
12:            $y_j \leftarrow$ QMinStocConv($\mathcal{O}_f^{\mathcal{T},\alpha}, L/K$)▷ Corollary 3.2 with $|T_i|/K$ queries
13:            for $l \leftarrow 0, 1, \ldots, L/K$ do
14:                $x_{t+l} \leftarrow x_{2^{i-1}}$               ▷ Onliner learner's output at time $t+l$
15:            end for
16:            if $f(y_j) < f(x_{2^i})$ then
17:                $x_{2^i} \leftarrow y_j$
18:            end if
19:        end for
20:    end for
21: end procedure
```

Proof of Theorem 3.3. We prove that Algorithm 14 satisfies Theorem 3.3.

Intuitively, we divide the T rounds into $m+1$ intervals where $m \leftarrow \lfloor \log_2 T \rfloor$, such that $[T] = \bigcup_{i=1}^{m+1} \mathcal{T}_i$ and $\mathcal{T}_i := \{2^{i-1}, 2^{i-1}+1, \ldots, 2^i - 1\}$ for each $i \in [m]$. When executing in the interval $\mathcal{T}_i$, the output required by the online learner is always $x_t = x_{2^{i-1}}$, the x at the end of the last interval. On the other hand, the queries in the current interval are applied to running the quantum stochastic convex optimization algorithm in Corollary 3.2 and output a nearly-optimal solution with probability at least $1 - O(1/T)$. With $|\mathcal{T}_i| = 2^{i-1}$ queries at hand, we divide them into $\log(TR)$ repeats of Corollary 3.2, each using $2^{i-1}/\log(TR)$ queries in the quantum algorithm. As a result, each repeat $j \in [\log(TR)]$ outputs a value $\tilde{x}_{2^i,j}$ such that

$$f(\tilde{x}_{2^i,j}) - \min_{x \in \mathcal{K}} f(x) \leq \tilde{O}(n^5 \log(TR)/2^{i-1})$$

with probability at least 0.8. We take $x_{2^i} := \arg\min_{j\in[\log T]} f(\tilde{x}_{2^i,j})$. With probability at least $1 - 0.8^{\log(TR)} = 1 - O(1/TR)$, we have

$$f(x_{2^i}) - \min_{x\in\mathrm{K}} f(x) \le \tilde{O}(n^5 \log(TR)/2^i). \tag{3.11}$$

Going through all $i \in [m+1]$ intervals, by the union bound, with probability at least

$$1 - (m+1)\cdot O\Big(\frac{1}{TR}\Big) = 1 - O\Big(\frac{\log T}{TR}\Big),$$

we have

$$f(x_{2^i}) - \min_{x\in\mathrm{K}} f(x) \le \tilde{O}(n^5 \log(TR)/2^{i-1}) \quad \forall i \in [m+1]. \tag{3.12}$$

In all, we get that the regret bound as desired:

$$\begin{aligned}\mathcal{R}_T &= \mathbb{E}\left[\sum_{t=1}^{T}(f(x_t) - f^*)\right] \\ &\le \left(1 - O\Big(\frac{\log T}{TR}\Big)\right)\cdot \sum_{i=1}^{m+1} 2^{i-1}\cdot \tilde{O}\Big(\frac{n^5 \log TR}{2^{i-1}}\Big) + O\Big(\frac{\log T}{TR}\Big)\cdot T \cdot LR \\ &= \tilde{O}(n^5 \log(T)\log(TR)) + O(L\log T) \\ &= \tilde{O}(n^5 \log(T)\log(TR)).\end{aligned}$$

□

3.5 Classical Approach for Optimizing Approximately Convex Functions

In this section, we introduce the classical approach [BLNR15] for the optimization of approximately convex functions as in Eq. (3.1).

3.5.1 Low level: Hit-and-Run for approximate log-concave distributions

The Hit-and-Run walk uses a unidimensional rejection sampler to sample a point from the distribution π_g restricted to a line ℓ. The following lemma shows the performance guarantee of the unidimensional sampler:

Algorithm 15 Hit-and-Run walk

```
1: procedure HitAndRun($\pi_0$, $\pi_g$, $\Sigma$, $m$)    ▷ $\pi_g$ is the target distribution on $\mathcal{K}$ induced by a nonnegative function $g$, $\Sigma$ is a linear transformation
2:     $\mathbf{x}_0 \leftarrow$ sample from $\pi_0$
3:     Choose accuracy parameter $\epsilon_\ell$
4:     for $i \leftarrow 1, \dots, m$ do
5:         $\mathbf{u} \leftarrow$ uniformly sample from the surface of ellipse given by $\Sigma$ acting on sphere
6:         $\ell(t) := \mathbf{x}_{i-1} + t\mathbf{u}$, compute $[\mathbf{s}, \mathbf{t}] \leftarrow \ell \cap \mathcal{K}$
7:         $\mathbf{x}_i \leftarrow$ UniSampler($g$, $\beta$, $[\mathbf{s}, \mathbf{t}]$, $\epsilon_\ell$)
8:     end for
9:     return $\mathbf{x}_m$
10: end procedure
```

Lemma 3.11 (Unidimensional rejection sampler, [BLNR15, Lemma 5])**.** *Given $\beta = O(1)$. Let g be a β-log-concave function and ℓ be a bounded line segment on $\mathcal{K}$. For $\epsilon \in (0, e^{-2\beta}/2)$, Algorithm 16 outputs a point $\mathbf{x} \in \ell$ with a distribution $\widetilde{\pi}_\ell$ such that*

$$d_{\mathrm{TV}}(\widetilde{\pi}_\ell, \pi_g|_\ell) \leq 3e^{2\beta}\epsilon.$$

Moreover, the algorithm requires $\widetilde{O}(1)$ evaluations of the function g.

The following theorem gives the mixing time of the standard Hit-and-Run walk for an approximate log-concave distribution, where we assume that in each step we directly sample from the restricted distribution $\pi_g|_\ell$.

Theorem 3.12 (Mixing time of Hit-and-Run for approximate log-concave distribution, [BLNR15, Theorem 4])**.** *Let π_g be the stationary measure associated with the Hit-and-Run walk based on a $\beta/2$-approximately log-concave function g, and let $\sigma^{(0)}$ be an initial distribution with ℓ_2-warmness $M := \|\sigma^{(0)}/\pi_g\|$. There is a universal constant C such that for any $\gamma \in (0, 1/2)$, if*

$$m \geq Cn^2 \frac{e^{6\beta}R^2}{r^2} \log^4\left(\frac{e^\beta MnR}{r\gamma^2}\right) \log\left(\frac{M}{\gamma}\right),$$

then m steps of the Hit-and-Run random walk based on g yield

$$d_{\mathrm{TV}}(\sigma^{(m)}, \pi_g) \leq \gamma.$$

Algorithm 16 Unidimensional rejection sampler

```
1: procedure InitP($g$, $\beta$, $\ell = [\mathbf{s}, \mathbf{t}]$)
2:     while true do
3:         $\mathbf{x}_1 \leftarrow \frac{3}{4}\mathbf{s} + \frac{1}{4}\mathbf{t}$, $\mathbf{x}_2 \leftarrow \frac{1}{2}\mathbf{s} + \frac{1}{2}\mathbf{t}$, $\mathbf{x}_3 \leftarrow \frac{1}{4}\mathbf{s} + \frac{3}{4}\mathbf{t}$
4:         if $|\log(g(\mathbf{x}_1)) - \log(g(\mathbf{x}_3))| > \beta$ then
5:             $\mathbf{t} \leftarrow \mathbf{x}_3$ if $g(\mathbf{x}_1) > g(\mathbf{x}_3)$; $\mathbf{s} \leftarrow \mathbf{x}_1$ otherwise
6:         else if $|\log(g(\mathbf{x}_1)) - \log(g(\mathbf{x}_2))| > \beta$ then
7:             $\mathbf{t} \leftarrow \mathbf{x}_2$ if $g(\mathbf{x}_1) > g(\mathbf{x}_2)$; $\mathbf{s} \leftarrow \mathbf{x}_1$ otherwise
8:         else if $|\log(g(\mathbf{x}_2)) - \log(g(\mathbf{x}_3))| > \beta$ then
9:             $\mathbf{t} \leftarrow \mathbf{x}_3$ if $g(\mathbf{x}_2) > g(\mathbf{x}_3)$; $\mathbf{s} \leftarrow \mathbf{x}_2$ otherwise
10:        else
11:            return $\mathbf{p} \leftarrow \arg\max_{\mathbf{x} \in \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}} g(\mathbf{x})$
12:        end if
13:    end while
14: end procedure
15: procedure BinSearch($g$,$\mathbf{x}_l$, $\mathbf{x}_r$, $V_l$, $V_r$)
16:    while true do
17:        $\mathbf{x}_m \leftarrow (\mathbf{x}_l + \mathbf{x}_r)/2$
18:        if $g(\mathbf{x}_m) > V_r$ then
19:            $\mathbf{x}_r \leftarrow \mathbf{x}_m$
20:        else if $g(\mathbf{x}_m) < V_l$ then
21:            $\mathbf{x}_l \leftarrow \mathbf{x}_m$
22:        else
23:            return $\mathbf{x}_m$
24:        end if
25:    end while
26: end procedure
27: procedure InitE($g$, $\beta$, $\ell = [\mathbf{s}, \mathbf{t}]$, $\mathbf{p}$, $\epsilon_\ell$)
28:    if $g(\mathbf{s}) \geq \frac{1}{2}e^{-\beta}\epsilon_\ell g(\mathbf{p})$ then
29:        $\mathbf{e}_0 \leftarrow \mathbf{s}$
30:    else
31:        $\mathbf{e}_0 \leftarrow$ BinSearch$(g, \mathbf{s}, \mathbf{p}, \frac{1}{2}e^{-\beta}\epsilon_\ell g(\mathbf{p}), \epsilon_\ell g(\mathbf{p}))$
32:    end if
33:    if $g(\mathbf{t}) \geq \frac{1}{2}e^{-\beta}\epsilon_\ell g(\mathbf{p})$ then
34:        $\mathbf{e}_1 \leftarrow \mathbf{t}$
35:    else
36:        $\mathbf{e}_1 \leftarrow$ BinSearch$(g, \mathbf{p}, \mathbf{t}, \frac{1}{2}e^{-\beta}\epsilon_\ell g(\mathbf{p}), \epsilon_\ell g(\mathbf{p}))$
37:    end if
38:    return $\mathbf{e}_0, \mathbf{e}_1$
39: end procedure
```

Algorithm 17 Unidimensional rejection sampler (Continue)

```
1: procedure UniSampler($g$, $\beta$, $\ell = [\mathbf{s}, \mathbf{t}]$, $\epsilon_\ell$)
2:     $\mathbf{p} \leftarrow$ InitP($g$, $\beta$, $\ell$)
3:     $\mathbf{e}_0, \mathbf{e}_1 \leftarrow$ InitE($g$, $\beta$, $\ell$, $\mathbf{p}$, $\epsilon_\ell$)
4:     while true do
5:         $\mathbf{x} \leftarrow \mathrm{Uniform}([\mathbf{e}_0, \mathbf{e}_1])$, $r \leftarrow \mathrm{Uniform}([0, 1])$
6:         if $r \le g(\mathbf{x})/(e^{3\beta} g(\mathbf{p}))$ then
7:             return $\mathbf{x}$
8:         end if
9:     end while
10: end procedure
```

The next theorem shows the closeness between the output distribution of Algorithm 15 and the target distribution π_g. Due to the unidimensional rejection sampler (Algorithm 16), the stationary distribution of Algorithm 15 may not be exactly π_g. Nevertheless, we can still show that it will not deviate a lot.

Theorem 3.13 (The effect of the rejection sampler, [BLNR15, Theorem 5])**.** *Let π_g, $\sigma^{(0)}$ be defined as in Theorem 3.12. Let $\hat{\sigma}^{(m)}$ denote the output distribution of Algorithm 15 with initial distribution $\hat{\sigma}^{(0)}$ in m steps. Let ϵ_ℓ be the accuracy parameter for the unidimensional rejection sampler (Algorithm 16). Then, we have*

$$d_{\mathrm{TV}}(\hat{\sigma}^{(m)}, \sigma^{(m)}) \le m\epsilon_\ell + 2 d_{\mathrm{TV}}(\hat{\sigma}^{(0)}, \sigma^{(0)}).$$

In particular, for $\gamma \in (0, 1/e)$, suppose $d_{\mathrm{TV}}(\hat{\sigma}^{(0)}, \sigma^{(0)}) \le \gamma/8$. Let $s \in (0, 1)$ be such that $H_s \le \gamma/4$, where H_s is defined to be:

$$H_s := \sup_{A \subset \mathcal{K} : \pi_g(A) \le s} |\pi_g(A) - \sigma^{(0)}(A)|.$$

Then, there is a constant C' such that, if we take $\epsilon_\ell := \gamma e^{-2\beta}/(12m)$ and

$$m \ge C' n^2 \frac{e^{6\beta} R^2}{r^2} \log^4\left(\frac{e^{\beta} n R}{rs}\right) \log(1/s),$$

we have

$$d_{\mathrm{TV}}(\hat{\sigma}^{(m)}, \pi_g) \le \gamma.$$

3.5.2 Mid level: rounding into isotropic position

The following lemma rounds a β-log-concave distribution to near-isotropic position.

Lemma 3.14 (Rounding β-log-concave distribution, [BLNR15, Lemma 9])**.** *Let π be a β-log-concave distribution in $\mathbb{R}^n$. By taking $N = \Theta(n \log n)$ i.i.d. samples $\mathbf{x}_1, \ldots, \mathbf{x}_n$ from π, we have*

$$\frac{1}{2} \leq \sigma_{\min}\Big(\frac{1}{N}\sum_{i\in[N]} \mathbf{x}_i \mathbf{x}_i^\top\Big) \leq \sigma_{\max}\Big(\frac{1}{N}\sum_{i\in[N]} \mathbf{x}_i \mathbf{x}_i^\top\Big) \leq \frac{3}{2}$$

holds with probability at least $1 - n^{-O(1)}$.

3.5.3 High level: simulated annealing

At high level, we run a simulated annealing for a series of functions:

$$h_i(x) := \exp(-f(x)/T_i), \quad \text{and} \quad g_i(x) := \exp(-F(x)/T_i),$$

where f, F satisfy Eq. (3.1) and $\{T_i\}_{i\in[K]}$ are parameters to be chosen later.

Algorithm 18 Simulated annealing

```
1:  procedure SimAnnealing($K$, $\{T_i\}_{i\in[K]}$)
2:      $N \leftarrow \Theta(n \log n)$                              ▷ The number of strands
3:      $X_0^j \sim \mathrm{Uniform}(\mathcal{K})$ for $j = 1, \ldots, N$
4:      $\mathcal{K}_0 \leftarrow \mathcal{K}$, $\Sigma_0 \leftarrow I$
5:      $m \leftarrow \widetilde{O}(n^3)$                                  ▷ Theorem 3.13
6:      for $i \leftarrow 1, \ldots, K$ do
7:          $\Sigma_i' \leftarrow$ the rounding linear transformation for $\{X_{i-1}^j\}_{j\in[N]}$
8:          $\Sigma_i \leftarrow \Sigma_i' \circ \Sigma_{i-1}$
9:          for $j \leftarrow 1, \ldots, N$ do
10:             $X_i^j \leftarrow$ HitAndRun($X_{i-1}^j, \pi_{g_i}, \Sigma_i, m$)       ▷ Algorithm 15
11:         end for
12:     end for
13:     return $\arg\min_{i\in[K], j\in[N]} F(X_i^j)$
14: end procedure
```

Lemma 3.15 (The warmness of annealing distributions, [BLNR15, Lemma 8])**.** *Let $g(x) = exp(-F(x))$ be a β-log-concave function. Let π_{g_i} be a distribution with density proportional to $g_i(x) = \exp(-F(x)/T_i)$, supported on $\mathcal{K}$. Let $T_i := T_{i-1}\Big(1 - \frac{1}{\sqrt{n}}\Big)$. Then,*

$$\|\pi_{g_i}/\pi_{g_{i+1}}\| \le C_\gamma = 5\exp(2\beta/T_i).$$

Theorem 3.16 (Sample Guarantee for the simulated annealing, [BLNR15, Theorem 6])**.** *Fix a target accuracy $\gamma \in (0, 1/e)$ and let g be an $\beta/2$-approximately log-concave function in $\mathbb{R}^n$. Suppose the simulated annealing algorithm (Algorithm 18) is run for $K = \sqrt{n}\log(1/\rho)$ epochs with temperature parameters $T_i = (1-1/\sqrt{n})^i$ for $0 \le i \le K$. If the Hit-and-Run with the unidimensional sampling scheme (Algorithm 15) is run for $m = \widetilde{O}(n^3)$ number of steps prescribed in Theorem 3.13, the algorithm maintains that*

$$d_{\mathrm{TV}}(\hat{\sigma}_i^{(m)}, \pi_{g_i}) \le e\gamma$$

for each $i \in [K]$, where $\hat{\sigma}_i^{(m)}$ is the distribution of the m-th step of Hit-and-Run. Here, m depends polylogarithmically on $1/\rho$.

Then, we have the following optimization guarantee for the simulated annealing procedure:

Theorem 3.17 (Optimization guarantee for the simulated annealing, [BLNR15, Corollary 1])**.** *Suppose F is approximately convex and $|F - f| \le \epsilon/n$ as in Eq. (3.1). The simulated annealing method with $K = \sqrt{n}\log(n/\epsilon)$ epochs produces a random point X such that*

$$\mathbb{E}[f(X)] - \min_{\mathbf{x}\in\mathcal{K}} f(\mathbf{x}) \le \epsilon,$$

and thus,

$$\mathbb{E}[F(X)] - \min_{\mathbf{x}\in\mathcal{K}} F(\mathbf{x}) \le 2\epsilon.$$

Furthermore, the number of oracle queries required by the method is $\widetilde{O}(n^{4.5})$.

3.6 Quantum Speedup for Optimizing Approximately Convex Functions

As we discussed in previous section, there are three levels for the optimization algorithm. The goal of this section is to prove Theorem 3.1, where we improve the classical query complexity $\widetilde{O}(n^{4.5})$ (Theorem 3.17) to quantum query complexity $\widetilde{O}(n^3)$. The main idea is to use quantum walk algorithm (introduced in Section 2.8) to speed-up the low level such that each sample can be generated with less queries.

3.6.1 Quantum speedup for low-level

In this section, we show how to use the quantum walk algorithm to speedup the sampling procedure in the simulated annealing process. According to the framework (Corollary 2.26), we first show that the each Markov chain's stationary distribution in the annealing process is a warm-start for its adjacent chains, and the Markov chains are slowly-varying. Then, we show how to implement the quantum walk operator for the Hit-and-Run walk. Finally, we prove the quantum speedup from $\widetilde{O}(n^3)$ classical query complexity to $\widetilde{O}(n^{1.5})$ quantum query complexity.

Warmness and overlap for the stationary distributions. We first show that π_{g_i} is a warm-start for $\pi_{g_{i+1}}$, and vice versa.

By lemma 3.15, we know that $\|\pi_{g_i}/\pi_{g_{i+1}}\| \leq 5\exp(2\beta/T_i)$. Similarly, we can also bound $\|\pi_{g_{i+1}}/\pi_{g_i}\|$:

Lemma 3.18. *Let $g(x) = exp(-F(x))$ be a β-log-concave function. Let π_{g_i} be a distribution with density proportional to $g_i(x) = \exp(-F(x)/T_i)$, supported on $\mathcal{K}$. Let $T_i := T_{i-1}\Big(1 - \frac{1}{\sqrt{n}}\Big)$. Then,*

$$\|\pi_{g_{i+1}}/\pi_{g_i}\| \leq 8\exp(2\beta/T_{i+1}).$$

Proof. Define $Y(a) := \int_{\mathcal{K}} \exp(-F(x)a)\mathrm{d}x$. Then, we have

$$\begin{aligned}\|\pi_{g_{i+1}}/\pi_{g_i}\| &= \frac{\int_{\mathcal{K}} \exp(-F(x)(2/T_{i+1} - 1/T_i))\mathrm{d}x \cdot \int_{\mathcal{K}} \exp(-F(x)/T_i)\mathrm{d}x}{\left(\int_{\mathcal{K}} \exp(-F(x)/T_{i+1})\mathrm{d}x\right)^2} \\ &= \frac{Y(2/T_{i+1} - 1/T_i)Y(1/T_i)}{Y(1/T_{i+1})^2}.\end{aligned}$$

Define $G(x,t) := g(x/t)^t$. Then, we have

$$\begin{aligned}G(\lambda x + (1-\lambda)x', \lambda t + (1-\lambda)t') &= g\Big(\frac{\lambda x + (1-\lambda)x'}{\lambda t + (1-\lambda)t'}\Big)^{\lambda t + (1-\lambda)t'} \\ &= g\Big(\frac{\lambda t}{\lambda t + (1-\lambda)t'}\frac{x}{t} + \frac{(1-\lambda)t'}{\lambda t + (1-\lambda)t'}\frac{x'}{t'}\Big)^{\lambda t + (1-\lambda)t'} \\ &\geq \exp(-\beta(\lambda t + (1-\lambda)t')) \cdot g\Big(\frac{x}{t}\Big)^{\lambda t} \cdot g\Big(\frac{x'}{t'}\Big)^{(1-\lambda)t'} \\ &= \exp(-\beta(\lambda t + (1-\lambda)t')) \cdot G(x,t)^{\lambda} \cdot G(x',t')^{1-\lambda} \\ &= (\exp(-\beta t)G(x,t))^{\lambda} \cdot (\exp(-\beta t')G(x',t'))^{1-\lambda},\end{aligned}$$

where the inequality follows from g is β-log-concave.

By Prékopa–Leindler inequality (Theorem 3.19), it implies that

$$\int_{\mathcal{K}} G(x, \lambda t + (1-\lambda)t')\mathrm{d}x \geq \Big(\int_{\mathcal{K}} \exp(-\beta t)G(x,t)\mathrm{d}x\Big)^{\lambda} \cdot \Big(\int_{\mathcal{K}} \exp(-\beta t')G(x,t')\mathrm{d}x\Big)^{1-\lambda}.$$

Note that

$$\int_{\mathcal{K}} G(x,t)\mathrm{d}x = \int_{\mathcal{K}} g\Big(\frac{x}{t}\Big)^t \mathrm{d}x = t^n \int_{\mathcal{K}} g(x)^t \mathrm{d}x = t^n \int_{\mathcal{K}} \exp(-F(x)t)\mathrm{d}x = t^n Y(t).$$

Hence, for $\lambda = \frac{1}{2}$, we have

$$\Big(\frac{t+t'}{2}\Big)^{2n} Y\Big(\frac{t+t'}{2}\Big)^2 \geq \exp(-\beta(t+t')/2) \cdot t^n Y(t) \cdot t'^n Y(t'),$$

which implies that

$$\frac{Y(t)Y(t')}{Y(\frac{t+t'}{2})^2} \leq \exp\Big(\frac{\beta(t+t')}{2}\Big) \cdot \Big(\frac{(t+t')^2/4}{tt'}\Big)^n. \tag{3.13}$$

By taking $t = 2/T_{i+1} - 1/T_i$ and $t' = 1/T_i$, we have

$$\begin{aligned}
\|\pi_{g_{i+1}}/\pi_{g_i}\| \leq & \frac{Y(2/T_{i+1}-1/T_i)Y(1/T_i)}{Y(1/T_{i+1})^2} \\
\leq & \exp(2\beta/T_{i+1}) \cdot \Big(\frac{(1/T_{i+1})^2}{(2/T_{i+1}-1/T_i)(1/T_i)}\Big)^n \\
= & \exp(2\beta/T_{i+1}) \cdot \Big(\frac{1}{(2-(1-1/\sqrt{n}))(1-1/\sqrt{n})}\Big)^n \\
= & \exp(2\beta/T_{i+1}) \cdot \Big(1+\frac{1}{n-1}\Big)^n \\
\leq & \exp(2\beta/T_{i+1}) \cdot \exp(n/(n-1)) \\
\leq & 8\exp(2\beta/T_{i+1}),
\end{aligned}$$

where the third step follows from $T_{i+1} = T_i(1-\frac{1}{\sqrt{n}})$.

The lemma is then proved. $\square$

Remark 3.1. Since we assume that $|F(x) - f(x)| \leq \epsilon/n$ in Eq. (3.1), i.e., $\beta = \epsilon/n$, by Lemmas 3.15 and 3.18, we know that the warmness $M := \max\{\|\pi_{g+i}/\pi_{g_{i+1}}\|, \|\pi_{g_{i+1}}/\pi_{g_i}\|\}$ can be bounded by $O(\exp(2\epsilon/(nT_{i+1})))$. Since we choose the final temperature $T_k = \epsilon/n$, we get that $M = O(1)$. Therefore, it satisfies the warmness condition in Corollary 2.26.

Theorem 3.19 (Prékopa–Leindler inequality, [Pré71, Pré73])**.** *Let $0 < \lambda < 1$ and let $f, g, h : \mathbb{R}^n \to [0,\infty)$ be measurable functions. Suppose that these functions satisfy*

$$h(\lambda x + (1-\lambda)y) \geq f(x)^\lambda \cdot g(y)^{1-\lambda} \quad \forall x, y \in \mathbb{R}^n.$$

Then, we have

$$\int_{\mathbb{R}^n} h(x)\mathrm{d}x \geq \Big(\int_{\mathbb{R}^n} f(x)\mathrm{d}x\Big)^\lambda \cdot \Big(\int_{\mathbb{R}^n} g(x)\mathrm{d}x\Big)^{1-\lambda}.$$

Lemma 3.20 (Bound distribution overlap)**.** *Let $g(x) = \exp(-F(x))$ be a β-log-concave function. Let π_{g_i} be a distribution with density proportional to $g_i(x) = \exp(-F(x)/T_i)$, supported on $\mathcal{K}$. Let $T_i := T_{i-1}\Big(1-\frac{1}{\sqrt{n}}\Big)$. Then,*

$$\langle \pi_i | \pi_{i+1} \rangle \geq \exp(-(\beta/T_{i+1}+1)/2).$$

Proof. We can write the overlap as follows:

$$\begin{aligned}
\langle \pi_{g_i} | \pi_{g_{i+1}} \rangle &= \frac{\int_{\mathcal{K}} \sqrt{g_i(x) g_{i+1}(x)} \mathrm{d}x}{(\int_{\mathcal{K}} g_i(x) \mathrm{d}x)^{1/2} \cdot (\int_{\mathcal{K}} g_{i+1}(x) \mathrm{d}x)^{1/2}} \\
&= \frac{\int_{\mathcal{K}} \exp(-F(x)(1/T_i + 1/T_{i+1})/2) \mathrm{d}x}{(\int_{\mathcal{K}} \exp(-F(x)/T_i) \mathrm{d}x)^{1/2} \cdot (\int_{\mathcal{K}} \exp(-F(x)/T_{i+1}) \mathrm{d}x)^{1/2}} \\
&= \frac{Y((1/T_i + 1/T_{i+1})/2)}{Y(1/T_i)^{1/2} Y(1/T_{i+1})^{1/2}},
\end{aligned}$$

where $Y(t) := \int_{\mathcal{K}} \exp(-F(x)t) \mathrm{d}x$.

By Eq. (3.13), we have

$$\begin{aligned}
\frac{Y(1/T_i) Y(1/T_{T_{i+1}})}{Y((1/T_i + 1/T_{i+1})/2)^2} &\leq \exp(\beta(1/T_i + 1/T_{i+1})/2) \cdot \Big(\frac{(1/T_i + 1/T_{i+1})^2/4}{1/(T_i T_{i+1})} \Big)^n \\
&= \exp(\beta(2 - 1/\sqrt{n})/(2T_{i+1})) \cdot \Big(1 + \frac{1}{4(n - \sqrt{n})} \Big)^n \\
&\leq \exp(\frac{1}{4} \frac{\sqrt{n}}{\sqrt{n} - 1}) \cdot \exp(\beta / T_{i+1}) \\
&\leq \exp(\beta / T_{i+1} + 1),
\end{aligned}$$

where the second step follows from $T_{i+1} = T_i(1 - 1/\sqrt{n})$.

Therefore,

$$\langle \pi_{g_i} | \pi_{g_{i+1}} \rangle \geq \exp(-(\beta / T_{i+1} + 1)/2).$$

□

Remark 3.2. By taking $\beta = \epsilon/n$ and $T_i \geq \epsilon/n$ in Lemma 3.20, we have for any $i \in [K-1]$, the overlap can be upper-bounded by:

$$\langle \pi_{g_i} | \pi_{g_{i+1}} \rangle \geq \exp(-(\beta / T_K + 1)/2) = e^{-1}.$$

Implementing the quantum walk operator. We introduce how to implement the quantum walk update operator U such that:

$$U|x\rangle|0\rangle = \int_{\mathcal{K}} \sqrt{P_{x,y}} |x\rangle|y\rangle \mathrm{d}y,$$

where P is the stochastic transition matrix for the Hit-and-Run walk.

Given an input state $|x\rangle$. We first prepare an n-dimensional Gaussian state in an ancilla register:

$$|x\rangle|0\rangle \longrightarrow |x\rangle \int_{\mathbb{R}^n} (2\pi)^{-n/4}|z\rangle \mathrm{d}z.$$

Then, by normalizing z and applying the linear transformation Σ in another quantum register, we get that

$$|x\rangle \int_{\mathbb{R}^n} (2\pi)^{-n/4}|z\rangle \left|\frac{\Sigma z}{\|z\|}\right\rangle \mathrm{d}z.$$

If we un-compute the $|z\rangle$ register, we get that (ignoring the normalization factor):

$$|x\rangle \int_{\Sigma\mathbb{S}^n} |u\rangle \mathrm{d}u.$$

Next, we coherently compute the two end-points of $\ell \cap \mathcal{K}$ for $\ell(t) := x + ut$ in the ancilla registers:

$$\int_{\Sigma\mathbb{S}^n} \mathrm{d}u|x\rangle|u\rangle|0\rangle \longrightarrow \int_{\Sigma\mathbb{S}^n} \mathrm{d}u|x\rangle|u\rangle|s,t\rangle$$

We coherently simulate the unidimensional sampler (Algorithm 16). More specifically, we can compute the points p, e_0, e_1 in ancilla registers:

$$\int_{\Sigma\mathbb{S}^n} \mathrm{d}u|x\rangle|u\rangle|s,t,p,e_0,e_1\rangle$$

Then, we prepare two unifrom distribution states in the next two ancilla qubits:

$$\int_{\Sigma\mathbb{S}^n} \mathrm{d}u|x\rangle|u\rangle|s,t,p,e_0,e_1\rangle \int_{[0,1]^2} |r',r\rangle \mathrm{d}r'\mathrm{d}r$$

And the next proposed point y can be computed via $y := e_0 + r'(e_1 - e_0)$:

$$\int_{\Sigma\mathbb{S}^n} \mathrm{d}u|x\rangle|u\rangle|s,t,p,e_0,e_1\rangle \int_{[0,1]^2} |r',r\rangle \mathrm{d}r'\mathrm{d}r|y\rangle$$

Then, we check the condition $r \leq g(y)/(3^\beta g(p))$ by querying the evaluation oracle twice and use an ancilla qubit to indicate whether it is satisfied:

$$\int_{\Sigma\mathbb{S}^n}\int_{[0,1]^2} \mathrm{d}u\mathrm{d}r'\mathrm{d}r|x\rangle|u\rangle|s,t,p,e_0,e_1\rangle|r',r\rangle|y\rangle|b\rangle,$$

where $b \in \{0,1\}$. Then, we post-select[4] the last qubit for $b = 1$. By un-computing the registers for $u, s, t, p, e_0, e_1, r, r'$, we get the desired state:

$$|x\rangle \int_{\mathcal{K}} \sqrt{P_{x,y}} |y\rangle.$$

By Lemma 3.11, we get that this procedure (including the post-selection cost) takes $O(1)$ oracle queries with high probability. Therefore, we get the following lemma:

Lemma 3.21 (Implementation cost of the quantum walk update operator)**.** *For the Hit-and-Run walk (Algorithm 15) with the unidimensional sampler (Algorithm 16), the quantum walk update operator U can be implemented by querying the evaluation oracle $O(1)$ times.*

$\widetilde{O}(n^{1.5})$-query quantum algorithm. We have the following theorem:

Theorem 3.22 (Quantum speedup for the Hit-and-Run sampler)**.** *Let $\gamma \in (0, 1/e)$. Let π_g be the stationary measure associated with the Hit-and-Run walk based on a $\beta/2$-approximately log-concave function g. Let $T_i = (1 - 1/\sqrt{n})^i$ for $0 \le i \le K$ be the annealing schedule. Suppose we use the quantum walk to implement the Hit-and-Run walk (Algorithm 15). Then, for each $0 \le i \le K-1$, given a state $|\pi_{g_i}\rangle$, we can produce a state $|\hat{\sigma}_i^{(m)}\rangle$ such that*

$$\| |\pi_{g_{i+1}}\rangle - |\hat{\sigma}_i^{(m)}\rangle \|_2 \le O(\gamma),$$

using $m = \widetilde{O}(n^{1.5})$ calls for the evaluation oracle.

Proof. We use the quantum walk framework in Theorem 3.5.

For the warmness, by Remark 3.1, we know that in this annealing schedule, $\|\pi_{g_i}/\pi_{g_{i+1}}\|$ and $\|\pi_{g_{i+1}}/\pi_{g_i}\|$ are upper-bounded by some constants.

[4]We can measure the last qubit. If the measurement outcome is 0, we reinitialize the r, r' registers and re-preapre $|y\rangle$ and $|b\rangle$. We repeat this process until we measure $b = 1$.

Then, by Theorem 3.12, we get that the number of steps for evolving from π_{g_i} to $\pi_{g_{i+1}}$ and from $\pi_{g_{i+1}}$ to π_{g_i} is $\widetilde{O}(n^3)$ classically. The proof of Theorem 3.13 implies that the stationary distribution of the Hit-and-Run walk with unidimensional sampler is very close to the original Markov chain, only causing a constant blowup to the total variation distance. Thus, for $\gamma' = O(\gamma)$, we have $t_1(\gamma'), t_2(\gamma') = \widetilde{O}(n^3)$ in Theorem 3.5.

By Remark 3.2, we know that in this annealing schedule, the adjacent distributions have a big overlap. In particular, we have $|\langle \pi_{g_i}|\pi_{g_{i+1}}\rangle| \geq \Omega(1)$, satisfying the condition in Theorem 3.5.

Therefore, by Theorem 3.5, we get that the state $|\hat{\sigma}_i^{(m)}\rangle$ satisfying $\||\pi_{g_{i+1}}\rangle - |\hat{\sigma}_i^{(m)}\rangle\|_2 \leq O(\gamma)$ can be prepared using $\widetilde{O}(n^{1.5})$ calls to the controlled walk operators.

By Lemma 3.21, each call to the quantum walk operator can be implemented with $O(1)$ query to the evaluation oracle. Hence, the total query complexity is $\widetilde{O}(n^{1.5})$.

The theorem is then proved. $\square$

3.6.2 Non-destructive rounding in the mid-level

In the middle level, we need to compute the linear transformation Σ_i that rounds the β-logconcave distribution to near-isotropic position. Moreover, we are given access to N copies of the quantum states $|\pi_{g_i}\rangle$ and we will compute Σ_i in a non-destructive way.

Classically, by Lemma 3.14, we can take

$$\Sigma_i'(x^1, \ldots, x^N) := \frac{1}{N}\sum_{i=1}^{N} x^j i^{j\top},$$

where x_i^j is the j-th independent sample from π_{g_i}. Then, the linear transformation in the i-th iteration is Σ_i composite with the linear transformation in the $(i-1)$-th iteration, i.e.,

$$\Sigma_i(x_1, \ldots, x_n) := \Sigma_i'(x_1, \ldots, x_N) \cdot \Sigma_{i-1}.$$

In quantum, we can use a quantum circuit to simulate the classically computation for Σ_i' and Σ_i coherently, which computes the following superposition state:

$$\int_{\mathcal{K}} \mathrm{d}x^1 \cdots \int_{\mathcal{K}} \mathrm{d}x^N \prod_{j=1}^{N} \sqrt{\pi_{g_i}(x^j)} \cdot \left|x^1\right\rangle \cdots \left|x^N\right\rangle |\Sigma_i(x_1, \ldots, x^n)\rangle. \tag{3.14}$$

That is, the first N quantum registers contain N copies of the state $|\pi_{g_i}\rangle$, and the last quantum register contains the linear transformation Σ_i. If we directly measure the last register, we can get the desired matrix, but the coherence of the quantum states $|\pi_{g_i}\rangle$ are also destroyed.

To resolve this issue, we use the following theorem of Harrow and Wei:

Theorem 3.23 (Non-destructive amplitude estimation, [HW20]). *Let P be an observable. Given state $|\psi\rangle$ and reflections $R_\psi = 2|\psi\rangle\langle\psi| - I$ and $R = 2P - I$, and any $\eta > 0$, there exists a quantum algorithm that outputs $\widetilde{a}$, an approximation to $a := \langle\psi|P|\psi\rangle$, so that*

$$|a - \widetilde{a}| \leq 2\pi \frac{a(1-a)}{M} + \frac{\pi^2}{M^2}.$$

with probability at least $1 - \eta$ and $O(\log(1/\eta)M)$ uses of R_ψ and R. Morover the algorithm restores the state $|\psi\rangle$ with probability at least $1 - \eta$.

Then, we can create $O(\log N)$ copies of the state in Eq. (3.14), and non-destructively estimate the mean of the last quantum register via the procedure in [CCH+19, HW20]. More specifically, we start from $\widetilde{O}(N)$ copies of the states $\left|\pi_{g_{i-1}}\right\rangle$, and evolve them to $|\pi_{g_i}\rangle$. In the same time, the reflection operator R can be approximately implemented by Lemma 2.23. Then, the mean value can be estimated by Theorem 3.23. Note that we can estimate all the coordinates of Σ_i in the same time using the non-destructive mean estimation quantum circuit. And we get that the success probability of this procedure is at least $1 - 1/\operatorname{poly}(N)$. After that, the states in the first N registers will be restored. Therefore, we get that:

Lemma 3.24 (Non-destructive rounding). *For $i \in [K]$, the linear transformation Σ_i at the i-iteration of the annealing process (Algorithm 18) can be obtained using $\widetilde{O}(N)$ copies of the states $\left|\pi_{g_{i-1}}\right\rangle$, with query complexity $\widetilde{O}(N \cdot \mathcal{C})$ where $\mathcal{C}$ is the cost of evolving $\left|\pi_{g_{i-1}}\right\rangle$ to $|\pi_{g_i}\rangle$. Moreover, the states $\left|\pi_{g_{i-1}}\right\rangle$ will be recovered with high probability.*

3.6.3 Proof of Theorem 3.1

Algorithm 19 Quantum speedup for approximately convex optimization.

```
1: procedure QSimAnnealing($K$, $\{T_i\}_{i\in[K]}$)
2:     $N \leftarrow \widetilde{O}(n)$                                         ▷ The number of strands
3:     Prepare $N$ (approximately) copies of $|\pi_0\rangle$, denoted as $|\widetilde{\pi}_0^{(1)}\rangle, \ldots, |\widetilde{\pi}_0^{(N)}\rangle$, where
       $\pi_0 = \text{Uniform}(\mathcal{K})$
4:     for $i \leftarrow 1, \ldots, K$ do
5:         Use the $N$ copies of the state $|\pi_{i-1}\rangle$ to nondestructively obtain the linear
           transformation $\Sigma_i$. Let $|\hat{\pi}_{i-1}^{(1)}\rangle, \ldots, |\hat{\pi}_{i-1}^{(N)}\rangle$ denote the post-measurements states ▷
           Lemma 3.24
6:         Apply quantum walk with $\Sigma_i$ to evolve the states $|\hat{\pi}_{i-1}^{(1)}\rangle, \ldots, |\hat{\pi}_{i-1}^{(N)}\rangle$ to
           $|\widetilde{\pi}_i^{(1)}\rangle, \ldots, |\widetilde{\pi}_i^{(N)}\rangle$                              ▷ Theorem 3.22
7:     end for
8:     $x_K^j \leftarrow$ measure the final state $|\widetilde{\pi}_K^{(j)}\rangle$ for $j \in [N]$
9:     return $\arg\min_{j\in[N]} F(x_K^j)$
10: end procedure
```

Proof of Theorem 3.1. The quantum algorithm for optimizing an approximately convex function is given in Algorithm 19. By Theorem 3.22 and Lemma 3.24, we know that it has the same optimization guarantee as the classical procedure (Algorithm 18). Thus, we take $K = \sqrt{n}\log(n/\epsilon)$. And the output x_* of QSimAnnealing procedure satisfies:

$$F(x_*) - \min_{x\in\mathcal{K}} F(x) \leq O(\epsilon)$$

with high probability.

Then, consider the query complexity. We have $K = \sqrt{n}\log(n/\epsilon)$ stages in the annealing process. In each iteration, the quantum walk has query complexity $\mathcal{C} = \widetilde{O}(n^{1.5})$ by Theorem 3.22. Thus, the query cost of Line 5 is $\widetilde{O}(N\mathcal{C}) = \widetilde{O}(n^{2.5})$. Also, the query cost of Line 6 is also $\widetilde{O}(n^{2.5})$. Therefore, the total query complexity of the annealing procedure is

$$K \cdot \widetilde{O}(n^{2.5}) = \widetilde{O}(n^3).$$

□

Chapter 4: Early Fault-Tolerant Ground-State Energy Estimation

4.1 Introduction

When will quantum computers solve valuable problems that are out of reach for state-of-the-art classical approaches? To understand this future moment of quantum advantage, we must identify what computational problems are most apt and determine what quantum algorithms will be able solve them in the nearest time frame. Despite some recent challenges being illuminated [LZUC22], estimating the ground state energy of quantum systems [AGDLHG05] remains one of the leading contenders for the first realization of practial quantum advantage. Solving this problem efficiently with a quantum computer would be of high value to areas including combustion [GRB$^+$20], batteries [KLP$^+$22, DCR$^+$22], and catalysts [GWL$^+$22]. Considering that the progress in quantum hardware has consistently improved [Gam22], we are urged to investigate: what are the minimal quantum resources needed to realize quantum advantage with ground state energy estimation?

4.1.1 Previous methods for ground state energy estimation

Towards realizing ground state energy estimation on earlier quantum computers, researchers have developed algorithms that reduce the required resource costs of the algorithms, including number of qubits, gates, and ancillas. Most of these algorithms are based on the variational quantum eigensolver (VQE) [PMS$^+$14]. These algorithms do not have performance guarantees and recent works have identified roadblocks for the practicality of such approaches through the measurement problem [GRB$^+$20, JKG$^+$22] and issues with optimization [MBS$^+$18, AK22]. Yet, quantum algorithms with performance guarantees [AGDLHG05, BGB$^+$18, DLT22] demand unfortunately-large quantum resources, requiring hundreds of logical qubits and over billions of operations per circuit (e.g. greater than 10^{10} T gates [KLP$^+$22]). The error correction overhead needed to run such quantum circuits is far beyond what can be realized on today's hardware. Towards realizing practical quantum advantage sooner, we develop quantum algorithms in the goldilocks regime of having provable performance guarantees while also exponentially reducing the required number of operations.

The operations involved in many GSEE algorithms, including ours, are controlled time evolution operations, $c\text{-}\exp(\mathbf{i}Ht)$. For the algorithms considered, the total number of operations per circuit and the circuit depth are proportional (ignoring logarithmic factors) to the evolution time t. We will refer to this measure as the *circuit depth*. The circuit depth required by a GSEE algorithm is typically costed in terms of ϵ, the target accuracy of the ground state energy estimate, and η, a lower bound on the overlap of the input state ρ with the ground state[1]. In contrast to previous GSEE methods, the costs of our GSEE algorithms will depend on Δ, a lower bound on the energy gap (i.e. the difference between the smallest and next-smallest eigenvalue of H), typically governs the performance of ground state *preparation* methods [DLT22]. With these parameters established we are able to describe the costs of previous methods for GSEE that aim to minimize the quantum resources. Recent work [DLT22] has developed ground state energy estimation algorithms with circuit depths scaling as $\tilde{O}(1/\epsilon)$, which require just a single ancilla qubit, and involve no costly circuit operations beyond controlled time evolutions. That algorithm requires running the circuits multiple times leading to a total runtime of $\tilde{O}(\epsilon^{-1}\eta^{-1})$, which achieves the so-called Heisenberg limit scaling in the runtime with respect to ϵ. Other recent work [WBC21a] combined the ideas in [LT22] with the principles of randomized Hamiltonian evolution (QDRIFT) [Cam19] to propose a ground state energy estimation algorithm which trades off between number of non-Clifford gates and runtime.

[1] An important caveat for all known ground state energy estimation methods (c.f Table I of [DLT22]) is that if η is extremely small, then we have little hope of accurately estimating the ground state energy. Thus, it is common to assume that the Hamiltonian of interest admits a good ground state approximation.

All of these methods employ quantum circuits whose depth scales inversely with the target accuracy ϵ. This circuit depth cost may put some important problem instances out of reach for early fault-tolerant quantum computers. The question addressed by this chapter is: *is it possible to exponentially improve the accuracy-dependence scaling of circuit depth in ground state energy estimation?*

4.1.2 Summary of main results

We develop and analyze low-depth ground state energy estimation (GSEE) algorithms with high accuracy for which the circuit depth scales exponentially better than $\tilde{O}(1/\epsilon)$. As is typical, the circuit depth and quantum runtime of the algorithm is measured in terms of the number of controlled evolution operations c-exp$\,(-2\pi\mathbf{i}H)$ referred to as *Hamiltonian evolution time.*

Theorem 4.1 (Low-depth GSEE, informal version of Theorem 4.14)**.** *Let H be a Hamiltonian with spectral gap at least Δ. Suppose we can prepare an initial state ρ such that the overlap with the ground state satisfies $\langle E_0|\rho|E_0\rangle \geq \eta$. Given Δ, η, and sufficiently small ϵ, there exists an algorithm to estimate the ground state energy within accuracy ϵ with high probability such that:*

- *The circuit depth, measured in maximal Hamiltonian evolution time, is*

$$\mathcal{T}_{\max} = \mathcal{O}(\Delta^{-1} \cdot \operatorname{poly}\log \epsilon^{-1}\eta^{-1}\Delta). \tag{4.1}$$

- *The quantum runtime, measured in total Hamiltonian evolution time, is*

$$\mathcal{T}_{\text{tot}} = \mathcal{O}(\eta^{-2}\epsilon^{-2}\Delta \cdot \operatorname{poly}\log \epsilon^{-1}\eta^{-1}\Delta). \tag{4.2}$$

Using the costs established in Theorem 4.1, Table 4.1 provides resource estimates that show the reduction in gates per circuit for molecules of industrial interest. The reduction in gates per circuit affords a reduction in the fault-tolerant overhead required

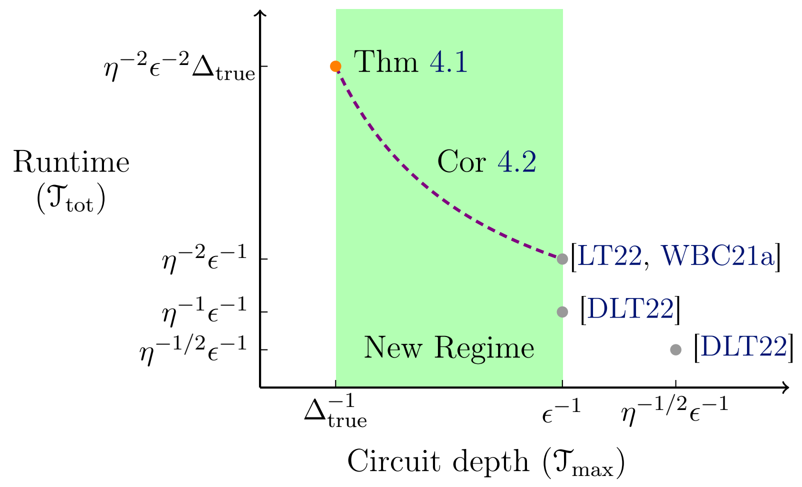


Figure 4.1: This figure shows the landscape of early fault-tolerant GSEE algorithms plotted according to their runtime and circuit depth measured in terms of total evolution time ($\mathcal{T}_{\text{tot}}$) and maximal evolution time ($\mathcal{T}_{\max}$), respectively. The green region indicates the new low-depth regime introduced in this work. The orange dot corresponds to the Δ^{-1}-depth GSEE algorithm (Theorem 4.1) when $\Delta = \Delta_{\text{true}}^{-1}$, and the curve shows the smooth trade off between $\mathcal{T}_{\max}$ and $\mathcal{T}_{\text{tot}}$ described in Corollary 4.2. We also remark that the right-most dot which shows an algorithm in [DLT22] requires multiple ancilla qubits and multi-qubit controlled operations, whereas the algorithms in this work and [LT22, WBC21a] only use a single ancilla qubit. For simplicity, we have ignored all the poly-logarithmic factors.

Molecule	Gap (mHa)	Gap Lower Bound (mHa)	Gate Reduction [KLP+22]	Gate Reduction [LT22]
EC	264 ± 20	244	$43\times$	$16\times$
PF_6^6	468 ± 20	448	$78\times$	$28\times$

Table 4.1: This table displays estimated circuit cost savings afforded by Algorithm 21 for two molecules relevant to battery design analyzed in previous work [KLP+22]. For these molecules in the cc-pVDZ basis, we can estimate the energy gaps using EOM-CCSD calculations with ORCA software [Nee12, Nee18]. The target accuracy considered in [KLP+22] was $\epsilon = 1$ mHa. The standard approach to quantum phase estimation (ignoring the cost due to imperfect ground state preparation) uses a circuit with $2/\epsilon$ applications of c-$\exp(2\pi \mathbf{i} H)$ to achieve an ϵ accurate estimate with high probability. We include the various logarithmic factors (c.f. Algorithm 21) and set a conservative input state overlap value of $\eta = 1/1000$. In the last column we give cost reductions relative to recent methods [LT22], which use $2/\pi\epsilon$ applications of c-$\exp(2\pi \mathbf{i} H)$.

to implement ground state energy estimation. These reductions may help to bring such problem instances within reach of earlier fault-tolerant quantum architectures, potentially realizing quantum advantage sooner.

For some molecules, it might be the case that the runtime of this low depth algorithm is too high to outperform state-of-the-art classical methods for solving the same problem. Our second main result (c.f. Corollary 4.15) is that we can trade circuit depth for total runtime reduction. This gives a means of speeding up the overall algorithm.

Corollary 4.2. *Let H be a Hamiltonian with spectral gap Δ_{true}. Suppose we can prepare an initial state ρ such that the overlap with the ground state satisfies $\langle E_0|\rho|E_0\rangle \geq \eta$. Then for arbitrary $\alpha \in [0,1]$, given Δ_{true}, η and sufficiently small ϵ, there exists an algorithm to estimate the ground state energy within accuracy ϵ with high probability such that:*

- *The circuit depth, measured in maximal Hamiltonian evolution time, is*

$$\mathcal{T}_{\max} = \widetilde{\mathcal{O}}(\epsilon^{-\alpha}\Delta_{\text{true}}^{-1+\alpha}). \tag{4.3}$$

- *The quantum runtime, measured in total Hamiltonian evolution time, is*

$$\mathcal{T}_{\text{tot}} = \widetilde{\mathcal{O}}(\eta^{-2}\epsilon^{-2+\alpha}\Delta_{\text{true}}^{1-\alpha}). \tag{4.4}$$

Through the era of early fault-tolerant quantum computing, as quantum architectures are able to realize deeper quantum circuits, the trade-off in Corollary 4.2 may lead to a crossover point into quantum advantage.

4.1.3 Technical overview

Before introducing the method, we define the ground state energy estimation problem. Suppose we are given a classical description of a quantum Hamiltonian H. This Hamiltonian has (unknown) spectral decomposition $H = \sum_{j=0}^{N-1} E_j |E_j\rangle\langle E_j|$, where

$E_0 < E_1 \leq E_2 \leq ... \leq E_{N-1}$ are the eigenvalues of H, and the $|E_j\rangle$'s are orthonormal eigenstates of H. Let ρ be an easy-to-prepare state (of the same dimension as H). Let $p_j \coloneqq \langle E_j| \rho |E_j\rangle$ be the overlap between ρ and $|E_j\rangle$, for $0 \leq j \leq N-1$. We assume that two numbers $\eta \in (0,1)$ and $\Delta > 0$ are given such that $p_0 = \langle E_0| \rho |E_0\rangle \geq \eta$ and $E_1 - E_0 \geq \Delta$. Our goal is to estimate E_0 with accuracy ϵ and confidence $1-\delta$, i.e. to output a sample from a random variable $\hat{E}_0$ such that the failure probability satisfies

$$\mathbb{P}\left[|\hat{E}_0 - E_0| > \epsilon\right] < \delta, \tag{4.5}$$

for given small $\epsilon > 0$ and $\delta \in (0,1)$. Furthermore, we want to achieve this by using limited-depth quantum circuits and classical post-processing of the quantum measurement outcome data.

Time signals from Hadamard tests. In our GSEE algorithm, the role of the quantum computer is simply to provide statistical estimates of $\mathrm{tr}\left[\rho e^{-\mathbf{i}H\tau}\right]$. The quantum circuit we use to generate these estimates is known as a Hadamard test and is shown in Figure 4.3. Labeling this outcome $\mathbf{b} \in \{+1, -1\}$, the average value of $\mathbf{b}$ output by the Hadamard test circuit is

$$\mathbb{E}[\mathbf{b}] = \mathrm{Re}\left[\mathrm{tr}\left[\rho e^{-\mathbf{i}H\tau}\right]\right], \tag{4.6}$$

when $W = I$ and it is equal to the imaginary part when $W = S^\dagger$ where S is the phase gate. It is helpful to view the quantity $\mathrm{tr}\left[\rho e^{-\mathbf{i}H\tau}\right]$ as a complex-valued time signal, with τ being the time. This time signal encodes information about the eigenvalues of H and the density operator ρ. In particular, if we can determine how this signal depends on the ground state energy E_0, then we might be able to estimate E_0 from the time signal. Although we are unable to exactly determine the time signal, we can estimate the real and imaginary parts of the signal at any time τ to within any desired accuracy using sufficiently many Hadamard test measurement outcomes as described above. The time cost of each Hadamard test is proportional to τ and the total time cost will depend on how many Hadamard tests, or samples, we take over the different chosen times τ.

Filtering the spectrum Here we introduce the method for estimating and processing the signals from the Hadamard test data. The Fourier transform (or frequency signal) of to the ideal time signal tr $\left[\rho e^{-\mathbf{i}H\tau}\right]$ is equal to the so-called spectral measure of H associated with the initial state ρ and is given by

$$p(x) \coloneqq \sum_{j=0}^{N-1} p_j \delta(x - E_j). \tag{4.7}$$

Although $p(x)$ itself cannot be determined exactly from Hadamard test data, we explain how to accurately estimate any convolution of $p(x)$ with a *filter function* $f(x)$. For our purposes, the filter function is used as a tool for organizing the time signal data from a limited time window into useful information about the spectrum of H. We briefly explain how to evaluate (or, rather, estimate) the complex number $(f * p)(x)$ for any given value x using low-depth Hadamard test circuits.

Three key features make low-depth convolution estimation possible. First, the convolution can be expressed as a linear combination of $\hat{p}(t) = \mathrm{tr}\left[\rho e^{-2\pi \mathbf{i} H t}\right]$,

$$(f * p)(x) = \int_{-\infty}^{\infty} \hat{f}(t)\hat{p}(t) e^{2\pi \mathbf{i} x t} dt, \tag{4.8}$$

where $\hat{f}(t)$ denotes the Fourier transform of $f(x)$. Second, these traces can be estimated from the Hadamard test data as shown in Eq. 4.6. Third, the circuit depth of each Hadamard test is proportional to t. We can limit the circuit depth used in the algorithm and still obtain an accurate estimate of the convolution by judiciously truncating the integral approximation

$$\int_{-\infty}^{\infty} \hat{f}(t)\hat{p}(t) e^{2\pi \mathbf{i} x t} dt \approx \int_{-T}^{T} \hat{f}(t)\, \mathrm{tr}\left[\rho e^{-2\pi \mathbf{i} H t}\right] e^{2\pi \mathbf{i} x t} dt. \tag{4.9}$$

As explained in detail in Algorithm 20, the strategy we use to estimate $(f * p)(x)$ uses a so-called multi-level Monte Carlo approach. An unbiased estimate of $(f * p)(x)$ is constructed by first (classically) sampling a time t in $[-T, T]$ drawn from a distribution proportional to $|\hat{f}_T(t)|$. Conditioned on this outcome t, a sample is then drawn from each of the real ($W = I$) and the imaginary ($W = S^\dagger$) part Hadamard tests with time

t, returning X and Y, respectively. From these outcome data (t, X, Y), we construct a random variable whose average value is equal to $(f * p)(x)$,

$$Z(x; t, X, Y) = \|\hat{f}_T\|_1 e^{2\pi \mathbf{i}(tx+\phi(t))} \cdot (X + \mathbf{i}Y). \tag{4.10}$$

where $e^{\mathbf{i}2\pi\phi(t)} = \hat{f}_T(t)/|\hat{f}_T(t)|$. It is important to note that the samples can be generated ahead of time; the choice of where to evaluate $(f * p)(x)$ can be made after this data is gathered.

Designing the filters The filter function plays two roles in determining the algorithm performance. First, the shape of the filter determines how easily the ground state energy can be determined from $(f * p)(x)$. Second, the smoothness of the filter determines the severity of the truncation T that can be withstood, and therefore the minimal viable circuit depth. This second role is what affords the exponential reduction in circuit depth of our method.

Estimating the ground state energy from the convolution can be understood through an example. In [LT22] they choose $f(x)$ to be a periodic Heaviside function[2]. As shown in Figure 4.2, this particular choice of convolution results in a series of steps, the first of which is located at the ground state energy E_0. Their algorithm proceeds by using a binary search to locate this first step. The drawback of this approach is that in order to resolve this first step to accuracy ϵ, the truncation must be $\tilde{O}(1/\epsilon)$. This means that the circuit depth of the Hadamard test scales as $\tilde{O}(1/\epsilon)$. We design a filter function and energy estimation strategy that requires a time window that scales as $O(\log 1/\epsilon)$. This corresponds to an exponential improvement in the circuit depth dependence on the accuracy.

The key observation for the design of low-depth filter functions is as follows. If a filter function satisfies the following properties, then it can isolate the minimum

[2] $\Theta(x) = \begin{cases} 1 & \text{if } x \in [2k\pi, (2k+1)\pi) \\ 0 & \text{if } x \in [(2k-1)\pi, 2k\pi) \end{cases} \quad \forall k \in \mathbb{Z}.$

eigenvalue from the others well and the corresponding convolution can be evaluated easily:

1. The filter function $f(x)$ has an exponentially-decaying tail, i.e., $|f(x)| = \exp(-\Omega(|x|))$ for sufficiently large x. This enables the filter function to "almost" eliminate the interference of other eigenvalues to the peak around E_0.

2. The filter function's Fourier transform $\hat{f}(t)$ also has an exponentially-decaying tail, i.e., $|\hat{f}(t)| = \exp(-\Omega(|t|))$ for sufficiently large t. This allows f to be well-approximated by a band-limited function, which means that the maximal evolution time in the Hadamard tests will be small.

Based on this observation, a natural choice are filters based on Gaussian functions. As shown in Figure 4.2, we choose a Gaussian derivative filter g_σ defined as

$$g_\sigma(x) := -\frac{1}{\sqrt{2\pi}\sigma^3} x e^{-\frac{x^2}{2\sigma^2}}.$$

Since the Gaussian derivative filter has an exponentially-decaying tail, $(g_\sigma * p)(x)$ resembles $p_0 g_\sigma(x - E_0)$ in a neighborhood of E_0. In particular, the unique zero point of $g_\sigma * p$ in this region is close to E_0. Instead of the Gaussian function itself, the Gaussian derivative filter is chosen because its slope at the location of the ground state energy enables better resolution compared to locating the peak of the Gaussian (with respect to the methods we analyzed).

Ground state energy estimation algorithm Using the Gaussian derivative filter we describe the algorithm for ground state energy estimation that proves Theorem 4.1. This is the first GSEE algorithm that uses $\widetilde{\mathcal{O}}(\Delta^{-1})$-depth quantum circuits to achieve accuracy ϵ. A detailed presentation of the algorithm can be found in Algorithm 21. The inputs to the algorithm are the Hamiltonian H, a lower bound on its gap Δ, a lower bound on the ground state overlap η of the input state ρ, a rough estimate $\widetilde{E}_0$ of E_0 with $O(\Delta)$ accuracy, the required accuracy ϵ, and confidence $1-\delta$. The output of the algorithm is an estimate of E_0. The steps of the algorithm are as follows:

1. **Configure filter** Set the Gaussian derivative filter function to have width $\sigma \in \widetilde{O}(\Delta)$ and choose the truncation of the filter to be $T \in \widetilde{O}(1/\sigma) = \widetilde{O}(1/\Delta)$ (this limits the circuit depth).

2. **Estimate convolution** (c.f. Algorithm 20) For a grid of $M \in \widetilde{O}(\Delta/\epsilon)$ evenly-spaced energies centered at $\widetilde{E}_0$ with width $\widetilde{O}(\Delta)$, estimate the Gaussian derivative convolution at each grid point.

   (a) To estimate the Gaussian derivative convolution at each point, for $\widetilde{O}(\eta^{-2}\epsilon^{-2}\Delta^2)$ rounds, draw a time $t \in [-T, T]$ with probability proportional to $|\hat{f}_T(t)|$ and run depth t real and imaginary Hadamard tests to generate binary samples X and Y.

   (b) Compute $Z(x; t, X, Y)$ (see Eq. 4.10) for each sample and average to output the convolution estimate at x.

3. **Estimate zero-crossing** Among the M convolution estimates, find the estimate closest to zero and report the corresponding energy as the ground state energy estimate.

To realize the results in Corollary 4.2, we choose Δ between Δ_{true} and $1/\epsilon$.

4.1.4 Future directions

In this chapter we have introduced a framework for implementing ground state energy estimation using tunable-depth quantum circuits. As shown in Figure 4.1, the algorithms developed in this work are applicable to maximum circuit depths ranging from $\widetilde{\mathcal{O}}(1/\Delta)$ to $\widetilde{\mathcal{O}}(1/\epsilon)$. We leave to future work the development of tunable-depth quantum algorithms outside of this region. While we have made progress in establishing upper bounds over a range of circuit depths, an important future direction is to establish lower bounds on depth-limited ground state energy estimation. These directions would further the research program of characterizing the performance and limitations of using depth-limited quantum computers to estimate properties of Hamiltonians.

Our work helps to establish the paradigm of developing quantum algorithms using the tools of classical signal processing [LT22, ZWJ22, WSJ22]. Here, the quantum computer is understood to generate a stochastic signal that encodes properties of a matrix of interest. This stochastic signal can be processed to learn the matrix properties of interest. The signal processing paradigm is well-suited to developing algorithms for early fault-tolerant quantum computers. Quantum computations with such architectures will be error prone, generating noisy signals. The tools of classical signal processing have been designed to handle such noisy signals and can aid in the design and analysis of robust quantum algorithms [WKJC21, KKPJ21, KKJ22].

One requirement of the algorithm is that a lower bound on the energy gap must be specified. There exist quantum chemistry methods for estimating the gap (e.g. using the ORCA software [Nee12, Nee18] as we did for our numerical comparisons). However, such estimates can become inaccurate for large systems. It may be helpful to incorporate a step into the quantum algorithm that estimates this gap. Although this estimation is computationally hard in general [Amb14], many physical systems of interest have structures that make the estimation feasible.

In this chapter, we did not consider the impact of implementation error on the performance of the algorithm. We expect that our algorithm is able to tolerate some degree of variation between the ideal Hadamard tests and the implemented Hadamard tests. Building off of recent work [KKJ22], we believe the algorithm can be operated so as to accommodate such deviations. We leave for future work the investigation of robust quantum algorithms for ground state energy estimation.

The methods introduced here may help to bring the target of useful quantum computing closer to the present. Yet, there is still much work needed to carry out detailed resource estimations that predict the onset of quantum advantage using methods such as those we have introduced. More broadly, our hope is that this work contributes to the general understanding of how to use quantum computers given practical constraints on their capabilities and might inspire the development of quantum algorithms in other application domains.

4.2 Estimating ground state energy via Gaussian derivative filtering

In this section, we propose a strategy for GSEE based on Gaussian derivative filtering. In Section 4.2.1, we define the Gaussian derivative function and prove a nice property of the convolution between this filter and the spectral measure p. In Section 4.2.2, we show how this property leads to a strategy for GSEE. In Section 4.2.3, we prove that the Gaussian derivative function can be approximated by a band-limited function, which is crucial for efficient evaluation of the convolution.

4.2.1 Convolving the spectral measure with a Gaussian derivative filter

Let us start by defining the Gaussian derivative function and demonstrating its properties. Specifically, let $\sigma > 0$ be arbitrary, and let $f_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$ be a Gaussian function. The Fourier transform of f_σ is

$$\hat{f}_\sigma(\xi) = e^{-\frac{1}{2}(\sigma\pi\xi)^2}. \tag{4.11}$$

Now consider the derivative of f_σ, i.e.,

$$g_\sigma(x) \coloneqq f'_\sigma(x) = -\frac{1}{\sqrt{2\pi}\sigma^3} x e^{-\frac{x^2}{2\sigma^2}}. \tag{4.12}$$

Then the Fourier transform of g_σ is

$$\hat{g}_\sigma(\xi) = 2\pi\mathbf{i}\xi \hat{f}_\sigma(\xi) = 2\pi\mathbf{i}\xi e^{-\frac{1}{2}(\sigma\pi\xi)^2}. \tag{4.13}$$

The following properties of g_σ and $\hat{g}_\sigma$ will be useful:

Fact 4.3 (Properties of the Gaussian derivative function)**.**

1. $g_\sigma(0) = 0$.

2. $|g_\sigma(x)|$ *is even, increases monotonically in* $(-\infty, -\sigma] \cup [0, \sigma]$, *and decreases monotonically in* $[-\sigma, 0] \cup [\sigma, \infty)$.

3. $g_\sigma(x)$ *decays exponentially to* 0 *as* $x \to \pm\infty$.

4. $\hat{g}_\sigma(\xi)$ *decays exponentially to* 0 *as* $\xi \to \pm\infty$.

Now let us consider the convolution between the filter g_σ and the spectral measure p:

$$(g_\sigma * p)(x) = \sum_{j=0}^{N-1} p_j g_\sigma(x - E_j) = -\frac{1}{\sqrt{2\pi}\sigma^3} \sum_{j=0}^{N-1} p_j (x - E_j) e^{-\frac{(x-E_j)^2}{2\sigma^2}}. \tag{4.14}$$

It turns out that if σ is appropriately chosen, then $|(g_\sigma * p)(x)|$ is small only if x is close to E_0, assuming x is at most $O(\sigma)$-away from E_0:

Lemma 4.4. *Let* $c = \sqrt{2\ln(10/9)} \approx 0.45904$, *and let* Δ *and* η *be as in the problem formulation in the main text. Suppose* $\epsilon > 0$ *is small enough such that* $\epsilon \le c \cdot \min\left(\frac{0.9\Delta}{\sqrt{2\ln(9\Delta\epsilon^{-1}\eta^{-1})}}, 0.2\Delta\right)$. *Then for*

$$\sigma := \min\left(\frac{0.9\Delta}{\sqrt{2\ln(9\Delta\epsilon^{-1}\eta^{-1})}}, 0.2\Delta\right), \tag{4.15}$$

we have

- $|(g_\sigma * p)(x)| < \dfrac{0.6\epsilon p_0}{\sqrt{2\pi}\sigma^3}$, $\forall x \in [E_0 - 0.5\epsilon, E_0 + 0.5\epsilon]$.
- $|(g_\sigma * p)(x)| > \dfrac{0.8\epsilon p_0}{\sqrt{2\pi}\sigma^3}$, $\forall x \in [E_0 - 0.5\sigma, E_0 - \epsilon) \cup (E_0 + \epsilon, E_0 + 0.5\sigma]$.

Proof. Note that our choice of σ and the condition on ϵ imply that $\epsilon \le c\sigma < 0.5\sigma$. As a consequence, we do have $E_0 - 0.5\sigma < E_0 - \epsilon$ and $E_0 + \epsilon < E_0 + 0.5\sigma$. Thus, the interval in the second bullet is well-defined. Moreover, we have

$$\begin{aligned}
|g_\sigma(0.9\Delta)| &= \frac{1}{\sqrt{2\pi}\sigma^3} 0.9\Delta e^{-\frac{0.81\Delta^2}{2\sigma^2}} \\
&\le \frac{1}{\sqrt{2\pi}\sigma^3} 0.1\epsilon\eta \qquad \text{(by the property } \sigma \le \tfrac{0.9\Delta}{\sqrt{2\ln(9\Delta\epsilon^{-1}\eta^{-1})}} \text{ in Eq. (4.15))} \\
&\le \frac{1}{\sqrt{2\pi}\sigma^3} 0.1\epsilon p_0, \qquad (4.16)
\end{aligned}$$

where the last step follows from $p_0 \ge \eta$.

We prove the first and the second parts of the lemma below.

Part I. For any $x \in [E_0 - 0.5\epsilon, E_0 + 0.5\epsilon]$, we have

$$\begin{aligned}
|(g_\sigma * p)(x)| &= \left| p_0 g_\sigma(x - E_0) + \sum_{j=1}^{N-1} p_j g_\sigma(x - E_j) \right| \qquad \text{(by Eq. (4.14))} \\
&\le p_0 \left| g_\sigma(x - E_0) \right| + \sum_{j=1}^{N-1} p_j \left| g_\sigma(x - E_j) \right| \\
&\le p_0 \left| g_\sigma(x - E_0) \right| + \max_{1 \le j \le N-1} \left| g_\sigma(x - E_j) \right|. \qquad (4.17)
\end{aligned}$$

The first term in Eq. (4.17) can be bounded as follows:

$$\begin{aligned}
|g_\sigma(x - E_0)| &\le |g_\sigma(0.5\epsilon)| \qquad \text{(by } |x - E_0| \le 0.5\epsilon < \sigma \text{ and Property 2 in Fact 4.3)} \\
&= \frac{1}{\sqrt{2\pi}\sigma^3} 0.5\epsilon e^{-\frac{0.25\epsilon^2}{2\sigma^2}} \qquad \text{(by Eq. (4.12))} \\
&\le \frac{1}{\sqrt{2\pi}\sigma^3} 0.5\epsilon. \qquad (4.18)
\end{aligned}$$

To upper bound the second term in Eq. (4.17), first note that for each $j \ge 1$,

$$\begin{aligned}
|x - E_j| &\ge E_j - E_0 - 0.5\epsilon \qquad \text{(since } x \in [E_0 - 0.5\epsilon, E_0 + 0.5\epsilon]) \\
&\ge \Delta - 0.5\epsilon \qquad \text{(since } E_j - E_0 \ge E_1 - E_0 \ge \Delta) \\
&> 0.9\Delta \qquad \text{(by the assumption } \epsilon \le 0.2c\Delta < 0.1\Delta) \\
&> \sigma, \qquad (4.19)
\end{aligned}$$

where the last step follows from the property $\sigma \leq 0.2\Delta$ in Eq. (4.15). Then we obtain

$$\begin{aligned}
|g_\sigma(x - E_j)| &< |g_\sigma(0.9\Delta)| && \text{(by Eq. (4.19) and Property 2 in Fact 4.3)} \\
&\leq \frac{1}{\sqrt{2\pi\sigma^3}} 0.1\epsilon p_0, && (4.20)
\end{aligned}$$

where the second step follows from Eq. (4.16).

Combining Eqs. (4.17), (4.18), and (4.20), we get that for $x \in [E_0 - 0.5\epsilon, E_0 + 0.5\epsilon]$,

$$|(g_\sigma * p)(x)| < p_0 \cdot \frac{1}{\sqrt{2\pi\sigma^3}} 0.5\epsilon + \frac{1}{\sqrt{2\pi\sigma^3}} 0.1\epsilon p_0 = \frac{0.6\epsilon p_0}{\sqrt{2\pi\sigma^3}}. \tag{4.21}$$

Part II. For any $x \in [E_0 - 0.5\sigma, E_0 - \epsilon) \cup (E_0 + \epsilon, E_0 + 0.5\sigma]$, we have

$$\begin{aligned}
|(g_\sigma * p)(x)| &= \left| p_0 g_\sigma(x - E_0) + \sum_{j=1}^{N-1} p_j g_\sigma(x - E_j) \right| && \text{(by Eq. (4.14))} \\
&\geq p_0 \left| g_\sigma(x - E_0) \right| - \sum_{j=1}^{N-1} p_j \left| g_\sigma(x - E_j) \right| \\
&\geq p_0 \left| g_\sigma(x - E_0) \right| - \max_{1 \leq j \leq N-1} \left| g_\sigma(x - E_j) \right|. && (4.22)
\end{aligned}$$

The first term in Eq. (4.22) can be lower bounded as follows:

$$\begin{aligned}
|g_\sigma(x - E_0)| &> |g_\sigma(\epsilon)| && \text{(by } \epsilon < |x - E_0| \leq 0.5\sigma \text{ and Property 2 in Fact 4.3)} \\
&= \frac{1}{\sqrt{2\pi\sigma^3}} \epsilon e^{-\frac{\epsilon^2}{2\sigma^2}} && \text{(by Eq. (4.12))} \\
&\geq \frac{1}{\sqrt{2\pi\sigma^3}} \epsilon e^{-\frac{c^2}{2}} && \text{(by the assumption } \epsilon \leq c\sigma) \\
&\geq \frac{1}{\sqrt{2\pi\sigma^3}} 0.9\epsilon, && (4.23)
\end{aligned}$$

where the last step follows from $c = \sqrt{2 \ln(10/9)}$.

To upper bound the second term in Eq. (4.22), note that for each $j \geq 1$,

$$\begin{aligned}
|x - E_j| &\geq E_j - E_0 - 0.5\sigma && \text{(since } x \in [E_0 - 0.5\sigma, E_0 - \epsilon) \cup (E_0 + \epsilon, E_0 + 0.5\sigma]) \\
&\geq \Delta - 0.5\sigma && \text{(since } E_j - E_0 \geq E_1 - E_0 \geq \Delta) \\
&\geq 0.9\Delta > \sigma, && (4.24)
\end{aligned}$$

where the last two inequalities follow from the property $\sigma \leq 0.2\Delta$ in Eq. (4.15). Then we obtain

$$\begin{aligned} |g_\sigma(x - E_j)| &\leq |g_\sigma(0.9\Delta)| \qquad \text{(by Eq. (4.24) and Property 2 in Fact 4.3)} \\ &\leq \frac{1}{\sqrt{2\pi\sigma^3}} 0.1\epsilon p_0, \end{aligned} \tag{4.25}$$

where the last step follows from Eq. (4.16).

Combining Eqs. (4.22), (4.23), and (4.25), we get that for $x \in [E_0 - 0.5\sigma, E_0 - \epsilon) \cup (E_0 + \epsilon, E_0 + 0.5\sigma]$,

$$|(g_\sigma * p)(x)| > p_0 \cdot \frac{1}{\sqrt{2\pi\sigma^3}} 0.9\epsilon - \frac{1}{\sqrt{2\pi\sigma^3}} 0.1\epsilon p_0 = \frac{0.8\epsilon p_0}{\sqrt{2\pi\sigma^3}}. \tag{4.26}$$

The lemma is thus proved. □

4.2.2 Basic strategy for ground state energy estimation

Lemma 4.4 prompts us to develop the following strategy for estimating ground state energy. We first obtain an estimate $\tilde{E}_0$ of E_0 such that $\tilde{E}_0$ is $O(\sigma)$-close to E_0 with high probability. Then we find a point at which $|(g_\sigma * p)|$ has small value in a region $[\tilde{E}_0 - O(\sigma), \tilde{E}_0 + O(\sigma)]$. Using Lemma 4.4 we can prove that this point is ϵ-close to E_0 with high probability. Formally, we have

Lemma 4.5. *Let Δ, η, ϵ and δ be as in the problem formulation in the main text. Suppose ϵ satisfies the condition in Lemma 4.4. Let σ be defined as Eq. (4.15). Suppose $\widetilde{E}_0$ is a random variable such that*

$$\mathbb{P}\left[|\widetilde{E}_0 - E_0| > \frac{\sigma}{4}\right] < \frac{\delta}{2}. \tag{4.27}$$

Let $M := \lceil \sigma/\epsilon \rceil + 1$, and let $x_j := \widetilde{E}_0 - 0.25\sigma + (0.5\sigma/M) \cdot (j-1)$ for $j \in [M]$. Suppose $h_1, h_2, \ldots, h_M$ are random variables such that

$$\mathbb{P}\left[\forall j \in [M] : |(g_\sigma * p)(x_j) - h_j| \leq \frac{0.1\epsilon\eta}{\sqrt{2\pi\sigma^3}}\right] \geq 1 - \frac{\delta}{2}. \tag{4.28}$$

Let $j^* = \arg\min_{1 \leq j \leq M} |h_j|$. Then we have

$$\mathbb{P}\left[|x_{j^*} - E_0| > \epsilon\right] < \delta. \tag{4.29}$$

Proof. By our assumptions about $\tilde{E}_0$ and $h_1, h_2, \dots, h_M$ and the union bound, we get that the following events happen simultaneously with probability at least $1-\delta$:

- $|\widetilde{E}_0 - E_0| \le 0.25\sigma$.
- $|(g_\sigma * p)(x_j) - h_j| \le \frac{0.1\epsilon\eta}{\sqrt{2\pi}\sigma^3}$, $\forall j \in [M]$.

In this case, we have $x_0, x_1, \dots, x_M \in [\tilde{E}_0 - 0.25\sigma, \tilde{E}_0 + 0.25\sigma] \subseteq [E_0 - 0.5\sigma, E_0 + 0.5\sigma]$. Then by Lemma 4.4, we have that

- If $|x_j - E_0| \le 0.5\epsilon$, then
$$|h_j| \le |(g_\sigma * p)(x_j)| + |(g_\sigma * p)(x_j) - h_j| < \frac{0.6\epsilon p_0}{\sqrt{2\pi}\sigma^3} + \frac{0.1\epsilon\eta}{\sqrt{2\pi}\sigma^3} \le \frac{0.7\epsilon p_0}{\sqrt{2\pi}\sigma^3}. \tag{4.30}$$

- If $|x_j - E_0| > \epsilon$, then
$$|h_j| \ge |(g_\sigma * p)(x_j)| - |(g_\sigma * p)(x_j) - h_j| > \frac{0.8\epsilon p_0}{\sqrt{2\pi}\sigma^3} - \frac{0.1\epsilon\eta}{\sqrt{2\pi}\sigma^3} \ge \frac{0.7\epsilon p_0}{\sqrt{2\pi}\sigma^3}. \tag{4.31}$$

Meanwhile, note that $x_1 \le E_0 \le x_M$, and $|x_{j+1} - x_j| \le 0.5\epsilon$, $\forall j \in [M-1]$. So there exists some $j_0 \in [M]$ such that $|x_{j_0} - E_0| \le 0.5\epsilon$. This implies that $|h_{j^*}| \le |h_{j_0}| < \frac{0.7\epsilon p_0}{\sqrt{2\pi}\sigma^3}$, which in turn implies that $|x_{j^*} - E_0| \le \epsilon$. This lemma is thus proved. $\square$

It remains to show how to generate the random variables $\tilde{E}_0$ and $h_1, h_2, \dots, h_M$ that satisfy the conditions Eqs. (4.27) and (4.28) respectively. To obtain $\tilde{E}_0$, we use the GSEE algorithm in [LT22] which takes $\widetilde{O}(\epsilon^{-1})$ maximal Hamiltonian evolution time to achieve ϵ-accuracy. Since $\widetilde{E}_0$ only needs $\frac{\sigma}{4}$-accuracy, this step has $\widetilde{O}(\sigma^{-1})$ maximal evolution time. To obtain $h_1, h_2, \dots, h_M$, we first introduce the band-limited version of g_σ, denoted as $g_{\sigma,T}$, in Section 4.2.3, and prove that $(g_\sigma * p)(x) \approx (g_{\sigma,T} * p)(x)$ for a small T. Then we design a data structure ConvEval in Section 4.3 such that this data structure can evaluate $g_{\sigma,T} * p$ at the points $x_1, x_2, \dots, x_M$ with high accuracy and confidence after appropriate initialization.

4.2.3 Gaussian derivative filters with bounded band-limits

In order to efficiently evaluate $g_\sigma * p$ at any given point, we truncate the spectrum of g_σ and construct a T-bandlimit version $g_{\sigma,T}$ such that

$$(g_\sigma * p)(x) \approx (g_{\sigma,T} * p)(x), \quad \forall x \in \mathbb{R}. \tag{4.32}$$

Specifically, we define $g_{\sigma,T}$ by restricting $\hat{g}_\sigma$ to $[-T, T]$ and performing the inverse Fourier transform:

$$g_{\sigma,T}(x) := \int_{-T}^{T} \hat{g}_\sigma(\xi) e^{2\pi \mathbf{i} x\xi} d\xi. \tag{4.33}$$

Clearly, $g_{\sigma,T} \to g_\sigma$ as $T \to \infty$. The following lemma shows how to choose T such that $g_{\sigma,T}$ can approximate g_σ in L_∞-norm:

Lemma 4.6. *Let $\epsilon_1 > 0$ be arbitrary. Then for*

$$T := \pi^{-1}\sigma^{-1}\sqrt{2\ln\left(8\pi^{-1}\epsilon_1^{-1}\sigma^{-2}\right)}, \tag{4.34}$$

we have

$$|g_\sigma(x) - g_{\sigma,T}(x)| \leq \frac{\epsilon_1}{2}, \, \forall x \in \mathbb{R}. \tag{4.35}$$

Proof. By the Fourier inversion theorem, we have

$$\begin{aligned} |g_\sigma(x) - g_{\sigma,T}(x)| =& \left| \int_{-\infty}^{-T} \hat{g}_\sigma(\xi) e^{2\pi \mathbf{i}\xi x} d\xi + \int_{T}^{+\infty} \hat{g}_\sigma(\xi) e^{2\pi \mathbf{i}\xi x} d\xi \right| \\ \leq& \int_{-\infty}^{-T} |\hat{g}_\sigma(\xi)| \, d\xi + \int_{T}^{+\infty} |\hat{g}_\sigma(\xi)| \, d\xi \\ =& 2\int_{T}^{+\infty} 2\pi\xi e^{-\frac{1}{2}(\sigma\pi\xi)^2} d\xi \\ =& \frac{4}{\sigma^2\pi} e^{-\frac{1}{2}\sigma^2\pi^2T^2}. \end{aligned} \tag{4.36}$$

By solving the inequality

$$\frac{4}{\sigma^2\pi} e^{-\frac{1}{2}\sigma^2\pi^2T^2} \leq \frac{\epsilon_1}{2}, \tag{4.37}$$

we get that it suffices to take

$$T \geq \pi^{-1}\sigma^{-1}\sqrt{2\ln\left(8\pi^{-1}\epsilon_1^{-1}\sigma^{-2}\right)}. \tag{4.38}$$

□

The following claim shows that the L_∞-approximation for g_σ implies the L_∞-approximation for $g_\sigma * p$.

Claim 4.7. *Let T be defined as in Lemma 4.6. Then we have*

$$|(g_\sigma * p)(x) - (g_{\sigma,T} * p)(x)| \leq \frac{\epsilon_1}{2}, \quad \forall x \in \mathbb{R}.$$

Proof. For any $x \in \mathbb{R}$, we have

$$\begin{aligned} |(g_\sigma * p)(x) - (g_{\sigma,T} * p)(x)| &= \left|\int_{-\infty}^{\infty} (g_\sigma(z) - g_{\sigma,T}(z))p(x-z)dz\right| \\ &\leq \int_{-\infty}^{\infty} |g_\sigma(z) - g_{\sigma,T}(z)|\, p(x-z)dz \\ &\leq \frac{\epsilon_1}{2}\int_{-\infty}^{\infty} p(x-z)dz \\ &= \frac{\epsilon_1}{2}, \end{aligned} \tag{4.39}$$

where the first step follows from the definition of convolution, the section step follows from the triangle inequality, the third step follows from Lemma 4.6, and the last step follows from the property of Dirac delta function. □

Claim 4.7 implies that in order to estimate $(g_\sigma * p)(x)$ within ϵ_1-accuracy, it suffices to evaluate $(g_{\sigma,T} * p)(x)$ within $0.5\epsilon_1$-accuracy, which can be achieved by the method in Section 4.3.

4.3 Complexity of Evaluating the convolution

In this section, we focus on evaluating the convolution between a filter function f and the spectral measure p to within ϵ-additive error. In Section 4.3.1, we develop an evaluation method for general filter functions with bounded band-limits. Then in Section 4.3.2, we apply the method to the Gaussian derivative filter used in our GSEE algorithm.

4.3.1 Evaluating the convolution via Hadamard tests

For the sake of generality, we will not restrict to a specific filter f but consider arbitrary filters with bounded band-limits. Specifically, for a parameter $T > 0$, let f_T be a function with band-limit T, i.e.,

$$f_T(x) = \int_{-T}^{T} \hat{f}_T(t) e^{2\pi \mathbf{i} tx} dt, \tag{4.40}$$

where $\hat{f}_T$ is the Fourier transform of f_T and satisfies $\hat{f}_T(t) = 0$ for all $t \in (-\infty, -T) \cup (T, +\infty)$. Furthermore, we require that $\hat{f}_T$ is either continuous in $[-T, T]$ or a weighted sum of Dirac delta functions (i.e., f_T has a discrete spectrum). Here we will state the results for the former case, and the reader can easily generalize them to the latter case.

Given such a function f_T, we can define a probability density ν in terms of its Fourier weights:

$$\nu(t) = \frac{|\hat{f}_T(t)|}{\|\hat{f}_T\|_1}, \quad \forall t \in [-T, T]. \tag{4.41}$$

Moreover, let $\phi(t)$ be the phase of $\hat{f}_T(t)$, i.e., $\hat{f}_T(t) = |\hat{f}_T(t)| e^{\mathbf{i} 2\pi \phi(t)}$. Then we have that

$$f_T(x) = \int_{-T}^{T} \|\hat{f}_T\|_1 e^{2\pi i (tx + \phi(t))} \nu(t) dt. \tag{4.42}$$

Now given a quantum state ρ, a Hamiltonian H and a parameter $t \in [-T, T]$, we define two random variables $\mathbf{X}_t$ and $\mathbf{Y}_t$ as follows. Let $\mathbf{b}_I$ and $\mathbf{b}_{S^\dagger}$ be the measurement outcome of the circuit in Figure 4.3 with $\tau = 2\pi t$ and $W = I$ or $S^\dagger$ (where S is the phase gate), respectively. Then we define $\mathbf{X}_t = (-1)^{\mathbf{b}_I}$ and $\mathbf{Y}_t = (-1)^{\mathbf{b}_{S^\dagger}}$.

As mentioned in the main text, we have that

$$\mathbb{E}\left[\mathbf{X}_t\right] = \mathrm{Re}\left(\mathrm{tr}\left[\rho e^{-2\pi \mathbf{i} H t}\right]\right), \quad \mathbb{E}\left[\mathbf{Y}_t\right] = \mathrm{Im}\left(\mathrm{tr}\left[\rho e^{-2\pi \mathbf{i} H t}\right]\right). \tag{4.43}$$

Now given a point $x \in \mathbb{R}$, we define the random variable $\mathbf{Z}(x)$ as follows. Let $\mathbf{t}$ be a random variable with probability density function ν. Then we define

$$\mathbf{Z}(x) := \|\hat{f}_T\|_1 \cdot e^{2\pi \mathbf{i}(\mathbf{t}x + \phi(\mathbf{t}))} \cdot (\mathbf{X}_{\mathbf{t}} + \mathbf{i}\mathbf{Y}_{\mathbf{t}}). \tag{4.44}$$

It turns out that $\mathbf{Z}(x)$ is an unbiased estimator of the convolution $f_T * p$ at point x:

Lemma 4.8. *For the random variable* $\mathbf{Z}(x)$ *defined as Eq.* (4.44)*, we have that*

$$\mathbb{E}[\mathbf{Z}(x)] = (f_T * p)(x), \quad \forall x \in \mathbb{R}. \tag{4.45}$$

Proof. Let us first consider the conditional expectation $\mathbb{E}[\mathbf{Z}(x)|\mathbf{t} = t]$ for some $t \in [-T, T]$. By Eq. (4.43) and the definition of $\mathbf{Z}(x)$ in Eq. (4.44), we get

$$\begin{aligned} \mathbb{E}[\mathbf{Z}(x)|\mathbf{t} = t] &= \mathbb{E}\left[\|\hat{f}_T\|_1 e^{2\pi \mathbf{i}(\mathbf{t}x + \phi(\mathbf{t}))}(\mathbf{X}_{\mathbf{t}} + \mathbf{i}\mathbf{Y}_{\mathbf{t}})\middle|\mathbf{t} = t\right] \\ &= \|\hat{f}_T\|_1 e^{2\pi \mathbf{i}(t_0 x + \phi(t))}\, \mathrm{tr}\left[\rho e^{-2\pi \mathbf{i} H t}\right]. \end{aligned} \tag{4.46}$$

By the law of total expectation, we have

$$\begin{aligned} \mathbb{E}[\mathbf{Z}(x)] &= \int_{-T}^{T} \mathbb{E}[\mathbf{Z}(x)|\mathbf{t} = t] \cdot \mathbb{P}\left[\mathbf{t} = t\right] dt \\ &= \int_{-T}^{T} \|\hat{f}_T\|_1 e^{2\pi \mathbf{i}(tx + \phi(t))}\, \mathrm{tr}\left[\rho e^{-2\pi \mathbf{i} H t}\right] \nu(t) dt \\ &= \int_{-T}^{T} \hat{f}_T(t) e^{2\pi \mathbf{i} t x}\, \mathrm{tr}\left[\rho e^{-2\pi \mathbf{i} H t}\right] \mathrm{d}t, \end{aligned} \tag{4.47}$$

where the last step follows from the definition of ν in Eq. (4.41) and the definition of $\phi(t)$.

It remains to prove that the above expression indeed coincides with $f_T * p(x)$. Indeed, we have that:

$$\begin{aligned}(f_T * p)(x) &= \int_{-\infty}^{\infty} p(x-y) f_T(y) \mathrm{d}y \\ &= \int_{-\infty}^{\infty} \int_{-T}^{T} p(x-y) \hat{f}_T(t) e^{2\pi \mathbf{i} t y} \mathrm{d}t \mathrm{d}y \\ &= \int_{-T}^{T} \hat{f}_T(t) \mathrm{d}t \int_{-\infty}^{\infty} p(x-y) e^{2\pi \mathbf{i} t y} \mathrm{d}y. \end{aligned} \tag{4.48}$$

By the definition of $p(x)$ in Eq. (4.7), we have that

$$\int_{-\infty}^{\infty} p(x-y) e^{2\pi \mathbf{i} t y} \mathrm{d}y = \int_{-\infty}^{\infty} \sum_{k\geq 0} p_k \delta(x-y-E_k) e^{2\pi \mathbf{i} t y} \mathrm{d}y = \sum_{k\geq 0} p_k e^{2\pi \mathbf{i} t (x-E_k)}, \tag{4.49}$$

where the last step follows from the integration of Dirac delta function. Then, it implies that

$$(f_T * p)(x) = \int_{-T}^{T} \hat{f}_T(t) \mathrm{d}t \cdot \sum_{k\geq 0} p_k e^{2\pi \mathbf{i} t (x-E_k)} = \int_{-T}^{T} \hat{f}_T(t) e^{2\pi \mathbf{i} t x} \operatorname{tr}\left[\rho e^{-2\pi \mathbf{i} H t}\right] \mathrm{d}t, \tag{4.50}$$

where the last step follows from $\operatorname{tr}\left[\rho e^{-2\pi \mathbf{i} H t}\right] = \sum_{k\geq 0} p_k e^{-2\pi \mathbf{i} t E_k}$.

Comparing Eqs. (4.47) and (4.50), we conclude that $\mathbb{E}\left[\mathbf{Z}(x)\right] = (f_T * p)(x)$ for all $x \in \mathbb{R}$. The lemma is thus proved.

$\square$

With Lemma 4.8 established, it is now straightforward to analyze how many samples we need to estimate the function $f_T * p$ at various points within a target accuracy.

Lemma 4.9 (Sample complexity of the convolution evaluation)**.** *Let $\{(t^{(i)}, X^{(i)}, Y^{(i)})\}_{i=1}^{S}$ be S i.i.d. samples such that $t^{(i)} \sim \nu$, $X^{(i)} \sim \mathbf{X}_{t_i}$ and $Y^{(i)} \sim \mathbf{Y}_{t_i}$, where ν is defined as Eq. (4.41), and $\mathbf{X}_t$ and $\mathbf{Y}_t$ are the measurement outcome of the circuit in Figure 4.3 with $\tau = 2\pi t$ and $W = I$ or S, respectively. Let $x_1, x_2, \ldots, x_M \in \mathbb{R}$ be arbitrary.*

For each $j \in [M]$, let $\overline{Z}_j$ be defined as follows:

$$\overline{Z}_j := \frac{\|\hat{f}_T\|_1}{S} \sum_{i=1}^{S} e^{2\pi \mathbf{i}(t^{(i)} x_j + \phi(t^{(i)}))} \cdot (X^{(i)} + \mathbf{i} Y^{(i)}). \tag{4.51}$$

Then for any $\epsilon_1 > 0$ and $\delta_1 \in (0,1)$, letting

$$S := \left\lceil \frac{\|\hat{f}_T\|_1^2 \ln(4M/\delta_1)}{\epsilon_1^2} \right\rceil, \tag{4.52}$$

we have

$$\mathbb{P}\left[\forall j \in [M] : |\overline{Z}_j - (f_T * p)(x_j)| \le \epsilon_1\right] \ge 1 - \delta_1. \tag{4.53}$$

Proof. Recall that $\mathbf{Z}(x) = \|\hat{f}_T\|_1 \cdot e^{2\pi \mathbf{i}(\mathbf{t}x + \phi(\mathbf{t}))} \cdot (\mathbf{X_t} + \mathbf{i}\mathbf{Y_t})$ for any $x \in \mathbb{R}$. Then $\bar{Z}_j$ is the empirical mean of S i.i.d. samples of $\mathbf{Z}(x_j)$ that correspond to $\{(t^{(i)}, X^{(i)}, Y^{(i)})\}_{i=1}^S$, for each $j \in [M]$. Since $\mathbf{X_t}$ and $\mathbf{Y_t}$ take values in $\{1, -1\}$, we know that $Re\,(\mathbf{Z}(x))$ and $Im\,(\mathbf{Z}(x))$ take values in $[-\|\hat{f}_T\|_1, \|\hat{f}_T\|_1]$. It then follows from Hoeffding's inequality [Hoe63] that for our choice of S in Eq. (4.52), for any $j \in [M]$, it holds that

$$\mathbb{P}\left[|Re\left(\overline{Z}_j\right) - \mathbb{E}[Re\,(\mathbf{Z}(x_j))]| > \frac{\epsilon_1}{\sqrt{2}}\right] < \frac{\delta_1}{2M}, \tag{4.54}$$

$$\mathbb{P}\left[|Im\left(\overline{Z}_j\right) - \mathbb{E}[Im\,(\mathbf{Z}(x_j))]| > \frac{\epsilon_1}{\sqrt{2}}\right] < \frac{\delta_1}{2M}. \tag{4.55}$$

Then by the triangle inequality and union bound, we get

$$\mathbb{P}\left[|\overline{Z}_j - \mathbb{E}[\mathbf{Z}(x_j)]| > \epsilon_1\right] < \frac{\delta_1}{M}. \tag{4.56}$$

Meanwhile, by Lemma 4.8, we know that

$$\mathbb{E}[\mathbf{Z}(x_j)] = (f_T * p)(x_j). \tag{4.57}$$

Thus, we have

$$\mathbb{P}\left[|\overline{Z}_j - (f_T * p)(x_j)| > \epsilon_1\right] < \frac{\delta_1}{M}. \tag{4.58}$$

By a union bound over all $j \in [M]$, we get that

$$\mathbb{P}\left[\exists j \in [M] : |\overline{Z}_j - (f_T * p)(x_j)]| > \epsilon_1\right] < \delta_1. \tag{4.59}$$

□

Remark 4.1. Note that $\overline{Z}_j$ is a complex number in general, but $(f_T * p)(x_j)$ is real provided that f_T is real. In this case, we can re-define $\overline{Z}_j$ as the real part of the right-hand side of Eq. (4.51) and Lemma 4.9 will still hold. We envision that in some scenarios, it is useful to have a complex filter f_T, and hence define $\overline{Z}_j$ as Eq. (4.51) for the sake of generality.

Now we give a data structure ConvEval in Algorithm 20 for evaluating the convolution $f_T * p$ at multiple points.

Lemma 4.9 immediately implies that:

Corollary 4.10. *Let $x_1, x_2, \ldots, x_M \in \mathbb{R}$ be arbitrary. Suppose the data structure* ConvEval *is initialized with parameters $(f_T, \epsilon, \delta, M)$. Let h_j be the output of the procedure* ConvEval.Eval(x_j) *for $j \in [M]$. Then we have*

$$\mathbb{P}\left[\forall j \in [M] : |(f_T * p)(x_j) - h_j| \le \epsilon\right] \ge 1 - \delta. \tag{4.60}$$

Lemma 4.11 (Running time of the convolution evaluation data structure)**.** *Suppose the data structure* FilterSampler *runs in $\mathcal{O}(1)$-time. Then, the data structure* ConvEval *in Algorithm 20 has the following running times:*

- *Procedure* Init$(f_T, \epsilon, \delta, M)$ *has $\mathcal{O}(T)$ maximal evolution time and $\mathcal{O}(ST)$ total evolution time, where $S = \mathcal{O}(\epsilon^{-2}\|\hat{f}_T\|_1^2 \log \delta^{-1} M)$.*
- *Procedure* Eval(x) *has $\mathcal{O}(S)$ classical post-processing time.*

Proof. The ConvEval.Init procedure runs the Hadamard test circuit $2S$ times to get the samples $\left\{(x^{(i)}, y^{(i)})\right\}_{i=1}^{S}$. Since the filter function f_T has spectrum bounded in $[-T, T]$, the maximal evolution time is $2\pi T$ and the total evolution time is at most $4\pi ST$.

The ConvEval.Eval procedure then uses the S samples to compute the estimate of $(f_T * p)(x)$. Moreover, the computation is classical and elementary. □

4.3.2 Application to Gaussian derivative filters

In this section, we apply the data structure ConvEval to the band-limited Gaussain derivative filter $g_{\sigma,T}$:

$$g_{\sigma,T}(x) = \int_{-T}^{T} \hat{g}_\sigma(\xi) e^{2\pi \mathbf{i} x\xi} d\xi = 2\pi\mathbf{i} \int_{-T}^{T} \xi e^{-\frac{1}{2}(\sigma\pi\xi)^2 + 2\pi\mathbf{i}x\xi} d\xi. \tag{4.61}$$

To apply Lemma 4.9, we first bound the L_1-norm of its spectrum.

Claim 4.12. *Let $g_{\sigma,T}$ be defined as Eq. (4.61). Then we have $\|\hat{g}_{\sigma,T}\|_1 \leq \frac{4}{\pi\sigma^2}$.*

Proof. By the fact that $\hat{g}_{\sigma,T}(\xi) = \hat{g}_\sigma(\xi)\mathbf{1}_{|\xi|\leq T}$ and direct calculation, we obtain

$$\|\hat{g}_{\sigma,T}\|_1 \leq \|\hat{g}_\sigma\|_1 = \int_{-\infty}^{+\infty} \left| 2\pi\mathbf{i}\xi e^{-\frac{1}{2}(\sigma\pi\xi)^2} \right| d\xi = \int_0^{+\infty} 4\pi\xi e^{-\frac{1}{2}(\sigma\pi\xi)^2} d\xi = \frac{4}{\pi\sigma^2}. \tag{4.62}$$

□

Then we get the following corollary on the sample complexity of evaluating $g_{\sigma,T} * p$ on M points.

Corollary 4.13. *Let $\epsilon_1 > 0$, $\delta_1 \in (0,1)$ and $x_1, x_2, \ldots, x_M \in \mathbb{R}$ be arbitrary. Suppose the data structure* ConvEval *is initialized with parameters $(g_{\sigma,T}, \epsilon_1, \delta_1, M)$. Let h_j be the output of the procedure* ConvEval.Eval(x_j) *for $j \in [M]$. Then we have*

$$\mathbb{P}\left[\forall j \in [M] : |(g_{\sigma,T} * p)(x_j) - h_j| \leq \epsilon_1\right] \geq 1 - \delta_1. \tag{4.63}$$

Furthermore, it take $S = \mathcal{O}(\epsilon_1^{-2}\sigma^{-4}\log M/\delta_1)$ samples from Hadamard tests to obtain h_1, h_2, ..., h_M.

Proof. Claim 4.12 implies $\|\hat{g}_{\sigma,T}\|_1 = \mathcal{O}(\sigma^{-2})$. Thus the procedure ConvEval.Init$(g_{\sigma,T}, \epsilon_1, \delta_1, M)$ draws $S = \mathcal{O}(\epsilon_1^{-2}\|\hat{g}_{\sigma,T}\|_1^2 \log M/\delta_1) = \mathcal{O}(\epsilon_1^{-2}\sigma^{-4}\log M/\delta_1)$ samples from Hadamard tests. Then Eq. (4.63) follows immediately from Corollary 4.10. □

4.4 Main Theorem

In this section, we describe our main results about ground state energy estimation.

In this section, we describe our main results about ground state energy estimation. We first present a $\widetilde{O}(\Delta^{-1})$-depth algorithm for GSEE in Algorithm 21. Then we prove the following theorem:

Theorem 4.14 (Ground state energy estimation). *Let $H = \sum_{j=0}^{N-1} E_j|E_j\rangle\langle E_j|$ be a Hamiltonian such that $E_0 < E_1 \le E_2 \le \cdots \le E_{N-1}$ are the eigenvalues of H, and the $|E_j\rangle$'s are orthonormal eigenstates of H. Suppose we are given access to the Hamiltonian evolution $e^{\mathbf{i}Ht}$ for any $t \in \mathbb{R}$. Let $\Delta > 0$ be given such that $\Delta \le E_1 - E_0$. Moreover, suppose we can prepare a state ρ such that $\langle E_0|\,\rho\,|E_0\rangle \ge \eta$ for known $\eta > 0$.*

Then, for sufficiently small $\epsilon > 0$ and any $\delta \in (0,1)$, there exists an algorithm that estimates E_0 within accuracy ϵ with probability at least $1-\delta$ such that:

- *The maximal Hamiltonian evolution time is $\widetilde{\mathcal{O}}(\Delta^{-1})$;*
- *The total Hamiltonian evolution time is $\widetilde{\mathcal{O}}(\eta^{-2}\epsilon^{-2}\Delta)$;*
- *The classical running time is $\widetilde{\mathcal{O}}(\eta^{-2}\epsilon^{-3}\Delta^3)$.*

Proof. **Algorithm:** Suppose $\epsilon > 0$ is small enough such that it satisfies the condition in Lemma 4.4. We first run the algorithm in [LT22] to obtain a coarse estimate $\widetilde{E}_0$ of E_0 such that $\widetilde{E}_0$ is $\sigma/4$-close to E_0 with probability at least $1-\delta/2$, where σ is defined as Eq. (4.15). Then we run Algorithm 21 with parameters $(\epsilon, \delta, \widetilde{E}_0, \eta)$ and return its output x_{j^*} as the final estimate of E_0.

Correctness: By construction, $\tilde{E}_0$ satisfies Eq. (4.27) in Lemma 4.5. Meanwhile, by Claim 4.7 and the choice of T in Algorithm 21, we have

$$|(g_\sigma * p)(x) - (g_{\sigma,T} * p)(x)| \leq \frac{\tilde{\epsilon}}{2}, \quad \forall x \in \mathbb{R}, \tag{4.64}$$

Meanwhile, since ConvEval is initialized with parameters $(g_{\sigma,T}, \tilde{\epsilon}/2, \delta/2, M)$ in Algorithm 21, by Corollary 4.13, we get

$$\mathbb{P}\left[\forall j \in [M] : |(g_{\sigma,T} * p)(x_j) - h_j| \leq \frac{\tilde{\epsilon}}{2}\right] \geq 1 - \frac{\delta}{2}. \tag{4.65}$$

Then it follows from Eqs. (4.64) and (4.65) and the triangle inequality that

$$\mathbb{P}\left[\forall j \in [M] : |(g_\sigma * p)(x_j) - h_j| \leq \tilde{\epsilon}\right] \geq 1 - \frac{\delta}{2}, \tag{4.66}$$

which coincides with Eq. (4.28) in Lemma 4.5, given the choice of $\tilde{\epsilon}$ in Algorithm 21. Now with both of its conditions met, Lemma 4.5 implies that the output of Algorithm 21, i.e., x_{j^*}, is ϵ-close to E_0 with probability at least $1-\delta$, as desired.

Cost analysis: First, we run the algorithm in [LT22] to obtain $\widetilde{E}_0$. Since $\sigma = \widetilde{\Omega}(\Delta)$, this step has maximal evolution time $\widetilde{\mathcal{O}}(\Delta^{-1})$, total evolution time $\widetilde{\mathcal{O}}(\Delta^{-1}\eta^{-2})$, and classical post-processing time $\widetilde{\mathcal{O}}(\Delta^{-1}\eta^{-2})$.

Then, in Line 5 of Algorithm 21, we run ConvEval.Init$(g_{\sigma,T}, \tilde{\epsilon}/2, \delta/2, M)$ to initialize the data structure ConvEval in Algorithm 20. We choose the parameters as follows:

- $\tilde{\epsilon} = \Omega(\epsilon\eta\sigma^{-3}) = \widetilde{\Omega}(\epsilon\eta\Delta^{-3})$,
- $T = \widetilde{\mathcal{O}}(\sigma^{-1}) = \widetilde{\mathcal{O}}(\Delta^{-1})$,
- $M = \Theta(\sigma\epsilon^{-1}) = \widetilde{\Theta}(\Delta\epsilon^{-1})$.

Thus, by Corollary 4.13, we have

$$S = \widetilde{\Theta}(\epsilon_1^{-2}\sigma^{-4}) = \widetilde{\Theta}(\epsilon^{-2}\eta^{-2}\Delta^6 \cdot \Delta^{-4}) = \widetilde{\Theta}(\epsilon^{-2}\eta^{-2}\Delta^2). \tag{4.67}$$

The for-loop in Procedure ConvEval.Init of Algorithm 20 draws S samples from the Hadamard test circuit. The sampling process has the maximal evolution time $2\pi T = \widetilde{\mathcal{O}}(\Delta^{-1})$ and total evolution time at most $\mathcal{O}(TS) = \widetilde{\mathcal{O}}(\epsilon^{-2}\eta^{-2}\Delta)$.

Next, in Line 6 of Algorithm 21, we call the procedure ConvEval.Eval M times to evaluate the convolutions at $x_1, x_2, \ldots, x_M$. Each evaluation takes $\mathcal{O}(S) = \widetilde{\mathcal{O}}(\epsilon^{-2}\eta^{-2}\Delta^2)$ classical time. Hence, this step takes $\mathcal{O}(MS) = \widetilde{\mathcal{O}}(\Delta\epsilon^{-1} \cdot \epsilon^{-2}\eta^{-2}\Delta^2) = \widetilde{\mathcal{O}}(\epsilon^{-3}\eta^{-2}\Delta^3)$ classical time.

Combining these steps together, we get that the whole GSEE algorithm takes:

- maximal evolution time $\widetilde{\mathcal{O}}(\Delta^{-1})$,
- total evolution time $\widetilde{\mathcal{O}}(\Delta^{-1}\eta^{-2} + \epsilon^{-2}\eta^{-2}\Delta) = \widetilde{\mathcal{O}}(\epsilon^{-2}\eta^{-2}\Delta)$, and
- classical post-processing time $\widetilde{\mathcal{O}}(\epsilon^{-3}\eta^{-2}\Delta^3)$,

as claimed. □

As described in the introduction, it is favorable to be able to reduce the total evolution time at the cost of increased maximal evolution time (or circuit depth). This allows one to make the most use of the available circuit depth afforded by the quantum architecture. Such a feature is desirable in the era of early fault-tolerant quantum computing where there is likely to be a limit to the available coherence of the device [Ton22]. Fortunately, this feature follows directly from the above theorem and we present it as a corollary below. Note that in Theorem 4.14, Δ is merely a lower bound on the true spectral gap $\Delta_{\text{true}} \coloneqq E_1 - E_0$ of Hamiltonian H, not necessarily Δ_{true} itself. In fact, Δ can range from $\widetilde{\mathcal{O}}(\epsilon)$ (in order to satisfy the condition in Lemma 4.4) to Δ_{true}. By setting $\Delta = \widetilde{\mathcal{O}}(\epsilon^{\alpha}\Delta_{\text{true}}^{1-\alpha})$ with $\alpha \in [0, 1]$, we obtain:

Corollary 4.15. *Let $H = \sum_{j=0}^{N-1} E_j |E_j\rangle\langle E_j|$ be a Hamiltonian such that $E_0 < E_1 \leq E_2 \leq \cdots \leq E_{N-1}$ are the eigenvalues of H, and the $|E_j\rangle$'s are orthonormal eigenstates of H. Let $\Delta_{\text{true}} = E_1 - E_0$ be the spectral gap of H. Suppose we are given access to* *the Hamiltonian evolution* $e^{\mathbf{i}Ht}$ *for any* $t \in \mathbb{R}$*. Moreover, suppose we can prepare a state* ρ *such that* $\langle E_0| \rho |E_0\rangle \geq \eta$ *for known* $\eta > 0$*.*

Then for any $\alpha \in [0,1]$*, for sufficiently small* $\epsilon > 0$ *and any* $\delta \in (0,1)$*, there exists an algorithm that estimates* E_0 *within accuracy* ϵ *with probability at least* $1-\delta$ *such that:*

- *The maximal Hamiltonian evolution time is* $\widetilde{\mathcal{O}}(\epsilon^{-\alpha}\Delta_{\text{true}}^{-1+\alpha})$*;*
- *The total Hamiltonian evolution time is* $\widetilde{\mathcal{O}}(\eta^{-2}\epsilon^{-2+\alpha}\Delta_{\text{true}}^{1-\alpha})$*;*
- *The classical running time is* $\widetilde{\mathcal{O}}(\eta^{-2}\epsilon^{-3+\alpha}\Delta_{\text{true}}^{3-\alpha})$*.*

In particular, setting $\alpha = 0$ or 1 leads to:

- $\Delta = \Delta_{\text{true}}$, for which Theorem 4.14 yields an algorithm with maximal evolution time $\widetilde{\mathcal{O}}(\Delta_{\text{true}}^{-1})$ and total evolution time $\widetilde{\mathcal{O}}(\eta^{-2}\epsilon^{-2}\Delta_{\text{true}})$; or
- $\Delta = \widetilde{\mathcal{O}}(\epsilon)$, for which Theorem 4.14 yields an algorithm with maximal evolution time $\widetilde{\mathcal{O}}(\epsilon^{-1})$ and total evolution time $\widetilde{\mathcal{O}}(\eta^{-2}\epsilon^{-1})$ (i.e., the Heisenberg limit).

For general $\Delta = \widetilde{\mathcal{O}}(\epsilon^{\alpha}\Delta_{\text{true}}^{1-\alpha})$ with $\alpha \in [0,1]$, Theorem 4.14 yields an algorithm with maximal evolution time $\widetilde{\mathcal{O}}(\epsilon^{-\alpha}\Delta_{\text{true}}^{-1+\alpha})$ and total evolution time $\widetilde{\mathcal{O}}(\eta^{-2}\epsilon^{-2+\alpha}\Delta_{\text{true}}^{1-\alpha})$. In other words, tuning Δ between the two extremes gives a trade-off between the circuit depth and total runtime of the algorithm.

4.5 Comparison to the Approach of [LT22]

The main advantage of our approach compared to [LT22] is in the minimal evolution time required to achieve a desired precision. Indeed, in their approach the evolution time scales inverse linearly with the desired precision. For our approach, the minimal evolution time is dictated by the reciprocal of the energy gap of the Hamiltonian and any additional precision we wish to attain only causes a poly-logarithmic factor in the evolution time. Of course, this comes at the expense of a higher sample complexity at smaller evolution times. This trade-off between the evolution time and the sample complexity is discussed in Corollary 4.15.

Note that this improvement in the minimal evolution time comes from two conceptual differences in our approach compared to [LT22]: the choice of the filter function (Heaviside versus Gaussian derivative) and how we then infer the value of the ground state energy from the convolution (jump versus 0 of derivative).

Both our approach and that of [LT22] require a truncated approximation of the underlying filter function to implement the algorithm with only finite-time evolutions. However, as the Heaviside function has a jump at 0, the degree of the Fourier series necessarily has to increase the better we want the approximation outside a small neighborhood of 0 to be. For instance, in [LT22] they find an approximation $F_{d,\epsilon}$ such that for $d = \mathcal{O}(\epsilon^{-1}\log\epsilon^{-1}\delta^{-1})$ and

$$F_{d,\epsilon}(x) = \frac{1}{\sqrt{2\pi}}\sum_{k=-d}^{d}\hat{F}_{d,\epsilon,k}e^{ikx}, \tag{4.68}$$

we have[3]

1. $-\frac{\delta}{2} \leq F_{d,\epsilon}(x) \leq 1+\delta$ for all $x \in \mathbb{R}$.

2. $|F_{d,\epsilon}(x) - \Theta(x)| \leq \delta$ for all $x \in [-\pi+\epsilon, -\epsilon] \cup [\epsilon, \pi-\epsilon]$.

Note that the evolution time required to implement the representation in Eq. (4.68) is $\mathcal{O}(d)$. This scales logarithmically with the precision with which we approximate the Heaviside function outside of the intervals $[-\pi+\epsilon, -\epsilon] \cup [\epsilon, \pi-\epsilon]$ and inverse-polynomially in size of the interval around 0, $(-\epsilon, \epsilon)$, where we are not guaranteed that the two functions are close.

[3]note that [LT22] adopts different notations. There δ is the precision with which we approximate the ground state energy, and here it is ϵ.

The approach of [LT22] consists of finding the smallest point x such that $(p * F_{d,\epsilon})(x) \geq \eta$. If we were convolving with the Heaviside function, this would correspond to the ground state energy. But we will now argue that the neighborhood around 0 for which we have the approximation can shift where the jump occurs. Indeed, note that for $x \in [E_0 - \epsilon, E_0 + \epsilon]$ and $\epsilon \leq \Delta$ we have that:

$$\begin{aligned}(p * F_{d,\epsilon})(x) = (p * \Theta)(x) + \int_{-\epsilon}^{\epsilon} p(x-y)(F_{d,\epsilon}(y) - \Theta(y))dy \\ + \int\limits_{[-1,1]\setminus[-\epsilon,\epsilon]} p(x-y)(F_{d,\epsilon}(y) - \Theta(y))dy\end{aligned}$$

Note that as $|F_{d,\epsilon}(x) - \Theta(x)| \leq \delta$ for all $x \in [-\pi + \epsilon, -\epsilon] \cup [\epsilon, \pi - \epsilon]$ we have that

$$\left| \int\limits_{[-1,1]\setminus[-\epsilon,\epsilon]} p(x-y)(F_{d,\epsilon}(y) - \Theta(y))dy \right| \leq \delta. \tag{4.69}$$

However, as we only have the promise that $-\frac{\delta}{2} \leq F_{d,\epsilon}(x) \leq 1 + \delta$ for points in $[-\epsilon, \epsilon]$, we will not be able to infer the precise point of the jump at a precision larger than $\mathcal{O}(\epsilon)$ with this approach. This is because the residual integral term (i.e. the one over $(-\epsilon, \epsilon)$ in Eq. (4.69)) will cause fluctuations in this interval and we will not be able to pin down the jump.

On the other hand, by choosing our filter to be Gaussian derivatives, we are able to obtain a good approximation everywhere on the real line. Furthermore, by choosing the zeros of the derivative as criteria, we only need to make sure that the standard deviation is small enough to separate different eigenvalues. This way we obtain a smaller maximal evolution time.

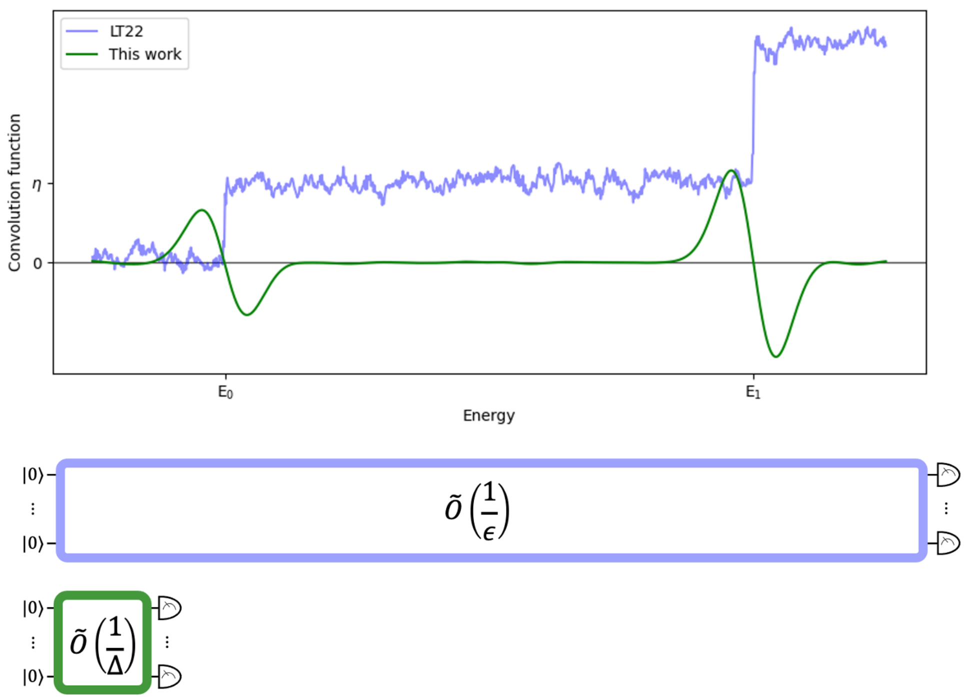


Figure 4.2: This figure compares the convolution functions and circuit depths used in the ground state energy estimation method of LT22 [LT22] and the method developed here. The LT22 method uses a Heaviside convolution, while our method uses a Gaussian derivative convolution. Their method requires a steep jump in the convolution function, which necessitates $\tilde{O}(1/\epsilon)$-depth circuits. Our method only requires that the contribution of the excited state energies to the convolution function does not interfere too much with that of the ground state energy. This affords the use of a less-steep convolution function, which only requires $\tilde{O}(1/\Delta)$-depth circuits. The trade-off is that our method requires more samples, leading to an increased total runtime.

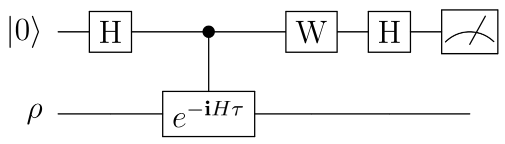


Figure 4.3: Hadamard test circuit parameterized by the Hamiltonian evolution time τ. H is the Hadamard gate and W is either I or $S^\dagger$, where S is the phase gate.

Algorithm 20 Convolution evaluation data structure.

```
 1: data structure FilterSampler
 2:     Init($f_T$)                                   ▷ Initialize for the filter $f_T$
 3:     Sample()                        ▷ Sample $\xi \in \mathbb{R}$ with probability $\propto |\hat{f}_T(\xi)|$
 4:     Norm()                                        ▷ Return $\|\hat{f}_T\|_1$
 5: end data structure
 6:
 7: data structure ConvEval
 8: members
 9:     $\mathcal{C}(t, W)$      ▷ Run the circuit in Figure 4.3 with $\tau = 2\pi t$ and $W = I$ or $S^\dagger$
10:     $\{(t^{(i)}, z^{(i)})\}_{i\in[S]} \subset \mathbb{R} \times \mathbb{C}$          ▷ Fourier samples
11:     FilterSampler FS                              ▷ Filter function's sampler
12: end members
13:
14: procedure Init($f_T$, $\epsilon$, $\delta$, $M$)      ▷ $\epsilon$ is the target accuracy, $\delta$ is the tolerable
    failure probability, $M$ is the maximal number of points at which the convolution
    is evaluated
15:     FS.Init($f_T$)
16:     $L \leftarrow$ FS.Norm()
17:     $S \leftarrow \left\lceil \frac{L^2 \ln(4M/\delta)}{\epsilon^2} \right\rceil$                          ▷ Lemma 4.9
18:     for $i \leftarrow 1, 2, \ldots, S$ do
19:         $t^{(i)} \leftarrow$ FS.Sample()
20:         $x^{(i)} \leftarrow \mathcal{C}(t^{(i)}, I)$                       ▷ Hadamard test
21:         $y^{(i)} \leftarrow \mathcal{C}(t^{(i)}, S^\dagger)$                      ▷ Hadamard test
22:         $z^{(i)} \leftarrow L \cdot e^{2\pi \mathbf{i} \phi(t^{(i)})}(x^{(i)} + \mathbf{i} y^{(i)})$
23:     end for
24: end procedure
25:
26: procedure Eval($x$)                  ▷ Approximate $(f_T * p)(x)$ within accuracy $\epsilon$
27:     $\overline{Z} \leftarrow \frac{1}{S} \sum_{i\in[S]} e^{2\pi \mathbf{i} t^{(i)} x} \cdot z^{(i)}$
28:     return $\overline{Z}$
29: end procedure
30: end data structure
```

Algorithm 21 Low-depth ground state energy estimation algorithm.

```
1: procedure GSEE($\epsilon$, $\delta$, $\widetilde{E}_0$, $\Delta$, $\eta$)
2:     $\sigma \leftarrow \min\left(\frac{0.9\Delta}{\sqrt{2\ln(9\Delta\epsilon^{-1}\eta^{-1})}}, 0.2\Delta\right)$
3:     $M \leftarrow \lceil \sigma/\epsilon \rceil + 1$, $\tilde{\epsilon} \leftarrow \frac{0.1\epsilon\eta}{\sqrt{2\pi}\sigma^3}$
4:     $T \leftarrow \pi^{-1}\sigma^{-1}\sqrt{2\ln(8\pi^{-1}\tilde{\epsilon}^{-1}\sigma^{-2})}$      ▷ Filter band-limit (Lemma 4.6)
5:     ConvEval.Init($g_{\sigma,T}$, $\tilde{\epsilon}/2$, $\delta/2$, $M$)      ▷ Algorithm 20
6:     for $j = 1, 2, \ldots, M$ do
7:         $x_j \leftarrow \widetilde{E}_0 - 0.25\sigma + (0.5\sigma/M) \cdot (j-1)$
8:         $h_j \leftarrow$ ConvEval.Eval($x_j$)      ▷ Algorithm 20
9:     end for
10:    $j^* \leftarrow \arg\min_{1\le j\le M} |h_j|$.
11:    return $x_{j^*}$
12: end procedure
```

Chapter 5: Early Fault-Tolerant Ground-State Property Estimation

5.1 Introduction

One of the primary applications of quantum computing is the simulation of materials and molecules, which are inherently quantum mechanical. It is hoped that future powerful quantum computers will be used in the development of materials and drug discovery [CRAG18]. Although they have yet to realize commercial application, quantum computers have been improving at a rapid rate, increasing the demand for quantum algorithms with high-impact use cases. To date, the main focus of quantum algorithm development for quantum chemistry and materials has been on ground state energy estimation [CRO+19]. This problem is mathematically formulated as estimating the lowest eigenvalue of the Hamiltonian matrix that characterizes the physical system. One of the first quantum chemistry applications of quantum computers was to use quantum phase estimation for estimating the ground state energy of small molecules [AGDLHG05]. More recently, the variational quantum eigensolver algorithm [PMS+14] was developed to use near-term intermediate-scale quantum (NISQ) computers to solve the ground state energy estimation problem.

However, in characterizing materials or analyzing small molecules for drug discovery, one often needs to estimate properties of the ground state beyond just the energy. These include transport properties [MW92], electric dipole moments [Jen17], and molecular forces [OSS+19]. Such properties depend on expectation values of observables O with respect to the ground state of a Hamiltonian H. The problem of estimating such quantities was studied in [Amb14, GY19, GPY20], showing that it is even harder, in a complexity theoretic sense, than the ground state energy estimation problem in general. A straightforward approach to estimating ground state properties is to first (approximately) prepare the ground state, from which properties can be estimated. Many algorithms (e.g. [PW09, GTC19, LT20a]) have been developed for ground state preparation. However, these algorithms only work for idealistic quantum computers, and the quantum circuit depths involved in these methods are too deep to even be implemented on early fault-tolerant quantum computers. Another approach to preparing ground states that is more amenable to near-term quantum computers is to use the variational quantum eigensolver algorithm [MMS+19, OSS+19]. However, recent work has suggested that VQE alone is not practical for solving problems of industrial relevance [GRB+20]; estimation methods which are more efficient (e.g. [WKJC21]) than prepare and measure estimation, as used in VQE, seem necessary in order for quantum computers to compete with state-of-the-art methods in quantum chemistry and materials. Further issues with the variational quantum eigensolver and its variants are that there are no guarantees on the quality of the output ground state and that heuristic optimization methods struggle to prepare high-fidelity ground states.

This motivates the development of quantum algorithms for ground state property estimation (GSPE) which are both reliable *and* able to be run on near-term quantum computers (e.g. early fault-tolerant quantum devices) with the following characteristics: (1) The circuit depth (or the maximal Hamiltonian evolution time) is small even with the price of increasing the total circuit size (or evolution time). (2) The number of logical qubits is limited. The early fault-tolerant model captures the challenges of building a large-scale long-time coherent quantum device, while also being able to solve many important problems with provable performance guarantees [BMN+21, BOM+21, Cam21, LT22, Lay22]. The central question that this chapter addresses is then:

Is it possible to estimate ground state properties of a Hamiltonian reliably using early fault-tolerant quantum computers?

In this chapter, we provide an affirmative answer to this question. Furthermore, we propose an algorithm for the ground state property estimation using low-depth quantum circuits. The main theorem is stated as follows:

Theorem 5.1 (Main theorem, informal)**.** *Given a Hamiltonian H and an observable O. Suppose we have access to a unitary U_I that prepares a state $|\phi_0\rangle$ that has non-trivial overlap with the ground state $|\psi_0\rangle$ of H. Then, there exists an algorithm to estimate $\langle\psi_0| O |\psi_0\rangle$ with high accuracy and low-depth: the maximal Hamiltonian evolution time is $\widetilde{O}(\gamma^{-1})$, where γ is the spectral gap of H.*

We make a few remarks about our main result. First, we note that the maximal evolution time, which is the maximal length of time we need to perform coherent time evolution, can roughly determine the depth of the quantum circuit. Our result achieves a nearly-linear dependence on γ^{-1} and only poly-logarithmic on the accuracy ϵ^{-1}, which improves the $\widetilde{O}(\epsilon^{-1})$ maximal evolution time in the ground state energy estimation algorithms [Som19, LT22, CBKC21, Ral21]. Second, our result does not violate the Heisenberg limit because the total evolution time still depends on poly(ϵ^{-1}). Third, similar to almost all prior works in ground state preparation and energy estimation (e.g. [Som19, LT20a, LT22]), we need the assumption that the initial state has some *nontrivial overlap* with the ground state, as otherwise the problem will become computationally intractable. Last, we consider the Hamiltonian as a black-box, which is a common model in this field. To implement our algorithm, for sparse local Hamiltonian, we can use the current state-of-the-art Hamiltonian simulation methods [BCC+15, LC17, CMN+18, LC19] with gate complexity depending linearly in the evolution time and logarithmically in the accuracy.

Comparison to the straightforward method. We can compare our algorithm with the straightforward approach of GSPE that first prepares the ground state and then applies quantum phase estimation (QPE) to estimate the ground state property.

- In the first step, to achieve an ϵ-accuracy for the estimation, the ground state need to be prepared with fidelity at least $1-\epsilon$ using the methods in [GTC19, LT20a], which have circuit depth $\widetilde{O}(\gamma^{-1}\eta^{-1})$ where η is the overlap between the initial state and the ground state.

- In the second step, QPE [KOS07, Ral21] requires circuit depth $\widetilde{O}(\epsilon^{-1})$ for an ϵ-accuracy estimation for the ground state property.

Therefore, this straightforward approach has circuit depth $\widetilde{O}(\gamma^{-1}\eta^{-1}+\epsilon^{-1})$, while our algorithm has circuit depth $\widetilde{O}(\gamma^{-1})$. Furthermore, they also need many (i.e., $\omega(1)$) additional ancilla qubits for preparing the ground state, while we only use one ancilla qubit. Our algorithm has a great advantage when the Hamiltonian's spectral gap is much larger than the estimation accuracy, making it easier to be implemented in the early fault-tolerant devices.

Organization. In Section 5.2 we formally state the problem of ground state property estimation. In Section 5.3 we review the method developed in [LT22] for estimating ground state energies. In the next three sections we explain our main algorithms and give an analysis for their performances starting from the simplest case and building to the most-involved, general case. Section 5.4 presents the case of a unitary observable which commutes with the Hamiltonian. Section 5.5 presents the case of a unitary observable which does not necessarily commute with the Hamiltonian. Section 5.6 describes the case of a general observable. Then, Section 5.7 gives two applications of the ground state property estimation algorithm. Section 5.8 gives a discussion of the results and presents some open questions.

5.2 Ground State Property Estimation Problem

In this section, we will formally define the ground state property estimation problem. This problem was initially studied by Ambainis [Amb14] as the approximate simulation problem (APX-SIM), and he proved that APX-SIM is $\mathsf{P}^{\mathsf{QMA}[\log]}$-complete[1].

[1] $\mathsf{P}^{\mathsf{QMA}[\log]}$ contains the problems with polynomial-time classical algorithms that are allowed to make $O(\log n)$ queries to an oracle solving a promise problem in QMA.

Problem 5.2 (Approximate simulation ($\mathsf{APX\text{-}SIM}$), [Amb14])**.** Given a k-local Hamiltonian H, an ℓ-local observable O, and real numbers a, b, ϵ such that $b-a \geq 1/\operatorname{poly}(n)$, and $\epsilon \geq 1/\operatorname{poly}(n)$, for n the number of qubits the Hamiltonian H acts on, decide:

- **Yes case:** H has a ground state $|\psi_0\rangle$ such that $\langle\psi_0| O |\psi_0\rangle \leq a$,
- **No case:** for any state $|\psi\rangle$ with $\langle\psi| H |\psi\rangle \leq \lambda_0 + \epsilon$ where λ_0 is the ground state energy of H, it holds that $\langle\psi_0| O |\psi_0\rangle \geq b$.

In the follow-up works, $\mathsf{APX\text{-}SIM}$ was shown to be $\mathsf{P}^{\mathsf{QMA[log]}}$-complete even for 5-local Hamiltonian and 1-local observable [GY19], and also for some physics models like 2D Heisenberg model and 1D nearest-neighbor, translationally invariant model [GPY20, WBG20]. However, these previous studies only focused on the decision version of this problem. For the purpose of designing efficient algorithms, we first define the "search version" of $\mathsf{APX\text{-}SIM}$ as follows:

Problem 5.3 (Search version of $\mathsf{APX\text{-}SIM}$)**.** Given a Hamiltonian H, an (local) observable O, and $\epsilon \in (0, 1)$, with $\Omega(1)$ probability, estimate $\langle\psi_0| O |\psi_0\rangle$ with an additive/multiplicative error at most ϵ.

In general, Problem 5.3 will not be more tractable than Problem 5.2. Thus, we may need some prior information about the Hamiltonian H and its ground state. Motivated by the widely used variational quantum eigensolver (VQE) [PMS$^+$14, MRBAG16] and the Hartree-Fock method [SO12] in quantum chemistry, it is often the case that for many real-world Hamiltonians, we are able to efficiently prepare an initial state $|\phi_0\rangle$ that has a nontrivial overlap with the ground state. Moreover, we assume that the Hamiltonian H has a nontrivial spectral gap, where a large family of Hamiltonians in practice satisfy this condition. With these assumptions, we formally define the ground state property estimation problem as follows:

Problem 5.4 (Ground state property estimation (GSPE))**.** Given a Hamiltonian H with spectral gap γ and ground state $|\psi_0\rangle$, an observable O, a unitary U_I such that it prepares an initial state $|\phi_0\rangle$ with $|\langle\phi_0|\psi_0\rangle|^2 \geq \eta$, and $\epsilon \in (0,1)$, estimate $\langle\psi_0| O |\psi_0\rangle$ with an additive/multiplicative error at most ϵ with $\Omega(1)$ probability.

Remark 5.1. We notice that when $O = H$, Problem 5.4 becomes the ground state energy estimation problem. Moreover, the prior knowledge of a large overlap for the initial state is required for all quantum algorithms with provable performance guarantees (e.g. [GTC19, LT20a, LT22]). It is also worth noting that even with these assumptions, it is unlikely to use a purely classical algorithm to estimate the ground state energy or property to high precision (unless $\mathsf{P} = \mathsf{BQP}$) [GLG22].

We propose a high-accuracy, early fault-tolerant quantum algorithm for GSPE that satisfies the following properties:

- The maximal evolution time depends *logarithmically* on the accuracy ϵ and overlap η.
- In addition to the Hamiltonian evolution and observable implementation, it only uses one additional ancilla qubit.

5.3 An Overview of the Low-Depth Ground State Energy Estimation

In this section, we provide a brief overview of the low-depth ground state energy estimation algorithm proposed by Lin and Tong [LT22]. Our algorithms are inspired by this algorithm and use it as a subroutine.

More specifically, they showed that:

Theorem 5.5 ([LT22]). *Given a Hamiltonian H with eigenvalues in the interval $[-\pi/3, \pi/3]$ and its ground state $|\psi_0\rangle$ has energy λ_0. And suppose we can prepare an initial state $|\phi_0\rangle$ such that $p_0 \geq \eta$ for some known η, where $p_0 := |\langle\phi_0|\psi_0\rangle|^2$. Then, for any $\epsilon, \nu \in (0,1)$, there exists an algorithm that estimates λ_0 with an additive error ϵ with probability $1-\nu$, by running a parameterized quantum circuit with the* *maximum quantum evolution time $\widetilde{O}(\epsilon^{-1})$ and the expected total quantum evolution time $\widetilde{O}(\epsilon^{-1}\eta^{-2})$.*

The pseudo-code of their algorithm is given in Algorithm 22.

The main technique of their algorithm is a classical post-processing procedure that extracts information from the following Hadamard test circuit (Figure 5.1).

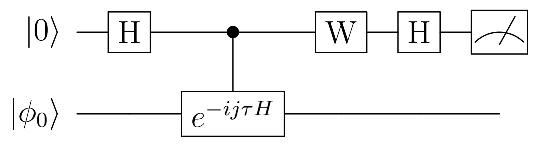


Figure 5.1: Quantum circuit parameterized by j. H is the Hadamard gate and W is either I or a phase gate. A detailed analysis of this circuit is given in Appendix C.1.1.

Let the initial state $|\phi_0\rangle$ be expanded as $|\phi_0\rangle = \sum_k \alpha_k |\psi_k\rangle$ in the eigen-basis of H and let $p_k := |\alpha_k|^2$ be the overlap with the k-th eigenstate. They considered the overlaps $p_0, p_1, \ldots$ as a density function:

$$p(x) := \sum_k p_k \delta(x - \lambda_k) \quad \forall x \in [-\pi, \pi]. \tag{5.1}$$

Then, the cumulative distribution function (CDF) $C(x) := \int_{-\pi}^{x} p(t)\mathrm{d}t$ can be expressed as a convolution of $p(x)$ and the 2π-periodic Heaviside function $H(x)$, which is 0 in $[(2k-1)\pi, 2k\pi)$ and 1 in $[2k\pi, (2k+1)\pi)$ for any $k \in \mathbb{Z}$. Thus, $C(x)$ is also a periodic function, which makes it convenient to apply the Fourier approximation. They showed that $H(x)$ can be approximated by a low-Fourier degree function $F(x)$ in the intervals $[-\pi+\delta, -\delta]$ and $[\delta, \pi-\delta]$. Then, they defined the approximated cumulative distribution function (ACDF) as $\widetilde{C}(x) := (F \star p)(x)$ and proved that

$$C(x-\delta) - \eta/8 \leq \widetilde{C}(x) \leq C(x+\delta) + \eta/8 \quad \forall x \in [-\pi/3, \pi/3]. \tag{5.2}$$

Moreover, for each x, we have

$$\widetilde{C}(x) = \sum_{|j|\le d} \hat{F}_j e^{ijx} \cdot \langle \phi_0 | e^{-ijH} | \phi_0 \rangle \,, \tag{5.3}$$

where $\hat{F}_j$ is the Fourier coefficient of $F(x)$. Note that $\langle \phi_0 | e^{-ijH} | \phi_0 \rangle$ can be estimated via the parameterized quantum circuit (Figure 5.1). Hence, we can estimate the ACDF at every point in $[-\pi/3, \pi/3]$. Moreover, they showed that the multi-level Monte Carlo method can be applied here to save the number of samples needed to achieve a high-accuracy estimation (Line 21).

Therefore, we can estimate the ground state energy λ_0 by locating the first non-zero point of the CDF $C(x)$, which is $\eta/8$-approximated by the ACDF $\widetilde{C}(x)$. Since we assume that $p_0 \ge \eta$, the approximation error and the estimation error of $\widetilde{C}(x)$ can be tolerated, and we can find λ_0 via a robust binary search (Line 18).

We note that the maximal evolution time of this algorithm corresponds to the Fourier degree of $F(x)$, which is $\widetilde{O}(\epsilon^{-1})$ by the construction, making their algorithm suitable for early fault-tolerant quantum devices. More details of this algorithm and the proofs are given in Appendix C.1.

5.4 Algorithm for Commutative Case

In this section, we consider a easier case that O is unitary and commutes with the Hamiltonian H, and give a two-step quantum-classical hybrid algorithm for Problem 5.4. More specifically, suppose the initial state can be expanded in the eigenbasis as follows: $|\phi_0\rangle = \sum_k c_k |\psi_k\rangle$ with $p_k := |c_k|^2$. We note that $\{|\psi_k\rangle\}$ is also an eigenbasis of O since O and H commute. In Step 1, we run [LT22]'s algorithm to estimate the ground state energy λ_0 and the overlap between the initial state and the ground state p_0. In Step 2, we construct a similar CDF function for the density $\sum_k O_k p_k \delta(x - \lambda_k)$, where $O_k := \langle \psi_k | O | \psi_k \rangle$. If we evaluate the CDF at λ_0, we can obtain an estimate of O_0.

5.4.1 Step 1: estimate the initial overlap

We first run the procedure EstimateGSE (Algorithm 22) to estimate the ground state energy λ_0 with an additive error ϵ. Let $x^\star$ be the output. We remark that $x^\star$ satisfy $C(x^\star+\tau\epsilon)\geq p_0$ and $C(x^\star-\tau\epsilon)=0$. However, we can only extract p_0 from the ACDF $\widetilde{C}(x)$, which satisfies:

$$C(x-\tau\epsilon)-\eta/8\leq \widetilde{C}(x)\leq C(x+\tau\epsilon)+\eta/8 \quad \forall x\in[-\pi/3,\pi/3]. \tag{5.4}$$

If $[x-\tau\epsilon, x+\tau\epsilon]$ contains a "jump" of $C(x)$, i.e., an eigenvalue λ_k, then the approximation error of $\widetilde{C}(x)$ will be large.

Hence, we say a point x is "good" for λ_k if $[x-\tau\epsilon, x+\tau\epsilon]$ is contained in $[\tau\lambda_k, \tau\lambda_{k+1})$. It is easy to see that $\widetilde{C}(x)$ will be an $\eta/8$-additive approximation of $\sum_{j\leq k} p_k$ if x is good. Our goal is to find an x_{good} that is good for λ_0, and estimating $\widetilde{C}(x_{\mathsf{good}})$ gives the overlap p_0. The following claim gives a way to construct x_{good} using the spectral gap of H.

Claim 5.6 (Construct x_{good})**.** *Let γ be the spectral gap of the Hamiltonian H. For any $\epsilon\in(0,\gamma/4)$, $x^\star+\tau\gamma/2$ is good for λ_0, where $x^\star$ is the output of* EstimateGSE(ϵ,η) *(Algorithm 22).*

Proof. We know that $x^\star$ satisfies:

$$x^\star-\tau\epsilon<\tau\lambda_0\leq x^\star+\tau\epsilon. \tag{5.5}$$

Then, we have

$$x^\star+\tau\gamma/2>\tau\lambda_0-\tau\epsilon+\tau\gamma/2>\tau\lambda_0+\tau\epsilon. \tag{5.6}$$

We also have

$$\begin{aligned} x^\star+\tau\gamma/2 &< \tau\lambda_0+\tau\epsilon+\tau\gamma/2 && (5.7)\\ &\leq \tau(\lambda_1-\gamma)+\tau\epsilon+\tau\gamma/2 \\ &= \tau\lambda_1+\tau(\epsilon-\gamma/2) \\ &< \tau\lambda_1-\tau\epsilon. && (5.8)\end{aligned}$$

Therefore, $x^\star$ is good for λ_0. □

We note that in [LT22], the ACDF's approximation error is chosen to be $\eta/8$. We may directly change it to $\epsilon\eta/8$ without significantly changing the circuit depth, since by Lemma C.6 the degree of F can only blowup by a log factor of ϵ.

Lemma 5.7 (Estimating the overlap)**.** *For any $\epsilon_0, \nu \in (0,1)$, the overlap $p_0 := |\langle\phi_0|\psi_0\rangle|^2$ can be estimated with multiplicative error $1 \pm O(\epsilon_0)$ with probability $1 - \nu$ by running the quantum circuit (Figure 4.3) $\widetilde{O}(\epsilon_0^{-2}\eta^{-2})$ times with expected total evolution time $\widetilde{O}(\gamma^{-1}\epsilon^{-2}\eta^{-2})$ and maximal evolution time $O(\gamma^{-1})$.*

Proof. By Claim 5.6, if we set the additive error of ground state energy λ_0 to be $O(\gamma)$, then we can construct an x_{good} that is good for λ_0. By Theorem 5.5, it can be done with maximum quantum evolution time $\widetilde{O}(\gamma^{-1})$ and the expected total quantum evolution time $\widetilde{O}(\gamma^{-1}\eta^{-2})$. Notice that we need to take $d = O(\delta^{-1}\log(\delta^{-1}\epsilon_0^{-1}\eta^{-1}))$ (Line 3 in Algorithm 22) to make $\widetilde{C}(x_{\mathsf{good}})$ be an $O(\epsilon_0\eta)$-approximation of p_0, where $\delta = \tau\gamma$.

Next, we estimate $\widetilde{C}(x_{\mathsf{good}})$ with additive error $\eta\epsilon$ with probability $1-\nu$. We have an unbiased estimator

$$\overline{G}(x; \mathbf{Z}, \mathbf{J}) = \mathcal{F}\mathbf{Z}e^{i\theta_{\mathbf{J}} + \mathbf{J}x} \tag{5.9}$$

for $\widetilde{C}(x)$, where $\mathbf{Z} := X + iY$ is measured from the Hadamard test, and $\mathbf{J}$ is a random variable for the Hamiltonian evolution time sampled proportional to the Fourier weight of F, i.e., $\Pr[\mathbf{J} = j] = |\hat{F}_j|/\mathcal{F}$ for $-d \le j \le d$ and $\mathcal{F} := \sum_{|j|\le d} |\hat{F}_j|$.

We can show that $\overline{G}(x; \mathbf{Z}, \mathbf{J})$ has variance $O(\log^2(d))$, and one estimate can be obtained with evolution time $\widetilde{O}(\tau d/\log(d))$ in expectation. If we repeatedly sample $\overline{G}(x; \mathbf{Z}, \mathbf{J})$ and take the mean of them, then by Chebyshev's inequality, the sample complexity is $\widetilde{O}(\epsilon_0^{-2}\eta^{-2}\nu^{-2})$ to have an additive error $O(\epsilon_0\eta)$ with probability $1-\nu$.

Instead, we can use the so-called "median-of-means" trick to reduce the sample complexity. More specifically, let $N_g = O(\log(1/\nu))$ and $K = O(\epsilon_0^{-2})$. We first partition $m = N_g K$ samples $(Z_1, J_1), \ldots, (Z_m, J_m)$ into N_g groups of size K. Then, for any $i \in [N_g]$, the i-th group mean is

$$\overline{G}_i := \frac{1}{K} \sum_{j=1}^{K} \overline{G}(x; Z_{(i-1)K+j}, J_{(i-1)K+j}). \tag{5.10}$$

The final estimator is given by the median of these group means, i.e.,

$$\overline{G}(x) := \text{median}(\overline{G}_1, \ldots, \overline{G}_{N_g}). \tag{5.11}$$

By Chernoff bound, it is easy to see that $\overline{G}(x)$ has an additive error at most $(\eta\epsilon_0)$ with probability $1-\nu$. It will imply that multiplicative error is at most $1 \pm O(\epsilon_0)$ since $p_0 = \Theta(\eta)$. And the sample complexity of $\overline{G}(x)$ is $\widetilde{O}(\epsilon_0^{-2}\eta^{-2})$. Hence, the expected total evolution time is $\widetilde{O}(\gamma^{-1}\epsilon_0^{-2}\eta^{-2})$. Since we run the same quantum circuit to estimate $\overline{G}(x)$, the maximal evolution time is still $\widetilde{O}(\gamma^{-1})$. $\square$

5.4.2 Step 2: estimate the O-weighted CDF

To estimate the expectation value of O, consider the following quantum circuit:

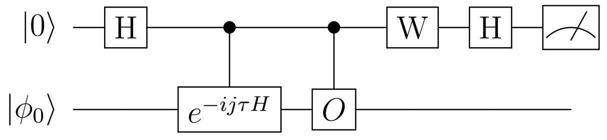


Figure 5.2: Quantum circuit parameterized by j. H is the Hadamard gate and W is either I or a phase gate S.

Define the random variables X_j, Y_j be as follows: for $W = I$, $X_j := 1$ if the outcome is 0, and $X_j := -1$ if the outcome is 1. For $W = S$, $Y_j := -1$ if the outcome is 0, and $Y_j := 1$ if the outcome is 1.

Then, we have the following claim on the expectation of the random variables X_j, Y_j:

Claim 5.8 (A variant of Hadamard test). *For any $j \in \mathbb{Z}$, the random variable $X_j + iY_j$ is an un-biased estimator for $\langle\phi_0| Oe^{-ij\tau H} |\phi_0\rangle$.*

The proof is deferred to Appendix C.1.1.

We can expand $\langle\phi_0| Oe^{-ij\tau H} |\phi_0\rangle$ in the eigenbasis of H (which is also an eigenbasis of O):

$$\begin{aligned}\langle\phi_0| Oe^{-ij\tau H} |\phi_0\rangle &= \sum_{k,k'} c_k^* c_{k'} e^{-ij\tau\lambda_k} \langle\psi_k| O |\psi_k'\rangle \\ &= \sum_k p_k O_k e^{-ij\tau\lambda_k}, \end{aligned} \tag{5.12}$$

where the last step follows from the simultaneous diagonalization of O and H, and $O_k := \langle\psi_k| O |\psi_k\rangle$. We may assume that $|O_k| \le 1$ for any $k \in \mathbb{N}$.

Inspired by the ground state energy estimation algorithm in [LT22], we define the O-weighted "density function" for the observable as follows:

$$p_O(x) := \sum_k p_k O_k \delta(x - \tau\lambda_k). \tag{5.13}$$

Note that $p_O(x)$ can be negative at some points.

Suppose the eigenvalues of τH is within $[-\pi/3, \pi/3]$. Then, we define the O-weighted CDF and ACDF for $p_O(x)$ similar to [LT22]:

$$C_O(x) := (H * p_O)(x), \quad \widetilde{C_O}(x) := (F * p_O)(x), \tag{5.14}$$

where H is the 2π-periodic Heaviside function and $F = F_{d,\delta}$ is the Fourier approximation of H constructed by Lemma C.6. It is easy to verify that $C_O(x)$ equals to $\sum_k p_k O_k \mathbf{1}_{x \ge p_k O_k}$ for any $x \in [-\pi/3, \pi/3]$.

The following lemma gives an unbiased estimator for the O-weighted ACDF.

Lemma 5.9 (Estimating the O-weighted ACDF). *For any $x \in [-\pi, \pi]$, there exists an unbiased estimator $\overline{G_O}(x)$ for the O-weighted ACDF $\widetilde{C_O}(x)$ with variance $\widetilde{O}(1)$.*

Furthermore, $\overline{G_O}(x)$ runs the quantum circuit (Figure 5.2) with expected total evolution time $O(\tau d/\log(d))$, where d is the Fourier degree of F .

Proof. $\widetilde{C_O}(x)$ can be expanded in the following way:

$$\begin{aligned}
\widetilde{C_O}(x) &= (F * p_O)(x) \qquad (5.15)\\
&= \int_{-\pi}^{\pi} F(x-y)p_O(y)\mathrm{d}y\\
&= \sum_{|j|\le d}\int_{-\pi}^{\pi}\hat{F}_j e^{ij(x-y)}p_O(y)\mathrm{d}y\\
&= \sum_{|j|\le d}\hat{F}_j e^{ijx}\int_{-\pi}^{\pi}p_O(y)e^{-ijy}\mathrm{d}y\\
&= \sum_{|j|\le d}\hat{F}_j e^{ijx}\sum_k p_k O_k e^{-ij\tau\lambda_k}\\
&= \sum_{|j|\le d}\hat{F}_j e^{ijx}\cdot\langle\phi_0|\, Oe^{-ij\tau H}\,|\phi_0\rangle\,, \qquad (5.16)
\end{aligned}$$

where the third step follows from the Fourier expansion of $F(x-y)$, the fifth step follows from the property of Dirac's delta function, and the last step follows from the definition of p_k and the eigenvalues of matrix exponential.

Define an estimator $G(x;\mathbf{J},\mathbf{Z})$ as follows:

$$G(x;\mathbf{J},\mathbf{Z}) := \mathcal{F}\cdot\mathbf{Z}e^{i(\theta_{\mathbf{J}}+\mathbf{J}x)}, \qquad (5.17)$$

where θ_j is defined by $\hat{F}_j = |\hat{F}_j|e^{i\theta_j}$, $\mathbf{Z} = X + iY$ measured from the quantum circuit (Figure 5.2) with parameter $j = \mathbf{J}$, and $\mathcal{F} = \sum_{|j|\le d}|\hat{F}_j|$.

Then, we show that $G(x;\mathbf{J},\mathbf{Z})$ is un-biased:

$$\begin{aligned}
\mathbb{E}[G(x;\mathbf{J},\mathbf{Z})] &= \sum_{|j|\le d}\mathbb{E}\left[(X_j+iY_j)e^{i(\theta_j+jx)}|\hat{F}_j|\right] \qquad (5.18)\\
&= \sum_{|j|\le d}\hat{F}_j e^{ijx}\cdot\mathbb{E}\left[X_j+iY_j\right]\\
&= \sum_{|j|\le d}\hat{F}_j e^{ijx}\cdot\langle\phi_0|\, Oe^{-ij\tau H}\,|\phi_0\rangle\\
&= \widetilde{C}(x), \qquad (5.19)
\end{aligned}$$

where the third step follows from Claim 5.8. Moreover, the variance of G can be upper-bounded by:

$$\begin{aligned}\mathbf{Var}[G(x;\mathbf{J},\mathbf{Z})] =&~ \mathbb{E}[|G(x;\mathbf{J},\mathbf{Z})|^2] - |\,\mathbb{E}[G(x;\mathbf{J},\mathbf{Z})]|^2 \quad (5.20)\\ \leq&~ \mathbb{E}[|G(x;\mathbf{J},\mathbf{Z})|^2]\\ \leq&~ 2\mathcal{F}^2, \quad (5.21)\end{aligned}$$

where the third step follows from $|e^{i(\theta_J+Jx)}| = 1$, and the last step follows from $X_j, Y_j \in \{\pm 1\}$. By Lemma C.6, we know that $|\hat{F}_j| = O(1/|j|)$. Hence, we have $\mathcal{F} = \sum_{|j|\leq d} O(1/|j|) = O(\log d)$. Thus, $\mathbf{Var}[G(x;\mathbf{J},\mathbf{Z})] = O(\log^2(d))$.

The expected total evolution time is

$$\mathcal{T}_{\mathsf{tot}} := \mathbb{E}[|J|] = \tau \sum_{|j|\leq d} |j| \cdot \frac{|\hat{F}_j|}{\mathcal{F}} = O(\tau d/\log(d)). \quad (5.22)$$

The lemma is then proved. □

The following lemma shows that the O-weighted CDF $C_O(x)$ can be approximated by the O-weighted ACDF $\widetilde{C_O}(x)$:

Lemma 5.10 (Approximating the O-weighted CDF)**.** *For any $\epsilon > 0$, $0 < \delta < \pi/6$, let $F(x) := F_{d,\delta}(x)$ constructed by Lemma C.6 with approximation error $\eta\epsilon/8$. Then, for any $x \in [-\pi/3, \pi/3]$, it holds that:*

$$C_O(x-\delta) - \eta\epsilon/8 \leq \widetilde{C_O}(x) \leq C_O(x+\delta) + \eta\epsilon/8. \quad (5.23)$$

The proof is very similar to Lemma C.7, so we omit it here.

We can take $\delta := \tau\gamma/5$ and let $x_{\mathsf{good}} := x^\star + \tau\gamma/2$. Then, by Claim 5.6, we know that x_{good} is good for λ_0, i.e., $[x_{\mathsf{good}} - \tau\gamma, x_{\mathsf{good}} + \tau\gamma] \subset (\tau\lambda_0, \tau\lambda_1)$. Hence, $\widetilde{C}_O(x_{\mathsf{good}})$ satisfies

$$\left|\widetilde{C}_O(x_{\mathsf{good}}) - p_0 O_0\right| \leq \eta\epsilon/8. \quad (5.24)$$

The following lemma shows how to estimate $\widetilde{C_O}(x_{\mathsf{good}})$, which is very similar to Lemma 5.7.

Lemma 5.11 (Estimating p_0O_0)**.** *For any $\epsilon_1, \nu \in (0,1)$, p_0O_0 can be estimated with multiplicative error $1 \pm O(\epsilon_1)$ with probability $1-\nu$ by runs the quantum circuit (Figure 4.3) $\widetilde{O}(\epsilon_1^{-2}\eta^{-2})$ times with expected total evolution time $\widetilde{O}(\gamma^{-1}\epsilon_1^{-2}\eta^{-2})$ and maximal evolution time $O(\gamma^{-1})$.*

5.4.3 Putting it all together

In this section, we will put the components together and prove the following main theorem, which gives an algorithm for the ground state property estimation.

Theorem 5.12 (Ground state property estimation with commutative observable, Restate)**.** *Suppose $p_0 \geq \eta$ for some known η, and let $\gamma > 0$ be the spectral gap of the Hamiltonian. Then, for any $\epsilon, \nu \in (0,1)$, the ground state property $\langle\psi_0| O |\psi_0\rangle$ can be estimated within additive error at most ϵ with probability $1-\nu$, such that:*

1. *the number of times running the quantum circuits (Figure 4.3 and 5.2) is $\widetilde{O}(\epsilon^{-2}\eta^{-2})$,*
2. *the expected total evolution time is $\widetilde{O}(\gamma^{-1}\epsilon^{-2}\eta^{-2})$,*
3. *the maximal evolution time is $\widetilde{O}(\gamma^{-1})$.*

Proof. By Lemma 5.7, we obtain an estimate $\overline{p_0}$ for p_0 with the guarantee that

$$\left|\overline{p_0} - p_0\right| \leq O(\eta\epsilon_0), \tag{5.25}$$

where ϵ_0 will be chosen shortly.

By Lemma 5.11, we obtain an estimate $\overline{p_0O_0}$ for p_0O_0 with the guarantee that

$$\left|\overline{p_0O_0} - p_0O_0\right| \leq O(\eta\epsilon_1), \tag{5.26}$$

where ϵ_1 will be chosen shortly.

Then, we have

$$\begin{aligned}
\left|\frac{\overline{p_0 O_0}}{\overline{p_0}} - O_0\right| &= \left|\frac{\overline{p_0 O_0}}{\overline{p_0}} - \frac{p_0 O_0}{\overline{p_0}} + \frac{p_0 O_0}{\overline{p_0}} - \frac{p_0 O_0}{p_0}\right| && (5.27)\\
&\le \frac{|\overline{p_0 O_0} - p_0 O_0|}{\overline{p_0}} + |p_0 O_0| \left|\frac{1}{\overline{p_0}} - \frac{1}{p_0}\right| \\
&\le \frac{O(\eta\epsilon_1)}{p_0 - O(\eta\epsilon_0)} + |p_0 O_0| \left|\frac{1}{p_0 - O(\eta\epsilon_0)} - \frac{1}{p_0}\right| \\
&\le \frac{O(\eta\epsilon_1)}{\eta - O(\eta\epsilon_0)} + |p_0 O_0| \left|\frac{1}{p_0 - p_0 O(\epsilon_0)} - \frac{1}{p_0}\right| \\
&\le O(\epsilon_1)(1 - O(\epsilon_0)) + |O_0|(1 + O(\epsilon_0) - 1) \\
&\le O(\epsilon_0 + \epsilon_1), && (5.28)
\end{aligned}$$

where the second step follows from the triangle inequality, the third step follows from Eqs. (5.25) and (5.26), the third step follows from $p_0 \ge \eta$, the fifth step follows from $\frac{1}{1-x} \le 1 + O(x)$ for $x \in (0, 1)$.

Hence, if we take $\epsilon_0 = \epsilon_1 = O(\epsilon)$, we will achieve additive error at most ϵ.

For the success probability, we can make Eq.(5.25) hold with probability $1-\nu/2$ in Lemma 5.7 and Eq.(5.26) hold with probability $1 - \nu/2$ in Lemma 5.11. Then, by the union bound, we get a good estimate with probability at least $1 - \nu$.

The computation costs follow directly from Lemma 5.7 and Lemma 5.11. And the proof of the theorem is then completed. □

5.5 Algorithm for General Unitary Observables

In this section, we will prove the following theorem for unitary observables in the general case:

Theorem 5.13 (Ground state property estimation with general unitary observable). *Suppose $p_0 \ge \eta$ for some known η and the spectral gap of the Hamiltonian H is at least γ. For any $\epsilon, \nu \in (0, 1)$, there exists an algorithm for estimating the ground state* *property $\langle\psi_0| O |\psi_0\rangle$ within additive error at most ϵ with probability at least $1-\nu$, such that:*

1. *the expected total evolution time is $\widetilde{O}(\gamma^{-1}\epsilon^{-2}\eta^{-2})$*
2. *the maximal evolution time is $\widetilde{O}(\gamma^{-1})$.*

In the following parts, we will first introduce the 2-d O-weighted density function and CDF, which extend the commuting observables to the general case. Then, we will show how to combine them with the overlap estimation in Section 5.4.1 for proving Theorem 5.13.

5.5.1 2-d O-weighted density function and CDF

Let $|\phi_0\rangle = \sum_k c_k |\psi_k\rangle$ where $|c_k|^2 = p_k$. In general, O and H may not commute. Hence, we consider a more symmetric form of expectation: $\langle\phi_0| e^{-ij\tau H} O e^{-ij'\tau H} |\phi_0\rangle$, which can be expanded in the eigenbasis of H as follows:

$$\begin{aligned}\langle\phi_0| e^{-ij\tau H} O e^{-ij'\tau H} |\phi_0\rangle &= \sum_{k,k'} c_k^* c_{k'} e^{-ij\tau\lambda_k} e^{-ij'\tau\lambda_{k'}} \langle\psi_k| O |\psi_k'\rangle \\ &= \sum_{k,k'} c_k^* c_{k'} e^{-ij\tau\lambda_{k'}} e^{-ij'\tau\lambda_{k'}} \langle\psi_k| O |\psi_{k'}\rangle \end{aligned} \tag{5.29}$$

Similar to the commutative case, we define a 2-d O-weighted density function:

$$p_{O,2}(x,y) := \sum_{k,k'} c_k^* c_{k'} O_{k,k'} \delta(x - \tau\lambda_k)\delta(y - \tau\lambda_{k'}), \tag{5.30}$$

where $O_{k,k'} := \langle\psi_k| O |\psi_{k'}\rangle$. Then, define the corresponding 2-d O-weighted CDF function as follows:

$$C_{O,2}(x) := (H_2 * p_{O,2})(x,y), \tag{5.31}$$

where $H_2(x,y) := H(x)\cdot H(y)$, the 2-d 2π-periodic Heaviside function.

We first justify that $C_{O,2}$ is indeed a CDF of $p_{O,2}$ in $[-\pi/3, \pi/3]$:

$$
\begin{aligned}
C_2(x,y) &= \int_{-\pi}^{\pi}\int_{-\pi}^{\pi} H_2(x-u, y-v)p(u,v)dudv && (5.32)\\
&= \sum_{k,k'} c_k^* c_{k'} O_{k,k'} \cdot \int_{-\pi}^{\pi}\int_{-\pi}^{\pi} H_2(x-u,y-v)\delta(u-\tau\lambda_k)\delta(v-\tau\lambda_{k'})dudv\\
&= \sum_{k,k'} c_k^* c_{k'} O_{k,k'} \cdot H(x-\tau\lambda_k)H(y-\tau\lambda_{k'})\\
&= \sum_{k,k'} c_k^* c_{k'} O_{k,k'} \cdot \mathbf{1}_{x\geq \tau\lambda_k, y\geq\tau\lambda_{k'}}\\
&= \sum_{\substack{k:\tau\lambda_k\leq x,\\ k':\tau\lambda_{k'}\leq y}} c_k^* c_{k'} O_{k,k'}. && (5.33)
\end{aligned}
$$

Hence, the definition of $C_{O,2}$ is reasonable.

Then, we show that $C_{O,2}$ can be approximated similar to the 1-d case. Let $F_2(x)$ be the 2-d approximated Heaviside function, i.e.,

$$F_2(x,y) := F(x)\cdot F(y). \tag{5.34}$$

The 2-d O-weighted approximated CDF (ACDF) is defined to be

$$\widetilde{C_{O,2}}(x,y) := (F_2 * p_{O,2})(x,y). \tag{5.35}$$

The following lemma shows that $\widetilde{C_{O,2}}(x,y)$ is close to $C_{O,2}(x',y')$ for some (x',y') close to (x,y).

Lemma 5.14 (Approximation ratio of the 2-d O-weighted ACDF)**.** *For any $\epsilon > 0$, $0 < \delta < \pi/6$, let $F_2(x,y) := F_{d,\delta}(x)\cdot F_{d,\delta}(y)$ constructed by Lemma C.6. Then, for any $x, y \in [-\pi/3, \pi/3]$, the 2-d O-weighted ACDF $\widetilde{C_{O,2}}(x,y) = (F_2 * p_{O,2})(x,y)$ satisfies:*

$$C_{O,2}(x-\delta, y-\delta) - 2\epsilon \leq \widetilde{C_{O,2}}(x,y) \leq C_{O,2}(x+\delta, y+\delta) + 2\epsilon. \tag{5.36}$$

Proof. By (2) in Lemma C.6, we have

$$|F(x) - H(x)| \leq \epsilon \quad \forall x \in [-\pi+\delta, -\delta] \cup [\delta, \pi-\delta], \tag{5.37}$$

which implies that for all $x, y \in [-\pi+\delta, -\delta] \cup [\delta, \pi-\delta]$,

$$\begin{aligned}
|F_2(x,y) - H_2(x,y)| &\le |F(x)F(y) - H(x)H(y)| && (5.38)\\
&= |F(x)F(y) - F(x)H(y) + F(x)H(y) - H(x)H(y)| \\
&\le F(x)|F(y) - H(y)| + H(y)|F(x) - H(x)| \\
&\le (F(x) + H(y))\epsilon \\
&\le 2\epsilon, && (5.39)
\end{aligned}$$

where the last step follows from $F(x) \in [0,1]$ by (1) in Lemma C.6. Furthermore, we also have for $x \in [-\delta, \delta]$, $y \in [-\pi+\delta, -\delta]$,

$$\begin{aligned}
|F_2(x,y) - H_2(x,y)| &\le |F(x)F(y) - H(x)H(y)| && (5.40)\\
&= |F(x)F(y)| && (H(y) = 0)\\
&\le F(y) \\
&\le \epsilon. && (5.41)
\end{aligned}$$

Similarly, for $x \in [-\pi+\delta, -\delta]$, $y \in [-\delta, \delta]$,

$$|F_2(x,y) - H_2(x,y)| \le \epsilon. \qquad (5.42)$$

Define $F_{L,2} := F_2(x-\delta, y-\delta)$ such that

$$\begin{aligned}
|F_{L,2}(x) - H_2(x)| \le 2\epsilon \quad \forall (x,y) &\in [-\pi+2\delta, 0] \times [-\pi+2\delta, \pi] && (5.43)\\
&\cup [-\pi+2\delta, \pi] \times [-\pi+2\delta, 0] \\
&\cup [2\delta, \pi] \times [2\delta, \pi].
\end{aligned}$$

For $\widetilde{C_{L,2}}(x,y) := (F_{L,2} * p_{O,2})(x,y)$, we have $\widetilde{C_{L,2}}(x,y) = \widetilde{C_{O,2}}(x-\delta, y-\delta)$.

Let $p_2 := p_{O,2}$. Then, for any $x, y \in [-\pi/3, \pi/3]$, we have

$$\left|C_{O,2}(x,y) - \widetilde{C_{L,2}}(x,y)\right| = \left|\int_{-\pi}^{\pi}\int_{-\pi}^{\pi} p_2(x-u,y-v)(H_2(u,v) - F_{L,2}(u,v))dudv\right| \tag{5.44}$$

$$\begin{aligned}
&\le \int_{-\pi}^{\pi}\int_{-\pi}^{\pi} p_2(x-u,y-v)|H_2(u,v) - F_{L,2}(u,v)|dudv \\
&= \left(\int_{-\pi}^{0}\int_{-\pi}^{\pi} + \int_{0}^{\pi}\int_{-\pi}^{0} + \int_{2\delta}^{\pi}\int_{2\delta}^{\pi}\right) p_2(x-u,y-v))|H_2(u,v) - F_{L,2}(u,v)|dudv \\
&\quad + \left(\int_{0}^{2\delta}\int_{0}^{\pi} + \int_{0}^{\pi}\int_{0}^{2\delta} - \int_{0}^{2\delta}\int_{0}^{2\delta}\right) p_2(x-u,y-v))|H_2(u,v) - F_{L,2}(u,v)|dudv \\
&\le 2\epsilon \cdot \left(\int_{-\pi}^{0}\int_{-\pi}^{\pi} + \int_{0}^{\pi}\int_{-\pi}^{0} + \int_{2\delta}^{\pi}\int_{2\delta}^{\pi}\right) p_2(x-u,y-v)dudv \\
&\quad + \left(\int_{0}^{2\delta}\int_{0}^{\pi} + \int_{0}^{\pi}\int_{0}^{2\delta} - \int_{0}^{2\delta}\int_{0}^{2\delta}\right) p_2(x-u,y-v))|H_2(u,v) - F_{L,2}(u,v)|dudv \\
&\le 2\epsilon + \left(\int_{0}^{2\delta}\int_{0}^{\pi} + \int_{0}^{\pi}\int_{0}^{2\delta} - \int_{0}^{2\delta}\int_{0}^{2\delta}\right) p_2(x-u,y-v))|H_2(u,v) - F_{L,2}(u,v)|dudv \\
&\le 2\epsilon + \left(\int_{0}^{2\delta}\int_{0}^{\pi} + \int_{0}^{\pi}\int_{0}^{2\delta} - \int_{0}^{2\delta}\int_{0}^{2\delta}\right) p_2(x-u,y-v)dudv \\
&= 2\epsilon + \left(\int_{x-2\delta}^{x}\int_{y-\pi}^{y} + \int_{x-\pi}^{x}\int_{y-2\delta}^{y} - \int_{x-2\delta}^{x}\int_{y-2\delta}^{y}\right) p_2(u,v)dudv \qquad (5.45) \\
&= 2\epsilon + C_{O,2}(x,y) - C_{O,2}(x-2\delta, y-2\delta),
\end{aligned}$$

where the second step follows from Cauchy-Schwarz inequality, the third step follows from partitioning the integration region, the forth step follows from Eq. (5.43) and the fact that $p(x,y)$ is supported in $[-\pi/3,\pi/3] \times [-\pi/3,\pi/3]$ and $\delta < \pi/6$ (see Figure 5.3 (a)), the fifth step follows from $p_{O,2}(x)$ is a density function, the last step follows from $C_{O,2}(x)$ is the CDF of $p_{O,2}(x)$ in $[-\pi,\pi]\times[-\pi,\pi]$ and $x,y \in [-\pi/3,\pi/3]$ (see Figure 5.3 (b)).

Hence, we have

$$\begin{aligned}
\widetilde{C_{L,2}}(x,y) &\ge C_{O,2}(x,y) - (2\epsilon + C_{O,2}(x,y) - C_{O,2}(x-2\delta,y-2\delta)) \\
&= C_{O,2}(x-2\delta,y-2\delta) - 2\epsilon, \qquad (5.46)
\end{aligned}$$

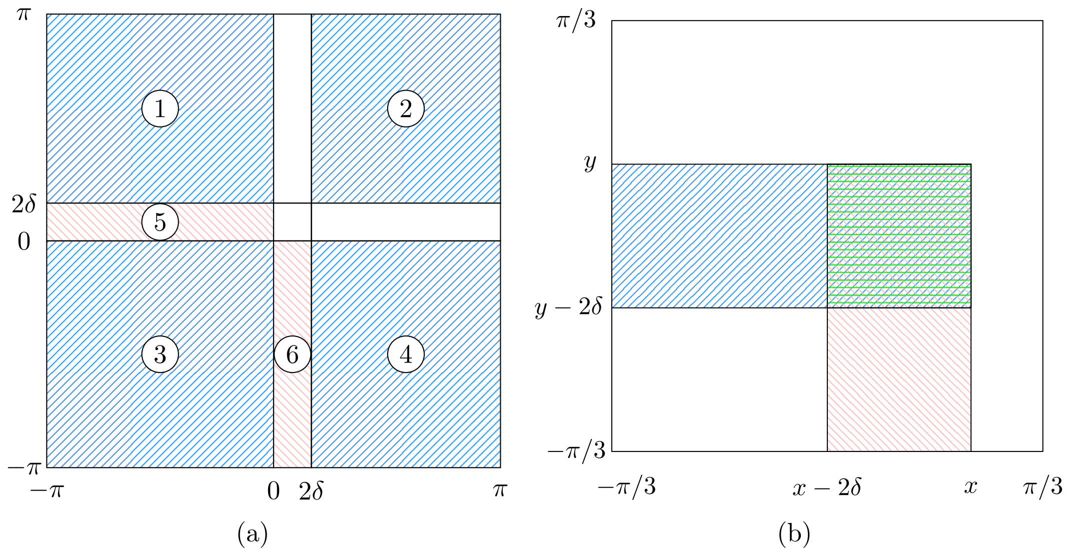


Figure 5.3: (a) is the integral region for Eq. (5.44), where the integral in regions 1-6 can be upper bounded by Eq. (5.43). (b) is the integral region for Eq. (5.45).

which proves the first inequality:

$$\widetilde{C_{O,2}}(x-\delta, y-\delta) \geq C_{O,2}(x-2\delta, y-2\delta) - 2\epsilon. \tag{5.47}$$

Similarly, we can define $F_{R,2} := F_2(x+\delta, y+\delta)$ and $\widetilde{C_{R,2}}(x,y) := (F_{R,2} * p_2)(x,y)$. We can show that

$$\left| C_{O,2}(x,y) - \widetilde{C_{R,2}}(x,y) \right| \leq 2\epsilon + C_{O,2}(x+2\delta, y+2\delta) - C_{O,2}(x,y), \tag{5.48}$$

which gives

$$\widetilde{C_{O,2}}(x+\delta, y+\delta) \leq C_{O,2}(x+2\delta, y+2\delta) + 2\epsilon. \tag{5.49}$$

The lemma is then proved. $\square$

5.5.2 Estimating the 2-d ACDF

We use the following parameterized quantum circuit to estimate the 2-d O-weighted ACDF $\widetilde{C_{O,2}}(x,y)$.

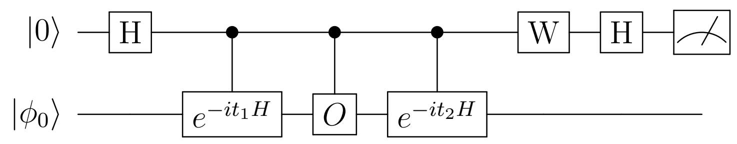


Figure 5.4: Quantum circuit parameterized by t_1, t_2. H is the Hadamard gate and W is either I or a phase gate S.

Lemma 5.15 (Estimate 2-d O-weighted ACDF)**.** *For any $x, y \in [-\pi/3, \pi/3]$, for any $\epsilon, \nu \in (0,1)$, we can estimate $\widetilde{C_{O,2}}(x,y)$ with additive error $\eta\epsilon$ with probability $1-\nu$ by running the quantum circuit (Figure 5.4) $O(\epsilon^{-2}\eta^{-2}\log(1/\nu))$ times with maximal evolution time $\widetilde{O}(\gamma^{-1})$ and total expected evolution time $\widetilde{O}(\gamma^{-1}\epsilon^{-1}\eta^{-1})$.*

Proof. $\widetilde{C_{O,2}}(x,y)$ can be expanded in the following way:

$$\begin{aligned}
\widetilde{C_{O,2}}(x,y) &= (F_2 * p_2)(x,y) \qquad (5.50)\\
&= \int_{-\pi}^{\pi}\int_{-\pi}^{\pi} F_2(x-u, y-v)p_2(u,v)dudv \\
&= \sum_{|j|\le d, |j'|\le d} \int_{-\pi}^{\pi}\int_{-\pi}^{\pi} \hat{F}_j\hat{F}_{j'}e^{ij(x-u)}e^{ij'(y-v)}p_2(u,v)dudv \\
&= \sum_{|j|\le d, |j'|\le d} \hat{F}_j\hat{F}_{j'}e^{i(jx+j'y)}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi} p_2(u,v)e^{-iju}e^{-ij'v}dudv \\
&= \sum_{|j|\le d, |j'|\le d} \hat{F}_j\hat{F}_{j'}e^{i(jx+j'y)}\sum_{k,k'} c_k^* c_k O_{k,k'}e^{-ij\tau\lambda_k}e^{-ij'\tau\lambda_{k'}} \\
&= \sum_{|j|\le d, |j'|\le d} \hat{F}_j\hat{F}_{j'}e^{i(jx+j'y)} \cdot \langle\phi_0| e^{-ij\tau H}Oe^{-ij'\tau H}|\phi_0\rangle, \qquad (5.51)
\end{aligned}$$

To estimate $\langle\phi_0| e^{-ij\tau H}Oe^{-ij'\tau H}|\phi_0\rangle$, we use the multi-level Monte Carlo method. Define a random variables J, J' with support $\{-d, \cdots, d\}$ such that

$$\Pr[J=j, J'=j'] = \frac{|\hat{F}_j||\hat{F}_{j'}|}{\mathcal{F}^2}, \qquad (5.52)$$

where $\mathcal{F} := \sum_{|j|\le d} |\hat{F}_j|$. Then, let $Z := X_{J,J'} + iY_{J,J'} \in \{\pm 1 \pm i\}$. Define an estimator $\overline{G_2}(x; J, J', Z)$ as follows:

$$\overline{G_2}(x, y; J, Z) := \mathcal{F}^2 \cdot Z e^{i(\theta_J + Jx)} e^{i(\theta_{J'} + J'y)}, \tag{5.53}$$

where θ_j is defined by $\hat{F}_j = |\hat{F}_j| e^{i\theta_j}$, and similar definition for $\theta_{j'}$. Then, we show that $\overline{G_2}(x, y; J, Z)$ is un-biased:

$$\begin{aligned}
\mathbb{E}[\overline{G_2}(x, y; J, J', Z)] = & \sum_{|j|\le d, |j'|\le d} \mathbb{E}\left[(X_{j,j'} + iY_{j,j'}) e^{i(\theta_j + jx)} e^{i(\theta_{j'} + j'y)} |\hat{F}_j||\hat{F}_{j'}|\right] & (5.54)\\
= & \sum_{|j|\le d, |j'|\le d} \hat{F}_j \hat{F}_{j'} e^{ijx} e^{ij'y} \cdot \mathbb{E}\left[X_{j,j'} + iY_{j,j'}\right] \\
= & \sum_{|j|\le d, |j'|\le d} \hat{F}_j \hat{F}_{j'} e^{ijx} e^{ij'y} \cdot \langle \phi_0 | e^{-ij\tau H} O e^{-ij'\tau H} | \phi_0 \rangle \\
= & \widetilde{C_2}(x, y), & (5.55)
\end{aligned}$$

where the third step follows from Claim C.1. Moreover, the variance of $\overline{G_2}$ can be upper-bounded by:

$$\begin{aligned}
\mathbf{Var}[\overline{G_2}(x, y; J, J', Z)] = & \ \mathbb{E}[|\overline{G_2}(x, y; J, J', Z)|^2] - |\,\mathbb{E}[\overline{G_2}(x, y; J, J', Z)]|^2 & (5.56)\\
\le & \ \mathbb{E}[|\overline{G_2}(x, y; J, J', Z)|^2] \\
= & \ \mathcal{F}^4 \cdot \mathbb{E}[|X_{J,J'} + iY_{J,J'}|^2] \\
= & \ 2\mathcal{F}^4, & (5.57)
\end{aligned}$$

where the third step follows from $|e^{i(\theta_J + Jx)}| = |e^{i(\theta_{J'} + J'y)}| = 1$, and the last step follows from $X_{j,j'}, Y_{j,j'} \in \{\pm 1\}$.

By Lemma C.6, we know that $\mathcal{F} = \widetilde{O}(1)$. Hence, we have for all $x, y \in [-\pi/3, \pi/3]$,

$$\mathbb{E}[\overline{G_2}(x, y)] = \widetilde{C_{O,2}}(x, y), \quad \text{and} \quad \mathbf{Var}[\overline{G_2}(x, y)] = \widetilde{O}(1). \tag{5.58}$$

Then, using median-of-means estimator, we can obtain an ϵ-additive error estimate of $\widetilde{C_{O,2}}(x, y)$ with probability $1 - \nu$ using $O(\epsilon^{-2}\eta^{-2}\log(1/\nu))$ samples.

The maximal evolution time is $2d = \widetilde{O}(\gamma^{-1})$. And the expected evolution time for one trial is

$$\tau \sum_{|j|,|j'|\le d} (j+j')\frac{|\hat{F}_j||\hat{F}_{j'}|}{\mathcal{F}^2} = 2\tau \sum_{|j|\le d} j\frac{|\hat{F}_j|}{\mathcal{F}} = O(\tau d/\log(d)). \tag{5.59}$$

Hence, the total expected evolution time is $\widetilde{O}(\gamma^{-1}\epsilon^{-2}\eta^{-2})$.

The lemma is then proved. $\square$

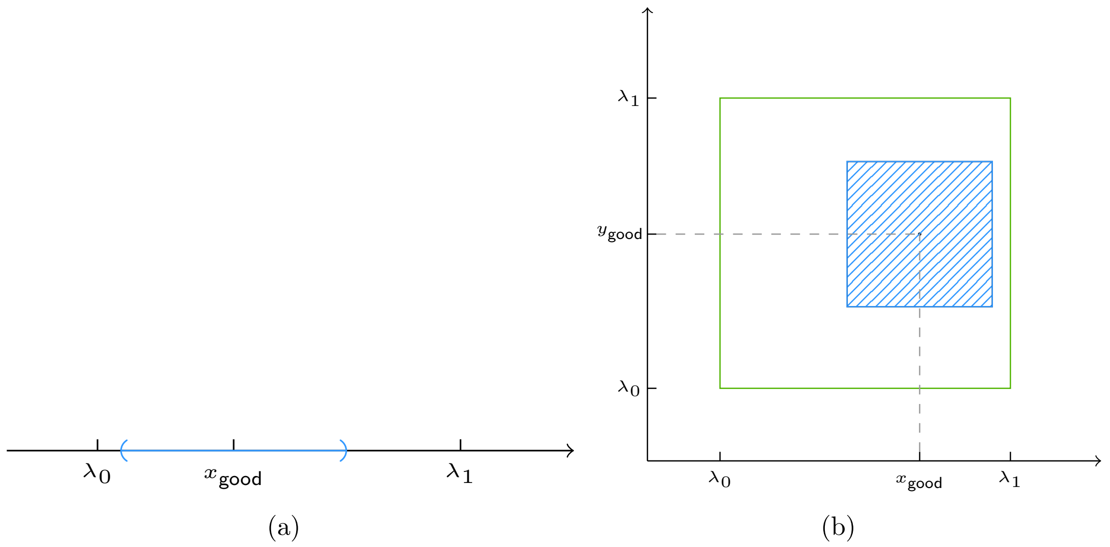


Figure 5.5: (a) shows a point that is *good* for λ_0, where the blue interval is the approximation region such that $\widetilde{C_O}(x_{\text{good}})$ is close to $C(x)$ for some x in this interval. (b) shows a good point in the 2-d case, where in the green square, the 2-d O-weighted CDF $C_{O,2}$ takes the same value $C_{O,2}(\lambda_0, \lambda_0)$. And the blue square is the approximation region of $(x_{\text{good}}, y_{\text{good}})$ such that $\widetilde{C_{O,2}}(x_{\text{good}}, y_{\text{good}})$ is close to some $C_{O,2}(x, y)$ in this region.

Similar to the 1-d case, we can construct a "good" point for (λ_0, λ_0) via the following claim.

Claim 5.16 (Construct 2-d good point)**.** *Let γ be the spectral gap of the Hamiltonian H. Let $x_{\text{good}} := x^\star + \tau\gamma/2$ where $x^\star$ is the output of* ESTIMATEGSE$(\gamma/8, \tau, \eta, \nu/10)$

(Algorithm 22). Then, $(x_{\mathsf{good}}, x_{\mathsf{good}})$ is good for (λ_0, λ_0). In particular, for any $\epsilon \in (0, 1)$, if the approximation error of $F(x)$ is set to be $\epsilon\eta$, then

$$\left|\widetilde{C_{O,2}}(x_{\mathsf{good}}, x_{\mathsf{good}}) - C_{O,2}(\lambda_0, \lambda_0)\right| \leq 2\epsilon\eta. \tag{5.60}$$

Proof. By Claim 5.6, we know that x_{good} is good for λ_0, i.e., $[x_{\mathsf{good}} - \delta, x_{\mathsf{good}} + \delta]$ is contained in $[\lambda_0, \lambda_1)$. It also holds in the 2-d case for $(x_{\mathsf{good}}, x_{\mathsf{good}})$. Then, by Lemma 5.15, we have

$$C_{O,2}(x_{\mathsf{good}} - \delta, x_{\mathsf{good}} - \delta) - 2\epsilon\eta \leq \widetilde{C_{O,2}}(x_{\mathsf{good}}, x_{\mathsf{good}}) \leq C_{O,2}(x_{\mathsf{good}} + \delta, x_{\mathsf{good}} + \delta) + 2\epsilon\eta. \tag{5.61}$$

The claim then follows from $C_{O,2}(x, y) = C_{O,2}(\lambda_0, \lambda_0)$ for any $(x, y) \in [\lambda_0, \lambda_1) \times [\lambda_0, \lambda_1)$. □

5.5.3 Putting it all together

The main algorithm for the ground state property estimation will first estimate the ground state energy λ_0 and the overlap p_0, which are described in Section 5.4.1. Then, by Lemma 5.15 and Claim 5.16, the weighted expectation $p_0 O_0$ can also be estimated. Hence, we will obtain an estimate for $O_0 = \langle\psi_0| O |\psi_0\rangle$.

Proof of Theorem 5.13. We first analyze the estimation error of Algorithm 24. By Lemma 5.7, $\overline{p_0}$ (Line 19) has additive error at most $O(\eta\epsilon)$. By Lemma 5.15 and Claim 5.16, $\overline{p_0 O_0}$ (Line 24) has additive error at most $O(\eta\epsilon)$. Then, by a similar error propagation analysis in Theorem 5.12, we get that

$$\left|\frac{\overline{p_0 O_0}}{\overline{p_0}} - O_0\right| \leq O(\epsilon). \tag{5.62}$$

For the success probability, Algorithm 24 has three components: estimate ground state energy, estimate p_0, and estimate $p_0 O_0$. By our choice of parameters, each of them will fail with probability at most $\nu/3$. Hence, Algorithm 24 will succeed with probability at least $1 - \nu$.

The maximal evolution time and the total expected evolution time follows from Theorem 5.5, Lemma 5.7, and Lemma 5.15. ☐

5.6 Handling Non-Unitary Observables

One may notice that Algorithm 24 works only for unitary observables because it needs to use the circuit in Figure 5.4 to estimate $\langle\phi_0| e^{-it_2 H} O e^{-it_1 H} |\phi_0\rangle$ for certain $t_1, t_2 \in \mathbb{R}$, in which controlled-O must be a unitary operation. In this section, we show that under reasonable assumptions this algorithm can be modified to estimate the ground state property $\langle\psi_0| O |\psi_0\rangle$ where O is a general observable.

Before we present this result, one may wonder why it is necessary. After all, we can always decompose O into a linear combination of Pauli strings $O = \sum_{\vec{s}} w_{\vec{s}} P_{\vec{s}}$, and use Algorithm 24 to estimate each term $\mu_{\vec{s}} := \langle\psi_0| P_{\vec{s}} |\psi_0\rangle$ individually, and return $\sum_{\vec{s}} w_{\vec{s}} \mu_{\vec{s}}$ as the result. While this strategy works in principle, it might be not efficient enough to be practical, depending on the weights $w_{\vec{s}}$'s of Pauli strings in the linear expansion of O.

Alternatively, one can fix the issue of Algorithm 24 by designing a procedure for estimating $\langle\phi_0| e^{-it_2 H} O e^{-it_1 H} |\phi_0\rangle$ for arbitrary non-unitary O. Such quantities are utilized in the same way as before. We have followed this approach and found that it is possible when there is a block-encoding of O. Namely, suppose O is an n-qubit observable with $\|O\| \le 1$ and U is an $(n+m)$-qubit unitary operator such that

$$(\langle 0^m| \otimes I) U (|0^m\rangle \otimes I) = \alpha^{-1} O \tag{5.63}$$

for some $\alpha \ge \|O\|$. More details about the block-encoding model can be found in [CGJ19, LC19, GSLW19, Ral20]. Then we can still perform Hadamard test for U to estimate $\langle\phi_0| e^{-it_2 H} O e^{-it_1 H} |\phi_0\rangle$ for arbitrary $t_1, t_2 \in \mathbb{R}$. The main theorem of this section is stated below:

Theorem 5.17 (Ground state property estimation with block-encoded observable)**.** *Suppose $p_0 \ge \eta$ for some known η and the spectral gap of the Hamiltonian H is at* *least γ. Suppose we have access to the α-block-encoding of the observable O. For any $\epsilon, \nu \in (0,1)$, there exists an algorithm for estimating the ground state property $\langle\psi_0| O |\psi_0\rangle$ within additive error at most ϵ with probability at least $1-\nu$, such that:*

1. *the expected total evolution time is $\widetilde{O}(\gamma^{-1}\epsilon^{-2}\eta^{-2}\alpha^2)$,*
2. *the maximal evolution time is $\widetilde{O}(\gamma^{-1})$.*

Proof sketch of Theorem 5.17. The algorithm for handling non-unitary block-encoded observables is quite similar to Algorithm 24 for handling unitary observables, except that it relies on a different procedure to estimate $\langle\phi_0| e^{-it_2 H} O e^{-it_1 H} |\phi_0\rangle$ for arbitrary $t_1, t_2 \in \mathbb{R}$. Here we briefly describe this procedure and defer the detailed analysis to Appendix C.2.

Let $C\text{-}V := |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes V$ be the controlled-V operation for arbitrary unitary operator V. Let $|\phi_0\rangle$ be an arbitrary n-qubit state. Consider the following procedure (as illustrated in Figure 5.6:

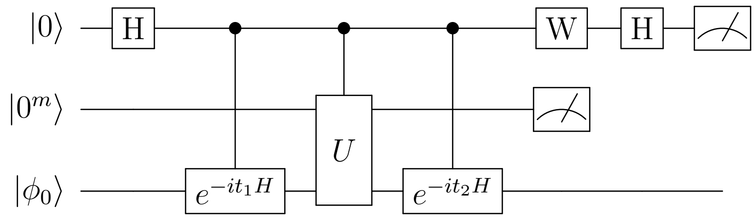


Figure 5.6: Quantum circuit parameterized by t_1, t_2. H is the Hadamard gate and W is either I or a phase gate S. U is the block-encoding of the non-unitary observable O.

1. Prepare the state $|0\rangle|0^m\rangle|\phi_0\rangle$.
2. Apply a Hadamard gate on the first register.

3. Apply a C-e^{-iHt_1} on the first and third registers.

4. Apply C-U on the current state, obtaining

$$\frac{1}{\sqrt{2}}\left(|0\rangle|0^m\rangle|\phi_0\rangle + |1\rangle U|0^m\rangle e^{-iHt_1}|\phi_0\rangle\right). \tag{5.64}$$

5. Measure the second register in the standard basis. If the outcome is not 0^m, then this procedure fails; otherwise, continue. The probability of this step succeeding is

$$p_{succ} = \frac{1+\alpha^{-2}\langle\phi_0|e^{iHt_1}O^2e^{-iHt_1}|\phi_0\rangle}{2}, \tag{5.65}$$

   and when this event happens, the state becomes

$$\frac{1}{\sqrt{2p_{succ}}}\left[|0\rangle|\phi_0\rangle + \alpha^{-1}|1\rangle Oe^{-iHt_1}|\phi_0\rangle\right]. \tag{5.66}$$

6. Apply a C-e^{-iHt_2} on the first and third registers. The state becomes

$$\frac{1}{\sqrt{2p_{succ}}}\left[|0\rangle|\phi_0\rangle + \alpha^{-1}|1\rangle e^{-iHt_2}Oe^{-iHt_1}|\phi_0\rangle\right]. \tag{5.67}$$

7. Apply $W = I$ or phase gate S on the first register.

8. Apply a Hadamard gate on the first register.

9. Measure the first register in the standard basis. Then if $W = I$, the (conditional) probability of getting outcome 0 is

$$\mathbb{P}[0|succ] = \frac{p_{succ} + \alpha^{-1}\operatorname{Re}[\langle\phi_0|e^{-iHt_2}Oe^{-iHt_1}|\phi_0\rangle]}{2p_{succ}}; \tag{5.68}$$

   if $W = S$, this probability is

$$\mathbb{P}[0|succ] = \frac{p_{succ} - \alpha^{-1}\operatorname{Im}[\langle\phi_0|e^{-iHt_2}Oe^{-iHt_1}|\phi_0\rangle]}{2p_{succ}}. \tag{5.69}$$

Now we define two random variables X and Y as follows. First, we run the above procedure with $W = I$ in step 7. If step 5 fails, $X = 0$; otherwise, if the measurement outcome is 0 or 1 in step 9, then $X = \alpha$ or $-\alpha$, respectively. One can show that X is an unbiased estimator of $\mathrm{Re}[\langle\phi_0| e^{-iHt_2} O e^{-iHt_1} |\phi_0\rangle]$, i.e.

$$\mathbb{E}[X] = \mathrm{Re}[\langle\phi_0| e^{-iHt_2} O e^{-iHt_1} |\phi_0\rangle]. \tag{5.70}$$

Y is defined similarly. We run the above procedure with $W = S$ in step 7. If step 5 fails, $Y = 0$; otherwise, if the measurement outcome is 1 or 0 in step 9, then $Y = \alpha$ or $-\alpha$, respectively. Then Y is an unbiased estimator of $\mathrm{Im}[\langle\phi_0| e^{-iHt_2} O e^{-iHt_1} |\phi_0\rangle]$, i.e.

$$\mathbb{E}[Y] = \mathrm{Im}[\langle\phi_0| e^{-iHt_2} O e^{-iHt_1} |\phi_0\rangle]. \tag{5.71}$$

It follows that $Z := X + iY$ is an unbiased estimator of $\langle\phi_0| e^{-iHt_2} O e^{-iHt_1} |\phi_0\rangle$, i.e.

$$\mathbb{E}[Z] = \langle\phi_0| e^{-iHt_2} O e^{-iHt_1} |\phi_0\rangle . \tag{5.72}$$

Note that $|Z|^2 = |X|^2 + |Y|^2 \leq 2\alpha^2$ with certainty.

Equipped with the above method for estimating $\langle\phi_0| e^{-iHt_2} O e^{-iHt_1} |\phi_0\rangle$ for arbitrary $t_1, t_2 \in \mathbb{R}$, we can now use the same strategy as in Lemma 5.15 to estimate $\widetilde{C_{O,2}}(x, y)$. The other components of Algorithm 24 remain intact. The analysis of this modified algorithm is almost the same as before, except that now we have

$$\mathbf{Var}[\overline{G_2}(x, y)] = \widetilde{O}(\alpha^2). \tag{5.73}$$

As a consequence, compared to Theorem 5.13, the total evolution time of this modified algorithm is larger by a factor of $O(\alpha^2)$, while its maximal evolution time is of the same order. □

5.7 Applications

In this section, we discuss some applications of our ground state property estimation algorithm. To define an application of the ground state property estimation algorithm, we must specify a Hamiltonian of interest H and an observable of interest O. An example application used in quantum chemistry and materials is the Green's function (see, e.g. [TAWL21]), where $O = a_i(z - (H - E_0)^{-1})a_j^\dagger$. In the following two sections we describe another example from quantum chemistry and materials as well as an example of a linear algebraic subroutine.

5.7.1 Charge density

The primary application of the technique is the estimation of ground state properties of physical systems. Here we describe how to compute the charge density of a molecule, which can be used to compute properties like electric dipole moments of a molecule [RGM+21]. From a second-quantized representation of the electronic system (assuming fixed positions of the nuclear positions), the charge density is determined from the one-particle reduced density matrix as,

$$\rho(\vec{r}) = -e \sum_{p,q} D_{p,q}\phi_p(\vec{r})\phi_q(\vec{r}), \tag{5.74}$$

where e is the electric constant, $D_{p,q}$ is the one-electron reduced density matrix (1RDM) of the ground state, and $\phi_q(\vec{r})$ are the basis wave functions chosen for the second-quantized representation of the electronic system [HJO14]. The 1RDM of the ground state is a matrix of properties of the ground state with each entry defined as

$$D_{p,q} = \langle\psi_0| a_p^\dagger a_q |\psi_0\rangle, \tag{5.75}$$

where a_p are annihilation operators. The operators involved in the 1RDM can each be expressed as a linear combination of unitary operators using the Majorana representation $a_p = \frac{1}{2}(\gamma_{2p} + i\gamma_{2p+1})$, where the γ_k are hermitian and unitary[2],

$$D_{p,q} = \frac{1}{4}\left(\langle\psi_0|\gamma_{2p}\gamma_{2q}|\psi_0\rangle - i\langle\psi_0|\gamma_{2p+1}\gamma_{2q}|\psi_0\rangle + i\langle\psi_0|\gamma_{2p}\gamma_{2q+1}|\psi_0\rangle + \langle\psi_0|\gamma_{2p+1}\gamma_{2q+1}|\psi_0\rangle\right). \tag{5.76}$$

[2]To implement this application on a quantum computer we must represent the unitaries as operations on qubits. For an n-electron system, using the Jordan-Wigner or Bravyi-Kitaev transformation [SRL12], each Majorana operator, and products thereof, can be represented as a Pauli string.

Accordingly, we may use the method of Section 5.5 to estimate each entry of the 1RDM and then obtain the charge density function of the ground state. As a point of comparison, we could alternatively use the variational quantum eigensolver algorithm to prepare an approximation to the ground state and then directly estimate each of the Pauli expectation values. However, there is no guarantee on whether a target accuracy for the ground state approximation can be achieved. Remarkably, the methods introduced in this chapter can be used to ensure a target accuracy in the estimation regardless of the quality of ground state approximation, though possibly at the cost of an increase in runtime.

5.7.2 Quantum linear system solver

In the seminal [HHL09] paper, a quantum algorithm is proposed to generate a quantum state approximately proportional to the solution of a linear system of equations. Namely, given a linear system $A\vec{x} = \vec{b}$, the algorithm produces a quantum state close to $|x\rangle := \frac{\sum_j x_j |j\rangle}{\sqrt{\sum_j |x_j|^2}}$, where x_j's are the entries of $\vec{x} = A^{-1}\vec{b}$. In fact, in many cases, we only need to know $\langle x| M |x\rangle$, where M is a linear operator. For example, in quantum mechanics, many features of $|x\rangle$ can be extracted in this way, including normalization, moments, etc. One approach to solve this problem is first solving the linear system using any quantum linear system solver [HHL09, CKS17, CGJ19, GSLW19] to obtain the state $|x\rangle$ and then performing the measurement of M. However, a shortcoming of this method is that most of the quantum linear system solvers require deep quantum circuits. And hence, the needed quantum resources may not be accessible in the near future.

Recently, a few quantum algorithms [BPLC$^+$19, HBR21, SSO19] were developed to solve linear systems of equations by encoding such a system into an effective Hamiltonian

$$H_G := A^\dagger(I - |b\rangle\langle b|)A, \tag{5.77}$$

whose ground state corresponds to the solution vector $|x\rangle$. We can combine this idea with our ground state property estimation algorithm to get a low-depth algorithm for estimating the properties of linear system solution. More specifically, suppose we can simulate the Hamiltonian H_G for some specified time and we know the normalization factor τ such that the eigenvalues of τH_G are in $[-\pi/3, \pi/3]$. For the operator M, we can assume that M can be decomposed into a linear combination of Pauli operators $M = \sum_{\ell=1}^{L} c_\ell \sigma_\ell$, or we assume that M is given in the block-encoding form. The estimation algorithm has two steps:

1. Run a quantum linear system algorithm (e.g. [SSO19], [AL22], or [LT20b]) with constant precision to prepare an initial state $|\phi_0\rangle$ such that $|\langle\phi_0|x\rangle|^2$ is $\Omega(1)$.

2. Using $|\phi_0\rangle$ from step 1 as the initial state, run Algorithm 24 to estimate $\langle x| M |x\rangle$ within ϵ-additive error for any $\epsilon \in (0, 1)$.

Step 1 takes $\tilde{O}(\kappa)$ time, where κ is the condition number of A. To analyze the computation cost of the second step, we need a lower-bound on the spectral gap of H_G. Since $\langle x| A^\dagger(I - |b\rangle\langle b|)A |x\rangle = 0$, we have $\lambda_0(H_G) = 0$. For the second smallest eigenvalue, since $H_G = A^\dagger A - A^\dagger |b\rangle\langle b| A$, by Weyl's inequality, we have

$$\begin{aligned}\lambda_1(H_G) &\geq \lambda_0(A^\dagger A) - \lambda_1(A^\dagger |b\rangle\langle b| A)\\ &= \lambda_0(A^\dagger A),\end{aligned} \tag{5.78}$$

where the second step follows from $A^\dagger |b\rangle\langle b| A$ is rank-1. Due to the normalization, the smallest (normalized) singular value of A is $\Omega(\kappa^{-1})$. Hence, we have $\gamma = \Omega(\kappa^{-2})$.

By Theorem 5.13, the maximal evolution time of the Hamiltonian will be $\widetilde{O}(\kappa^2)$. To further improve the circuit depth, we may apply the gap amplification technique [SB13, SSO19] to quadratically increase the spectral gap of H_G. Specifically, consider the following family of Hamiltonians:

$$\bar{H}'_G(s) := \sigma^+ \otimes \bar{A}^\dagger(s)(I - |\bar{b}\rangle\langle\bar{b}|) + \sigma^- \otimes (I - |\bar{b}\rangle\langle\bar{b}|)\bar{A}(s), \tag{5.79}$$

where $\sigma^{\pm} = (X \pm iY)/2$, $\bar{A}(s) := (1-s)Z \otimes I + sX \otimes A$, $\left|\bar{b}\right\rangle := \left|+\right\rangle \left|b\right\rangle$ and $s \in [0,1]$. Note that these Hamiltonians act on the original system and two ancilla qubits. Then we have

$$(\bar{H}'_G(s))^2 = \begin{bmatrix} \bar{H}_G(s) & 0 \\ 0 & (I - \left|\bar{b}\right\rangle \left\langle\bar{b}\right|)\bar{A}(s)\bar{A}^{\dagger}(s)(I - \left|\bar{b}\right\rangle \left\langle\bar{b}\right|) \end{bmatrix}, \tag{5.80}$$

where

$$\bar{H}_G(s) := \bar{A}^{\dagger}(s)(I - \left|\bar{b}\right\rangle \left\langle\bar{b}\right|)\bar{A}(s). \tag{5.81}$$

As shown in [SSO19], the eigenvalues of $\bar{H}'_G(s)$ are

$$\left\{0, 0, \pm\sqrt{\lambda_1(s)}, \pm\sqrt{\lambda_2(s)}, \dots\right\}, \tag{5.82}$$

where $\lambda_j(s)$'s are the nonzero eigenvalues of $\bar{H}_G(s)$. Furthermore, let $\left|x(s)\right\rangle$ be the unique ground state of $\bar{H}_G(s)$. Note that $\left|x(0)\right\rangle = \left|-\right\rangle\left|b\right\rangle$ and $\left|x(1)\right\rangle = \left|+\right\rangle\left|x\right\rangle$. Then the ground space of $\bar{H}'_G(s)$ is spanned by $\{\left|0\right\rangle\left|x(s)\right\rangle, \left|1\right\rangle\left|\bar{b}\right\rangle\}$. In addition, for $s = 1$, one can use Weyl's ineqality to show that $\lambda_1(1) \geq \kappa^{-2}$, which implies that the smallest nonzero eigenvalue of $\bar{H}'_G(1)$ is $\Omega(\kappa^{-1})$, as desired.

We can use the algorithm in [SSO19] to prepare a state that has $\Omega(1)$ overlap with $\left|0\right\rangle\left|x(1)\right\rangle = \left|0\right\rangle\left|+\right\rangle\left|x\right\rangle$ in $\tilde{O}(\kappa)$ time. Specifically, this algorithm starts with the state $\left|0\right\rangle\left|x(0)\right\rangle = \left|0\right\rangle\left|-\right\rangle\left|b\right\rangle$, performs a sequence of unitary operations of form $e^{-it_k\bar{H}'_G(s_k)}$ on it, and outputs a state ϵ-close to $\left|0\right\rangle\left|x(1)\right\rangle$ in $\tilde{O}(\kappa\epsilon^{-1})$ time. Here we set $\epsilon = \Theta(1)$ and the time cost of this procedure is $\tilde{O}(\kappa)$.

After obtaining a state $\left|\phi_0\right\rangle$ that has $\Omega(1)$ overlap with $\left|0\right\rangle\left|+\right\rangle\left|x\right\rangle$, we run Algorithm 24 on $\left|\phi_0\right\rangle$, $\bar{H}'_G(1)$ and $\tilde{M} := \left|0\right\rangle\left\langle 0\right| \otimes \left|+\right\rangle\left\langle +\right| \otimes M$ to estimate $\left\langle 0,+,x\right|\tilde{M}\left|0,+,x\right\rangle = \left\langle x\right|M\left|x\right\rangle$. Notice that since we know the ground state energy of $\bar{H}'_G(1)$ is zero, we do not need to first estimate the ground state energy using Algorithm 22. Instead, we directly evaluate the O-weighted CDF at zero. Therefore, by Theorem 5.17, we get the following result:

Corollary 5.18 (Quantum linear system solution property estimation)**.** *For a linear system $A\vec{x} = \vec{b}$, suppose A has singular values in $[-1, -1/\kappa] \cup [1/\kappa, 1]$ for $\kappa > 1$, and the eigenvalues of $\bar{H}'_G(1)$ (Eq. (5.79)) are in $[-\pi/3, \pi/3]$. Furthermore, suppose we can implement $e^{-it\bar{H}'_G(s)}$ (Eq. (5.79)) in $\tilde{O}(t)$ time for all $s \in [0, 1]$.*

Then, for any linear operator M given by its α-block encoding unitary U_M, and for any $\epsilon \in (0, 1)$, the expectation value $\langle x| M |x\rangle$ can be estimated with ϵ-additive error with high probability such that:

- *the depth of each circuit is $\widetilde{O}(\kappa)$.*
- *the expected total runtime is $\widetilde{O}(\kappa\epsilon^{-2}\alpha^2)$.*

For comparison, the algorithm in [SSO19] needs $\widetilde{O}(\kappa\epsilon^{-1})$ circuit depth to obtain a state that is ϵ-close to $|x\rangle$, which is larger than ours. Moreover, to estimate $\langle x| M |x\rangle$, even with amplitude estimation, it still needs $\Omega(\epsilon^{-1})$ copies of the state to achieve ϵ-additive error. Hence, its total runtime will be $\widetilde{O}(\kappa\epsilon^{-2})$, nearly matching our result (ignoring the dependence on the α factor).

5.8 Discussion and Outlook

We have shown a quantum-classical hybrid algorithm for estimating properties of the ground state of a Hamiltonian, such that the quantum circuit depth is relatively small and only poly-logarithmically depends on ϵ^{-1}. Therefore, the algorithm has a significant advantage in high-accuracy estimation, and it is possible to be implemented in early fault-tolerant devices. In practice, our algorithm can solve many important tasks by combining with some initial state preparation methods (e.g., VQE or QAOA). In this chapter, we provide two examples, one in quantum chemistry and another in solving linear systems. And we believe more applications will be explored in the future.

Another important direction is to improve the total evolution time of our algorithm which quadratically depends on ϵ^{-1}. The blowup comes from evaluating the O-weighted CDF in high precision and a trade-off between maximal evolution time and total evolution time. However, this does not meet the Heisenberg-limit of linear dependence on ϵ^{-1} for generic Hamiltonians [AA17]. In our main result (Theorem 5.13), the $\epsilon^{-2}\eta^{-2}$ factor comes from the number of samples needed to reduce the estimator's error to $O(\epsilon\eta)$. Amplitude estimation can be used to reduce this number of samples and the total evolution time. However, this comes at the cost of significantly increasing the maximal evolution time, which could require large fault-tolerant overheads for reliable implementation. A strategy to achieve improved performance that is more amenable to early fault tolerant quantum computers is to use recently introduced "enhanced sampling" techniques [WKJC21]. If λ characterizes the fidelity decay rate of the circuit as deeper circuits are used, then we would expect to need a maximal evolution time of $O(\lambda^{-1}\gamma^{-1})$ and an total evolution time of $O(\lambda\gamma^{-1}\epsilon^{-2}\eta^{-2})$. Note that because this approach incorporates the impact of error into the algorithm, the maximal evolution time is of no concern. Rather than being a cost that needs monitoring, the maximal evolution time is chosen by the algorithm to minimize the total evolution time. With this, we expect that as the quality of devices is improved, the performance of the algorithm improves proportionally. We note that a similar approach can also be applied to improve the total evolution time in [LT22] from $\widetilde{O}(\epsilon^{-1}\eta^{-2})$ to $\widetilde{O}(\lambda\epsilon^{-1}\eta^{-2})$.

This work fits into the paradigm of "beyond the ground state energy" and studies more general properties of the ground state. Can we go further beyond the ground state? Some prior works have explored the estimation of such kind of properties of Hamiltonian. For example, Brown, Flammia, and Schuch [BFS11] studied the density of states. Jordan, Gosset, and Love [JGL10] focused on the energy of excited states. Gharibian and Sikora [GS18] identified the energy barriers. Watson and Bausch [WB21] explored detecting phase transitions via order parameters. In general, for an unknown Hamiltonian, these estimation problems will be hard. An interesting open problem is, given some prior knowledge of the Hamiltonian, can we design efficient or low-depth quantum algorithms for estimating Hamiltonian properties beyond ground state?

Algorithm 22 Ground State Energy Estimation

```
1: procedure EstimateGSE($\epsilon, \tau, \eta, \nu$)
2:                                                        ▷ Initialization
3:     $d \leftarrow O(\delta^{-1}\log(\delta^{-1}\eta^{-1}))$, $\delta \leftarrow \tau\epsilon$
4:     for $i \leftarrow -d, \ldots, d$ do
5:         $\hat{F}_i \leftarrow \hat{F}_{d,\delta,i}$
6:         Compute $\theta_i$, the phase angle of $\hat{F}_i$
7:     end for
8:     $\mathcal{F} \leftarrow \sum_{|i|\le d} |\hat{F}_i|$
9:     $N_b \leftarrow \Omega(\log(1/\nu) + \log\log(1/\delta))$, $N_s \leftarrow O(\eta^{-2}\log^2(d))$
10:                                         ▷ Sampling from the quantum circuit
11:    for $k \leftarrow 1, \ldots, N_b N_s$ do
12:        Independently sample $J_k \sim [-d, d]$ with $\Pr[J_k = j] \propto |\hat{F}_j|$
13:        Measure $(X_k, Y_k)$ by running the quantum circuit with (Figure. 5.1) parameter $k$
14:        $Z_k \leftarrow X_k + iY_k$
15:    end for
16:                                                ▷ Classical post-processing
17:    $x_L \leftarrow -\pi/3$, $X_R \leftarrow \pi/3$
18:    while $x_R - x_L > 2\delta$ do                              ▷ Invert CDF
19:        $x_M \leftarrow (x_L + x_R)/2$
20:        for $r \leftarrow 1, \ldots, N_b$ do
21:            $\overline{G}_r \leftarrow \frac{\mathcal{F}}{N_s}\sum_{k=(r-1)N_s+1}^{rN_s} Z_k e^{i(\theta_{J_k} + J_k x_M)}$   ▷ Multi-level Monte Carlo method
22:        end for
23:        if $|\{r : \overline{G}_r \ge (3/4)\eta\}| \le N_b/2$ then
24:            $x_R \leftarrow x_M + (2/3)\delta$
25:        else
26:            $x_L \leftarrow x_M - (2/3)\delta$
27:        end if
28:    end while
29:    return $(x_L + x_R)/2$
30: end procedure
```

Algorithm 23 Ground State Property Estimation (Commutative Case)

```
 1: procedure EstimateGSProp($\epsilon, \tau, \eta, \gamma, \nu$)
 2:     $\delta \leftarrow O(\tau\gamma)$, $d \leftarrow O(\delta^{-1}\log(\delta^{-1}\epsilon^{-1}\eta^{-1}))$
 3:     for $j \leftarrow -d, \ldots, d$ do
 4:         Compute $\hat{F}_j := \hat{F}_{d,\delta,j}$ and $\theta_j$
 5:     end for
 6:                                         ▷ Estimate the ground state energy
 7:     $x^\star \leftarrow$ EstimateGSE($\gamma/8, \tau, \eta, \nu/10$)
 8:     $x_{\mathsf{good}} \leftarrow x^\star + \tau\gamma/2$
 9:                           ▷ Generate samples from the Hadamard test circuits
10:     $N_g \leftarrow O(\log(1/\nu))$, $K \leftarrow O(\epsilon^{-2})$
11:     for $k \leftarrow 1, \ldots, N_g K$ do
12:         Sample $(Z_k, J_k)$ from the quantum circuit (Figure 4.3)
13:         Sample $(Z'_k, J'_k)$ from the quantum circuit (Figure 5.2)
14:     end for
15:                                                          ▷ Estimate $p_0$
16:     for $i \leftarrow 1, \ldots, N_g$ do
17:         $\overline{G}_i \leftarrow \frac{1}{K}\sum_{j=1}^{K} \overline{G}(x_{\mathsf{good}}; Z_{(i-1)K+j}, J_{(i-1)K+j})$
18:     end for
19:     $\overline{p_0} \leftarrow \mathrm{median}(\overline{G}_1, \ldots, \overline{G}_{N_g})$
20:                                                       ▷ Estimate $p_0 O_0$
21:     for $i \leftarrow 1, \ldots, N_g$ do
22:         $\overline{G}'_i \leftarrow \frac{1}{K}\sum_{j=1}^{K} \overline{G}(x_{\mathsf{good}}; Z'_{(i-1)K+j}, J'_{(i-1)K+j})$
23:     end for
24:     $\overline{p_0 O_0} \leftarrow \mathrm{median}(\overline{G}'_1, \ldots, \overline{G}'_{N_g})$
25:     return $\overline{p_0 O_0}/\overline{p_0}$
26: end procedure
```

Algorithm 24 Ground State Property Estimation (General Case)

```
1: procedure EstimateGSProp($\epsilon, \tau, \eta, \gamma, \nu$)
2:     $\delta \leftarrow O(\tau\gamma)$, $d \leftarrow O(\delta^{-1}\log(\delta^{-1}\epsilon^{-1}\eta^{-1}))$
3:     for $j \leftarrow -d, \dots, d$ do
4:         Compute $\hat{F}_j := \hat{F}_{d,\delta,j}$ and $\theta_j$
5:     end for
6:                                             ▷ Estimate the ground state energy
7:     $x^\star \leftarrow$ EstimateGSE($\gamma/8, \tau, \eta, \nu/10$)
8:     $x_{\mathsf{good}} \leftarrow x^\star + \tau\gamma/2$
9:                                  ▷ Generate samples from the Hadamard test circuits
10:    $B \leftarrow O(\log(1/\nu))$, $K \leftarrow \widetilde{O}(\epsilon^{-2})$
11:    for $k \leftarrow 1, \dots, BK$ do
12:        Sample $(Z_k, J_k)$ from the quantum circuit (Figure 4.3)
13:        Sample $(Z''_k, J''_{k,1}, J''_{k,2})$ from the quantum circuit (Figure 5.4)
14:    end for
15:                                                              ▷ Estimate $p_0$
16:    for $i \leftarrow 1, \dots, B$ do
17:        $\overline{G}_i \leftarrow \frac{1}{K}\sum_{j=1}^{K} \overline{G}(x_{\mathsf{good}}; Z_{(i-1)K+j}, J_{(i-1)K+j})$
18:    end for
19:    $\overline{p_0} \leftarrow \mathrm{median}(\overline{G}_1, \dots, \overline{G}_B)$
20:                                                           ▷ Estimate $p_0 O_0$
21:    for $i \leftarrow 1, \dots, B$ do
22:        $\overline{G}''_i \leftarrow \frac{1}{K}\sum_{j=1}^{K} \overline{G_2}(x_{\mathsf{good}}, x_{\mathsf{good}}; Z''_{(i-1)K+j}, J''_{(i-1)K+j,1}, J''_{(i-1)K+j,2})$   ▷
   Eq. (5.53)
23:    end for
24:    $\overline{p_0 O_0} \leftarrow \mathrm{median}(\overline{G}''_1, \dots, \overline{G}''_B)$
25:    return $\overline{p_0 O_0}/\overline{p_0}$
26: end procedure
```

Chapter 6: QAOA for Network-Flow Optimization

6.1 Introduction

Combinatorial optimization (CO) tasks present many classically-hard computational problems, and abound in practical applications from vehicle routing to resource allocation, job scheduling, portfolio optimization, and integrated circuit layout. Finding optimal solutions to many practically relevant classes of CO problems is an NP-complete task, which is effectively intractable for large problems. In the past decades, quantum computers promise tantalizing speedups on certain classically-hard computational problems, such as integer factoring [Sho94] and structured searching [Gro96]. Unfortunately, barring an upheaval of complexity theoretic dogma, quantum optimization algorithms are not expected to efficiently yield optimal solutions to NP-hard problems. However, for classical optimization on hard problems, one typically aims for reasonable but suboptimal approximations, and tremendous effort has been put into improving the quality of approximate solutions. In a similar vein, there is widespread hope that quantum-heuristics could yield better approximate solutions than their classical counterparts.

This hope has been largely fueled by the introduction of the Quantum Approximate Optimization Algorithm (QAOA), a hybrid classical/quantum framework originally motivated as a variational spin-off of the Quantum Adiabatic Algorithm (QAA) [FGG14b]. QAOA consists of p-rounds of stroboscopic alternation between a classical cost Hamiltonian and a quantum mixing Hamiltonian, with time intervals for each evolution treated as variational parameters that are classically optimized. While it was initially suggested that even a single round ($p = 1$) QAOA could provide a quantum-improvement over classical state-of-the-art [FGG14a], the quantum/classical gap was quickly closed [BMO+15], and there is growing evidence [Has19, FGG20a, FGG20b, BKKT20] that p must generically scale with the problem-size in order to achieve improved approximate solutions. Due to the difficulty of analyzing

QAOA-performance at large-p, establishing rigorous evidence of quantum advantage remains elusive, and the practical value of QAOA will likely be decided empirically (like many successful classical heuristic methods).

Making QAOA into a successful quantum heuristic will require advances in problem encoding, and algorithm efficiency. A key weakness of traditional QAOA is that many relevant CO problems impose constraints among variables, which are not respected by the QAOA heuristic. A typical approach to QAOA would be to map a CO problem into a binary integer linear program (BILP), whose objective function is mapped to an Ising-like spin model that can be implemented on quantum hardware. In this formulation, constraints are typically softly enforced by adding a term to the cost Hamiltonian that energetically penalizes constraint violations. This approach is frequently inefficient, as it can result in exploration of an exponentially-large (in problem size) set of infeasible (constraint-violating) configurations, which has been shown to dramatically hamper performance [WRDR20].

An alternative technique is to modify the QAOA procedure to automatically satisfy constraints throughout the algorithm. In [HWO+19, WRDR20], this approach was used to tackle graph-coloring problems (among others), where a number-conserving mixing Hamiltonian was designed to preserve a one-hot encoding structure. Due to the intimate connection between symmetries and conservation laws, this highlights a connection between physical symmetries and constraints in CO problems, and suggests that physics-inspired solutions may be fruitful.

In this chapter, we exploit another common “symmetry" found in physical systems: gauge-invariance [1], to implement a constraint-satisfying mixer for network flow problems. Network flow problems are defined on graphs, where each link of a graph has a directed flow of “goods” that takes real or integer values. In practice, flow could represent an amount of vehicles, goods, communication packets, etc., being

[1]Strictly speaking, gauge invariance is not an ordinary symmetry, however the analogy is frequently useful.

transported through the network. Real-valued flow problems tend to admit classically efficient solutions via linear programming, whereas multi-commodity integer flow problems are often classically hard. Integer flow problems have a wide array of applications from vehicle routing, traffic congestion minimization, and package delivery, to communication network optimization. Each of these problem formulations share a common constraint structure: the amount of flow entering a vertex must match the total outgoing flow, plus (minus) a fixed amount at certain source (sink) nodes.

This flow structure is a discrete analog of Gauss law in electromagnetism: $\nabla \cdot \boldsymbol{E} = \rho$, where ρ is the charge density, if we re-interpret the electric field $\boldsymbol{E}$ as a flow emanating out of a node, and the charge ρ as the amount of sourced or sinked goods. The central idea of this chapter will be to exploit this analogy to develop a lattice quantum electrodynamics (QED) inspired QAOA-mixer that automatically preserves network-problem flow constraints.

The structure of the chapter is organized as follows: we briefly summarize QAOA from the generalized perspective advocated in [FGG14a], and review the structure of lattice-QED. We then establish a direct relationship to flow problems on finite-dimensional graphs, and define a constraint-preserving generalization of QAOA using a QED-style mixer. We numerically compare the performance of modified QED-QAOA and the original (X-mixer) QAOA on a (classically easy) flow maximization problem, and show that the quality of approximate solutions increases in a way that is consistent with exponential-in-problem size scaling. We then explore QED-mixer performance on classically-hard traffic congestion minimization problems, and study the behavior with increasing problem size and number of QAOA rounds. A key step in the algorithm is preparing an initial constraint-preserving state that is a quantum superposition including all constraint-preserving states. Unlike the original QAOA, where the X-mixer ground-state can be accomplished with a transversal set of single-qubit rotations, the QED-mixer ground-state is more complicated. We explore and compare multiple strategies for initial state preparation, and find, perhaps surprisingly, that the QED-mixer ground-state is not optimal, suggesting a departure from the adiabatic-algorithm reasoning often used to motivate QAOA.

6.2 Quantizing Network Flow Problems

To set the stage, we briefly review the constraint structure of network flow problems, introduce the specific problem types that we will use to illustrate the QED-inspired QAOA approach, and describe an implementation of their cost function as a quantum Hamiltonian acting on qudits.

6.2.1 Constraints in flow problems

A flow problem is defined on a graph $\mathcal{G}$ with vertices $\mathcal{V}$ and edges $\mathcal{E} = \{(u, v)|\ u, v \in \mathcal{V}$ are connected$\}$. Here $\mathcal{G}$ is required to be a directed graph by many versions of flow problems, such as max-flow problems, but can be undirected in other cases like EDP. We denote the total number of vertices as $|\mathcal{V}|$, and the number of edges as $|\mathcal{E}|$. On each edge, we define a flow: $f(u, v) \in \mathbb{F}$ taking value in some field $\mathbb{F}$, with $f(u, v) = -f(v, u)$. To facilitate implementation on discrete-leveled quantum computing systems, in this chapter we will specialize to integer flows of k-different commodities (i.e. $\mathbb{F} = \mathbb{Z}^k$). We define the vertex from which a commodity originates or terminates as a source or sink node respectively. We denote the sets of source and sink nodes as $\{s_i\}_{i=1}^k$ and $\{t_i\}_{i=1}^k$, and the amount of flow to be delivered for the i^{th} source-sink pair as d_i.

While there are a large variety of flow-problem formulations, they all share a common constraint structure. Namely, valid flows may begin and terminate only on source and sink nodes, respectively:

$$\sum_{v:(u,v)\in\mathcal{E}} f_i(u, v) = d_i(\delta_{u,s_i} - \delta_{u,t_i}) \quad \forall\ u \in \mathcal{V}. \tag{6.1}$$

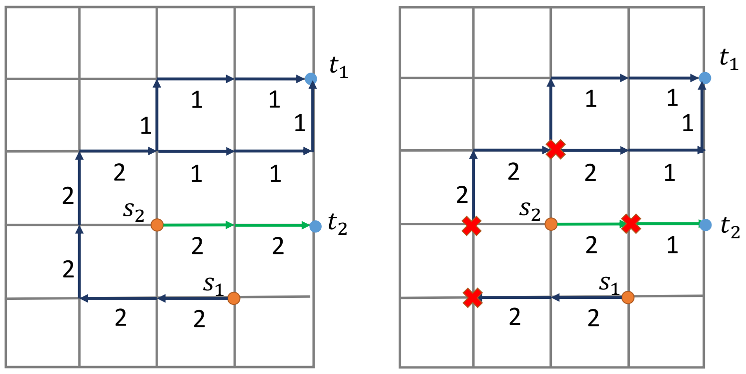


Figure 6.1: **Example flows on a 5×5 grid graph** A feasible network flow configuration (L) and an unfeasible configuration(R): the arrows stand for flow directions, and different flows are distinguished by colors with numbers representing the amount of flow on each edge(a certain assignment of the flows in the graph is called a *configuration*).

Figure. 6.1 illustrates selected examples of valid and invalid flow configurations.

In addition, many flow problems impose additional capacity constraints on how many of each type of commodities may flow through a particular edge:

$$\sum_{i \in [k]} |f_i(u,v)| \leq c(u,v) \quad \forall\ (u,v) \in \mathcal{E}, \tag{6.2}$$

where $c(u,v) \in \mathbb{Z}_+$ is referred to as the edge-capacity: the total amount of all types of flows cannot exceed the capacity on that edge.

Flow problems come in many varieties. Some, such as the single-commodity max flow problem, have efficient classical algorithms. However, many practical problems require introducing multiple commodities and imposing finite edge-capacities, which typically results in hard optimization problems. For example, the problem of maximizing capacitated integer flow was proven to be NP-complete even for only two source-sink pairs [EIS75].

6.2.1.1 Qudit encoding

To encode integer flow problems onto quantum hardware, we imagine using a register of $(2d_i + 1)$-level qudits (possibly encoded into ordinary qubits using, e.g. binary or one-hot encoding) for each commodity and each edge $(u, v) \in \mathcal{E}$, with the qudit computational basis states $\{|-d_i\rangle, \dots, |-1\rangle, |0\rangle, |1\rangle \dots |d_i\rangle\}$ indicating the amount of flow on that link [2] We note that, for this encoding the dimension of the entire Hilbert space is thus the same as the number of all possible configurations on the graph, which is $\prod_i (2d_i + 1)^{|\mathcal{E}|}$.

The total Hilbert space of this encoded system contains exponentially many infeasible configurations that violate the flow constraints (Eq. (6.1)). The precise ratio of feasible (flow-conserving) to infeasible (flow-violating) solutions varies by graph; however, it is generally exponentially small in $|\mathcal{V}|$. To see this, note that, the distance between a pair of randomly chosen source and sink points is typically poly($|\mathcal{V}|$), and for each valid path from source to sink, removing any edge along the path from source to sink would result in an infeasible solution, resulting in combinatorially many infeasible solutions for each feasible one.

6.2.1.2 Flow operators

We also introduce quantum flow operators on each edge $e \in \mathcal{E}$, and for each commodity type $i = 1 \dots k$:

$$E_e^{(i)} = \sum_{f=-d_i}^{d_i} f|f\rangle\langle f|_e \otimes \mathbb{1}_{e' \neq e} \tag{6.3}$$

where the symbol E anticipates an analogy with electric field operators in lattice-QED. Applying $E_e^{(i)}$ on a state would just return the amount of flow on edge e.

[2]One could partially enforce capacity constraints by restricting the basis to $\{|-c(u, v)\rangle \dots |c(u, v)\rangle\}$ for each commodity, however, this choice would conflict with our scheme for producing a valid initial state.

Furthermore, we denote the operator whose eigenstates are equal weighted superpositions of flow values as:

$$X_e^{(i)} = \sum_{f,f'=-d_i}^{d_i} |f'\rangle\langle f|_e \otimes \mathbb{1}_{e'\neq e} \tag{6.4}$$

which is the natural qudit analog of the Pauli-X operator.

We also define the total flow of all goods on edge $e \in \mathcal{E}$, as $E_e \equiv \sum_{i=1}^{k} E_e^{(i)}$, and similarly $X_e \equiv \sum_{i=1}^{k} X_e^{(i)}$. The conventional QAOA mixer is built from $H_M = -\sum_e X_e$, which indiscriminately mixes between feasible and infeasible solutions, and has a tendency to get "lost" in the exponentially larger infeasible parts of Hilbert space.

6.2.2 The edge-disjoint path problem

The main problem we will consider in this chapter is a variant of the traffic congestion minimization problem known as the edge-disjoint paths problem (EDP), often regarded as a particularly clean problem that characterizes the NP-hardness of flow optimization. Qualitatively, the frequently studied optimization version of EDP seeks to route k different commodities without "congestion", i.e., without multiple commodities flowing through the same edge:

EDP: *Given a undirected graph $G(V, E)$ with k source/sink-pairs (s_i, t_i), find k paths connecting s_i and t_i for all $i \in [k]$ such that the maximum of congestion in each edge is minimized.*

Since the maximum of congestion is a global function that is hard to implement on a quantum circuit, we can reformulate EDP's cost function by locally penalizing all congested edges instead:

$$\min \quad C \equiv \sum_{(u,v)\in E} \max\left\{0, \sum_{i\in[k]} |f_i(u,v)| - 1\right\} \quad s.t.$$
$$\sum_{v:(u,v)\in\mathcal{E}} f_i(u,v) = d_i(\delta_{u,s_i} - \delta_{u,t_i}) \quad \forall\, u \in \mathcal{V}. \tag{6.5}$$

In other words, in this version one aims at minimizing the total amount of congestion on all edges instead of the maximum. Notice that an optimal solution with $C = 0$ will still be a solution of the EDP problem (with no congestion), whereas $C > 0$ configurations may be regarded as approximate solutions of the relaxed EDP.

EDP has been shown to be NP-hard even with a rather modest scaling of commodity types ($k \sim \log|\mathcal{V}|$) [CL12]. We restrict our attention to EDPs on planar graph, where the problem remains NP-hard [CL12].

To convert the EDP cost-function into a quantum Hamiltonian, we reformulate the cost function into an analytic form, and introduce the EDP cost Hamiltonian (using the encoding described above):

$$H_{\text{C,EDP}} = \sum_{e\in E}\left[\frac{(2E_e^2 - 1)^2 - 1}{48}\right] \tag{6.6}$$

which has vanishing energy for non-congested links (with $E_e = 0, 1$) and penalizes higher congestion. The normalization is chosen such that minimally congested links with $E_e = \pm 2$ have one unit of energy cost. Eq. (6.6) is a reformulation of the cost function in Eq. (6.5), whose flow constraints will be dealt with in later sections.

6.2.3 The single source shortest path problem

For classical simulations, the fully unconstrained multi-commodity Hilbert space quickly becomes intractable. Therefore, to benchmark the modified QAOA performance against the original formulation, we also consider a much simpler class of single source shortest path problem (SSSP), which seek the shortest path (on a weighted graph) between a single source and sink with unit demand ($d = 1$):

SSSP: *Given a weighted undirected graph $G(\mathcal{V}, \mathcal{E})$, with weights $\{w_e : e \in \mathcal{E}\}$, and a single pair of source and sink vertices $s, t \in \mathcal{V}$, find the minimal length path connecting s and t where length is defined as the sum of the weights along the path.*

Notice that SSSP can be defined on either directed, undirected or mixed graphs, and we choose to study the undirected version for consistency with the study of EDP. Efficient classical algorithms for SSSP [Bel58, Dij59] are textbook-standard materials (see also [KS19] for a quantum algorithm for directed acyclic graphs). In this chapter, we do not aim to improve solution of SSSP, but only to use this problem as a benchmark to compare the performance of different QAOA mixers in the graph routing problem. Importantly, none of the QAOA strategies we test take advantage of the classically efficient solution, providing a fair comparison.

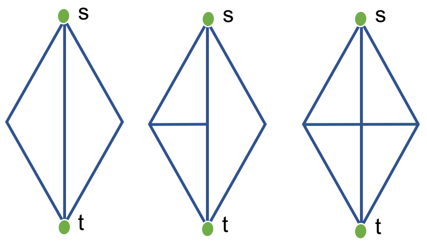


Figure 6.2: **Triangle graphs used in the study of SSSP problem** In SSSP simulations we look for the shortest (lowest-weight) path from the top node to the bottom node, where weight on each edge is randomly assigned.

Since SSSP is a direct analogy of EDP (at $k = 1$), we can use the same encoding scheme and write the cost Hamiltonian as

$$H_{\mathrm{C,SSSP}} = \sum_{e \in \mathcal{E}} w_e (E_e)^2 \tag{6.7}$$

where w_e denote the edge e's weight. Since only a single type of flow with demand 1 presents in the problem, the valid flow values are just -1,0,1 on any edge. The size of the Hilbert space of SSSP is thus $3^{|\mathcal{E}|}$, which, for large graphs, is far less than the $3^{k|\mathcal{E}|}$

# of Triangles	Total # States	# Feasible States	Feasible fraction
2	729	3	4.1×10^{-3}
3	2187	4	1.8×10^{-3}
4	6561	8	1.2×10^{-3}

Table 6.1: **A comparison between total and feasible Hilbert space dimension** Total Hilbert space dimension ($|\mathcal{H}_{\text{tot}}|$) and feasible sub-space dimension ($|\mathcal{H}_{\text{f}}|$), and ratio of feasible to total states $|\mathcal{H}_{\text{f}}|/|\mathcal{H}_{\text{tot}}|$.

(with $k \geq 2$) required for EDP, allowing us to classically simulate relatively larger instances.

With these problem classes in hand, we now turn to the task of modifying the QAOA algorithm to preserve the network flow constraints, beginning with a brief review of QAOA to set notation.

6.3 From QAOA to Lattice QED

QAOA is designed to sample from low-cost states of a cost Hamiltonian H_C which is diagonal in the computational basis, and represents the objective function of the optimization problem in question. In its original incarnation [FGG14a], the initial state $|\psi_0\rangle$, is chosen to be the ground-state of a mixing Hamiltonian $H_M = H_{M,X}$ with:

$$H_{M,X} = -\sum_i X_i, \tag{6.8}$$

which we will refer to as the "X-mixer." Subsequent generalizations [HWO+19] considered more complicated forms of H_M designed to preserve constraints of various forms. The algorithm proceeds by evolving:

$$|\psi_p(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle = \prod_{j=1}^{p} e^{-i\beta_j H_M} e^{-i\gamma_j H_C} |\psi_0\rangle \tag{6.9}$$

to generate a variational wave function characterized by real-parameters $\{\gamma_i\}$ and $\{\beta_i\}$ $(i = 1, 2, ...p)$, which are classically optimized (using the classical routine of ones choice) to minimize the expected cost: $\boldsymbol{\gamma}_*, \boldsymbol{\beta}_* = \arg\min \varepsilon_C$, where:

$$\varepsilon_C \equiv \langle \psi_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) | H_C | \psi_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) \rangle. \tag{6.10}$$

This biases the wave-function amplitude of $|\psi_p(\boldsymbol{\gamma}_*, \boldsymbol{\beta}_*)\rangle$ towards low-cost configurations, such that repeated sampling from this state preferentially yields low-cost solutions.

In the limit of infinite p, QAOA contains Quantum Adiabatic Algorithm (QAA) as a subset of possible solutions and is guaranteed to find the exact optimum. For hard problem instances, precisely following the adiabatic path may require p to grow super-exponentially with problem-size, but it is hoped that approximate short-cuts to this adiabatic solution may be variationally identified with far lower p.

To apply this formalism, one must first map the optimization problem variables onto qubits, such that the cost for each qubit configuration can be computed in a local manner. For constrained optimization problems, this often results in a wasteful encoding in which many qubit states do not satisfy the feasibility constraints. One possible strategy would be to energetically penalize the constraint violation by introducing a penalty term into H_C for unsatisfied constraints. While straightforward in its implementation, this strategy results in wasteful exploration of (typically exponentially many) configurations corresponding to infeasible solutions, degrading algorithm performance. An alternative option [HWO+19] is to identify an alternate mixing term H_M which automatically preserves the constraint structure. Then, if an initial state can be prepared to satisfy all constraints, the algorithm will only search inside the feasible subspace. In what follows, we focus on the constraints common to a large variety of network flow problems and show how to encode them into an appropriate constraint-preserving mixer inspired by lattice-QED.

6.3.1 Lattice QED Hamiltonian

The flow constraints described in Eq. (6.5) are of precisely the same form as Gauss's law for lattice-QED, if we interpret each commodity flow as a different "flavor" of electric field, and the corresponding sources and sinks as positive and negative charge d_i. This suggests that we can use gauge-invariant lattice-QED Hamiltonians to implement constraint-preserving mixers for the network flow QAOA. Here, we briefly review some relevant lattice-QED notations and formalisms. In what follows, we specialize them to planar graphs, although our construction generalizes to arbitrary finite-dimensional graphs (but would become infeasible for fully-connected graphs). For notational simplicity, we initially suppress the commodity ("flavor") label.

The Hamiltonian formulation of the (compact) lattice QED, on a planar graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, is defined by introducing discrete analogs of the continuum electric field $\boldsymbol{E}(\boldsymbol{r})$ and vector potential $\boldsymbol{A}(\boldsymbol{r})$. Specifically, a (gauge-redundant) Hilbert space is defined by electric field operators $E_{uv} = -E_{vu}$ on each edge $(u, v) \in \mathcal{E}$ whose eigenstates are denoted $|e_{uv}\rangle$ with $e_{uv} \in \mathbb{Z}$. Electric fields are oriented such that $E_{vu} = -E_{vu}$. The conjugate operator to E_{uv} is denoted by $e^{-iA_{uv}}$, which raises or lowers the electric field:

$$e^{iA_{uv}} E_{uv} e^{-iA_{uv}} = E_{uv} + 1, \tag{6.11}$$

and $[e^{-iA_{uv}}, E_{wx}] = 0$ for $(w, x) \neq (u, v)$.

Physical states are defined by projecting onto subspace that satisfies a lattice analog of the continuum Gauss' law $\nabla \cdot \boldsymbol{E}(\boldsymbol{r}) = \rho(\boldsymbol{r})$, i.e. $\sum_{u:(u,v)\in\mathcal{E}} E_{uv} = \rho_u$, which is precisely the same form as the flow constraint (Eq. (6.1)), provided that we equate the electrical charge with demand, d. The Gauss' law is equivalent to demanding invariance under gauge transformations:

$$e^{-iA_{uv}} \to e^{-i\phi_u} e^{-iA_{uv}} e^{i\phi_v} \tag{6.12}$$

$$|\psi\rangle \to e^{i\sum_{u\in\mathcal{V}} \phi_u \rho_u} |\psi\rangle \tag{6.13}$$

for any vertex-dependent phases $e^{i\phi_v} \in U(1)$.

A special role is played by gauge invariant, Wilson loop operators, $U_\Gamma = e^{-i\oint_\Gamma \vec{A}\cdot d\vec{\ell}}$, which measure the magnetic flux through a closed oriented loop Γ, where we use integral notation to indicate the product of $e^{-iA_{uv}}$ over all links (u, v) on the perimeter of Γ, with orientation along that of Γ. On planar graphs, which have trivial homology, an arbitrary Wilson loop can be decomposed into a product of small loop operators circling the elementary faces (plaquettes) of the graph, which we label by $\mathcal{F}$.

For dimensions $d > 2$, ordinary Maxwell electrodynamics emerges as the continuum and weak-coupling limit of the minimal gauge invariant Hamiltonian:

$$H_{\text{Maxwell}} = \frac{K}{2}\sum_{uv\in\mathcal{E}} E_{uv}^2 - \sum_{f\in\mathcal{F}}(U_f + U_f^\dagger) \tag{6.14}$$

where U_f denotes the Wilson loop encircling face f in the right-handed sense, and K is a coupling constant. The first term represents an electric field line tension, whereas the second gives an energy cost to magnetic flux (which produces quantum dynamics for electric fields). For $d = 2$, the lattice-QED systems is confined by monopole/instanton proliferation for any non-zero electric field line tension, $K > 0$.

6.3.2 QED-mixer for network flow problems

To obtain a flow-conserving mixer, one can nominally choose any gauge-invariant lattice-QED Hamiltonian, replacing electric field variables with flow variables. We introduce a separate electric-field “flavor" for each type of commodity indicated by a superscript parenthetical index: $E^{(i)}$ with $i = 1 \ldots k$. In practice, we will choose our mixing Hamiltonian as the minimal Maxwell Hamiltonian, since it contains only the minimal elementary Wilson loops, thereby simplifying its implementation. Furthermore, we will set the electric field tension K to zero, since the goal of a mixer Hamiltonian is to produce unbiased quantum tunneling between different flow configurations. Significant efforts have been devoted to developing various schemes for “qubitization" and quantum simulation of lattice gauge theories [MWR+14]. We will remain largely agnostic about the specific implementation details, however, it is crucial to truncate the range of electric field values to lie between $-c(u,v) \leq E_{u,v} \leq c(u,v)$. To this end, we modify the electric field raising operator $e^{-iA_{uv}}$ to annihilate $|c(u,v)\rangle$, without altering its action on other states. We refer to the resulting Hamiltonian:

$$H_{\mathrm{M,QED}} = -\sum_{i=1}^{k} \sum_{f \in \mathcal{F}} (U_f^{(i)} + \mathrm{h.c.}) \tag{6.15}$$

as the QED-mixer. We require that sufficiently many elementary faces/plaquettes $f \in \mathcal{F}$ are included to provide a complete basis of graph cycles, so that evolution under H_m can transfer any flow-configuration to any other flow configuration. This is easy to satisfy for planar graphs, one can readily verify that $O(|\mathcal{V}|)$ applications of H_m connect any any two flow configurations (see Appendix D.1). We note that the circuit complexity of implementing this mixing Hamiltonian grows length with number of minimal cycles.

6.3.2.1 Avoiding Isolated Loop Generation

As written, the QED-mixer does not allow any flow constraint violations. However, this mixer still suffers from a potential problem: it can crate isolated loops of circulating flow that do not connect to sources or sinks (see Figure. 6.3). These isolated loops satisfy all flow constraints, but do not correspond to a physically relevant solution. One option is to simply retain these isolated loops throughout the QAOA, and prune them from the final solutions via classical post-processing. A potential drawback is that is that isolated loops may incur unphysical costs, and on large graphs, each valid path can be dressed with exponentially many isolated loops, each of which could incur an unphysical cost penalty, masking the true cost of the “pruned" post-processed solution during the QAOA optimization. Throughout the remainder of this chapter, we will restrict our attention to problems with unit demand for each type of good. For this subclass of problems, we can avoid isolated

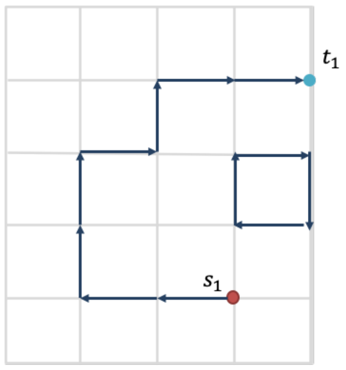


Figure 6.3: **A configuration with an isolated loop** Without the loop which is detached from the path, this would be a feasible solution. One could remove it easily, but having multiple isolated loops in a complicated graph would make such process hard to perform

loop creation by introducing further restrictions on the QED-mixer, which we call the restricted QED (RQED) mixer. In practice, this restriction will incur additional circuit complexity and may be undesirable. We will later compare the performance of the QED-mixer with and without restriction. The key step will be formulating a method to efficiently detect whether acting with U_f or $U_f^\dagger$ would create an isolated loop, depending on the graph property and specific problem. To avoid combinatorial blow-up of Hamiltonian terms, this detection must be done locally, which we do as follows. To determine whether adding electric field circulation around an elementary cycle of the graph adds an isolated loop, consider acting with $U_f^\dagger$ to add an electric field loop to a simple path and the following steps: Traverse the edge segments of the cycle in a counterclockwise fashion. For each vertex $v \in \mathcal{V}$, count the number of electric field lines entering ($E_{v,\text{in}}^{(i)}$) versus leaving ($E_{v,\text{out}}^{(i)}$) the node. Denote their difference-squared as

$$\mathsf{V}^{(i)} \equiv \sum_{j=1}^{\ell} \left(E_{v_j,\text{in}}^{(i)} - E_{v_j,\text{out}}^{(i)} \right)^2 , \tag{6.16}$$

where $v_1, \ldots, v_\ell$ are the nodes in the cycle. Notice that $(E_{v,\text{in}}^{(i)} - E_{v,\text{out}}^{(i)})^2$ can only take value 1 or 0. Since in our setting where maximum flow is 1, having two different direction of flows at the same node would suggest the node being used repeatedly,

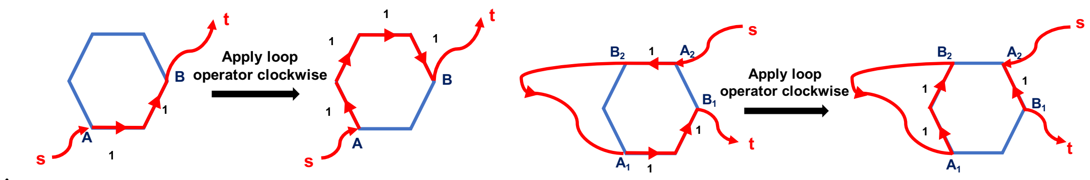


Figure 6.4: **An explanation of the "decision function"** For simplicity, we consider only one type of flow with max capacity 1. In both pictures, a flow (marked red) initially travels through the plaquette and then a loop operator is applied, increasing the flow on each edge on the plaquette by 1 in clockwise direction. The only difference between the pictures is that, the flow enters the loop twice at A_1 and A_2, and applying the operator resulted in redirected flow from A_1 to B_2, resulting in an isolated loop $A_1, B_2, ..., A_1$. To avoid such instances, we only apply the loop operator when exactly one continuous path of flow appears in the plaquette, which can be determined locally.

which further implies the configuration already contains an isolated loop. Imposing the Gauss' law constraint, $\mathsf{V}^{(i)}$ is equal to the total number of electric field lines entering or exiting the loop (if the loop does not contain a source/sink; or one could interpret a source as outside flow entering the loop and vice versa) without regard to sign (which is necessarily even). One can readily check that an isolated loop will be created unless $\mathsf{V}^{(i)} = 2$ (see Figure. 6.4 for sample instances).

With this in mind, we can then left-multiply $U_f^\dagger$ by a locally evaluable "decision function" to define a modified loop operator:

$$U_f^{(i)} \to \tilde{U}_f^{(i)} = \delta_{\mathsf{V}^{(i)},2} U_f^{(i)}, \tag{6.17}$$

which does not create isolated loops. Note that $\delta_{\mathsf{V}^{(i)},2}$ commutes with U so the multiplication order is arbitrary.

In practice, $\delta_{\mathsf{V}^{(i)},2}$ can be written as a polynomial with zeros at all even values of $\mathsf{V}^{(i)}$ other than 2:

$$\delta_{\mathsf{V}^{(i)},2} = \prod_{j=0,1...\ell;j\neq 1} \left(\frac{2j - \mathsf{V}^{(i)}}{2j-2}\right), \tag{6.18}$$

which permits implementation with circuit complexity $\sim$ poly(ℓ). For simple graph structures, such as grids, where the sizes of elementary cycles are bounded independent of the system size, imposing this restriction adds only constant circuit-depth overhead.

6.3.2.2 Initial state preparation

To begin the QAOA procedure, one must choose an initial state that is a quantum superposition with weights on all possible solutions. In the original formulation of QAOA, the initial state was chosen as the ground-state of the X-mixer Hamiltonian. This had two virtues: first, it ensured that QAOA could reduce to the quantum adiabatic algorithm in the limit of large step number, p. Second, this state is an equal weighted superposition of all computational states, and does not introduce an intrinsic bias.

In contrast, for QED-mixers, the mixer ground-state is no longer an equal-weight superposition. Moreover, it is not straightforward to implement the ground-state of the QED or RQED mixers. For these reasons, we consider alternative state preparation schemes. As a starting point, we assume that it is straightforward to greedily prepare a computational basis state that satisfies the flow-constraints (a detailed prescription will be given below for EDP problems).

Adiabatic ground-state preparation by reverse-annealing: One option would be to adiabatically prepare the QED or RQED mixer ground-state via adiabatic evolution from a classical Hamiltonian with the fixed computational basis state as the ground-state to the (R)QED mixer ground-state. However, generically, the QED mixer will have gapless photon-like excitations, whose gap scales to zero as $\sim 1/R$ where R is the graph radius (maximal distance between two nodes), such that this adiabatic ground-state preparation requires time $\sim \mathcal{O}(R)$. Moreover, we will see that starting from the ground-state of the mixer Hamiltonian actually leads to worse QAOA performance, due to the reasons we will discuss in later sections.

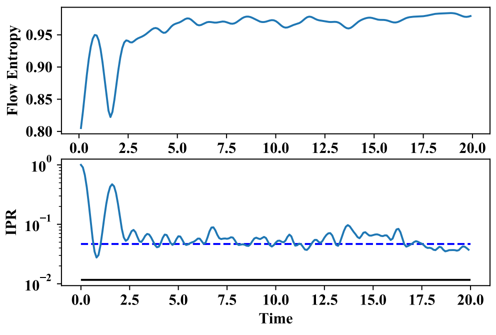
State preparation by mixer evolution: An alternative approach is to simply time-evolve the initial flow-constraint-preserving computational basis state with the mixing Hamiltonian for a certain amount of time, which spreads out the weight of the Hamiltonian onto other configurations. In analogy to photon propagation in electrodynamics, the flow should spread out ballistically (moving with constant velocity), covering the graph in time $\sim O(R)$. Hamiltonian simulation techniques can implement time-evolution for time t with performance that asymptotically tends to $O(t)$ [BCK15]. In practice, it may not be necessary to simulate continuous time evolution, but rather one could break H_M into local terms acting on disjoint sets of qubits and stroboscopically alternate among them to achieve similar results.

To numerically analyze the spreading of the wave function, we introduce the inverse participation ratio (IPR) test:

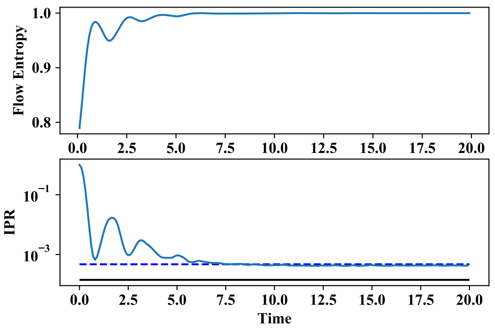
$$\mathrm{IPR} = \sum_i |\psi_i|^4, \tag{6.19}$$

where ψ_i is the amplitude of the wave-function in computational basis state i. IPR measure is inversely proportional to how evenly the wave-function spread-out over the computational basis states (i.e., among potential solutions to the optimization problem). The choice of power 4 here is because 2 would always give 1 and higher powers contain the same information about the wave function as 4th power does except for rare cases. When the wave-function is concentrated on a single state, IPR $= 1$; whereas an equal superposition of all states yields the minimal value of IPR $= 1/|\mathcal{H}|$, where $\mathcal{H}$ is the size of the Hilbert space (number of feasible solutions)

Figure. 6.5 shows the evolution of IPR with evolution under the RQED-mixer for single source-sink pairs on different sized square-grids. Since the RQED-Hamiltonian only evolves in the feasible solution space the test is done only within a constructed feasible subspace. The IPR decays from one, approximately saturating to a value close-to, but below the IPR for the mixer ground-state (blue dashed line), in characteristic time $t_{\mathrm{sat}} \sim O(R)$ (where R is the graph radius). In addition, the



(a)



(b)

Figure 6.5: **IPR and entropy test –** The figures show instances of IPR and flow entropy (normalized to its maximum) S in the feasible space for single-source 4×4 (top) and 5×5 (bottom) square lattices for RQED-mixer in real time evolution. In IPR tests, the black solid line stands for the minimum possible IPR value (equal superposition of all possible paths from s to t), and the blue dashed line shows the IPR for mixer ground state. All $s-t$ pairs and initial paths are drawn at random. At both sizes, the entropy curve characters the bumps and saturation in the IPR curve, suggesting itself as a good alternative of IPR.

IPR exhibits approximately periodic revival behaviors, which are most evident on the smaller 4×4 grid. As is well known from the study of Poincare recurrences, the period of these revivals becomes (doubly)-exponential in size of the graph, since the number of feasible solutions grows exponentially with the size of the graph, and can be safely neglected even for moderate graph sizes (indeed the oscillations are negligible already for the 5×5 grid.)

To prepare the initial state for subsequent QAOA iterations, we evolve the state until it just enters the saturation region where the IPR stabilizes to its long time value (e.g. in the 5×5-grid this occurs around $t_{\text{sat}} \approx 7.5$, see Figure. 6.5). In both tests, we observe that, inside the saturation region, the saturation-value of IPR lies below that of the mixer's ground-state, indicating that the mixer ground-state is more biased than the time-evolved state. This feature is natural since the evolved state is not low-energy and can be expected to contain additional configurational entropy.

In practice, IPR is challenging to measure as the Hilbert space size grows exponentially. Instead, one can determine the saturation time by monitoring local observables that act as witnesses for the IPR. Without loss of generality we focus on a single commodity case, since for multiple commodities, the Hilbert space is a tensor product of the single-commodity Hilbert spaces, with no inter-commodity interactions in the state preparation procedure. We examine the probability of observing unit flow (of either sign) on edge $e \in \mathcal{E}$ after evolution for time t under the mixing Hamiltonian

$$p_e(t) = \frac{\langle E_e^2 \rangle}{\sum_{e \in \mathcal{E}} \langle E_e^2 \rangle} \tag{6.20}$$

which can be estimated by sampling from the state in the computational basis.

We then define the (normalized) "flow entropy" as the von-Neumann entropy of this probability distribution:

$$S_f = -\frac{1}{|\mathcal{E}| \log 2} \sum_{e \in \mathcal{E}} p_e \log(p_e). \tag{6.21}$$

Larger $S_f \leq 1$ represents a more even distribution of paths. S_f saturates its maximal value of 1 when each link carries flow with equal probability. The flow entropy exhibits similar saturation behavior to the IPR, allowing one to measure the saturation time for a given graph. Crucially, to accurately estimate flow, the probability p_e needs only be measured to accuracy $\sim 1/|\mathcal{E}|$, which requires sampling cost $\sim |\mathcal{E}|^2$ that is polynomial in problem size (in contrast to the exponentially small IPR), allowing an efficient measurement to identify saturation time at which to stop the state preparation step.

6.3.3 Algorithm description

We are now ready to detail the steps of the modified QAOA for network flow problems. Given a directed graph $G(\mathcal{V}, \mathcal{E})$ as input (if the graph is undirected, simply choose an arbitrary orientation for the edges):

1. *Pre-process:* Identify a set of elementary faces (i.e. choose a basis of closed cycles) in G and store them. For a planar graph, this can be done classically in polynomial time [dBCvKO08].

2. *Hamiltonians simulation:* Choose a technique to simulate time-evolution under the cost and mixing Hamiltonians: H_C, H_M.

3. *Initial state preparation:* As described in Section 6.3.2.2, for each pair (s_i, t_i) given in the input, pick an arbitrary "seed" path, P_0 (which can be found efficiently by standard methods), and define the corresponding computational basis state as $|P_0\rangle$. Identify the saturation time t_{sat}, for the graph by the flow-entropy test described in the text. Then, simulate time-evolution under the mixing Hamiltonian to form the initial state: $|\psi_0\rangle = e^{-iH_M t_{\text{sat}}}|P_0\rangle$.

4. *Variational Optimization:* Following the original QAOA procedure, but replacing the the X-mixer with the (R)QED-mixer to avoid generated flow-constraint violations, find $\boldsymbol{\gamma}_*, \boldsymbol{\beta}_* = \arg\min \varepsilon_C$ using any desired classical minimization procedure,

5. *Post-process* Repeatedly sample from the optimized variational state $|\psi(\boldsymbol{\gamma}_*, \boldsymbol{\beta}_*)\rangle$, recording the best (lowest-cost) sample encountered as an approximate solution.

6.4 Numerical Simulation of Algorithm Performance

In this section, we present results from numerical simulation of QED-modified and standard QAOA of small-scale network flow problems. Due to the rapid growth of Hilbert space, $|\mathcal{H}| \sim O(3^{k|\mathcal{E}|})$, the accessible problem size is quite limited. In order to provide a meaningful comparison of the QED-mixer, we first consider the (classically easy) SSSP problem ($k = 1$), which will allow simulation of relatively larger graphs to enable a comparison of QED-mixer and X-mixer. We then simulate EDP problems with $k = 2$ on a grid graph for RQED-mixer only, where we can restrict our numerical simulation to the feasible solution space of size $\ll |\mathcal{H}|$.

For the original X-mixer, in each step, the variational parameters can be limited to $[0, 2\pi]$ for $\{\gamma_i\}$ and $[0, \pi]$ for $\{\beta_i\}$, due to the periodicity of evolution under Pauli strings. The QED-mixer has no such periodicity. However, to run the QED-mixer for longer times would require additional circuit depth with which additional rounds of QAOA with the X-mixer could have been performed. Hence, to make a fair comparison, we also restrict our variational parameter ranges for the QED-mixer to the same range as for the original X-mixer.

In all simulations, we first perform a global search with differential evolution, and then optimize with a local BFGS method [Fle13]. For both methods, we restrict the optimizer to at most 200 minimization steps to balance accuracy and efficiency.

To generate a larger collection of problems from a limited set of graph types and sizes, we generate random problem instances for each graph. For the SSSP problems we consider for each triangle graph in Figure. 6.2 with source-and-sink located at opposite corners, we generate random problem instances by drawing random weights w_e i.i.d. for each edge from the uniform distribution on the unit interval $[0, 1]$, and seeding the state-preparation step with a uniformly-randomly chosen path, $|P_0\rangle$. For the EDP problem, the edges are unweighted, so we further choose the source and sink locations uniformly at random on different-sized grid graphs.

6.4.1 Comparing mixers

To compare the performance of QAOA on network flow problems using different X-, QED-, and RQED-mixers, we adopt a metric called the approximation ratio (AR) [WRDR20], defined as:

$$\mathrm{AR}(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \frac{\langle \psi_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) | \Pi \left(C_{\max} - H_C \right) \Pi | \psi_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) \rangle}{C_{\max} - C_{\min}} \tag{6.22}$$

where $C_{\max}$ and $C_{\min}$ respectively represent the maximum and minimum costs from the set of feasible solutions, and Π is the projector into the feasible subspace, which ensures that only states without constraint violations and isolated loops are counted.

The approximation ratio indicates fractional of improvement compared to the worst case, normalized by the possible range of cost values, despite whether an EDP instance on a certain problem exists. In practice, we perform multiple independent runs to obtain average performance, namely, the average approximation ratio (AAR), as the indicator of QAOA performance. Similarly, the variational optimization of QAOA parameters is done with respect to the projected cost function:

$$\tilde{\varepsilon}_C(\boldsymbol{\gamma}, \boldsymbol{\beta}) := \langle \psi_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) | \Pi H_C \Pi | \psi_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) \rangle. \tag{6.23}$$

Whereas, by construction, the QED- and RQED-mixers automatically avoid flow-conservation violating constraints, flow-constraint violations can only be softly penalized by introducing an extra term to the cost function for the X-mixer:

$$H_{C,\text{penalty}} = \Delta \sum_{u \in \mathcal{V}, i} \left(\sum_{(u,v) \in \mathcal{E}} E^{(i)}_{(u,v)} - d_i(\delta_{u,s_i} - \delta_{u,t_i}) \right)^2 . \tag{6.24}$$

In principle, Δ introduces an extra hyperparameter that must be optimized. Generally, Δ should increase with problem size to avoid the tendency to lower cost by violating constraints. For the problem-sizes we simulate, the results are not very sensitive to the precise choice in Δ (Figure. 6.6), and we choose $\Delta = 1$ throughout for simplicity.

6.4.2 Mixer comparison on SSSP problems

We begin with a comparison of the performance between all three mixers: the X-, QED- and RQED-mixer, for approximately solving SSSP problems on different sized graphs. As expected, X-mixer exhibits substantially worse performance than the flow-constraint preserving QED mixers. For a single QAOA round, $p = 1$, the degradation in X-mixer's performance with increasing graph sizes tracks the decreasing trend of the ratio between the number of feasible solutions and the size of whole Hilbert space (as shown in Figure. 6.7).

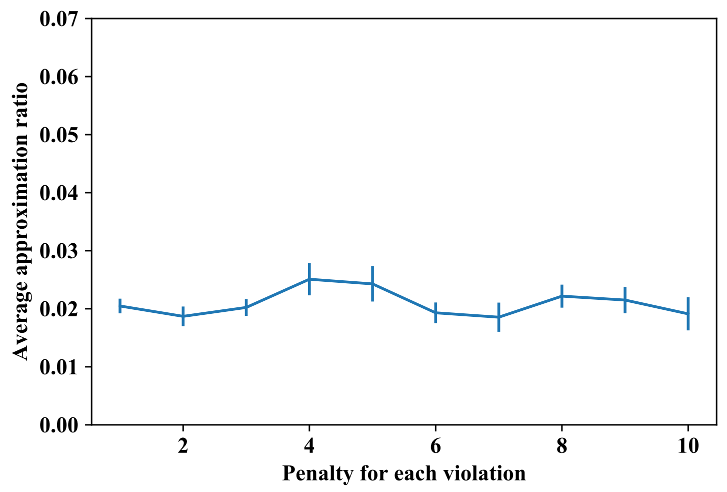


Figure 6.6: **Behavior of X-mixer QAOA with different penalties,** Δ for SSSP problem at $P = 1$. This result shows that the average behavior of X-mixer QAOA is fairly insensitive to the precise choice of the penalty coefficient Δ.

The unrestricted QED mixer initially matches the RQED-mixer on the smallest problem instances, for which the graphs are too small to permit isolated loop creation. As the graph size grows, the unrestricted QED mixer's AAR drops below that of the RQED-mixer. For the largest graphs, the QED-mixers' AAR approaches the value achieved by picking feasible paths at random, showing that isolated loop creation can substantially degrade the unrestricted QED-mixer performance at $p = 1$.

This shows that, although we start with a feasible solution, isolated loops can be created when using the QED-mixer in its original version. A multi-step QAOA simulation shows that, for the 2-triangle graph, the QED-mixers are able to solve the problem exactly at around $p = 3$, which is not surprising due to the small size of the problem.

6.4.3 EDP on undirected graphs

Even though a direct comparison between the X-mixer and the QED-mixers for EDP problems is expensive, we test out the performance of the QED-mixers alone on larger graphs by restricting the simulation to the feasible subspace to reduce the computational power required. In order to be able to compare performances at different graph sizes, we only consider EDP problems with $k = 2$ source-sink pairs. As shown in Figure. 6.8, even though the solution space size for 4×4 grid is typically 100 or more times (depending on the location of sinks and sources) than that of the 3×3-grid, the performance is only weakly affected – even after only a single QAOA round, $p = 1$, the AAR remains higher than 0.7. As a complementary to the results in IPR test, Figure. 6.9, shows how different initial state IPRs result in different outcomes in solving EDP on 3×3 grids. We observe that, the ground state preparation is not a necessity for our mixer, but the equal superposition state of all feasible solutions does serve as a best starting point of the three, followed closely by the initial state prepared by evolving a random configuration with the mixing Hamiltonian, which is with the IPR test.

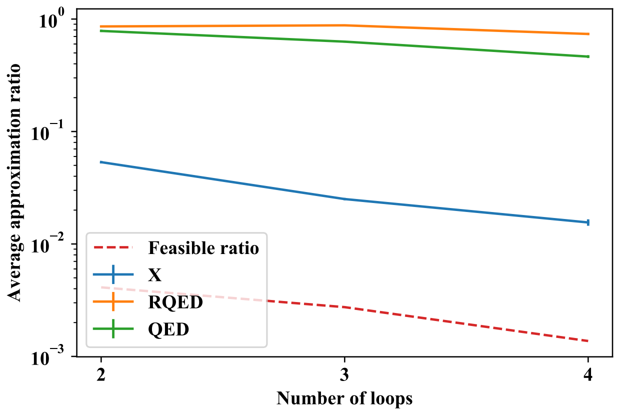
These results suggest that having an unbiased ergodic superposition of solutions is more advantageous than starting close to the mixer ground-state (for ordinary QAOA with the X-mixer, these coincide).

6.5 Discussion

In this chapter, we designed and simulated a QED-inspired QAOA algorithm to the flow network problems. In particular, we tested its performance with the EDP and SSSP problems. The biggest difference between routing problems and other typical QAOA benchmark problems (like MaxCut) is that the feasible solutions only consist of an exponentially small fraction of the whole solution space. The standard QAOA approach produces infeasible solutions with high probability. To resolve this issue, we proposed the RQED-mixer, which automatically ensures the satisfaction of flow constraints throughout the algorithm. By observing the analogy between Gauss' Law and those constraints, we theoretically and numerically demonstrated that the QED-mixer is a natural choice for the routing problem. Although implementing the RQED-mixer requires additional circuit complexity, the generating Hamiltonian is still local and the number of terms is still linear in problem size, and optimization purely within the feasible space makes the QAOA with RQED-mixer more likely to find nearly optimal solutions in comparison to the standard QAOA approach. Part of the simulation results showed that for SSSP problem, the average approximation ratio of RQED-mixer is significantly higher than the X-mixer. For the harder problem, EDP, our results also showed that QAOA with RQED-mixer can achieve high approximation ratio on different size instances, although our numerical simulations were necessarily limited to rather modest problem sizes.

Our experiments with different initial state strategies suggest an intriguing departure from the “shortcut-to-adiabaticity" mechanism typically used to motivate QAOA. Namely, QAOA is often motivated as a short-depth approximation to the adiabatic mapping from mixer to the cost of ground-state. However, we have seen that, at least on modest graph sizes available for classical simulation, starting with a more ergodic (less biased) superposition of initial states produces better results than starting in a low-energy state of the mixer, suggesting that a different mechanism than approximate adiabaticity is at play.

Whether the improved performance and superiority of the non-adiabatic operations extend to a larger problem size is an important question for future studies. However, the scope of classical simulation is limited due to the typical explosion of Hilbert space size with problem size. Analytic insights would be extremely valuable, although have often proved challenging beyond small-p. One possible approach is to investigate the locality of QAOA with the RQED-mixer. For standard QAOA with X-mixer, the locality was studied [FGG20a] to prove the performance of QAOA on the independent set problem, another famous NP-complete problem on graphs. Last but not least, it would be desirable to implement the algorithm on near-term quantum computers, as these devices begin to eclipse classical simulation [AAB+19].



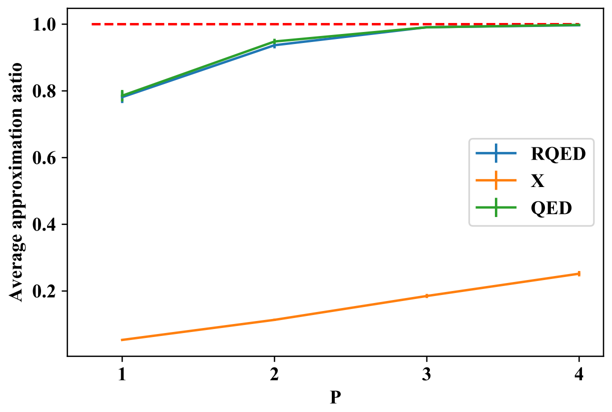


Figure 6.7: **Comparing different mixers Top:** Solving SSSP on different sized triangle graphs with different mixers; 120 runs performed for each mixer: weight on each edge is randomly drew from $[0, 1]$. **Bottom:** A multiple-step comparison: We compare the behavior of the 3 mixers in solving SSSP problem on the 2-triangle graph as plotted in Figure 6.2. Each point represents an average of 200 random instances. Notice that, for this particular graph, it is impossible for the unrestricted QED-mixer to create an isolated loop, making its performance almost identical to the RQED-mixer.

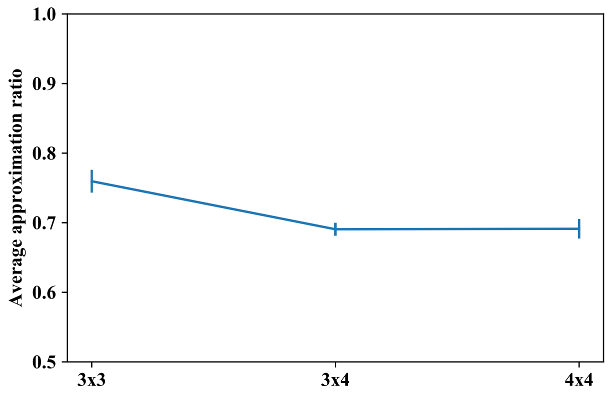


Figure 6.8: **RQED-Mixer Behavior at** $p = 1$ The simulation is done for 3×3, 3×4, 4×4 grids for a 2-pair EDP problem. 200 random problem instances are performed at each graph, by choosing the location of each source, sink, state preparation seed path at random.

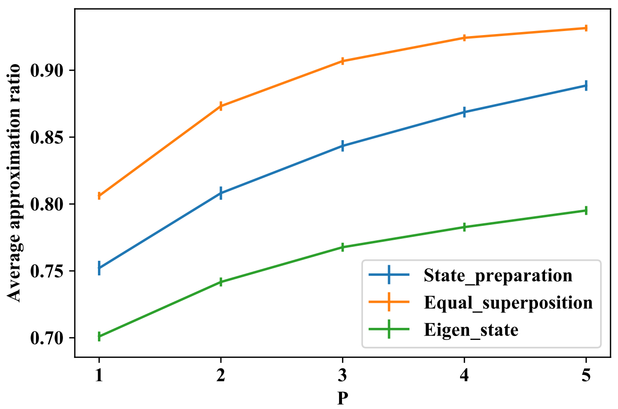


Figure 6.9: **RQED QAOA behavior in solving actual EDP problems with different initial states** We compare the effect of different choices of initial states on the RQED-mixer's performance, averaging over 200 random problem instances. “Eigenstate" stands for the ground state.

Chapter 7: Quantum Fine-Grained Complexity

7.1 Introduction

In the closest pair problem (CP), we are given a list of points in $\mathbb{R}^d$, and asked to find two that are closest. (See Fig. 7.1 for an illustration of this problem.) This is a fundamental problem in computational geometry and has been extensively studied. Indeed, CP is one of the standard examples in textbooks (such as [CLRS09] and [KT06]) to introduce the divide-and-conquer technique. Moreover, CP relates to problems that have critical applications in spatial data analysis and machine learning, such as empirical risk minimization [BIS17], point location [SH75, Bes98], time series motif mining [MKZ+09], spatial matching problems [WTFX07], and clustering [NTM01]. Therefore, any improvement on CP may imply new efficient algorithms for related applications.

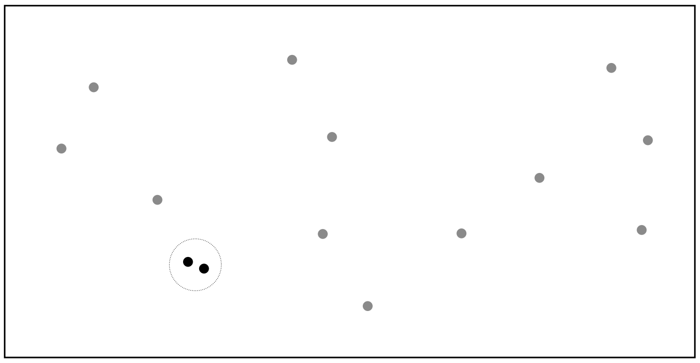


Figure 7.1: An instance of the CP, where the the closest pair is labeled in the circle.

Like with many other geometric problems, the hardness of CP rises as the dimension d increases. Shamos and Hoey gave the first $O(n \log n)$ deterministic algorithm in $\mathbb{R}^2$ by using Voronoi diagrams [SH75], improving on the trivial $O(n^2 d)$ upper bound. Then, Bentley and Shamos gave an algorithm with $2^{O(d)} n \log n$ running time via a divide-and-conquer approach [BS76]. A randomized algorithm by Khuller and Matias [KM95, Rab76] takes $2^{O(d)} n$ expected running time. A trivial lower bound for CP is $\Omega(n)$, since one must read all points to find the closest pair in the worst case. Yao showed an $\Omega(n \log n)$ lower bound for CP on the algebraic decision tree model [CC89].

When we consider CP in polylog(n) dimensions, the running time of all existing algorithms blows up to $\Omega(n^2)$, and thus it is unknown if there exists an algorithm matching the unconditional lower bounds. Nevertheless, under the *Strong Exponential Time Hypothesis (SETH)*, Karthik and Manurangsi [KM20a], and David et al. [DSL19], recently proved a conditional lower bound of $n^{2-o(1)}$ for CP in polylog(n) dimensions. This implies that the brute force approach is nearly optimal in polylog(n) dimensions unless SETH is false. SETH was introduced by Impagliazzo and Paturi [IP01], and is the assumption that for all $\epsilon > 0$, there exists an integer $k > 2$ such that no algorithm can solve k-SAT in time $O(2^{(1-\epsilon)n})$.

The main idea behind the results of [KM20a, DSL19] is to prove a "fine-grained" reduction from CNF-SAT to CP in polylog(n) dimensions. Fine-grained reductions are reductions between computational problems that keep track of the exact polynomial exponents. For instance, [KM20a] showed that CNF-SAT with $2^{n(1-o(1))}$ time is reducible to CP in polylog n dimensions with $n^{2-o(1)}$ time, and thus the lower bound for CP in polylog n dimensions is $n^{2-o(1)}$ unless SETH is false.

Surprisingly, to our knowledge, the quantum time complexity of CP was hardly investigated before. The trivial quantum algorithm for CP is to use Grover's search algorithm on all n^2 pairs, which takes $O(nd)$ time. Sadakane et al. [SST01] sketched a quantum algorithm that runs in $O(n^{1-1/(4\lceil d/2 \rceil)})$ time. Volpato and Moura [VM10] claimed a quantum algorithm that uses $O(n^{2/3})$ *queries*, but no analysis was given of the *running time*, and as we will see, the conversion from the query-efficient algorithm to a time-efficient algorithm is nontrivial. As for the lower bound, any quantum algorithm for CP needs $\Omega(n^{2/3})$ time, since Aaronson and Shi [AS04] proved such a lower bound for element distinctness, and CP contains element distinctness as a special case, where a closest pair has distance 0.

In this chapter, we resolve the quantum time complexity of CP. In constant dimensions, we observe that by using a quantum walk for element distinctness [Amb07, MNRS11], we can achieve $O(n^{2/3})$ queries for CP. However, to obtain the same time complexity, the algorithm needs some geometric data structure that supports fast updates and checking, and that—crucially—is "history-independent", i.e., the data structure is uniquely represented, disregarding the order of insertion and deletion. History-independence is essential since different representations of the same data would destroy quantum interference between basis states.

We propose a geometric data structure that is history-independent and that supports fast checking and updates. Our data structure works by discretizing $\mathbb{R}^d$ into hypercubes with length $\epsilon/\sqrt{d}$. Then, we use a hash table, skip lists, and a radix tree to maintain the locations of the points and hypercubes. This data structure is history-independent, and we can easily find pairs with distance at most ϵ with it. We then find the closest pair by a binary search. By using our data structure and a quantum walk [Amb07, MNRS11], we achieve quantum time complexity $\widetilde{O}(n^{2/3})$.

For CP in polylog(n) dimensions, one may expect a conditional lower bound under SETH. However, SETH fails when quantum algorithms are considered since a simple application of Grover's search algorithm on all assignments solves CNF-SAT in time $\widetilde{O}(2^{n/2})$. Furthermore, existing fine-grained reductions may require time greater than $O(2^{n/2})$.

In this chapter, we introduce the *Quantum Strong Exponential Time Hypothesis (QSETH)* and *quantum fine-grained reductions*. We define QSETH as follows.

Definition 7.1 (QSETH)**.** For all $\epsilon > 0$, there exists some $k \in \mathbb{N}$ such that there is no quantum algorithm solving k-SAT in time $O(2^{(1-\epsilon)\frac{n}{2}})$.

We then observe that the classical definition of fine-grained reductions cannot capture the features of quantum reductions such as superposed queries and speedups from quantum algorithms. For instance, a fine-grained reduction may reduce problem A to solving many instances of problem B and then output the best solution; in this case, one can use Grover's search algorithm to achieve a quadratic speedup. Therefore, instead of summing the running time over all instances as in Definition 7.9, we use a quantum algorithm which solves all instances in superposition and outputs the answer. We give a formal definition of quantum fine-grained reductions in Definition 7.12 and show that under QSETH, any quantum algorithm for CP in $\text{polylog}(n)$ dimensions requires $n^{1-o(1)}$ time. This implies that Grover's algorithm is optimal for the problem up to an $n^{o(1)}$ factor.

Intuitively, QSETH is the conjecture that applying Grover's search algorithm over all assignments in superposition is the optimal quantum algorithm for $\mathsf{CNF\text{-}SAT}$. This is similar to SETH, which says that a brute force search is optimal for $\mathsf{CNF\text{-}SAT}$. A series of works on $\mathsf{CNF\text{-}SAT}$ [Sch99, PPSZ05, PP10, Her15, SS17] shows that for some constant $c \in [1, 2]$, there exist (randomized) algorithms for n-variable k-SAT that run in time $2^{n(1-c/k)}$. As k grows, the running time of these algorithms approach 2^n. When k is small, however, there are algorithms with better running times. For instance, when $k = 3$, Schöning [Sch99] obtained an algorithm with $O(1.334^n)$ running time, which was later improved to $O(1.308^n)$ by Paturi et al. [PPSZ05]. However, none of the above mentioned algorithms have good running time on larger k's, so SETH remains a plausible conjecture.

When k is small enough, there are also quantum algorithms for k-SAT [Amb04, DKW05] running in time much less than $O(2^{n/2})$. However, these quantum algorithms mainly use Grover search to speed up the classical algorithms of [Sch99, PPSZ05], and thus do not perform well for large k, either. Therefore, we conjecture that for large enough k, no quantum algorithm can do much better than Grover search.

Finally, we study the bichromatic closest pair problem (BCP) and the orthogonal vector problem (OV). Briefly, OV is to find a pair of vectors that are orthogonal given a set of vectors in $\{0, 1\}^d \in \mathbb{R}^d$, and BCP is, given two sets A, B (representing two colors) of n points in $\mathbb{R}^d$, to find the pair (a, b) of minimum distance with $a \in A$ and $b \in B$.

We can summarize all of our results as follows.

Theorem 7.1 (Informal)**.** *Assuming QSETH, there is no quantum algorithm running in time $n^{1-o(1)}$ for* OV*,* CP*, and* BCP *when $d = \mathrm{polylog}(n)$.*

Theorem 7.2 (Informal)**.** *The quantum time complexity of* CP *in $O(1)$ dimensions*[1] *is $\widetilde{\Theta}(n^{2/3})$*[2]*.*

Theorem 7.3 (Informal)**.** *For any $\delta > 0$, there exists a quantum algorithm for* BCP *with $\widetilde{O}(n^{1-\frac{1}{2d}+\delta})$ running time. There exists a quantum algorithm which solves $(1+\xi)$-approximate* BCP *in time $\widetilde{O}(\xi^{-d} n^{2/3})$.*

Theorem 7.4 (Informal)**.** *The quantum time complexity of* OV *in $O(1)$ dimensions*[3] *is $\Theta(n^{1/2})$.*

Table 7.1 also summarizes what is known about upper and lower bounds on the classical and quantum time complexities of all of these problems.

Related work A recent independent work by Buhrman, Patro and Speelman [BPS19] also studied quantum strong exponential time hypothesis. They defined (a variant of) QSETH based on the hardness of testing properties on the set of satisfying assignments of a SAT formula, e.g., the parity of the satisfying assignments. Based on these hardness assumptions extended from the original QSETH, they gave conditional quantum lower bounds for OV, the Proofs of Useful Work [BRSV17] and the

[1]We actually give a slightly stronger result: the same time complexities still hold when $d = O\left(\frac{\log\log n}{\log\log\log n}\right)$.

[2]The $\widetilde{\Theta}$ notation is Θ with logarithmic factors hidden in both upper and lower bounds.

[3]The same time complexities still hold when $d = O(\log\log n)$.

[4]$\log^*(n) := \log^*(\log n) + 1$ for $n > 1$ and $\log^*(1) := 0$. Hence, $2^{O(\log^* n)}$ is an extremely slow-growing function.

	Dimension		Lower Bound	Upper Bound
CP	$\Theta(1)$	Classical	$\widetilde{\Omega}(n)$ [CC89]	$\widetilde{O}(n)$ [SH75, BS76, KM95]
		Quantum	$\mathbf{\Omega(n^{2/3})}$ Theorem 7.33	$\mathbf{\widetilde{O}(n^{2/3})}$ Corollary 7.32
	polylog n	Classical	$n^{2-o(1)}$ (Under SETH) [KM20a]	$O(n^2)$
		Quantum	$\mathbf{n^{1-o(1)}}$ **(Under QSETH)** Theorem 7.12	$\mathbf{\widetilde{O}(n)}$ Theorem 7.7
OV	$\Theta(1)$	Classical	$\Omega(n)$	$O(n)$ [Wil17]
		Quantum	$\mathbf{\Omega(n^{1/2})}$ Theorem 7.43	$\mathbf{O(n^{1/2})}$ Theorem 7.43
	polylog n	Classical	$n^{2-o(1)}$ (Under SETH) [Wil05]	$n^{2-o(1)}$ [AWY15, CW16]
		Quantum	$\mathbf{n^{1-o(1)}}$ **(Under QSETH)** Theorem 7.12	$\mathbf{\widetilde{O}(n)}$ Theorem 7.7
BCP	$\Theta(1)$	Classical	$\Omega(n)$	$O\left(n^{2-\frac{2}{\lceil d/2\rceil+1}+\delta}\right)$ [AESW91]
		Quantum	$\mathbf{\Omega(n^{2/3})}$ Theorem 7.42	$\mathbf{\widetilde{O}(n^{1-\frac{1}{2d}+\delta})}$ for BCP Theorem 7.41 $\mathbf{\widetilde{O}(\xi^{-d}n^{2/3})}$ for $(1+\xi)$-BCP Theorem 7.39
	$2^{O(\log^*(n))}$[4]	Classical	$n^{2-o(1)}$ (Under SETH) [Che18]	$n^{2-o(1)}$ [AWY15, CW16]
		Quantum	$\mathbf{n^{1-o(1)}}$ **(Under QSETH)** Theorem 7.18	$\mathbf{\widetilde{O}(n)}$ Theorem 7.7

Table 7.1: A summary of our quantum complexity results and comparison to classical results. The bold entries highlight our contributions in this chapter.

edit distance problem. In comparison, we formally define the *quantum fine-grained reductions* and prove lower bounds for CP, OV, and BCP under the original form of QSETH by showing the existence of quantum fine-grained reductions from CNF-SAT to the these problems.

7.1.1 Proof overview

For ease of presentation, some notations and descriptions will be informal here. Formal definitions and proofs will be given in subsequent sections.

We give an optimal (up to a polylogarithmic factor) quantum algorithm that solves CP for constant dimensions in time $\widetilde{O}(n^{2/3})$. First note that there exists a Johnson graph corresponding to an instance of CP, where each vertex corresponds to a subset of $n^{2/3}$ points of the input of CP, and two vertices are connected when the intersection of the two subsets (they are corresponding to) has size $n^{2/3}-1$. A vertex is marked if the subset it corresponds to contains a pair with distance at most ϵ. Then, the goal is to find a marked vertex on this Johnson graph and use binary search over ϵ to find the closest pair. Our algorithm for finding a marked vertex is based on the quantum walk search framework by Magniez et al. [MNRS11], which can be viewed as the quantum version of the Markov chain search on a graph (in our case, a Johnson graph). The complexity of this quantum walk algorithm is $O(\mathsf{S} + \frac{1}{\sqrt{\lambda}}(\frac{1}{\sqrt{\delta}}\mathsf{U} + \mathsf{C}))$, where λ is the fraction of marked states in the Johnson graph, δ is its spectral gap, S is the cost for preparing the algorithm's initial state, U is the cost for implementing one step of the quantum walk, and C is the cost for checking the solution. For our Johnson graph, $\lambda = n^{-2/3}$ and $\delta = n^{-2/3}$. If we consider only the query complexity, $\mathsf{S} = n^{2/3}$, $\mathsf{U} = O(1)$, and $\mathsf{C} = 0$. However, the time complexity for C is huge in the straightforward implementation, e.g., storing all points in an array according to the index order, as we need to check all the pairs from the $n^{2/3}$ points, which will kill the quantum speedup. To tackle this, we discretize the space into small hypercubes. With this discretization, it suffices to check $O((\sqrt{d})^d)$ neighbor hypercubes to find a pair with distance at most ϵ. To support the efficient neighborhood search, we need an efficient data structure.

Existing data structures do not meet our need. They either have prohibitive dependence on the dimension, such as $\Omega(n^{\lceil d/2 \rceil})$ time for constructing and storing Voronoi diagrams [Kle80], or do not have unique representation (i.e., they are history-dependent), such as fair-split trees and dynamic trees [Bes98]. Note that the requirement of unique representation is due to the fact that different representations of the same data would destroy the interference that quantum computation relies on. To solve this problem, we propose a uniquely represented data structure that can answer queries about ϵ-close pairs and insert/delete points efficiently. This data structure is based on a hash table, skip lists, and a radix tree. With this data structure, $\mathsf{U} = O(\log n)$ and $\mathsf{C} = O(1)$. Hence, we have the desired time complexity (see Section 7.4.2). We give another method for solving CP that only uses a radix tree as the data structure. With only a radix tree, the algorithm cannot handle cases with multiple solutions, and we need to subsequently reduce the size of the problem until there is at most one solution (see Section 7.4.3). These two quantum algorithms have the same time complexity.

Our quantum algorithm for solving approximate BCP follows the same spirit as that for CP, except that we use a finer discritization of the space (see Section 7.5.1). To solve BCP exactly, we need a history-independent data structure for nearest-neighbor search, but no such data structure is known. Instead, we adapt the nearest-neighbor search data structure by Clarkson [Cla88] to the quantum algorithm proposed by Buhrman et al. [BdWD+01] for element distinctness, which does not require history-independence of the data structure because in the algorithm of [BdWD+01], no insertions and deletions are performed once the data structure for a set of points is constructed (see Section 7.5.2). Sadakane et al. [SST01] sketched an algorithm for BCP with similar ideas and running time, but we give the first rigorous analysis.

To derive our quantum fine-grained complexity results for OV and CP when $d = \operatorname{polylog} n$ under QSETH, we first define quantum fine-grained reductions. In our definition, we consider problems whose input is given in the quantum query model, and allow the reduction to perform superposed queries and run quantum algorithms, e.g., amplitude amplification. The classical reductions from CNF-SAT to CP [KM20a, DSL19] and OV [WY14] are not "quantum fine-grained" under QSETH. These reductions fail because their running time exceeds $2^{n/2(1-\epsilon)}$, which is the conjectured time complexity for CNF-SAT under QSETH. Therefore, we cannot derive from them any non-trivial lower bounds for CP or OV based on QSETH. In the following, we use the advantages of quantum algorithms to make these reductions work.

There are two main obstacles in "quantizing" the fine-grained reductions under QSETH. The first obstacle is that the time cost for preparing the input of the problem we reduce to is already beyond the required running time. For instance, consider the reduction from CNF-SAT to OV. Let φ be a CNF-SAT instance on n variables and m clauses. The classical fine-grained reduction divides all n variables into two sets A and B of size $n/2$, and then maps all assignments for variables in A and B to two sets V_A and V_B of $2^{n/2}$ vectors each. It is obvious that the time for writing down V_A and V_B is already $\Theta(2^{n/2})$. Nevertheless, many quantum algorithms achieve sublinear query complexities by querying the input oracle in superposition. Hence, instead of first constructing the input of OV at once and then running the algorithm, we can simulate it "on-the-fly": whenever the OV's algorithm queries the input oracle with some superposition of indices, we use a quantum subroutine to realize the input oracle by mapping the query indices to the corresponding assignments in $\mathsf{CNF\text{-}SAT}$, and then to the corresponding vectors in V_A and V_B. This subroutine takes only $O(n)$ time, and therefore the quantum reduction, which has running time $O(n)$ times the running time of the OV algorithm, is quantum fine-grained.

Another difficulty in quantizing the fine-grained reductions is that some reduction needs to call the oracle multiple times, and the number of calls exceeds the required running time. However, it is possible to achieve quadratic speedup if these oracle calls are non-adaptive. For the reduction from BCP to CP, we can reduce a BCP instance to $n^{1.8+o(1)} \log n$ instances of CP, which is already larger than the conjectured $\Omega(n)$ quantum lower bound of BCP. By further studying the reduction, we find that the solution to BCP is the minimum of the solutions to the the constructed CP instances. Therefore, we can use the quantum minimum-finding algorithm to reduce the total time complexity to $\widetilde{O}(\sqrt{n^{1.8+\epsilon}} \cdot t_{\mathsf{CP}})$, which is enough to show that BCP is quantum fine-grained reducible to CP.

With the above-mentioned techniques, we quantize the classical fine-grained reductions, and show that $\mathsf{CNF\text{-}SAT}$, with conjectured lower bound $\Omega(2^{n/2})$, is quantum fine-grained reducible to OV and CP with lower bound $\Omega(n')$[5], when the dimension d is polylog(n').

7.2 Preliminaries

Definition 7.2 (Distance measure)**.** For any two vectors $a, b \in \mathbb{R}^d$, the distance between them in the ℓ_2-metric is denoted by $\|a - b\| = \left(\sum_{i=1}^{d} |a_i - b_i|^2\right)^{1/2}$. Their distance in the ℓ_0-metric (Hamming distance) is denoted by $\|a - b\|_0 = |\{i \in [d] : a_i \neq b_i\}|$, i.e., the number of coordinates on which a and b differ.

[5] n is the input size of $\mathsf{CNF\text{-}SAT}$, and n' is the input size of OV and CP.

7.2.1 Quantum query model

We consider the quantum query model in this work. Let $X := \{x_1, \dots, x_n\}$ be a set of n input points and $\mathcal{O}_X$ be the corresponding oracle. We can access the i-th data point x_i by making the query

$$|i\rangle |0\rangle \xrightarrow{\mathcal{O}_X} |i\rangle |x_i\rangle , \tag{7.1}$$

and we can make queries to elements in X in superposition. Note that $\mathcal{O}_X$ is an unitary transformation in the formula above. Hence, a quantum algorithm with access to $\mathcal{O}_X$ can be represented as a sequence of unitary transformations.

Consider a quantum algorithm $\mathcal{A}$ with access to an oracle $\mathcal{O}$ and a initial state $|0\rangle := |0\rangle_Q |0\rangle_A |0\rangle_W$, where the registers Q and A are for the queries and the answers from the oracle, and the register W is the working space which is always hold by $\mathcal{A}$. Then, we can represent the algorithm as

$$U_T \mathcal{O} U_{T-1} \cdots \mathcal{O} U_1 |0\rangle . \tag{7.2}$$

Let $|\psi\rangle_i = U_i \mathcal{O} \cdots \mathcal{O} U_1 |0\rangle := \sum_{i,z} |i\rangle_Q |0\rangle_A |z\rangle_W$ be the state right before applying the i-th $\mathcal{O}$, then

$$\mathcal{O} |\psi\rangle_i := \sum_{i,z} |i\rangle_Q |x_i\rangle_A |z\rangle_W . \tag{7.3}$$

7.2.2 Quantum subroutine for unstructured searching and minimum finding

Definition 7.3 (Unstructured search)**.** Given a set P of n elements in $\{0, 1\}$, decide whether there exists a 1 in P.

Theorem 7.5 (Grover's search algorithm [Gro96, NC10])**.** *There is a quantum algorithm for unstructured search with running time $O(\sqrt{n})$.*

By Theorem 7.5 and BBBV's argument [BBBV97], the quantum time complexity of unstructured search is $\Theta(\sqrt{n})$. We can also get a $\widetilde{O}(\sqrt{n})$ quantum algorithm for minimum finding by combining Grover's search algorithm and binary search.

Theorem 7.6 (Quantum minimum finding [DH96])**.** *There is a quantum algorithm that finds from a set of n elements with values in $\mathbb{R}$, the index of the minimum element of the set, with success probability $\frac{1}{2}$ and run time $\widetilde{O}(\sqrt{n})$.*

7.2.3 Problem definitions

In this subsection, we first formally define OV, CP, and BCP. Then we show the folklore algorithms for CP, BCP, and OV by Grover's algorithm, which run in time $\widetilde{O}(n)$.

Definition 7.4 (Orthogonal Vectors, OV)**.** Given two sets A, B of n vectors in $\{0,1\}^d$ as input, find a pair of vectors $a \in A$, $b \in B$ such that $\langle a, b\rangle = 0$, where the inner product is taken in $\mathbb{Z}$.[6]

We denote OV with input length n and dimension d as $\mathsf{OV}_{n,d}$. We will use this notation when we need to specify the parameters in the following sections.

Definition 7.5 (Closest Pair Problem, CP)**.** Given a set P of n points in $\mathbb{R}^d$ and a distance measure Δ, find a pair of distinct points $a, b \in P$ such that $\Delta(a,b)$ is the smallest among all distinct pairs in P.

Similar to OV, we denote CP with input length n and dimension d as $\mathsf{CP}_{n,d}$. We will use this notation when the parameters in the following sections are required to be specified. Note that in this work, we consider $\Delta(a,b) = \|a-b\|$ as the distance measure for CP and BCP.

Definition 7.6 (Bichromatic Closest Pair Problem, BCP)**.** Given two sets A, B of n points in $\mathbb{R}^d$ and a distance measure Δ, find a pair of points $a \in A$, $b \in B$ such that

$$\Delta(a,b) = \min_{a\in A, b\in B} \Delta(a,b). \tag{7.4}$$

[6]Our definition is slightly different than some of the literature, for example, [CW19], which is searching among pairs inside one set. Those two definitions are equivalent up to constant in complexities.

We also define an approximate version of BCP as follows.

Definition 7.7 ((1+ξ)-approximate Bichromatic Closest Pair Problem, (1+ξ)-BCP)**.** Given two sets A, B of n points $\in \mathbb{R}^d$ and a distance measure Δ, find a pair of points $a \in A$, $b \in B$ such that

$$\Delta(a, b) \leq (1 + \xi) \min_{a \in A, b \in B} \Delta(a, b). \tag{7.5}$$

Same as CP, we use $\mathsf{BCP}_{n,d}$ and $(1 + \xi)$-$\mathsf{BCP}_{n,d}$ to specify the parameters.

Definition 7.8 (Element Distinctness Problem, ED)**.** Let $f : [n] \to [m]$ be a given function. Decide whether there exist distinct $i, j \in [n]$ such that $f(i) = f(j)$.

For this problem, Ambainis [Amb07] gave a quantum algorithm with time complexity $\widetilde{O}(n^{2/3})$, which matches the lower bound proved by Aaronson and Shi [AS04] up to a polylogarithmic factor.

Theorem 7.7. *There are $\widetilde{O}(n)$-time quantum algorithms for* CP *and* BCP *when $d = O(\text{poly} \log n)$.*

Proof. We can solve CP and BCP by searching the minimum distance through all pairs by the algorithm of Theorem 7.6. There are $O(n^2)$ pairs and checking each pair took $O(d)$ time, so the total running time is $O(nd)$. For $d = O(\text{poly} \log n)$, the time complexity equals to $\widetilde{O}(n)$. □

7.2.4 Fine-grained complexity

As we have mentioned earlier in the introduction, a fine-grained reduction from problem P to Q with conjectured lower bounds $p(n)$ and $q(n)$, respectively, has the property that if we can improve the $q(n)$ time for Q, then we can also improve the $p(n)$ time for P. We give the formal definition by Williams [VW15] in below.

Definition 7.9 (Fine-grained reduction, [VW15]). Let $p(n)$ and $q(n)$ be non-decreasing functions of n. Problem P is (p,q)-reducible to problem Q, denoted as $(\mathsf{P},p) \leq_{\mathrm{FG}} (\mathsf{Q},q)$, if for every ϵ, there exist $\delta > 0$, an algorithm R for solving P with access to an oracle for Q, a constant d, and an integer $k(n)$, such that for every $n \geq 1$, the algorithm R takes any instance of P of size n and

- R runs in at most $d \cdot (p(n))^{1-\delta}$-time,
- R produces at most $k(n)$ instances of Q adaptively, that is, the jth instance X_j is a function of $\{(X_i, y_i)\}_{1\leq i<j}$ where X_i is the ith instance produced and y_i is the answer of the oracle for Q on instance X_i, and
- the sizes n_i of the instances X_i for any choice of oracle answers y_i obeys the inequality
$$\sum_{i=1}^{k(n)} (q(n_i))^{1-\epsilon} \leq d \cdot (p(n))^{1-\delta}. \tag{7.6}$$

Let $(\mathsf{P},p) \leq_{\mathrm{FG}} (\mathsf{Q},q)$ for some non-decreasing function $p(n)$ and $q(n)$. If for every $\epsilon > 0$, we can solve problem Q in time $q(n)^{1-\epsilon}$ with probability 1 for all input length n, then there exists a $\delta > 0$ such that we can solve the problem P in time $p(n)^{1-\delta}$ by Eq. (7.6).

Here are some known results about fine-grained reductions.

Theorem 7.8 ([KM20a, Wil05]).

$$(\mathsf{CNF\text{-}SAT}_n, 2^n) \leq_{\mathrm{FG}} (\mathsf{OV}_{n_1,d_1}, n_1^2) \leq_{\mathrm{FG}} (\mathsf{BCP}_{n_2,d_2}, n_2^2) \leq_{\mathrm{FG}} (\mathsf{CP}_{n_3,d_3}, n_3^2), \tag{7.7}$$

where $d_1 = \Theta(\log n_1)$, $d_2 = \Theta(\log n_2)$ *and* $d_3 = (\log n_3)^{\Omega(1)}$.

Remark 7.1. The second reduction from OV to BCP has been improved to $d_2 = 2^{O(\log^* n)}$ by Chen [Che18].

There are several plausible hypotheses in fine-grained complexity, which can imply conditional hardness results for many interesting problems. We first give the definition of the strong exponential time hypothesis (SETH).

Hypothesis 7.9 (Strong Exponential Time Hypothesis, SETH). For every $\epsilon > 0$, there exists a $k = k(\epsilon) \in \mathbb{N}$ such that no algorithm can solve k-SAT (i.e., satisfiability on a CNF of width k) in $O(2^{(1-\epsilon)m})$ time where m is the number of variables. Moreover, this holds even when the number of clauses is at most $c(\epsilon) \cdot m$ where $c(\epsilon)$ denotes a constant that depends only on ϵ.

Another popular conjecture is the orthogonal vector hypothesis (OVH):

Definition 7.10 (Orthogonal Vector Hypothesis, OVH)**.** For every $\epsilon > 0$, there exists a $c \geq 1$ such that $\mathsf{OV}_{n,d}$ requires $n^{2-\epsilon}$ time when $d = c \log n$.

Remark 7.2. Under SETH, we can have the following conclusions from Theorem 7.8:

- OVH is true.
- For all $\epsilon > 0$, there exists a $c > 0$ such that $\mathsf{BCP}_{n,c\log n}$ cannot be solved by any randomized algorithm in time $O(n^{2-\epsilon})$.
- For all $\epsilon > 0$, there exists a $c > 0$ such that $\mathsf{CP}_{n,(\log n)^c}$ cannot be solved by any randomized algorithm in time $O(n^{2-\epsilon})$.

7.2.5 The framework for quantum walk search

In this subsection, we review the quantum walk framework for the Markov chain search problem and demonstrate how to use it to solve the element distinctness problem. For simplicity, we use the transition matrix P to refer to a Markov chain, where $P = (p_{xy})_{x,y\in X}$ for X being the state space of P and p_{xy} being the transition probability from x to y. An irreducible and ergodic Markov chain has a unique stationary distribution π, which is also the unique eigenvector of P with eigenvalue 1. Let $M \subseteq X$ be a set of marked elements. In the Markov chain search problem, the objective is to find an $x \in M$. We can perform the following actions: setup, sampling from the π with cost S; update, making a transition with cost U, and checking whether the current state is marked or not with cost C. To solve the search problem classically, we perform a random walk as follows. Start from a point sampled from π and check if it is marked. If not, make a number of transitions on P until it mixes, and then check again. We then repeat this process until a marked state is found. The cost of this random walk algorithm is $O(\mathsf{S} + \frac{1}{\lambda}(\frac{1}{\delta}\mathsf{U} + \mathsf{C}))$, where $\lambda := |M|/|X|$ and δ is the spectral gap of P.

Quantum analogues of random walks, namely, quantum walks, have been developed for solving different problems. In 2003, Ambainis [Amb07] proposed a quantum walk algorithm for solving the element distinctness problem. His algorithm also solves the Markov chain search problem on the Johnson graph with cost $O(\mathsf{S} + \frac{1}{\sqrt{\lambda}}(\frac{1}{\sqrt{\delta}}\mathsf{U} + \mathsf{C}))$. In 2004, Szegedy [Sze04] gave a quantum walk algorithm for more generalized Markov chains with cost $O(\mathsf{S} + \frac{1}{\sqrt{\lambda\delta}}(\mathsf{U} + \mathsf{C}))$. We can view Szegedy's quantum walk as a quantum counterpart of a random walk, where one checks the state after each transition. Szegedy's quantum walk only detects the presence of a marked state, but cannot find one without extra costs. In 2006, Magniez et al. [MNRS11] proposed a quantum walk search framework that unified the advantages of the quantum walks in [Amb07] and [Sze04]. In this quantum walk framework, we can perform the following operations:

- **Setup:** with cost S. preparing the initial state $|\pi\rangle = \frac{1}{\sqrt{|X|}} \sum_x \sqrt{\pi_x}\,|x\rangle$.
- **Update:** with cost U. applying the transformation $|x\rangle\,|0\rangle \mapsto |x\rangle \sum_{y\in X} \sqrt{p_{xy}}\,|y\rangle$.
- **Checking:** with cost C, applying the transformation: $|x\rangle \mapsto \begin{cases} -|x\rangle & \text{if } x \in M \\ |x\rangle & \text{otherwise.} \end{cases}$

The main result of [MNRS11] is summarized as follows.

Lemma 7.10 ([MNRS11])**.** *Let P be an irreducible and ergodic Markov chain P on X. Let $M \subseteq X$ be a subset of marked elements. Let $\lambda := |M|/|X|$ and δ be the* *spectral gap of P. Then, there exists a quantum algorithm that with high probability, determines M is empty or finds an $x \in M$ with cost $O(\mathsf{S} + \frac{1}{\sqrt{\lambda}}(\frac{1}{\sqrt{\delta}}\mathsf{U} + \mathsf{C}))$.*

To solve the element distinctness problem, we define a Markov chain, following the work [Amb07, BJLM13, Jef14]. The state space X is all subsets of $[n]$ with size r. The Markov chain is based on the Johnson graph on X, where an edge is connecting S and S' if and only if $|S \cap S'| = r - 1$. The transition probability on each edge is hence $\frac{1}{r(n-r)}$. A state S is marked when there exist distinct $i, j \in S$ such the i^{th} and the j^{th} items are the same. The Markov chain has spectral gap $\delta \geq 1/r$ (see [Jef14]) and it is easy to verify that $\lambda \geq \binom{n-2}{r-2}/\binom{n}{r} = O(r^2/n^2)$. If we only consider the query complexity, the setup procedure costs r queries, the update procedure costs one query, and the checking procedure does not cost any query. Choosing $r = n^{2/3}$ yields the optimal query complexity $O(n^{2/3})$.

7.3 Quantum Fine-Grained Complexity

In this section, we give the formal definitions of the quantum fine-grained reduction and quantum strong exponential time hypothesis (QSETH). Moreover, we show that under QSETH, for $d = \text{polylog}(n)$, the lower bounds for $\mathsf{CP}_{n,d}$ and $\mathsf{OV}_{n,d}$ are $n^{1-o(1)}$, which nearly matches the upper bounds given in Theorem 7.7.

7.3.1 Quantum fine-grained reduction and QSETH

QSETH is defined based on the assumption that the best quantum algorithm for $\mathsf{CNF\text{-}SAT}$ is Grover search when the clause width k is large enough.

Hypothesis 7.11 (QSETH). For every $\epsilon > 0$, there exists a $k = k(\epsilon) \in \mathbb{N}$ such that no quantum algorithm can solve k-SAT (i.e., satisfiability on a CNF of width k) in $O(2^{(1/2-\epsilon)n})$ time where n is the number of variables. Moreover, this holds even when the number of clauses is at most $c(\epsilon)n$ where $c(\epsilon)$ denotes a constant that depends only on ϵ.

Obviously, the Grover search can solve $\mathsf{CNF\text{-}SAT}$ in $\widetilde{O}(2^{n/2})$. To the best of the our knowledge, there is no quantum algorithm that can do better than $O(2^{n/2})$ for any k.

We recall that in the quantum query model, the input of a problem is given by a quantum oracle. Specifically, let P be a problem, and X be an instance of P in the classical setting. Then, in the quantum query model, X will be given by an oracle $\mathcal{O}_X$. We will denote an algorithm or an oracle $\mathcal{A}$ with access to $\mathcal{O}_X$ by $\mathcal{A}(\mathcal{O}_X)$.

We say $\mathcal{A}_\epsilon$ is an ϵ-oracle for problem P, if for every instance $\mathcal{O}_X$, it holds that

$$\Pr[\mathcal{A}_\epsilon(\mathcal{O}_X) = \mathsf{P}(X)] \geq 1 - \epsilon, \tag{7.8}$$

and the running time is $O(1)$, where $\mathsf{P}(X)$ is the answer of X for problem P.

Definition 7.11 (Quantum oracles)**.** Let $X := \{x_1, \dots, x_n\}$ be an instance of some problem and $\mathcal{O}_X$ be the corresponding quantum oracle. To realize $\mathcal{O}_X$, we do not need to write down the whole X; instead, we can just design a quantum circuit to realize the mapping

$$|i\rangle\,|0\rangle \xrightarrow{\mathcal{O}_X} |i\rangle\,|x_i\rangle\,. \tag{7.9}$$

Definition 7.12 (Quantum fine-grained reduction)**.** Let $p(n)$ and $q(n)$ be nondecreasing functions of n. Let P and Q be two problems in the quantum query model and $\mathcal{A}_\epsilon$ be an ϵ-oracle for Q with error probability $\epsilon \leq 1/3$. P is quantum (p, q)-reducible to Q, denoted as $(\mathsf{P}, p) \leq_{\text{QFG}} (\mathsf{Q}, q)$, if for every ϵ, there exits a $\delta > 0$, and algorithm R with access to $\mathcal{A}_\epsilon$, a constant d, and an integer $k(n)$, such that for every $n \geq 1$, the algorithm R takes any instance of P of size n and satisfies the following:

- R can solve P with success probability at least $2/3$ in time at most $d \cdot p(n)^{1-\delta}$.
- R performs at most $k(n)$ quantum queries to A_ϵ. Specifically, in the j^{th} query, let $\mathbf{X}_j := \{X_{1,j}, X_{2,j}, \dots\}$ be a set instances of Q. Then, R realizes the oracles $\{\mathcal{O}_{X_{1,j}}, \mathcal{O}_{X_{2,j}}, \dots\}$ in superposition and applies A_ϵ to solve the instances.

- The following inequality holds.

$$\sum_{j=1}^{k(n)} c(\mathbf{X}_j) \cdot q(n_j)^{1-\epsilon} \leq d \cdot p(n)^{1-\delta},$$

  where $c(\mathbf{X}_j)$ is the time required for R to realize the oracles $\{\mathcal{O}_{X_{1,j}}, \mathcal{O}_{X_{2,j}}, \dots\}$ in superposition and $n_j := \max_i |X_{i,j}|$.

In Definition 7.12, the input of A_ϵ is given as a quantum oracle such that A_ϵ can be a quantum query algorithm with running time strictly less than the input size. Moreover, the quantum reduction R can realize quantum oracles $\{\mathcal{O}_{X_{1,j}}, \mathcal{O}_{X_{2,j}}, \dots\}$ in superposition, and thus the time required is $\max_i c(X_{i,j})$ (where $c(X_{i,j})$ is the time required to realize $\mathcal{O}_{X_{i,j}}$) instead of $\sum_i c(X_{i,j})$. This also allows R to use fast quantum algorithms to process the information of $A_\epsilon's$ output (e.g., amplitude amplification).

7.3.2 Lower bounds for CP, OV, and BCP in higher dimensions under QSETH

Here, we give nearly linear lower bounds for OV and CP under QSETH by showing that there exist quantum fine-grained reductions from SAT to these problems.

Theorem 7.12. *Assuming QSETH, for all $\epsilon > 0$, there exists a c such that $\mathsf{OV}_{n,c\log n}$ and $\mathsf{CP}_{n,(\log n)^c}$ cannot be solved by any quantum algorithm in time $O(n^{1-\epsilon})$.*

We prove Theorem 7.12 by showing that there exist quantum fine-grained reductions from CNF-SAT to OV, OV to BCP, and BCP to CP with desired parameters. We first give the reduction from CNF-SAT to OV as a warm-up.

Lemma 7.13.

$$(\mathsf{CNF\text{-}SAT}_n, 2^{n/2}) \leq_{\mathrm{QFG}} (\mathsf{OV}_{n_1,d_1}, n_1), \tag{7.10}$$

where $n_1 = 2^{n/2}$ and $d_1 = \Theta(n)$.

Proof. Let ϕ be a CNF formula with n variables and $m = \Theta(n)$ clauses. Let $\mathcal{A}$ be an algorithm for OV. We first recall the classical reduction. Let $\phi := \phi_1 \wedge \cdots \wedge \phi_m$. We divide the n variables into two sets A and B with $|A| = |B| = \frac{n}{2}$. Let $A := \{x_1, \ldots, x_{n/2}\}$ and $B := \{x_{n/2+1}, \ldots, x_n\}$. We let $S_A := \{a_1, \ldots, a_{2^{n/2}}\}$ be all assignments to A and $S_B := \{b_1, \ldots, b_{2^{n/2}}\}$ be all assignments to B. We describe two mappings $f_A : S_A \to \{0,1\}^m$ and $f_B : S_B \to \{0,1\}^m$ as follows:

$$f_A(a_i) = [\phi_1(a_i), \ldots, \phi_m(a_i)]^T, \text{ and} \tag{7.11}$$

$$f_B(b_i) = [\phi_1(b_i), \ldots, \phi_m(b_i)]^T, \tag{7.12}$$

where $\phi_j(a_i) = 0$ if a_i is a satisfied assignment for ϕ_j, and $\phi_j(a_i) = 1$ otherwise; we define $\phi_i(b_i)$ in the same way. Let $F_A := \{f_A(a_i) : \ i \in [2^{n/2}]\}$ and $F_B := \{f_B(b_i) : i \in [2^{n/2}]\}$. Then, it is obvious that if there exist $v \in F_A$ and $u \in F_B$ such that $\langle v, u \rangle = 0$, then ϕ is satisfiable. However, at first glance, this reduction with $O(2^{n/2})$ running time is not fine-grained since we require the cost of the reduction to be at most $2^{n(1-\delta)/2}$ for some $\delta > 0$ by Definition 7.12, but writing down elements in F_A and F_B already takes $\Omega(2^{n/2})$.

Nevertheless, as in Definition 7.11, a quantum fine-grained reduction only needs to realize the functions f_A and f_B, which takes $O(mkn)$ time where k is the width of clauses. This is much less than $O(2^{n(1-\delta)/2})$. More specifically, f_A and f_B are oracles for F_A and F_B, and for any quantum query to elements in F_A or F_B, the reduction can implement oracles f_A and f_B:

$$|e, x\rangle |0\rangle \xrightarrow{f_e} |e, x\rangle |f_e(x)\rangle, \tag{7.13}$$

where $e \in \{A, B\}$, and the time $c(f_e)$ for the reduction to implement f_e for one quantum query is at most $O(kmn)$. Finally, this reduction only uses one oracle (F_A, F_B). If there is an algorithm for OV which succeeds with probability $2/3$, we can boost the success probability of the reduction by repetition. Therefore, $(\mathsf{CNF\text{-}SAT}, 2^{n/2})$ is quantum reducible to $(\mathsf{OV}_{n_1,d_1}, n_1)$. $\square$

Then, to prove $(\mathsf{CNF\text{-}SAT}, 2^{n/2}) \leq_{\mathrm{QFG}} (\mathsf{CP}_{n_3,d_3}, n_3)$, we show that $(\mathsf{BCP}_{n_2,d_2}, n_2) \leq_{\mathrm{QFG}} (\mathsf{CP}_{n_3,d_3}, n_3)$ and $(\mathsf{OV}_{n_1,d_1}, n_1) \leq_{\mathrm{QFG}} (\mathsf{BCP}_{n_2,d_2}, n_2)$, where n_2, n_3, d_2, d_3 are some functions of n specified in the following lemmas.

Lemma 7.14. *For $d = \Theta(\log n)$,*

$$(\mathsf{BCP}_{n,d}, n) \leq_{\mathrm{QFG}} (\mathsf{CP}_{n',d'}, n'), \tag{7.14}$$

where $n' = n^{O(1)}$ and $d' = (\log n)^c$ for some constant c and all points have $\{0,1\}$ entries with the Hamming metric.

Remark 7.3. The points have coordinate entries in $\{0,1\}$, and the Hamming metric is equivalent to distance in ℓ_2-metric (up to power of 2) in this case. Therefore, in the proof of Lemma 7.14, we can consider the Hamming distance between points instead of ℓ_2 distance without loss of generality.

We first introduce the classical reductions in [KM20a] and some results we will use to prove Lemma 7.14.

Classical reduction We can consider an instance of BCP with two sets of points A and B as a weighted complete bipartite graph $K_{n,n}$, where the vertices are the points in these two sets and edges' weights are equal to the distances between the corresponding points. Then, solving BCP is equivalent to find an edge with the minimum weight in this graph. However, we cannot directly apply the algorithm for CP on this graph since there could be two points in the same set (no edge connecting them) that have a smaller distance than any pairs of points in two sets (connected by an edge). To overcome this difficulty, we can "stretch" the points to make the points in the same set far from each other, which is characterized by the contact dimension of a graph:

Definition 7.13 (Contact Dimension)**.** For any graph $G = (V, E)$, a mapping $\tau : V \to \mathbb{R}^d$ is said to realize G if for some $\beta > 0$, the following holds for every distinct vertices u, v:

$$\|\tau(u) - \tau(v)\|_2 = \beta \text{ if } \{u, v\} \in E, \qquad (7.15)$$
$$\|\tau(u) - \tau(v)\|_2 > \beta \text{ otherwise.}$$

The contact dimension of G, denoted by $\mathrm{cd}(G)$, is the minimum $d \in \mathbb{N}$ such that there exists $\tau : V \to \mathbb{R}^d$ realizing G.

That is, with the help of τ, we can restrict the optimal solution of CP to be the points connected by an edge in G. But we cannot realize the whole complete bipartite graph since $\mathrm{cd}(K_{n,n}) = \Theta(n)$, which makes the dimension of the CP instance too large. [KM20a] showed that we can realize a subgraph of $K_{n,n}$ and apply permutations to its vertices such that the union of these subgraphs cover $K_{n,n}$. In this way, BCP can be computed by solving CP on each subgraph and outputting the best solution. More specifically, the reduction in [KM20a] relies on the following theorem:

Theorem 7.15 (Theorem 4.2 in [KM20a])**.** *For every $0 < \delta < 1$, there exists a log-dense sequence $(n_i)_{i\in\mathbb{N}}$ such that, for every $i \in \mathbb{N}$, there is a bipartite graph $G_i = (A_i \dot\cup B_i, E_i)$ where $|A_i| = |B_i| = n_i$ and $|E_i| \geq \Omega(n_i^{2-\delta})$, such that $\mathrm{cd}(G_i) = (\log n_i)^{O(1/\delta)}$. Moreover, for all $i \in \mathbb{N}$, a realization $\tau : A_i \dot\cup B_i \to \{0,1\}^{(\log n_i)^{O(1/\delta)}}$ of G_i can be constructed in $n_i^{2+o(1)}$ time.*

The log-dense sequence is defined as follows:

Definition 7.14. A sequence $(n_i)_{i\in\mathbb{N}}$ of increasing positive integers is log-dense if there exists a constant $c \geq 1$ such that $\log n_{i+1} \leq c \cdot \log n_i$ for all $i \in \mathbb{N}$.

They also showed that, the permutations for covering the complete bipartite graph can be efficiently found, as shown in the following lemma.

Lemma 7.16 (Lemma 3.11 in [KM20a])**.** *For any bipartite graph $G(A \dot\cup B, E_G)$ where $|A| = |B| = n$ and $E_G \neq \emptyset$, there exist side-preserving permutations $\pi_1, \ldots, \pi_k$:*

$A \cup B \to A \cup B$ where $k \leq \frac{2n^2 \ln n}{|E_G|} + 1$ such that

$$\bigcup_{i \in [k]} E_{G_{\pi_i}} = E_{K_{n,n}}. \tag{7.16}$$

Moreover, such permutations can be found in $O(n^6 \log n)$ time.

Now, we are ready to state the quantum fine-grained reduction by "quantizing" the classical reduction.

Proof of Lemma 7.14. Let A, B be the two sets of input points of BCP. Suppose for BCP, there is an input oracle $\mathcal{O}_{\mathsf{BCP}}$ which, given an index, returns the corresponding point:

$$|b\rangle\,|i\rangle\,|0\rangle \xrightarrow{\mathcal{O}_{\mathsf{BCP}}} \begin{cases} |b\rangle\,|i\rangle\,|x_i\rangle & \text{if } b = 0, \\ |b\rangle\,|i\rangle\,|y_i\rangle & \text{if } b = 1, \end{cases} \tag{7.17}$$

where x_i is the i-th point in the set A and y_i is the i-th point in the set B. The sizes of A and B are both equal to n and each point is in $\{0,1\}^{d_1}$, where $d_1 = \Theta(\log n)$ is the dimension of BCP.

For CP, suppose there is a quantum algorithm $\mathcal{A}$ such that for m points in $\{0,1\}^{d_2}$ given by an oracle $\mathcal{M}_{CP}$, $\mathcal{A}^{\mathcal{M}_{CP}}$ returns the closest pair of these n points with probability at least $2/3$.

Then we need to transform $\mathcal{O}_{\mathsf{BCP}}$ to some oracles $\mathcal{M}_i$ for CP, such that by running $\mathcal{A}$ with $\mathcal{M}_i$ as input oracles, we can get the bichromatic closest pair between A and B. The reduction has four steps:

1. Pre-processing. We first follow the classical reduction to pre-process the input points of BCP. For some integer $n' \leq n^{0.1}$, we can partition A and B into n'-size subsets:

$$\begin{aligned} A &= A_1 \,\dot\cup\, \cdots \,\dot\cup\, A_r, \\ B &= B_1 \,\dot\cup\, \cdots \,\dot\cup\, B_r, \end{aligned} \tag{7.18}$$

where $r = \lfloor n/n' \rfloor$. Here, we assume that n is divisible by n'. It follows that

$$\mathsf{BCP}(A, B) = \min_{i,j \in [r]} \mathsf{BCP}(A_i, B_j). \tag{7.19}$$

Then, we use the algorithm in [KM20a] to construct k mappings $f_1, \ldots, f_k : [2n'] \to \{0,1\}^{d'}$ such that

$$\mathsf{BCP}(A_i, B_j) = \min_{t \in [k]} \mathsf{CP}(f_t(A_i) \cup f_t(B_j)) \quad \forall i, j \in [\lfloor n/n' \rfloor]. \tag{7.20}$$

More specifically, we pick n' to be the largest number in a log-dense sequence that is smaller than $n^{0.1}$. Then, we apply Theorem 7.15 to classically construct a bipartite graph $G(A \cup B, E)$ with n' vertices in each side and a realization τ. By choosing $\delta = \epsilon/2$ in Theorem 7.15, the graph G has $|E| = \Omega(n'^{2-\epsilon/2})$ edges. And we can get $2n'$ 0/1-strings of length $(\log n')^{O(2/\epsilon)}$:

$$\tau_i^A = \tau(u_i) \quad \forall u_i \in A, \quad \text{ and } \quad \tau_i^B = \tau(v_i) \quad \forall v_i \in B. \tag{7.21}$$

In order to cover the complete bipartite graph, we run the classical algorithm (Lemma 7.16) to find k permutations $\pi_1, \ldots, \pi_k : [n'] \to [n']$, where k is a parameter to be chosen later.

Then, we can define the mappings as follows:

$$f_t(u) = \begin{cases} x_v \circ \left(\tau^A_{\pi_t(w)}\right)^{d+1} & \text{if } 1 \le u \le n' \\ y_v \circ \left(\tau^B_{\pi_t(w)}\right)^{d+1} & \text{if } n' < u \le 2n' \end{cases} \quad \forall t \in [k], u \in [2n'], \tag{7.22}$$

where $\circ$ means string concatenation and $(s)^{d+1}$ denotes $d+1$ copies of the string s. For a point $p \in A_i \cup B_j$, $u \in [2n']$ is the index in this union-set, $v \in [n]$ is the index in the ground set A or B, and $w \in [n']$ is the index in the subset A_i or B_j. Further, if $1 \le u \le n'$, then $w := u$; otherwise, $w := u - n'$.

2. Oracle construction. For $i, j \in [r], t \in [k]$, we then construct the input oracle $\mathcal{M}_{i,j,t}$ for the problem $\mathsf{CP}(f_t(A_i) \cup f_t(B_j))$. For a query index $u \in [2n']$,

$$M_{i,j,t} \left|u\right\rangle \left|0\right\rangle = \left|u\right\rangle \left|f_t(u)\right\rangle . \tag{7.23}$$

With the help of the input oracle $\mathcal{O}_{\mathsf{BCP}}$, we can implement $\mathcal{M}_{i,j,t}$ in the following way:

1. Prepare an ancilla qubit $\left|b\right\rangle$ such that $b = 1$ if $u > n'$.
2. Transform $\left|u\right\rangle$ to $\left|v\right\rangle$, the index of the point in A or B, based on the value of b. Note that the index is unique. Hence, this transformation is unitary and can be easily achieved by a small quantum circuit.
3. Query $\mathcal{O}_{\mathsf{BCP}}$ with input $\left|b\right\rangle \left|v\right\rangle$. Assume $b = 0$. Then,

$$\left|b\right\rangle \left|v\right\rangle \left|0\right\rangle \xmapsto{\mathcal{O}_{\mathsf{BCP}}} \left|b\right\rangle \left|v\right\rangle \left|x_v\right\rangle . \tag{7.24}$$

4. Similar to the second step, the index w of the point in A_i and B_j can be computed from v by a unitary transformation:

$$\left|b\right\rangle \left|v\right\rangle \left|x_v\right\rangle \mapsto \left|b\right\rangle \left|w\right\rangle \left|x_v\right\rangle \tag{7.25}$$

5. Since each w corresponds to a unique string $\tau^A_{\pi_t(w)}$, we can attach $d+1$ copies of this string to the remaining quantum registers:

$$\left|b\right\rangle \left|w\right\rangle \left|x_v\right\rangle \mapsto \left|b\right\rangle \left|w\right\rangle \left|x_v\right\rangle \left|\left(\tau^A_{\pi_t(w)}\right)^{d+1}\right\rangle . \tag{7.26}$$

6. By recovering u from w, we get the final state:

$$\left|u\right\rangle \left|f_t(u)\right\rangle = \left|u\right\rangle \left|x_v, \left(\tau^A_{\pi_t(w)}\right)^{d+1}\right\rangle . \tag{7.27}$$

3. Query process By Eqs. (7.19) and (7.20), we have

$$\mathsf{BCP}(A,B) = \min_{i,j\in[r],t\in[k]} \mathsf{CP}(f_t(A_i)\cup f_t(B_j)). \tag{7.28}$$

Hence, we can use quantum minimum-finding algorithm in Theorem 7.7 over the sub-problems to find the minimum solution. For each sub-problem, we can run the algorithm for CP with $\mathcal{M}_{i,j,t}$ as the input oracle.

4. Post-processing. In case that n is not divisible by n', let the remaining points in A and B be A_{res}, B_{res}, respectively. Then, we can use Grover search to find the closest pair between A_{res} and B, and between B_{res} and A. Then, compare the answer to the previously computed result and pick the smaller one.

Correctness. In this reduction, we do not change the constructions of the mappings $\{f_i\}_{i\in[k]}$. By [KM20a], Eq. (7.28) is correct in the classical setting. Hence, it also holds in the quantum setting, and we can use Grover search to find the minimum solution. However, since the algorithm $\mathcal{A}$ for CP has success probability $2/3$, for each tuple $(i,j,t)\in[r]\times[r]\times[k]$, we need to run $\mathcal{A}^{\mathcal{M}_{i,j,t}}$ $O(\log n)$ times to boost the success probability to at least $1-\frac{1}{n}$. Then, by the union bound, the probability that all queries in the Grover search are correct is at least $99/100$. Hence, by Theorem 7.6, the overall success probability is at least $2/3$.

Running Time of the Reduction. The running time of the pre-processing step consists of two parts: (1) constructing the graph G and its realization τ; (2) finding k permutations. For the first part, by Theorem 7.15, it can be done in $n'^{2+o(1)}$ time. For the second part, we pick $k=O(\frac{2n'^2\log n'}{n'^{2-\epsilon/2}})=O(n'^{\epsilon/2}\log n')$, and by Lemma 7.16, it can be done in $O(n'^6\log n')$ time. Hence, the total running time of pre-processing step is $n'^{2+o(1)}+O(n'^6\log n')=\widetilde{O}(n^{0.6})$.

The oracle construction can be done "on-the-fly". More specifically, given the strings $\{\tau_i^A,\tau_i^B\}_{i\in[n']}$, and permutations $\{\pi_i\}_{i\in[k]}$, for each query index u, we can simulate the oracle $\mathcal{M}_{i,j,t}$ defined in Eq. (7.23) in $c(\mathcal{M}_{i,j,t}) = O(d_2) = (\log n')^{\Omega(1)} = \widetilde{O}(1)$ time.

In the query process, for each CP instance indexed by (i, j, t), suppose $\mathcal{A}^{\mathcal{M}_{i,j,t}}$ gets the answer in time $q(n') = n'$. Moreover, for each time $\mathcal{A}$ querying the input oracle $\mathcal{M}_{i,j,t}$, we need to spend $c(\mathcal{M}_{i,j,t})$ time to simulate the oracle. And we also have $O(\log n)$ runs for each instance. Hence, the total running time for each CP is at most

$$n'^{1-\epsilon} \cdot \widetilde{O}(1) \cdot O(\log n) = \widetilde{O}(n'^{1-\epsilon}). \tag{7.29}$$

Then, we use Grover's search algorithm over $r^2 \cdot k$ instances, which can be done by querying $\widetilde{O}(\sqrt{r^2 \cdot k})$ instances by Theorem 7.6. Therefore, for any $\epsilon > 0$, we have

$$\widetilde{O}(\sqrt{r^2 k}) \cdot q(n')^{1-\epsilon} \cdot c(\mathcal{M}_{i,j,t}) \cdot O(\log n) = \widetilde{O}(\sqrt{(n/n')^2 k} \cdot (n')^{1-\epsilon}) \tag{7.30}$$

$$\leq \widetilde{O}(n \cdot (n')^{-\epsilon}) \leq \widetilde{O}(n \cdot n^{-\epsilon/2}) \leq n^{1-\delta}, \tag{7.31}$$

where the first inequality follows from $k = O(n'^{\epsilon/2} \log n')$ as shown in [KM20a] and the last inequality follows by setting $\delta = \epsilon/10$.

For the post-processing step, the sizes of A_{res} and B_{res} are at most n'. The running time is

$$O(\sqrt{n \cdot n'} \cdot \log n) \leq \widetilde{O}(n^{0.55}). \tag{7.32}$$

Therefore, for any $\epsilon > 0$, there exists a $\delta > 0$ such that the Eq. (7.30) holds and the total reduction time is $O(n^{1-\delta})$. By Definition 7.12, BCP_{n,d_1} can be quantum fine-grained reduced to CP_{n,d_2}. This completes the proof of this lemma. $\square$

Finally, we show that $(\mathsf{OV}_{n,d}, n) \leq_{\mathrm{QFG}} (\mathsf{BCP}_{n,d'}, n)$ by quantizing the reduction in [KM20a] following the same idea.

Lemma 7.17. *For $d = \Theta(\log n)$,*

$$(\mathsf{OV}_{n,d}, n) \leq_{\mathrm{QFG}} (\mathsf{BCP}_{n,d'}, n), \tag{7.33}$$

where $d' = \Theta(\log n)$.

Proof. For an OV instance with sets of vectors A and B, let $\mathcal{O}_{\mathsf{OV}}$ be the input oracle such that:

$$\mathcal{O}_{\mathsf{OV}} |i\rangle |0\rangle = \begin{cases} |i\rangle |a_i\rangle & \text{if } i \in A, \\ |i\rangle |b_i\rangle & \text{if } i \in B. \end{cases} \tag{7.34}$$

where $a_i, b_i \in \{0,1\}^d$.

Then, similar to the classical reduction, we can construct mappings $f_A, f_B : \{0,1\}^d \to \{0,1\}^{5d}$ such that

$$f_A(a_i)_{5j-4:5j} = \begin{cases} 11000 & \text{if } a_i(j) = 0 \\ 00110 & \text{if } a_i(j) = 1 \end{cases} \quad \forall j \in [d], \tag{7.35}$$

and

$$f_B(b_i)_{5j-4:5j} = \begin{cases} 10100 & \text{if } b_i(j) = 0, \\ 01001 & \text{if } b_i(j) = 1. \end{cases} \quad \forall j \in [d]. \tag{7.36}$$

By the classical reduction, we have

$$\mathsf{OV}(A,B) = 1 \text{ if and only if } \mathsf{BCP}(f_A(A), f_B(B)) = 2d \tag{7.37}$$

under Hamming distance.

Also, note that we can simulate the input oracle $\mathcal{O}_{\mathsf{BCP}}$ by first querying the oracle $\mathcal{O}_{\mathsf{OV}}$ to get the vector, then applying the corresponding mapping f_A or f_B, which can be done in $c(\mathcal{O}_{\mathsf{BCP}}) = O(d)$ time. Let the running time of the algorithm for BCP be $q(n) = n$. Then for any $\epsilon > 0$,

$$q(n)^{1-\epsilon} \cdot c(\mathcal{O}_{\mathsf{BCP}}) = n^{1-\epsilon} \cdot \Theta(\log n) \leq n^{1-\delta} \tag{7.38}$$

for some small $\delta > 0$. Hence, by Definition 7.12, $(\mathsf{OV}_{n,d}, n) \leq_{\text{QFG}} (\mathsf{BCP}_{n,d'}, n)$. $\square$

Proof of Theorem 7.12. We can prove the theorem by contradiction following Lemma 7.13, Lemma 7.17, and Lemma 7.14. Specifically, suppose that there exists an $\epsilon > 0$, for all $d = \Theta(\log n)$, there exists a quantum algorithm which can solve OV in time $O(n^{1-\epsilon})$. Then, we can obtain a quantum algorithm for $\mathsf{CNF\text{-}SAT}$, which runs in time $O(2^{n/2(1-\epsilon)})$ by Lemma 7.13. This contradicts QSETH. The proof for CP is the same. □

7.3.3 Quantum lower bound for BCP in nearly-constant dimensions under QSETH

A byproduct of the previous subsection is a quantum lower bound for BCP in higher dimensions (i.e., $d = \text{polylog}(n)$) under QSETH (Lemma 7.17). In this subsection, we show that this quantum lower bound for BCP even holds for nearly-constant dimensions (i.e., $d = c^{\log^*(n)}$). The main result of this subsection is the following theorem.

Theorem 7.18. *Assuming QSETH, there is a constant c such that* BCP *in $c^{\log^*(n)}$ dimensions requires $n^{1-o(1)}$ time for any quantum algorithm.*

We will "quantize" the results by Chen [Che18] to prove this theorem. More specifically, we first show a quantum fine-grained self-reduction of OV from $\log n$ dimensions with binary entries to $2^{\log^*(n)}$ dimensions with integer entries ($\mathbb{Z}$-OV). Then, we give a quantum fine-grained reduction from $\mathbb{Z}$-OV to BCP in nearly-constant dimensions.

Definition 7.15 (Integral Orthogonal Vector, $\mathbb{Z}$-OV)**.** Given two sets A, B of n vectors in $\mathbb{Z}^d$, find a pair of vectors $a \in A$ and $b \in B$ such that $\langle a, b\rangle = 0$, where the inner product is taken in $\mathbb{Z}$.

We use $\mathbb{Z}$-$\mathsf{OV}_{n,d}$ to denote $\mathbb{Z}$-OV with n vectors of d dimension in each set. We then recap a theorem in [Che18]:

Theorem 7.19 ([Che18, Theorem 4.1])**.** *Let b, ℓ be two sufficiently large integers. There is a classical reduction $\psi_{b,\ell} : \{0,1\}^{b\cdot\ell} \to \mathbb{Z}^{\ell}$ and a set $V_{b,\ell} \subseteq \mathbb{Z}$, such that for* *every* $x, y \in \{0,1\}^{b\cdot\ell}$,

$$\langle x, y\rangle = 0 \;\Leftrightarrow\; \langle \psi_{b,\ell}(x), \psi_{b,\ell}(y)\rangle \in V_{b,\ell} \tag{7.39}$$

and

$$0 \le \psi_{b,\ell}(x)_i < \ell^{6^{\log^*(b)}\cdot b} \tag{7.40}$$

for all possible x *and* $i \in [\ell]$. *Moreover, the computation of* $\psi_{b,\ell}(x)$ *takes* $\mathrm{poly}(b\cdot\ell)$ *time, and the set* $V_{b,\ell}$ *can be constructed in* $O\left(\ell^{O(6^{\log^*(b)}\cdot b)}\cdot \mathrm{poly}(b\cdot\ell)\right)$ *time.*

Note that the size of $V_{b,\ell}$ is at most $\ell^{2\cdot 6^{\log^*(b)}\cdot b+1}$. The following lemma gives a quantum fine-grained reduction from OV to $\mathbb{Z}$-OV:

Lemma 7.20. *For* $d = \Theta(\log n)$,

$$(\mathsf{OV}_{n,d}, n) \le_{\mathrm{QFG}} (\mathbb{Z}\text{-}\mathsf{OV}_{n,d'}, n). \tag{7.41}$$

where $d' = 2^{O(\log^* n_2)}$.

Proof. Consider an $\mathsf{OV}_{n,d}$ with $d = c\cdot\log n$, where c is an arbitrary constant. We choose $\ell := 7^{\log^* n}$ and $b := d/\ell$. Then, we can apply Theorem 7.19 to get the mapping function $\psi_{b,\ell}$ and the set $V_{b,\ell}$. For each $v \in V_{b,\ell}$, we'll construct an instance of $\mathbb{Z}$-$\mathsf{OV}_{n,\ell+1}$ as follows:

1. Let $|i\rangle$ be the input query index of $\mathbb{Z}$-$\mathsf{OV}_{n,\ell+1}$.
2. Query $\mathsf{OV}_{n,d}$'s input oracle $\mathcal{O}_{\mathsf{OV}}$ and get the vector $|i, x\rangle$.
3. Compute the mapping $\psi_{b,\ell}$ and get $|i, x\rangle\,|\psi_{b,\ell}(x)\rangle$.
4. If $x \in A$, then attach 1 to the end of the register: $|i, x\rangle\,|\psi_{b,\ell}(x), 1\rangle$. If $x \in B$, then attach $-v$ to the end: $|i, x\rangle\,|\psi_{b,\ell}(x), -v\rangle$.
5. Use $\mathcal{O}_{\mathsf{OV}}$ to erase x and return the final input state $|i\rangle\,|\psi_{b,\ell}(x), 1\rangle$ or $|i\rangle\,|\psi_{b,\ell}(x), -v\rangle$.

For each instance, we can use the quantum oracle for $\mathbb{Z}$-$\mathsf{OV}_{n,\ell+1}$ to check the orthogonality. $\mathsf{OV}_{n,d}$ is YES if and only if there exists a YES-instance of $\mathbb{Z}$-$\mathsf{OV}_{n,\ell+1}$.

Correctness. The correctness follows from Eq. (7.39):

$$\langle x, y\rangle = 0 \Leftrightarrow \langle \psi_{b,\ell}(x), \psi_{b,\ell}(y)\rangle = v \in V_{b,\ell} \Leftrightarrow \langle [\psi_{b,\ell}(x),\ 1], [\psi_{b,\ell}(y),\ -v]\rangle = 0. \tag{7.42}$$

Reduction time. Note that for $\ell = 7^{\log^* n}$ and $b = d/\ell$, we have:

$$\begin{aligned}
\log\left(\ell^{O(6^{\log^*(d)}\cdot b)}\right) &= \log \ell \cdot O\left(6^{\log^*(d)} \cdot (d/\ell)\right) &(7.43)\\
&= O\left(\log^*(n) \cdot 6^{\log^* n} \cdot c \log n / 7^{\log^* n}\right) &(7.44)\\
&= o(\log n). &(7.45)
\end{aligned}$$

This implies that $|V_{b,\ell}| \leq \ell^{2\cdot 6^{\log^*(b)}\cdot b+1} \leq n^{o(1)}$. Hence, the number of $\mathbb{Z}\text{-}\mathsf{OV}_{n,\ell+1}$ instances is $n^{o(1)}$ and the running time for compute $V_{b,\ell}$ is $n^{o(1)}$. And for each input query, the oracle for $\mathbb{Z}\text{-}\mathsf{OV}_{n,\ell+1}$ can be simulated in $c(\mathcal{O}_{\mathbb{Z}\text{-}\mathsf{OV}}) = \mathrm{poly}(d) = \mathrm{poly}(\log n)$ time. We can show that for every $\epsilon > 0$, if $\mathbb{Z}\text{-}\mathsf{OV}_{n,\ell+1}$ can be decided in $n^{1-\epsilon}$ time, then

$$\sum_{v \in V_{b,\ell}} n^{1-\epsilon} \cdot c(\mathcal{O}_{\mathbb{Z}\text{-}\mathsf{OV}}) = n^{o(1)} \cdot n^{1-\epsilon} \cdot \mathrm{poly}(\log n) \leq n^{1-\delta} \tag{7.46}$$

for some $\delta > 0$, which satisfies the definition of quantum fine-grained reduction (Definition 7.12).

Therefore, $\mathsf{OV}_{n,O(\log n)}$ is quantum fine-grained reducible to $\mathbb{Z}\text{-}\mathsf{OV}_{n,2^{O(\log^*(n))}}$.

□

Then, we give a quantum fine-grained reduction from $\mathbb{Z}$-OV to BCP:

Lemma 7.21. *For $d = 2^{O(\log^* n)}$,*

$$(\mathbb{Z}\text{-}\mathsf{OV}_{n,d}, n) \leq_{\mathrm{QFG}} (\mathsf{BCP}_{n,d'}). \tag{7.47}$$

where $d' = d^2 + 2$.

Proof. We remark here that this proof closely follows that for Theorem 4.3 in [Che18]. We nonetheless give it here as some details are different.

For an $\mathbb{Z}\text{-}\mathsf{OV}_{n,d}$ instance with $(k \cdot \log n)$-bit entries, we construct a BCP instance as follows:

1. For $x \in A$, construct a vector $x' \in \mathbb{Z}^{d^2}$ such that $x'_{i,j} = x_i \cdot x_j$. Here, we index a d^2-dimensional vector by $[d] \times [d]$. Similarly, for $y \in B$, construct a vector $y' \in \mathbb{Z}^{d^2}$ such that $y'_{i,j} = -y_i \cdot y_j$.

2. Choose $W := (d^2 + 1) \cdot n^{4k}$. For each x', construct a vector $x'' \in \mathbb{R}^{d^2+2}$ such that
$$x'' = \left[x',\ \sqrt{W - \|x'\|_2^2},\ 0 \right]. \tag{7.48}$$
For each y', construct a vector $y'' \in \mathbb{R}^{d^2+2}$ such that
$$y'' = \left[y',\ 0,\ \sqrt{W - \|y'\|_2^2} \right]. \tag{7.49}$$

Then, we claim that the $\mathbb{Z}\text{-}\mathsf{OV}$ instance is YES if and only if the BCP instance has the minimum distance $\leq \sqrt{2W}$.

Correctness. First note that $\|x'\|_2^2 \leq d^2 \cdot (2^{k \log n})^4 = d^2 \cdot n^{4k}$. Hence, $W - \|x'\|_2^2 > 0$ and $W - \|y'\|_2^2 > 0$. For any x'' and y'' in the new constructed instance of BCP, we have
$$\begin{aligned} \|x'' - y''\|_2^2 &= \|x''\|_2^2 + \|y''\|_2^2 - 2 \cdot \langle x'', y'' \rangle && (7.50)\\ &= 2 \cdot W - 2 \cdot \langle x', y' \rangle && (7.51)\\ &= 2 \cdot W - 2 \cdot \sum_{(i,j) \in [d] \times [d]} x_i \cdot x_j \cdot (-y_j \cdot y_j) && (7.52)\\ &= 2 \cdot W + 2 \cdot (\langle x, y \rangle)^2. && (7.53) \end{aligned}$$

Hence,
$$\langle x, y \rangle = 0 \ \Leftrightarrow\ \|x'' - y''\|_2^2 = 2W. \tag{7.54}$$

Reduction time. We can see from the above description that the input mapping function is simple and can be computed by a small quantum circuit in $O(d^2) = O(2^{O(\log^*(n))})$ time. Hence, we have $c(\mathcal{O}_{\mathsf{BCP}}) = O(2^{O(\log^*(n))})$. Also, by Definition 7.12, it's easy to check that this is indeed a quantum fine-grained reduction from $\mathbb{Z}$-OV to BCP. $\square$

Now Theorem 7.18 follows immediately from Lemma 7.20 and Lemma 7.21:

Proof of Theorem 7.18. Let $\epsilon > 0$ be some constant. Suppose we can solve $\mathsf{BCP}_{n,c^{\log^*(n)}}$ in $n^{1-\epsilon}$ time for all constant $c > 0$. Then, by Lemma 7.20 and Lemma 7.21, we can also solve $\mathsf{OV}_{n,c' \log n}$ in $n^{1-\delta}$ time for some $\delta > 0$ and any $c' > 0$. However, this contradicts QSETH by Theorem 7.12. Therefore, assuming QSETH, there exists a constant c such that $\mathsf{BCP}_{n,c^{\log^*(n)}}$ requires $n^{1-o(1)}$ time. $\square$

7.4 Closest Pair in Constant Dimension

In this section, we show that there exist almost-optimal quantum algorithms for CP in constant dimension. The main result is the following theorem, which is a direct consequence of Corollary 7.32 and Theorem 7.33.

Theorem 7.22. *For any constant dimension, the quantum time complexity for* CP *is* $\widetilde{\Theta}(n^{2/3})$.

Our approach to solve CP is first reducing to the decision version of the problem, and then apply quantum walk algorithms to solve the decision version. We define the decision version of CP, CP_ϵ, as follows.

Definition 7.16 (CP_ϵ)**.** Given a set of points $P \subset \mathbb{R}^d$ and $\epsilon \in \mathbb{R}$, find a pair $a, b \in P$ such that $\|a - b\| \leq \epsilon$ if there is one and returns no is no such pair exists.

The reduction from CP to CP_ϵ is given by the following lemma.

Lemma 7.23. *Let m be the number of bits needed to encode each coordinate as a bit string and d be the dimension. Given an oracle $\mathcal{O}$ for CP_ϵ, there exists an algorithm $A^{\mathcal{O}}$ that runs in time and query complexity $O(m + \log d)$ that solves the CP.*

Proof. Let (P, δ) be an instance of the CP. We first pick an arbitrary pair $a_0, b_0 \in P$ and compute $\Delta(a_0, b_0)$. Then, we set ϵ to be $\Delta(a_0, b_0)/2$ and run the oracle $\mathcal{O}$ to check whether there exists a distinct pair with distance less than $\Delta(a_0, b_0)/2$ or not. If there exists such a pair, which we denote as (a_1, b_1), then we set $\epsilon = \Delta(a_1, b_1)$ and call $\mathcal{O}$ to check again. If there is no such pair, then we set $\epsilon = 3\Delta(a_0, b_0)/4$ and call $\mathcal{O}$. We run this binary search for $m + \log d$ iterations. Finally, the algorithm outputs the closest pair. $\square$

In classical setting, point location is an important step in solving the closest-pair problem, especially the dynamic version. For the quantum algorithm, as walking on the Markov chain, we repeatedly delete a point and add a new point. Hence, in each step, the first thing is to determine the location of the new added point.

For simplicity, we assume that $m = O(\log n)$, which is the number of digits of each coordinate of the points. By translation, we can further assume that all the points are lying in $[0, L]^d$, where $L = O(2^m) = \text{poly}(n)$.

Since we are considering CP_ϵ, one simple way of point location is to discretize the whole space into a hypergrid, which is defined as follows:

Definition 7.17. Let $d, \epsilon, L > 0$. A hypergrid $G_{d,\epsilon,L}$ in the space $[0, L]^d$ consists of all ϵ-boxes

$$g := [a_1, b_1) \times [a_2, b_2) \times \cdots \times [a_d, b_d), \tag{7.55}$$

such that $b_1 - a_1 = \cdots = b_d - a_d = \epsilon/\sqrt{d}$ [7], and a_i is divisible by ϵ for all $i \in [d]$.

[7] The diagonal length of an ϵ-box is ϵ.

For each point $p_i \in [0, L]^d$, we can identify the ϵ-box that contains it using the function $\mathrm{id}(p_i) : [0, L]^d \to \{0, 1\}^{d \log(L/\epsilon)}$:

$$\mathrm{id}(p_i) = \left(\lfloor p_i(1)/w \rfloor , \lfloor p_i(2)/w \rfloor , \ldots , \lfloor p_i(d)/w \rfloor \right), \tag{7.56}$$

where $w = \frac{\epsilon}{\sqrt{d}}$ is the width of the ϵ-box. The number of bits to store $\mathrm{id}(p_i)$ is $d \cdot \log(L/w) = O(d \cdot \log(L))$. Since all the points in an ϵ-box have the same id, we also use this $g(\mathrm{id}(p))$ to denote this ϵ-box containing p.

For the ease of our analysis, we define the neighbors of a hypergrid.

Definition 7.18. Let $\epsilon \in \mathbb{R}$. Let g_1, g_2 be two ϵ-boxes in a hypergrid where $\mathrm{id}(g_1) = (x_1, \ldots, x_d)$ and $\mathrm{id}(g_2) = (x'_1, \ldots, x'_d)$. We say that g_1 and g_2 are each other's ϵ-*neighbor* if

$$\sqrt{\sum_{i=1}^{d} \|x_i - x'_i\|^2} \le \epsilon \tag{7.57}$$

Note that the number of ϵ-neighbors of a ϵ-box is at most $(2\sqrt{d}+1)^d$. We also have the following observation:

Observation 7.24. *Let $p_1, p_2 \in [0, L]^d$ be any two distinct points.*

- *If p_1 and p_2 are in the same ϵ-box, then $\Delta(p_1, p_2) \le \epsilon$.*
- *If $\Delta(p_1, p_2) \le \epsilon$, then $g(\mathrm{id}(p_1))$ must be an ϵ-neighbor of $g(\mathrm{id}(p_2))$.*

To solve CP_ϵ with quantum walk, we need data structures to keep track of the pairs that have distance at most ϵ. The desired data structure should have size $\widetilde{O}(n^{2/3})$, insertion/deletion time $O(\log n)$, and one should be able to check whether there exist pairs of distance at most ϵ in time $O(\log n)$. In addition, as pointed out in [Amb07], the data structure should have the following two properties:

- the data structure should have the bounded *worst-case* performance rather than *average-case* performance;

- the representation of the data structure should be history-independent, i.e., the data is uniquely represented regardless of the order of insertions and deletions.

We need the first property since the data structure may take too long for some operations, and this is not acceptable. The second property is required because, otherwise, the interference of quantum states would be messed up. In [Amb07], a hash table and a skip list is used to for solving the element distinctness problem using quantum walks. In [BJLM13], a simpler data structure, namely, a radix tree, is used to achieve the same performance. More details of using a radix tree to solve the element distinctness can be found in [Jef14]. Similar to the quantum data structure model in [Amb07, BJLM13, Jef14], we need the *quantum random access gate* to efficiently access data from a quantum memory, whose operation is defined as:

$$|i, b, z_1, \ldots, z_m\rangle \mapsto |i, z_i, z_1, \ldots, z_{i-1}, b, z_{i+1}, z_m\rangle\,, \tag{7.58}$$

where $|z_1, \ldots, z_m\rangle$ is some data in a quantum memory with m qubits. We assume this operation takes $O(\log m)$ time.

In the remainder of this section, we present two quantum algorithms for solving CP_ϵ. The data structures of both versions are based on the *augmented radix tree*, which we discuss in detail in the following subsection.

7.4.1 Radix tree for at most one solution

The purpose of the augmented radix tree is to quickly locate the points in an ϵ-box given its id. An ordinary radix tree is a binary tree that organizes a set of keys which are represented as binary strings. Each edge is labeled by a substring of a key and each leaf is labeled by a key such that concatenating all the labels on the path from the root to a leaf yields the key for this leaf. In addition, for each internal node, the labels of the two edges connecting to two children start with different bit. Note that in this definition, we implicitly merge all internal nodes that have only one child.

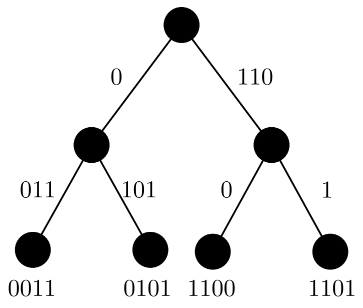


Figure 7.2: The uniquely represented radix tree that stores the keys $\{0011, 0101, 1100, 1101\}$.

The radix tree is uniquely represented for any set of keys. An example of a radix tree is shown as Fig. 7.2.

Our basic radix tree is essentially the one in [BJLM13, Jef14] with modification on the nodes' internal structure. We highlight the extra information stored in the radix tree. First we use a *local counter* to store the number of points in this ϵ-box; second, we use a *flag* in each leaf node to indicate whether there is a point in this ϵ-box that is in some pair with distance at most ϵ. The flag bit in an internal node is the OR of the ones in its children. The local counter in each internal node is the sum of the local counters in its children. We also store at most two points that are in the ϵ-box corresponding to this node. More precisely, let S be a subset of indices of the input points. We use $\tau(S)$ to denote the radix tree associated with S. Then, $\tau(S)$ consists of at most $r\lceil \log r \rceil$ nodes. Each node consists of the following registers:

$$\mathcal{D} \times \mathcal{M}_1 \times \mathcal{M}_2 \times \mathcal{M}_3 \times \mathcal{C} \times \mathcal{F} \times \mathcal{P}_1 \times \mathcal{P}_2, \tag{7.59}$$

where $\mathcal{D}$ stores the id of an ϵ-box for a leaf (and a substring of an id for an internal node) using $O(d \log(L/\epsilon))$ bits. $\mathcal{M}_1, \mathcal{M}_2$, and $\mathcal{M}_3$ use $O(\log n)$ bits to store the pointers to its parent, left child, and right child, respectively as well as the labels of the three edges connecting them to this node, $O(\log n)$ bits to store the labels of the three edges incident to it. $\mathcal{C}$ uses $O(\log n)$ bits to store the local counter. $\mathcal{F}$ stores the flag bit. $\mathcal{P}_1$ and $\mathcal{P}_2$ stores the coordinates of at most two points in this ϵ-box, which takes $O(d \log L)$ bits. The two points are stored in ascending order of their indices.

We need to pay attention to the layout of $\tau(S)$ in memory. We use three times more bits than needed to store $\tau(S)$, this will ensure that there are always more than 1/3 of the bits that are free. We divide the memory into cells where each cell is large enough to store one leaf node of $\tau(S)$. Besides $\tau(S)$, we also store a bitmap $\mathcal{B}$, which takes $O(\log n)$ bits to encode the current free cells (with "1" indicating occupied and "0" indicating free). To make the radix tree history-independent, we use a quantum state which is the uniform superposition of basis states $|\tau(S), B\rangle$ for all possible valid layout of $\tau(S)$ and it corresponds to the bitmap $\mathcal{B}$.

Insertion and deletion from $\tau(S)$ takes $O(\log n)$ time. Checking the presence of an ϵ-close pair takes constant time — we just need to read the flag bit in the root. Preparing the uniform superposition of all $i \in S$ can be done in $O(\log n)$ time by performing a controlled-rotation on each level of the radix tree where the angles are determined by the local counters in the two children of a node.

In the following subsections, we present the two versions of our algorithms. The first version invokes the quantum walk framework only once and its data structure maintains the existence of an ϵ-close pair. The second version uses a much simpler data structure, but it is only capable of handling CP_ϵ with a unique solution. Hence it requires invoking the quantum walk framework multiple times to solve the general CP_ϵ. These two quantum algorithms have almost the same time complexity.

7.4.2 Single-shot quantum walk with complicated data structure

To handle multiple solutions, our data structure is a composition of an augmented radix tree, a hash table, and a skip list. We give a high-level overview of our data structure as follows. Recall that by the discretization of the space into ϵ-boxes, it is possible that a pair of points in different ϵ-boxes have distance at most ϵ, but one only needs to check $(2\sqrt{d}+1)^d$ ϵ-neighbors to detect such a case. We maintain a list of points for each nonempty ϵ-box in an efficient way. A hash table is used to store the tuple (i, p_i) which is used to quickly find the point p_i, given its index i. The points are also stored in a skip list for each nonempty ϵ-box, ordered by its index i, which allows for quick insertion and deletion of points. Each ϵ-box is encoded into a unique key, and a radix tree is used to store such key-value pairs, where the value is associated with a skip list. The flag bits in this radix tree maintain the presence of an ϵ-close pair.

In the following, we present the details of the data structure and show it has all the desired properties.

Hash table. The hash table we use is almost the same as the one used in [Amb07], except that we do not store the $\lfloor \log r \rfloor$ counters in each bucket to facilitate the diffusion operator (which is handled easily here in the quantum walk on a Johnson graph). Our hash table has r buckets, where each bucket contains $\lceil \log n \rceil$ entries. We use a fixed hash function $h(i) = \lfloor ir/n \rfloor + 1$ to hash $\{1, \ldots, n\}$ to $\{1, \ldots, r\}$. That is, for $j \in [r]$, the j-th bucket contains the entries for (i, p_i) in ascending order of i, where $i \in S$ and $h(i) = j$.

The entry for (i, p_i) contains the tuple (i, p_i) and $\lceil \log n \rceil + 1$ pointers to other entries. These pointers are used in the skip list which we will describe below. The memory size of each entry is hence $O(\log^2 n + d \log L)$ and there are $O(r \log n)$ entries. Therefore, the hash table uses $O(rd \log^3(n + dL))$ qubits.

It is possible that more than $\lceil \log n \rceil$ points are hashed into the same bucket. However, as shown in [Amb07], this probability is small.

Skip list. The skip list we use closely follows that in [Amb07], except that the elements p_i in our skip list is ordered by its index i. We construct a skip list for each ϵ-box containing at least one point to store the points in it. For each $i \in S$, p_i belongs to exactly one skip list. Also, for $i \in S$, we randomly assign a level $\ell_i \in [0, \ldots, \ell_{\max}]$ where $\ell_{\max} = \lceil \log n \rceil$. The skip list associated with a ϵ-box has $\ell_{\max} + 1$ lists, where the level-ℓ list consists of all $i \in S$ such that $\ell_i \geq \ell$ and p_i is in this ϵ-box. Hence, the

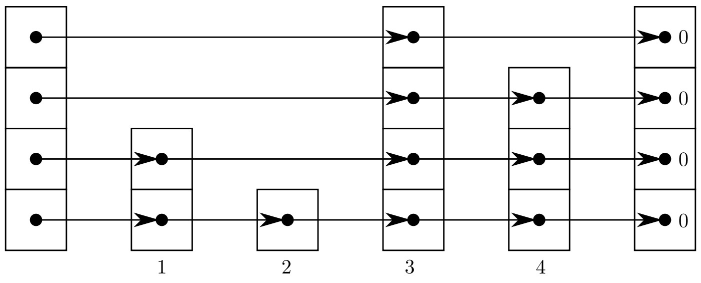


Figure 7.3: An example of a skip list that stores $\{1, 2, 3, 4\}$.

level-0 list consists of all $i \in S$ for p_i in this ϵ-box. Each element of the level-ℓ list has a specific pointer to the next element in this level, or to 0 if there is no next element. Each skip list contains a start entry that does not contain any (i, p_i) information but $\ell_{\max} + 1$ pointers to the first element of the each level. This start entry is stored in a leaf node of the augmented radix tree (which we will describe below) corresponding to this ϵ-box. In each skip list, we do not allocate memory for each node. Instead, each pointer is pointing to an entry of the hash table. The pointers are stored in the hash table (for the internal entries of each level) and in the radix tree (for the start entry). An example of a skip list is shown in Fig. 7.3.

Given $i \in S$, we can search for p_i as follows. We start from the start entry of the level-$\ell_{\max}$ list and traverse each element until we find the last element $j_{\ell_{\max}}$ such that $j_{\ell_{\max}} < i$. Repeat this for levels $\ell_{\ell_{\max}-1}, \ldots, \ell_0$ and at each level start from the element that ended the previous level. At level-0, we obtain the element j_0. Then, the next element of j_0 is where p_i should be located (if it is stored in this skip list) or be inserted.

Each $i \in S$ is randomly assigned a level ℓ_i at the beginning of computation that does not change during the computation. More specifically, $\ell_i = \ell$ with probability $1/2^{\ell+1}$ for $\ell < \ell_{\max}$ and with probability $1/2^{\ell_{\max}}$ for $\ell = \ell_{\max}$. This can be achieved using $\ell_{\max}$ hash functions $h_1, \ldots, h_{\ell_{\max}} : [n] \to \{0, 1\}$. In this way, each $i \in [n]$ has level $\ell < \ell_{\max}$ if $h_1(i) = \cdots = h_\ell(i) = 1$ but $h_{\ell+1}(i) = 0$; and it has level $\ell_{\max}$ if $h(i) = \cdots = h_{\ell_{\max}}(i) = 1$. In this quantum algorithm, we use an extra register to hold the state $|h_1, \ldots, h_{\ell_{\max}}\rangle$ which is initialized to a uniform superposition of all possible such functions from a d-wise independent family of hash functions (see [Amb07, Theorem 1]) for $d = \lceil 4 \log n + 1 \rceil$. During the execution of the quantum algorithm, a hash function from the hashing family is chosen depending on the state in this register.

At first glance, the skip list has the same role as the hash table – finding p_i given index i. However, they have very different purposes in our algorithm. Recall that each nonempty ϵ-box is associated with a skip list, which is used to quickly insert and delete a point in this ϵ-box. The number of points in this ϵ-box can be as small as one and as large as r (in the extreme case where all the points are in the same ϵ-box). Hence, we cannot afford to have a fixed length data structure (such as a hash table or a sorted array) to store these points. In addition, to support quick insertion and deletion, a skip list is a reasonable choice (against an ordinary list). The purpose of the hash table can be viewed as a uniquely represented memory storing all the r points that can be referred to by the skip lists.

Augmented radix tree. We augment the radix tree described in Section 7.4.1 to handle multiple solution. In this augmented radix tree, we do not need the registers $\mathcal{P}_1$ and $\mathcal{P}_2$. Instead, we use $\lceil \log n \rceil$ pointers $\mathcal{L}_1, \ldots, \mathcal{L}_{\lceil \log n \rceil}$ as the start entry of a skip list. These pointers uses $O(\log^2 n)$ bits. In addition, we use an *external counter* in the leaf nodes to record whether there is a point in other ϵ-boxes that is at most ϵ-away from a point in this ϵ-box, which uses $O(\log n)$ bits. More formally, let $\tau'(S)$ be the augmented radix tree associated with S. Each node of $\tau'(S)$ consists of the following registers

$$\mathcal{D} \times \mathcal{M}_1 \times \mathcal{M}_2 \times \mathcal{M}_3 \times \mathcal{E} \times \mathcal{C} \times \mathcal{F} \times \mathcal{E} \times \mathcal{L}_1 \times \cdots \times \mathcal{L}_{\lceil \log n \rceil}. \tag{7.60}$$

Next, we present how to perform the required operations on S with our data structure.

Checking for ϵ-close pairs. To check the existence of an ϵ-close pair, we just read the flag in the root of the radix tree. If the flag is set, there is at most one ϵ-close pair in S, and no such pairs otherwise. This operation takes $O(1)$ time.

Insertion. Given (i, p_i), we perform the insertion with the following steps:

1. Insert this tuple into the hash table.
2. Compute the id, $\mathrm{id}(p_i)$, of the ϵ-box which p_i belongs to. Denote this ϵ-box by $g(\mathrm{id}(p_i))$.
3. Using $\mathrm{id}(p_i)$ as the key, check if this key is already in $\tau'(S)$, if so, insert i into the skip list corresponding to $g(\mathrm{id}(p_i))$; otherwise, first create a uniform superposition of the addresses of all free cells into another register, then create a new tree node in the cell determined by this address register and insert it into the tree. The pointers for the start entry of the skip list is initially set to 0. Insert i into this skip list. Let $\tau'(S, g(\mathrm{id}(p_i))$ denote the leaf node in $\tau'(S)$ corresponding to $g(\mathrm{id}(p_i))$.
4. Increase the local counter $\mathcal{C}$ in $\tau'(S, g(\mathrm{id}(p_i)))$ by 1.
5. Use Procedure 25 to update the external counters $\mathcal{E}$ and flags $\mathcal{F}$ in $\tau'(S, g(\mathrm{id}(p_i)))$ as well as in the leaf nodes corresponding to the neighbor ϵ-boxes of $g(\mathrm{id}(p_i))$.

Note that the first step takes at most $O(\log n)$ time. The second step can be done in $O(d)$ time. In Lines 3 and 20, the number of ϵ-neighbors to check is at most $(2\sqrt{d}+1)^d$.

To obtain a uniform superposition of the addresses of all free cells, we first create a uniform superposition of all possible addresses to access to the bitmap $|\mathcal{B}\rangle$. We also use an auxiliary register that is initialized to $|0\rangle$. Then, the quantum random access gate defined in Eq. (7.58) is applied on the register holding the uniform superposition of all addresses, the auxiliary register, and the bitmap register, which is effectively a SWAP operation on the second register and the corresponding bit in $|\mathcal{B}\rangle$. The auxiliary register remains $|0\rangle$ if and only if the address in the first register is free. Since the size of memory space is chosen so that the probability of seeing a free cell is at least 1/3 (and we know exactly this probability based on how many cells have already been used), we can add an extra register and apply a one-qubit rotation to make the amplitude of the second register being $|0\rangle$ exactly 1/2. Hence, using *one* iteration of the oblivious amplitude amplification (which is a generalized version of Grover's search algorithm. See [BCC+17] and [MW05]) with the second register being the indicator, we obtain the uniform superposition of the addresses of all free cells. This cost if $O(\log n)$.

In [Amb07], it was shown that with high probability, insertion into the skip list can be done in $O(d + \log^4(n + L))$ time. Hence, with high probability, the insertion costs $O(d + \log^4(n + L) + d(2\sqrt{d} + 1)^d)$ time, where $O(d(2\sqrt{d} + 1)^d)$ is the time for checking neighbors. To further reduce the running time, we can just stop the skip list's insertion and deletion procedures after $O(d + \log^4(n + L))$ time, which only causes little error (see Lemma 7.26).

Deletion. Given (i, p_i), we perform the following steps to delete this tuple from our data structure.

1. Compute the id, $\mathrm{id}(p_i)$, of the ϵ-box which p_i belongs to, and denote this ϵ-box by $g(\mathrm{id}(p_i))$.

2. Using $\mathrm{id}(p_i)$ as the key, we find the leaf node in $\tau'(S)$ that is corresponding to $g(\mathrm{id}(p_i))$.

3. Remove i from the skip list, and decrease the local counter $\mathcal{C}$ in $\tau'(S, g(\mathrm{id}(p_i)))$ by 1.

4. Use Procedure 26 to update the external counters $\mathcal{E}$ and flags $\mathcal{F}$ in $\tau'(S, g(\mathrm{id}(p_i)))$ as well as in leaf nodes corresponding to the neighbor ϵ-boxes of $g(\mathrm{id}(p_i))$.

5. If the local counter $\mathcal{C} = 0$ in this leaf node, remove $\tau'(S, g(\mathrm{id}(p_i)))$ from $\tau'(S)$, and update the bitmap $\mathcal{B}$ in $\tau'(S)$ that keeps track of all free memory cells.

6. Remove (i, p_i) from the hash table.

Note that the first step can be done in $O(d)$ time. The second step can be done in $O(\log n)$ time. Procedure 26 has the same time complexity with Procedure 25. Hence, the cost for the deletion procedure is the same as that for insertion.

Finding a ϵ-close pair. We just read the flag in the root of the radix tree and then go to a leaf whose flag is 1. Check the local counter $\mathcal{C}$ of the node. if it is at least 2, output the first two elements in skip list. Otherwise, we find the ϵ-neighbor of the current node whose $C = 1$ and then output the points in that ϵ-neighbor and the current node.

Uniqueness. The uniqueness of our data structure follows from the analysis of [Amb07, BJLM13, Jef14]. More specifically, the hash table is always stored in the same way, as each $i \in S$ is stored in the same bucket for the fixed hash function and in each bucket, elements are stored in ascending order of i. The skip list is uniquely stored once the hash functions $h_1, \ldots, h_{\ell_{\max}}$ is determined. The shape of the radix tree is unique for S, but each node can be stored in different locations in memory. We use a uniform superposition of all possible memory organizations (by keeping track of the bitmap for free cells) to keep the quantum state uniquely determined by S.

Correctness. In the following, we argue that our data structure has the desired properties. First, we prove the correctness.

Lemma 7.25. *The flag bit in the root of $\tau'(S)$ is set if and only if there exist distinct $i, j \in S$ such that $|p_i - p_j| \leq \epsilon$.*

Proof. We show that after each insertion and deletion, the data structure maintains the following conditions, and then lemma follows.

1. The flag bit of each leaf node of $\tau'(S)$ is set if and only if either its local counter is at least 2, or its external counter is at least 1.

2. The external counter of a leaf node $\tau'(S, g')$ is nonzero if and only if g' contains only one point p, and there exists another p' in another ϵ-box g'' such that $|p - p'| \leq \epsilon$.

It is easy to check that the first condition is maintained for each insertion and deletion. We show the second condition holds during insertions and deletion. For insertions, we consider the first case where a point p is just inserted into the ϵ-grid g', and p is its only point. The first for-loop in Procedure 25 updates other ϵ-boxes that have only one point to maintain the second condition. We consider the second case where g' already contains p' and p is just inserted, then the external counter in g' should be 0, and the second for-loop in Procedure 25 updates other ϵ-boxes that have only one point using the information of p'. This maintains the second condition. For deletions, there are also two cases. First, consider p is the only one point in g' and it is just deleted. We use the first for-loop in Procedure 26 to update the ϵ-boxes that has only one point using the information of p to maintain the second condition. Second, there is another point p' left in g' after deleting p. In this case, we start to check the external counter in g'. We use the second for-loop in Procedure 26 to check other ϵ-boxes that have only one point using the information of p' and update the corresponding external counter to maintain the second condition. □

Imperfection of the data structures and error analysis. Our data structure is not perfect. As indicated by Ambainis [Amb07], there are two possibilities that it will fail. First, the hash table might overflow. Second, it might take more that $\lceil \log n \rceil$ time to search in a skip list. To resolve the first problem, we fix the number of entries in each bucket to be $\lceil \log n \rceil$ and treat any overflow as a failure. For the second problem, we stop the subroutine for accessing the skip list after $O(\log n)$ steps, and it causes an error in some cases. The original error analysis can be directly used in our case, as our hash table doesn't change the structure or the hash function, and our skip lists can be viewed as breaking the skip list in [Amb07] into several pieces (one for each nonempty ϵ-box), and each insertion/deletion only involves one of them. Hence, the discussion in [Amb07, Section 6] can be directly adapted to our case:

Lemma 7.26 (Adapted from [Amb07])**.** *Let $|\psi\rangle$ be the final state of our algorithm (with imperfect data structures) and $|\psi'\rangle$ be the final state with the perfect data structure. Then $\| \, |\psi\rangle - |\psi'\rangle \, \| \leq O(1/\sqrt{n})$.*

Sketch of proof. There are two places where the data structure may give error: first, the hash table may have overflow, and second, the algorithm cannot find the required element in the skip lists in the desired time. The original proof showed that the probability that any of these errors happens is negligible, and thus the two-norm distance between $|\psi\rangle$ and $|\psi'\rangle$ can be bounded. Here, our data structure combining hash table, skip list, and radix tree, only has errors from hash tables and skip lists. The radix tree which has no error can be viewed as applying additional unitaries on $|\psi\rangle$ and $|\psi'\rangle$, and this does not change the distance between the two states. Since the probability that the errors from hash tables and skip lists happen are negligible by following the same analysis in [Amb07], we can conclude that the two-norm distance between $|\psi\rangle$ and $|\psi'\rangle$ is small. □

Time complexity for CP_ϵ. We use the quantum walk framework reviewed in Section 7.2.5 to solve CP_ϵ. We first build the Johnson graph for CP_ϵ, which is similar to that for ED in Section 7.2.5. The vertices of the Johnson graph are $S \subseteq [n]$ with $|S| = n^{2/3}$ and S is marked if there exist distinct $i, j \in S$ such that $\Delta(p_i, p_j) \leq \epsilon$. We use $|S, d(S)\rangle$ to represent the quantum state corresponding to S, where $d(S)$ is the data structure of S defined in Section 7.4.1. Consider the three operations:

- **Steup:** with cost S, preparing the initial state
$$|\pi\rangle = \frac{1}{\sqrt{\binom{n}{n^{2/3}}}} \sum_{S \subseteq [n]:|S|=n^{2/3}} \sqrt{\pi_S} \, |S, d(S)\rangle \,. \tag{7.61}$$
- **Update:** with cost U, applying the transformation
$$|S, d(S)\rangle \, |0\rangle \mapsto |S, d(S)\rangle \sum_{S' \subseteq [n]:|S \cap S'|=n^{2/3}-1} \sqrt{p_{SS'}} \, |S', d(S')\rangle \,, \tag{7.62}$$
where $p_{SS'} = \frac{1}{n^{2/3}(n-n^{2/3})}$.
- **Checking:** with cost C, applying the transformation:
$$|S, d(S)\rangle \mapsto \begin{cases} -\,|S, d(S)\rangle & \text{if } S \in M \\ |S, d(S)\rangle & \text{otherwise,} \end{cases} \tag{7.63}$$
where M is the set of marked states.

We have the following result.

Theorem 7.27. *There exists a quantum algorithm that with high probability solves CP_ϵ with time complexity $O(n^{2/3}(d + \log^4(n+L) + d(2\sqrt{d}+1)^d))$.*

Proof. The Johnson graph has $\lambda \geq 1/n^{2/3}$ and the Markov chain has spectral gap $\delta \geq 1/n^{2/3}$. For the setup operation, $\mathsf{S} = O(n^{2/3}(d + \log^4(n+L) + d(2\sqrt{d}+1)^d))$, since preparing the uniform superposition for all $|S\rangle$ costs $O(\log n)$ Hadamard gates and we need to do $n^{2/3}$ insertions to prepare the data structure. Each insertion costs $O(d + \log^4(n+L) + d(2\sqrt{d}+1)^d)$. For the update operation, we can implement the quantum walk operator as described in [Jef14]: we use $|S, d(S)\rangle \, |i, j\rangle$ to represent

$|S, d(S)\rangle \, |S', d(S')\rangle$ where S' is obtained from S by adding index i and removing index j. Then the diffusion can be implemented by preparing a uniform superposition of all $i \in S$ and a uniform superposition of all $j \notin S$, which takes time $O(\log n)$, and the "SWAP" operation can be implemented by a unitary that maps $|S, d(S)\rangle \, |i, j\rangle$ to $|S', d(S')\rangle \, |j, i\rangle$. In this way, the update operation uses $O(1)$ insertion and deletion to construct $d(S')$ from $d(S)$, and hence $\mathsf{U} = O(d + \log^4(n + L) + d(2\sqrt{d} + 1)^d)$. The checking operation can be done in $O(1)$ time with the data structure. Therefore, by Lemma 7.10, the time complexity is $O(\mathsf{S} + \frac{1}{\sqrt{\lambda}}(\frac{1}{\sqrt{\delta}}\mathsf{U} + \mathsf{C})) = O(n^{2/3}(d + \log^4(n + L) + d(2\sqrt{d} + 1)^d))$. □

By Lemma 7.23, we have the following corollary.

Corollary 7.28. *There exists a quantum algorithm that with high probability solves* CP *with time complexity* $O(n^{2/3} \cdot (d + \log^4(n + L) + d(2\sqrt{d} + 1)^d) \cdot (m + \log d))$.

Remark 7.4. For $d = O(1)$ dimension and $m = O(\log n)$ digits of each coordinate of the points, the running time of the single-shot quantum algorithm is $O(n^{2/3} \cdot \log^5 n)$.

7.4.3 Multiple-trial quantum walks with simple data structure

In the previous subsection, we provide a quantum algorithm which solves CP_ϵ in time $O(n^{2/3}(d + \log^4 n + d(2\sqrt{d} + 1)^d))$, where the logarithmic cost is mainly from the cost of the skip list. In this section we present a quantum algorithm which only requires the radix tree, and thus improve the running time. The caveat is that, with only the radix tree data structure, the insertion would fail if there are more than one ϵ-close pairs. As a result, we need to keep shrinking the size of the problem using [Amb07, Algorithm 3] until there is a unique solution with high probability, and then run the $\widetilde{O}(n^{2/3})$ quantum algorithm for this unique-solution case.

In the following, we first show how to solve the unique-solution CP_ϵ, and then show the reduction from the multiple-solution case to the unique-solution case.

Lemma 7.29. *There exists a quantum algorithm that with high probability solves the unique-solution* CP_ϵ *with time complexity* $O(n^{2/3}(\log n + d(2\sqrt{d}+1)^d))$.

Data structure for unique-solution. We use the radix tree $\tau(S)$ for S defined in Section 7.4.1. In the following, we describe the necessary operations on $\tau(S)$.

Checking for ϵ-close pair. To check the existence of an ϵ-close pair, we just read the flag bit in the root of $\tau(S)$, which takes $O(1)$ time.

Insertion. Given (i, p_i), we perform the following steps for insertion. First compute the id, $\mathrm{id}(p_i)$, of the ϵ-box which p_i belongs to. Denote this ϵ-box by $g(\mathrm{id}(p_i))$. Using $\mathrm{id}(p_i)$ as the key, check if this key is already in $\tau(S)$. There are two cases:

- $\mathrm{id}(p_i)$ is already in $\tau(S)$: insert p_i into $\tau(S, g(\mathrm{id}(p_i)))$, increase the local counter in $\tau(S, g(\mathrm{id}(p_i)))$ by 1 and also set the flag. Then update the flag and local counter of the nodes along the path from $\tau(S, g(\mathrm{id}(p_i)))$ to the root.

- $\mathrm{id}(p_i)$ is not in $\tau(S)$: create a new leaf node for $\mathrm{id}(p_i)$ and insert it into $\tau(S)$. Insert p_i into this new leaf node, and increase the local counter in $\tau(S, g(\mathrm{id}(p_i)))$ by 1. Then, check the ϵ-neighbors g' of $\tau(S, g(\mathrm{id}(p_i)))$ that contains only one point p' and set both flags if p_i is ϵ-close to p', and in this case, update the flag bit and local counter on the nodes along the paths from $\tau(S, g(\mathrm{id}(p_i)))$ and g'.

Deletion. Given (i, p_i), we first compute the id, $\mathrm{id}(p_i)$ of the ϵ-box that p_i belongs to, and locate the corresponding leaf node $\tau(S, g(\mathrm{id}(p_i)))$. Decrease the local counter in $\tau(S, g(\mathrm{id}(p_i)))$ by 1 and update the local counter in the nodes along the path from $\tau(S, g(\mathrm{id}(p_i)))$ to the root. Check the number of points stored in $\tau(S, g(\mathrm{id}(p_i)))$. There are two possibilities:

- There are two points in $\tau(S, g(\mathrm{id}(p_i)))$: unset the flag in $\tau(S, g(\mathrm{id}(p_i)))$ and update the flag bit in the nodes along the path to the root and delete p_i from $\tau(S, g(\mathrm{id}(p_i)))$.

- p_i is the only point in $\tau(S, g(\mathrm{id}(p_i)))$: check the ϵ-neighbors g' of $\tau(S, g(\mathrm{id}(p_i)))$ that contains only one point p' and unset both flags if p_i is ϵ-close to p', and in this case, update the flag bit on the nodes along the path from $\tau(S, g(\mathrm{id}(p_i)))$ and g' to the root. Delete p_i from $\tau(S, g(\mathrm{id}(p_i)))$ and delete $\tau(S, g(\mathrm{id}(p_i)))$ from $\tau(S)$.

As in Section 7.4.2, we use a bitmap register $|\mathcal{B}\rangle$ to keep track of the free cells in $\tau(S)$. For insertion, we maintain a uniform superposition of all possible free cells to insert a new radix tree node. For deletion, we update the bitmap $|\mathcal{B}\rangle$ accordingly. This ensures the uniqueness of the quantum data structure.

The correctness of the data structure is straightforward, and the time complexity is $O(\log n + d(2\sqrt{d} + 1)^d)$ for both insertion and deletion. Also, preparing a uniform superposition for all $i \in S$ costs $O(\log n)$ using the local counter in each node. By a similar analysis of Theorem 7.27, we prove Lemma 7.29 as follows.

Proof of Lemma 7.29. The algorithm uses the framework in Lemma 7.10 with the data structure we just described in this subsection, where $\mathsf{U} = O(\log n + d(2\sqrt{d}+1)^d))$, $\mathsf{C} = O(1)$ and $\mathsf{S} = O(n^{2/3}(\log n + d(2\sqrt{d} + 1)^d))$. Therefore, the running time of the algorithm is as claimed. $\square$

Next, we show how to reduce multiple-solution CP_ϵ to unique-solution CP_ϵ. A high-level overview of Ambainis's reduction in [Amb07] is the following. We run the algorithm for unique-solution CP_ϵ several times on some random subsets of the given input. If the given subset contains solutions, then with constant probability there exists a subset which contains exactly one solution.

Definition 7.19 ([Amb07, INT05])**.** Let $\mathcal{F}$ be a family of permutations on $f : [n] \to [n]$. $\mathcal{F}$ is ϵ-approximate d-wise independent if for any $x_1, \dots, x_d \in [n]$ and for all $y_1, \dots, y_d \in [n]$,

$$\frac{1-\epsilon}{n \cdot (n-1) \cdot (n-d+1)} \leq \Pr\left[\bigwedge_{i=1}^{n} f_i(x_i) = y_i\right] \leq \frac{1+\epsilon}{n \cdot (n-1) \cdot (n-d+1)}. \tag{7.64}$$

Lemma 7.30 ([Amb07, INT05])**.** *Let n be an even power of a prime number. For any $t \leq n$, $\epsilon > 0$, there exists an ϵ-approximate t-wise independent family $\mathcal{F} = \{\pi_j | j \in [R]\}$ of permutations $\pi_j : [n] \to [n]$ such that:*

- $R = O\left(\left(n^{t^2} \cdot \epsilon^{-t}\right)^{3+o(1)}\right)$*;*
- *given i, j, $\pi_j(i)$ can be computed in time $O(t \log^2 n)$.*

The multiple-solution algorithm from [Amb07] is as follows:

We have the main result of this subsection:

Theorem 7.31. *There exists a quantum algorithm that with high probability solves CP_ϵ with time complexity $O(n^{2/3} \cdot (\log n + d(2\sqrt{d}+1)^d) \cdot \log^3 n) = O(n^{2/3} \cdot \log^4 n)$ for $d = O(1)$.*

Proof. We prove the running time of the algorithm here. For the correctness, one can check [Amb07] for the detail.

By Eq. (7.65), the size of T_{j+1} will be at most

$$\frac{4}{5} \cdot (1 + \frac{1}{8})|T_j| = \frac{9}{10}|T_j|. \tag{7.66}$$

Therefore, the while-loop takes at most $O(\log n)$ iterations in the worst case. Let $n_j = |T_j|$ be the size of the instance in the j-th iteration. Then, the unique-solution algorithm in Line 5 runs in $O(n_j^{2/3} \cdot (\log n_j + d(2\sqrt{d}+1)^d))$-time (Lemma 7.29), given an $O(1)$-time access to the set T_j. However, in Line 7 each element of the random permutation can be computed in time $O(\log^3 n)$ according to Lemma 7.30

with $t = 4\log n$, which means the unique-solution algorithm will take $O(\log^3 n)$ time for each query to T_j. Note that we will not actually compute the whole set T_{j+1}, as shown in Line 9, which takes too much time. Hence, the running time for the j-th iteration is $O(n_j^{2/3} \cdot (\log n_j + d(2\sqrt{d}+1)^d) \cdot \log^3 n)$. And the total running time for the while-loop is

$$\sum_{j=1}^{O(\log n)} O(n_j^{2/3} \cdot (\log n_j + d(2\sqrt{d}+1)^d) \cdot \log^3 n) \tag{7.67}$$

$$\leq O(n^{2/3} \cdot (\log n + d(2\sqrt{d}+1)^d) \cdot \log^3 n) \cdot \sum_{j=0}^{O(\log n)} \left(\frac{9}{10}\right)^{2j/3} \tag{7.68}$$

$$= O(n^{2/3} \cdot (\log n + d(2\sqrt{d}+1)^d) \cdot \log^3 n), \tag{7.69}$$

where the first inequality follows from $n_j \leq (\frac{9}{10})^{j-1} \cdot n$. Finally, Line 12 runs in time $O(n^{2/3} \log n)$. This completes the proof of the running time. $\square$

To conclude the quantum algorithms for solving CP in constant dimension, we have the following corollary that is a direct consequence of either Theorem 7.27 or Theorem 7.31.

Corollary 7.32. *For any $d = O(1)$, there exists a quantum algorithm that, with high probability, solves $\mathsf{CP}_{n,d}$ in time $\widetilde{O}(n^{2/3})$.*

7.4.4 Quantum lower bound for CP in constant dimensions

We can easily get an $\Omega(n^{2/3})$ lower bound for the quantum time complexity of CP in constant dimension by reducing the element distinctness problem (ED) to CP.

Theorem 7.33 (Folklore)**.** *The quantum time complexity of* CP *is $\Omega(n^{2/3})$.*

Proof. We reduce ED to one dimensional CP by mapping the point i with value $f(i)$ in ED the point $f(i) \in \mathbb{R}$ in CP. If the closest pair has distance zero, we know there is a collision $f(i) = f(j)$. If the closest pair has distance greater or equal to one, we know there is no collision. Therefore, ED's $\Omega(n^{2/3})$ query lower bound by [AS04] translates into $\Omega(n^{2/3})$ time lower bound for CP. $\square$

7.5 Bichromatic Closest Pair in Constant Dimensions

Classically, the bichromatic closest pair problem is harder than the closest pair problem. In constant dimension, the best algorithms for the closest pair problem are "nearly linear", while the algorithm by [AESW91] for bichromatic closest pair problem is "barely subquadratic", running in $O(n^{2-1/\Theta(d)})$-time. In quantum, we found that BCP is still harder than CP in constant dimension. In particular, we cannot adapt the quantum algorithm in previous section for solving BCP because the data structure cannot distinguish the points from two sets efficiently. We can only get a sub-linear time quantum algorithm for BCP using different approach, which is a quadratic speed-up for the classical algorithm.

Nevertheless, we show that we can find an approximate solution for BCP with multiplicative error $1+\xi$ with quantum time complexity $\widetilde{\Theta}(n^{2/3})$. The following theorem is a direct consequence of Theorems 7.39 and 7.42.

Theorem 7.34. *For any fixed dimension and error ξ, there is a quantum algorithm which can find an approximate solution for BCP with multiplicative error $1+\xi$ in time $\widetilde{O}(n^{2/3})$. Moreover, all quantum algorithms which can find an approximate solution for BCP with arbitrary multiplicative error requires time $\Omega(n^{2/3})$.*

Similar to solving CP, we reduce BCP to its decision version of the problem, and then apply quantum algorithms to solve the decision problem. We define the decision problem as BCP_ϵ.

Definition 7.20 (BCP_ϵ)**.** In BCP_ϵ, we are given two sets A, B of n points $\in \mathbb{R}^d$ and a distance measure Δ. The goal is to find a pair of points $a \in A$, $b \in B$ such that $\Delta(a,b) \leq \epsilon$ if it exists and returns no if no such pair exists.

To address the approximate version of BCP, we also define the approximation version of BCP_ϵ as follows:

Definition 7.21 ($(1+\xi)$-BCP_ϵ)**.** In $(1+\xi)$-BCP_ϵ, we are given two sets A, B of n points $\in \mathbb{R}^d$, a distance measure Δ, and ξ. The goal is to do the following

1. If there exists a pair of points $a \in A$, $b \in B$ such that $\Delta(a,b) \leq \epsilon$, output the pair (a,b).

2. If for all pairs of points $a \in A$, $b \in B$, $\Delta(a,b) > (1+\xi)\epsilon$, returns no.

Again, we consider $\Delta(a,b) = \|a-b\|$ as the distance measure in this work. We show that BCP reduce to BCP_ϵ in time $O(m + \log d)$, where m is the number of digits of each coordinate and d is the dimension.

Lemma 7.35. *Given an oracle $\mathcal{O}$ for $(1+\xi)$-BCP_ϵ, there exists an algorithm $A^{\mathcal{O}}$ that runs in time and query complexity $O(m + \log d)$ solves the $(1+\xi)$-BCP.*

Proof. Let (A, B, δ) be an instance of the $(1+\xi)$-BCP. We first pick an arbitrary pair $a_0 \in A, b_0 \in B$ and computes $\Delta(a_0, b_0)$. Then, we set ϵ to be $\Delta(a_0,b_0)/2$ and run the oracle $\mathcal{O}$ to check whether there exists a distinct pair which distance is less than $\Delta(a_0,b_0)/2$ or not. If there exists such a pair, which we denote as (a_1, b_1), then we set $\epsilon = \Delta(a_1, b_1)$ and call $\mathcal{O}$ to check again. If there is no such a pair, then we set $\epsilon = 3\Delta(a_0,b_0)/4$ and call $\mathcal{O}$. We continuously run this binary search for $m + \log d$ iterations. Finally, the algorithm outputs the bichromatic closest pair. $\square$

In the subsections, we present a quantum algorithm for solving $(1+\xi)$-BCP and a quantum algorithm for exact BCP. To complement the algorithmic results, we also give quantum lower bound for BCP.

7.5.1 Quantum algorithm for $(1+\xi)$-BCP

The quantum algorithm is based on the quantum walk framework on a tensor product of Johnson graphs. To begin with, we define the Johnson graphs J_A and J_B for A and B, respectively. The vertices of J_A, denoted by X_A, is defined as the set $\{S \subseteq A : |S| = n^{2/3}\}$. There is an edge connecting S and S' if and only if $|S \cap S'| = n^{2/3} - 1$. The Markov chain M_A is defined on X_A with $p_{SS'} = \frac{1}{n^{2/3}(n - n^{2/3})}$ when S and S' are connected by an edge. The Johnson graph for J_B for B and its corresponding Markov chain can be defined similarly. The *tensor product* $M_A \otimes M_B$ is defined as the Markov chain based on $X_A \times X_B$ defined as

$$X_A \times X_B := \{(S_A, S_B) : S_A \in X_A, S_B \in X_B\}, \tag{7.70}$$

with transition probability

$$p_{(S_A,S_B)(S'_A,S'_B)} = p_{S_A S'_A} \cdot p_{S_B S'_B}. \tag{7.71}$$

A state (S_A, S_B) is marked if there exists a pair $a \in S_A$ and $b \in S_B$ such that $\Delta(a,b) \le \epsilon$.

Now, we examine the properties of $M_A \otimes M_B$. It is easy to see that $\lambda = \frac{\binom{n-1}{n^{2/3}-1}^2}{\binom{n}{n^{2/3}}^2} = \frac{1}{n^{2/3}}$. Let δ_A and δ_B be the spectral gap of M_A and M_B respectively. As a result of [AB09, Lemma 21.17], $\delta \ge \min\{\delta_A, \delta_B\} = \frac{1}{n^{2/3}}$. By Lemma 7.10, the cost for solving $(1+\xi)$-BCP_ϵ is $O(\mathsf{S} + n^{1/3}(n^{1/3}\mathsf{U} + \mathsf{C}))$, where S, U and C are the cost of quantum operations defined in Section 7.4.2. Before describing the data structure to achieve meaningful S, U, and C, we first introduce a finer discretization scheme. In Section 7.4, we used a hypergrid consisting of ϵ-boxes. Here, we discretize the space $[0, L]^d$ as a hypergrid consisting of $\frac{\xi\epsilon}{2\sqrt{d}}$-boxes. The following lemma guarantees that distance between a $\frac{\xi\epsilon}{2\sqrt{d}}$-box and its ϵ-neighbor is at most $(1+\xi)\epsilon$.

Lemma 7.36. *Let g and g' be $\frac{\xi\epsilon}{2\sqrt{d}}$-boxes. If g and g' are ϵ-neighbors, then for all $p \in g$ and $p' \in g'$, $\Delta(p,p') \le (1+\xi)\epsilon$.*

Proof. Recall the definition of the id function in Eq. (7.56). $\mathrm{id}(g)$ can be treated as a point, and we can measure the distance between $\mathrm{id}(g)$ and other points. The lemma can be proven via the triangle inequality:

$$\Delta(p,p') \le \Delta(p, \mathrm{id}(g)) + \Delta(\mathrm{id}(g), \mathrm{id}(g')) + \Delta(p', \mathrm{id}(g') \le \frac{\xi\epsilon}{2} + \epsilon + \frac{\xi\epsilon}{2} \le (1+\xi)\epsilon. \tag{7.72}$$

□

In our algorithm, we need to search for all ϵ-neighbors that contain the other color to report an ϵ-close pair (with an multiplicative error ξ). It's easy to see that the number of neighbors of a box is bounded in terms of d and ξ:

Claim 7.37. *For each $\frac{\xi\epsilon}{2\sqrt{d}}$-box, the number of ϵ-neighbors is at most $\left(4\sqrt{d}/\xi + 1\right)^d$.*

Based on this finer discretization scheme, we use the data structure defined in Section 7.4.2 but with simple modifications on the radix tree. Instead of using $\mathcal{L}_1, \ldots, \mathcal{L}_{\lceil \log n \rceil}$ as the start entry of the skip list, we use $\lceil \log n \rceil$ pointers for both sets A and B. We also need local counters $\mathcal{C}^A$ and $\mathcal{C}^B$ for both colors. Now, each node in the radix tree has the following registers:

$$\begin{aligned} &\mathcal{D} \times \mathcal{M}_1 \times \mathcal{M}_2 \times \mathcal{M}_3 \times \mathcal{E}^A \times \mathcal{E}^B \times \mathcal{C}^A \times \mathcal{C}^B \times \\ &\mathcal{F} \times \mathcal{L}_1^A \times \cdots \times \mathcal{L}_{\lceil \log n \rceil}^A \times \mathcal{L}_1^B \times \cdots \times \mathcal{L}_{\lceil \log n \rceil}^B. \end{aligned} \tag{7.73}$$

The points in A (or B, respectively) is organized by the skip list for A (or B, respectively). The insertion and deletion operations are similar to the data structure in Section 7.4.2, but in the procedure for updating the local and external counters and checking ϵ-neighbors, we need to consider points of the other color. We formally describe the two procedures as follows.

Insertion. Given a point (i, p_i, x), where $x \in \{A, B\}$ denotes the color. We perform the insertion with the following steps:

1. Insert this tuple into the hash table corresponding to x.

2. Compute the id, $\mathrm{id}(p_i)$, of the $\frac{\xi\epsilon}{\sqrt{d}}$-box which p_i belongs to and denote it by $g(\mathrm{id}(p_i))$.

3. Using $\mathrm{id}(p_i)$ as the key, check if this key is already in $\tau'(S)$, if so, insert i into the skip list for color x corresponding to $g(\mathrm{id}(p_i))$; otherwise, first create a uniform superposition of the addresses of all free cells into another register, then create

a new tree node in the cell determined by this address register and insert it into the tree. The pointer for the start entry of the skip list is initially set to 0. Insert i into this skip list. Let $\tau'(S, g(\mathrm{id}(p_i))$ denote the leaf node in $\tau'(S)$ corresponding to $g(\mathrm{id}(p_i))$.

4. Increase the local counter $\mathcal{C}^x$ in $\tau'(S, g(\mathrm{id}(p_i)))$ by 1.

5. Use Procedure 28 to update the external counters $\mathcal{E}^x, \mathcal{E}^{\bar{x}}$ (here $\bar{x}$ denotes the other color than x) and flags $\mathcal{F}$ in $\tau'(S, g(\mathrm{id}(p_i)))$, the leaf nodes which are corresponding to the ϵ-neighbors of $g(\mathrm{id}(p_i))$, and their parent nodes.

Note that the first step takes at most $O(\log n)$ time. The second step can be done in $O(d)$ time. In Lines 4 and 20, the number of ϵ-neighbors to check is at most $(\frac{4\sqrt{d}}{\xi} + 1)^d$ by Claim 7.37.

Deletion. Given (i, p_i, x), we perform the following steps to delete this tuple from our data structure.

1. Compute the id, $\mathrm{id}(p_i)$, of the $\frac{\epsilon\xi}{\sqrt{d}}$-box which p_i belongs to and denote it by $g(\mathrm{id}(p_i))$.

2. Using $\mathrm{id}(p_i)$ as the key, we find the leaf node in $\tau'(S)$ that is corresponding to $g(\mathrm{id}(p_i))$.

3. Remove i from the skip list for color x, and decrease the local counter $\mathcal{C}^x$ in $\tau'(S, g(\mathrm{id}(p_i)))$ by 1.

4. Use Procedure 26 to update the external counters $\mathcal{E}^x$ and $\mathcal{E}^{\bar{x}}$ (here $\bar{x}$ denote the other color than x) and flags $\mathcal{F}$ in $\tau'(S, g(\mathrm{id}(p_i)))$ as well as in leaf nodes corresponding to the ϵ-neighbors of $g(\mathrm{id}(p_i))$.

5. If both local counters $\mathcal{C}^x, \mathcal{C}^{\bar{x}}$ in this leaf node are 0, remove $\tau'(S, g(\mathrm{id}(p_i)))$ from $\tau'(S)$, and update the bitmap $\mathcal{B}$ in $\tau'(S)$ that keeps track of all free memory cells.

6. Remove (i, p_i, x) from the hash table.

Note that the first step can be done in $O(d)$ time. The second step can be done in $O(\log n)$ time. Procedure 29 has the same time complexity with Procedure 28. Hence, the cost for the deletion procedure is the same with that for insertion.

Checking for $(1+\xi)\epsilon$-close pairs. To check the existence of an $(1+\xi)\epsilon$-close pair, we just read the flag in the root of the radix tree. If the flag is set, there is at most one ϵ-close pair in S, and no such pairs otherwise. This operation takes $O(1)$ time.

Finding a $(1+\xi)\epsilon$-close pair. We just read the flag in the root of the radix tree and then go to a leaf which flag is 1. Check the local counters of the node. If both local counters are at least 1, output the first elements in skip lists for A and the first element in the skip list for B. Otherwise, check the external counters. Suppose $\mathcal{E}^A$ is non-zero. Then we find the ϵ-neighbor of the current node whose $\mathcal{C}^B > 0$ and output the first point in the skip list of A of the current node and the first element in the skip list of B of the ϵ-neighbor.

We have the following result.

Theorem 7.38. *For any fixed dimension and fixed ξ, there exists a quantum algorithm that, with high probability, can solve $(1+\xi)$-BCP_ϵ in time $O(n^{2/3}(d + \log^4(n + L) + d(\frac{4\sqrt{d}}{\xi} + 1)^d))$.*

Proof. The proof closely follows the analysis for Theorem 7.27, and the correctness of the data structure and the time complexity of its operations follow from the discussion in Section 7.4.2. Note that our algorithm will output a pair which belong to the same $\frac{\xi\epsilon}{2\sqrt{d}}$-box or two of them that are ϵ-neighbors. Based on Lemma 7.36, two points which corresponding hyercubes are ϵ-neighbors have distance at most $(1+\xi)\epsilon$. Therefore, our algorithm could output a pair of points which distance is at most $(1+\xi)\epsilon$. Another difference is that here we need to search at most $(4\sqrt{d}/\xi + 1)^d$ neighbors during insertions and deletions. As a result, $\mathsf{U} = O(d + \log^4(n+L) + d(4\sqrt{d}/\xi + 1)^d)$, and $\mathsf{S} = O(n^{2/3}(d + \log^4(n+L) + d(4\sqrt{d}/\xi + 1)^d)$. Again, $\mathsf{C} = O(1)$, $\delta \geq 1/n^{2/3}$, and $\lambda \geq 1/n^{2/3}$. Therefore, by Lemma 7.10, the total cost is $O(\mathsf{S} + \frac{1}{\sqrt{\lambda}}(\frac{1}{\sqrt{\delta}}\mathsf{U} + \mathsf{C})) = O(n^{2/3}(d + \log^4(n+L) + (4\sqrt{d}/\xi + 1)^d))$. $\square$

By Lemma 7.35 and the above Theorem 7.38, we have the following theorem:

Theorem 7.39. *For an fixed dimension and fixed ξ, there exists a quantum algorithm that, with high probability, can solve $(1+\xi)$-BCP in time $\widetilde{O}(n^{2/3})$.*

7.5.2 Quantum algorithm for solving BCP exactly

In this subsection, we present a quantum algorithm for solving BCP exactly. The main idea of this algorithm is to partition A into smaller subsets. Then we build data structures which support nearest-neighbor search on all subsets in superposition. We use the quantum minimum finding algorithm to find the smallest distances from B to each subset, among which we use the quantum minimum finding algorithm again to find the smallest distance.

Unlike the data structure for solving CP, the data structure for BCP does not have to be uniquely represented, as no insertion and deletion are performed in this algorithm. The data structure can have expected running time instead of the worst-case running time. The total worst-case running time can be bounded by standard techniques. The nearest-neighbor search data structure we use is from [Cla88], and is reformulated in the following lemma.

Lemma 7.40 ([Cla88])**.** *For any fixed dimension, there exists a data structure for n points in $\mathbb{R}^d$ that can be built in expected time complexity $O(n^{\lceil d/2 \rceil + \delta})$ for arbitrarily small δ and the nearest-neighbor search can be performed in worst-case time complexity $O(\log n)$.*

This data structure is based on the Voronoi diagram and its triangulation in higher dimensions. Using this data structure, we have a quantum algorithm for solving BCP exactly, which yields the following theorem.

Theorem 7.41. *There exists a quantum algorithm that, with high probability, solves* BCP *for dimension d with time complexity $\widetilde{O}\left(n^{1-\frac{1}{2d}+\delta}\right)$ for arbitrarily small δ.*

Proof. We first partition A into $\lceil n/r \rceil$ subsets $S_1, \ldots, S_{\lceil n/r \rceil}$, where $|S_i| = r$ for $i \in \big[\lceil n/r \rceil\big]$. (The value of r will be determined later). For all $i \in \big[\lceil n/r \rceil\big]$, we can find a closest pair between S_i and B as follows. First, a data structure as in Lemma 7.40 for S_i is built in expected time $O\big(r^{\lceil d/2 \rceil + \delta}\big)$, which supports nearest-neighbor search in time $O(\log n)$. Then, we use the quantum minimum finding subroutine (Theorem 7.6) which uses the distance reported by the nearest-neighbor search as the oracle. The closest pair between S_i and B can be found in time complexity $\widetilde{O}(\sqrt{n})$. Note that the time complexity for building the data structure is not bounded for the worst case. However, using Markov's inequality, we know that with high probability, say, at least $9/10$, the time complexity is bounded by $O\big(r^{\lceil d/2 \rceil + \delta}\big)$. Hence, fixing a constant $c \geq 10$, and stop the data structure construction after $c \cdot r^{\lceil d/2 \rceil + \delta}$ steps. With at most $1/10$ probability, the construction will fail and this event can be detected by checking the solution returned by the quantum minimum finding subroutine. We run $O(\log n)$ instances of above procedure in parallel and use take the quantum minimum of all the $O(\log n)$ results. The probability that all these instances fail is at most $(1/10)^{O(\log n)} = O(1/n)$. We refer to the above procedure as the "inner search", and its time complexity is $O\big(r^{\lceil d/2 \rceil + \delta} + \sqrt{n}\big)$.

Next, we use the distance of the output of the inner search as the oracle and perform another quantum minimum finding subroutine for $i \in \big[\lceil n/r \rceil\big]$. We refer to this procedure as the "outer search". The probability that the closest pair between A and B lies in S_i and B is r/n. As a result, the number of the oracle queries for the quantum minimum finding subroutine is $\widetilde{O}(\sqrt{n/r})$. The time complexity for each query is $O\big(r^{\lceil d/2 \rceil + \delta} + \sqrt{n}\big)$. Therefore, the total time complexity is $\widetilde{O}\big((r^{\lceil d/2 \rceil + \delta} +$

$\sqrt{n}) \cdot \sqrt{n/r}\big)$. A simple calculation shows that this achieves the minimum (ignoring the δ term in the exponent) when $r = n^{1/d}/(d-1)^{2/d}$, which yields the total time complexity

$$\widetilde{O}\left(n^{1-\frac{1}{2d}+\delta}\right). \tag{7.74}$$

The failure probability for each query is at most $O(1/n)$. Therefore, the total failure probability is at most $O(\sqrt{n/r}/n) = O(n^{-(1/2-1/2d)})$ for $d > 1$, which can be smaller than any constant. □

7.5.3 Quantum lower bound for BCP in constant dimensions

Now, we give a lower bound for $(1+\xi)$-BCP, which trivially holds for BCP.

Theorem 7.42. *The quantum query complexity for solving* BCP *is* $\Omega(n^{2/3})$*. Furthermore, the quantum query complexity for solving* $(1+\xi)$-BCP *with an arbitrary* ξ *is also* $\Omega(n^{2/3})$*.*

Proof. Recall that we have shown in Section 7.4.4 that ED reduces to CP by viewing ED as one-dimensional CP with the minimum distance 0. It is not hard to see that ED also reduces to approximate CP with multiplicative error $1+\xi$ since 0 times $1+\xi$ is still 0. For simplicity, we denote approximate CP with multiplicative error $1+\xi$ as $(1+\xi)$-CP. Given a set S as a $(1+\xi)$-CP instance, we choose $A, B \subset S$ uniformly at random such that $A = S \setminus B$ and $|A| = |B|$. Then, with $1/2$ probability, a closest pair in S has one point in A and another in B. Therefore, if (a, b) be a valid solution for $(1+\xi)$-BCP on (A, B), (a, b) is also a a valid solution for $(1+\xi)$-CP on S with probability $1/2$.

It is obvious that following the same proof, CP reduces to BCP. Hence, the quantum query complexity for BCP and $(1+\xi)$-BCP are both $\Omega(n^{2/3})$. This completes the proof. □

7.6 Orthogonal Vectors in Constant Dimensions

Theorem 7.43. *The time complexity of* $\mathsf{OV}_{n,d}$ *(Definition 7.4) in quantum query model is* $\Theta(\sqrt{n})$ *when the dimension* d *is constant .*

Proof. We show lower and upper bounds for $\mathsf{OV}_{n,d}$:

Lower bound We reduce the search problem to an instance of 2-dimensional OV. Let all vectors in A be $(0,1)$. We map an element of the search instance with value 0 as a vector in B with value $(0,1)$ in $\mathsf{OV}_{n,2}$, and 1 as $(1,0)$. An orthogonal pair must contain the vector in B with value $(1,0)$ in this construction. Therefore, if we find an orthogonal pair, we find the corresponding marked (value 1) element in the search instance. The $\Omega(\sqrt{n})$ lower bound of Grover's search algorithm gives an $\Omega(\sqrt{n})$ lower bound to $\mathsf{OV}_{n,d}$.

Upper bound The vectors only have 2^d possible values, $\{0,1\}^d$, in the d-dimensional OV. For a particular value $v \in \{0,1\}^d$, we can use Grover search to check whether there exist vector $a \in A$ such that $a = v$ in time $O(\sqrt{n})$, and similarly for vectors in B. Therefore we can, for all $v \in \{0,1\}^d$, check whether there exist $a \in A$ such that $a = v$ and $b \in B$ such that $b = v$ in $O(2^{d+1}\sqrt{n})$ time, recording the results as two 2^d bit strings S_A and S_B. Then we check all 2^{2d} pairs of values (v,w) whether $\langle v,w\rangle = 0$, $S_A(v) = 1$, and $S_B(w) = 1$. When we found such a pair (v,w), we use Grover's search algorithm again to output a corresponding pair of vectors. The total running time is $O(2^{d+1}\sqrt{n} + 2^{2d} + 2\sqrt{n}) = \widetilde{O}(\sqrt{n})$. □

Algorithm 25 Updating nodes for insertion.

```
1: procedure UpdateIns($i$, $p_i$)    ▷ The leaf node in $\tau'(S)$ corresponding to the $\epsilon$-box $g(\mathrm{id}(p_i))$, denoted by $\tau'(S, g(\mathrm{id}(p_i)))$.
2:     if the local counter $\mathcal{C} = 1$ in $\tau'(S, \mathrm{id}(p_i))$ then
3:         for all $\epsilon$-box $g'$ that is a $\epsilon$-neighbor (see Definition 7.18) of $g(\mathrm{id}(p_i))$ where the local counter $\mathcal{C} = 1$ in $\tau'(S, g')$ and the distance between $p_i$ and the point in $g'$ is at most $\epsilon$ do
4:             Increase the external counter $\mathcal{E}$ of $\tau'(S, g')$ by 1
5:             Increase the external counter $\mathcal{E}$ of $\tau'(S, g(\mathrm{id}(p_i)))$ by 1
6:             if the external counter $\mathcal{E}$ in $\tau'(S, g')$ was increased from 0 to 1 then
7:                 Set the flag $\mathcal{F}$ in $\tau'(S, g')$
8:                 Update the flag $\mathcal{F}$ in the nodes along the path from $\tau'(S, g')$ to the root of $\tau'(S)$
9:             end if
10:        end for
11:        if the external counter $\mathcal{E} > 1$ in $\tau'(S, g(\mathrm{id}(p_i)))$ then
12:            Set the flag $\mathcal{F}$ in $\tau'(S, g(\mathrm{id}(p_i)))$
13:            Update the flag $\mathcal{F}$ in the nodes along the path from $\tau'(S, \mathrm{id}(p_i))$ to the root of $\tau'(S)$
14:        end if
15:    else if the local counter $\mathcal{C} = 2$ in $\tau'(S, \mathrm{id}(p_i))$ then
16:        Set the flag $\mathcal{F}$ in $\tau'(S, g(\mathrm{id}(p_i)))$
17:        Update the flag $\mathcal{F}$ in the nodes along the path from $\tau'(S, g(\mathrm{id}(p_i)))$ to the root of $\tau'(S)$
18:        Set the external counter $\mathcal{E} = 0$ in $\tau'(S, \mathrm{id}(p_i))$
19:        Let $i'$ be the other index (than $i$) stored in the skip list corresponding to $g(\mathrm{id}(p_i))$
20:        for all $\epsilon$-box $g'$ that is a $\epsilon$-neighbor of $g(\mathrm{id}(p_i))$ where the local counter $\mathcal{C} = 1$ in $\tau'(S, g')$ and the distance between $p_{i'}$ and the point in $g'$ is at most $\epsilon$ do
21:            Decrease the external counter of $\tau'(S, g')$ by 1
22:            if the external counter $\mathcal{E}$ in $\tau'(S, g')$ was decreased from 1 to 0 then
23:                Unset the flag $\mathcal{F}$ in $\tau'(S, g')$
24:                Update the flag $\mathcal{F}$ in the nodes along the path from $\tau'(S, g')$ to the root of $\tau'(S)$
25:            end if
26:        end for
27:    end if
28: end procedure
```

Algorithm 26 Updating nodes for deletion.

```
1: procedure UpdateDel($i$, $p_i$)    ▷ The leaf node in $\tau'(S)$ corresponding to the $\epsilon$-box $g(\mathrm{id}(p_i))$, denoted by, $\tau'(S, g(\mathrm{id}(p_i)))$.
2:     if the local counter $\mathcal{C} = 0$ in $\tau'(S, \mathrm{id}(p_i))$ then
3:         Unset the flag $\mathcal{F}$ in $\tau'(S, g(\mathrm{id}(p_i)))$
4:         Update the flag $\mathcal{F}$ in the nodes along the path from $\tau'(S, \mathrm{id}(p_i))$ to the root of $\tau'(S)$
5:         Set the external counter $\mathcal{E} = 0$ in $\tau'(S, \mathrm{id}(p_i))$
6:         for all $\epsilon$-box $g'$ that is a $\epsilon$-neighbor (see Definition 7.18) of $g(\mathrm{id}(p_i))$ where the local counter $\mathcal{C} = 1$ in $\tau'(S, g')$ and the distance between $p_i$ and the point in $g'$ is at most $\epsilon$ do
7:             Decrease the external counter $\mathcal{E}$ of $\tau'(S, g')$ by 1
8:             if the external counter $\mathcal{E}$ in $\tau'(S, g')$ was decreased from 1 to 0 then
9:                 Unset the flag $\mathcal{F}$ in $\tau'(S, g')$
10:                Update the flag $\mathcal{F}$ in the nodes along the path from $\tau'(S, g')$ to the root of $\tau'(S)$
11:            end if
12:        end for
13:    else if the local counter $\mathcal{C} = 1$ in $\tau'(S, \mathrm{id}(p_i))$ then
14:        Let $i'$ be the only index stored in the skip list corresponding to $g(\mathrm{id}(p_i))$
15:        for all $\epsilon$-box $g'$ that is a $\epsilon$-neighbor of $g(\mathrm{id}(p_i))$ where the local counter $\mathcal{C} = 1$ in $\tau'(S, g')$ and the distance between $p_{i'}$ and the point in $g'$ is at most $\epsilon$ do
16:            Increase the external counter $\mathcal{E}$ of $\tau'(S, g')$ by 1
17:            Increase the external counter $\mathcal{E}$ of $\tau'(S, g(\mathrm{id}(p_i)))$ by 1
18:            if the external counter $\mathcal{E}$ in $\tau'(S, g')$ was increased from 0 to 1 then
19:                Set the flag $\mathcal{F}$ in $\tau'(S, g')$
20:                Update the flag $\mathcal{F}$ in the nodes along the path from $\tau'(S, g')$ to the root of $\tau'(S)$
21:            end if
22:        end for
23:        if the external counter $\mathcal{E} = 0$ in $\tau'(S, g(\mathrm{id}(p_i)))$ then
24:            Unset the flag $\mathcal{F}$ in $\tau'(S, g(\mathrm{id}(p_i)))$
25:            Update the flag $\mathcal{F}$ in the nodes along the path from $\tau'(S, \mathrm{id}(p_i))$ to the root of $\tau'(S)$
26:        end if
27:    end if
28: end procedure
```

Algorithm 27 The algorithm for multiple ϵ-close pair

```
1: procedure MULTICP($S$, $\epsilon$)                                    ▷ $|S| = n$.
2:     Let $T_1 = S$ and $j = 1$
3:     while $|T_j| > n^{2/3}$ do
4:         Run the algorithm described in Lemma 7.29 on $T_j$, and Measure the final
   state.
5:         If there is a pair with distance less than $\epsilon$, output the pair and stop
6:         Let $q_j$ be an even power of a prime with $|T_j| \leq q_j \leq (1 + \frac{1}{8})|T_j|$.
7:         Select a random permutation $\pi_j$ on $[q_j]$ from the $\frac{1}{n}$-approximately $4 \log n$-
   wise independent family of permutations as in Lemma 7.30
8:         Let
               $$T_{j+1} := \left\{ \pi_1^{-1} \cdot \pi_2^{-1} \cdots \pi_j^{-1}(i), \quad i \in \left[\left[ \frac{4q_j}{5} \right]\right] \right\}. \qquad (7.65)$$
9:         $j \leftarrow j + 1$
10:    end while
11:    if $|T_j| \leq n^{\frac{2}{3}}$ then
12:        Run Grover's search algorithm on $T_j$ for a pair with distance at most $\epsilon$
13:    end if
14: end procedure
```

Algorithm 28 Updating nodes for insertion for the bichromatic case.

```
1: procedure UpdateInsBCP($i$, $p_i$, $x$)  ▷ the leaf node in $\tau'(S)$ corresponding to $g(\mathrm{id}(p_i))$, denoted by $\tau'(S, g(\mathrm{id}(p_i)))$.
2:     Let $\bar{x} \in \{A, B\}$ and $\bar{x} \neq x$
3:     if $\mathcal{C}^x = 1$ in $\tau'(S, \mathrm{id}(p_i))$ and $\mathcal{C}^{\bar{x}} = 0$ then
4:         for all $\epsilon$-neighbor $g'$ (see Definition 7.18) of $g(\mathrm{id}(p_i))$ where $\mathcal{C}^{\bar{x}} \geq 1$ in $\tau'(S, g')$ do
5:             Increase $\mathcal{E}^x$ of $\tau'(S, g')$ by 1
6:             Increase $\mathcal{E}^{\bar{x}}$ of $\tau'(S, g(\mathrm{id}(p_i)))$ by 1
7:             if $\mathcal{E}^x$ in $\tau'(S, g')$ was increased from 0 to 1 then
8:                 Set the flag $\mathcal{F}$ in $\tau'(S, g')$
9:                 Update the flags $\mathcal{F}$ in the nodes along the path from $\tau'(S, g')$ to the root of $\tau'(S)$
10:            end if
11:        end for
12:        if $\mathcal{E}^{\bar{x}} \geq 1$ in $\tau'(S, g(\mathrm{id}(p_i)))$ then
13:            Set the flag $\mathcal{F}$ in $\tau'(S, g(\mathrm{id}(p_i)))$
14:            Update the flags $\mathcal{F}$ in the nodes along the path from $\tau'(S, \mathrm{id}(p_i))$ to the root of $\tau'(S)$
15:        end if
16:    else if $\mathcal{C}^x = 1$ and $\mathcal{C}^{\bar{x}} \geq 1$ in $\tau'(S, g(\mathrm{id}(p_i)))$ then
17:        Set the flag $\mathcal{F}$ in $\tau'(S, g(\mathrm{id}(p_i)))$
18:        Update the flags $\mathcal{F}$ in the nodes along the path from $\tau'(S, g(\mathrm{id}(p_i)))$ to the root of $\tau'(S)$
19:        Set $\mathcal{E}^{\bar{x}} = 0$ in $\tau'(S, \mathrm{id}(p_i))$
20:        for all $g'$ that is an $\epsilon$-neighbor of $g(\mathrm{id}(p_i))$ where the the local counter $\mathcal{C}^{\bar{x}} \geq 1$ in $\tau'(S, g')$ do
21:            Decrease $\mathcal{E}^x$ of $\tau'(S, g')$ by 1
22:            if $\mathcal{E}^x$ in $\tau'(S, g')$ was decreased from 1 to 0 then
23:                Unset the flag $\mathcal{F}$ in $\tau'(S, g')$
24:                Update the flags $\mathcal{F}$ in the nodes along the path from $\tau'(S, g')$ to the root of $\tau'(S)$
25:            end if
26:        end for
27:    end if
28: end procedure
```

Algorithm 29 Updating nodes for deletion for the bichromatic case.

```
1: procedure UpdateDelBCP($i$, $p_i$, $x$) ▷ the leaf node in $\tau'(S)$ corresponding to $g(\mathrm{id}(p_i))$, which we denote as $\tau'(S, g(\mathrm{id}(p_i)))$.
2:     Let $\bar{x} \in \{A, B\}$ and $\bar{x} \neq x$
3:     if $\mathcal{C}^x$ and $\mathcal{C}^{\bar{x}}$ in $\tau'(S, \mathrm{id}(p_i)) = 0$ then
4:         Unset the flag $\mathcal{F}$ in $\tau'(S, g(\mathrm{id}(p_i)))$
5:         Update the flags $\mathcal{F}$ in the nodes along the path from $\tau'(S, \mathrm{id}(p_i))$ to the root of $\tau'(S)$
6:         Set $\mathcal{E}^x = 0$ and $\mathcal{E}^{\bar{x}} = 0$ in $\tau'(S, \mathrm{id}(p_i))$
7:         for all $g'$ that is an $\epsilon$-neighbor (see Definition 7.18) of $g(\mathrm{id}(p_i))$ where the local counter $\mathcal{C}^{\bar{x}} \geq 1$ in $\tau'(S, g')$ do
8:             Decrease $\mathcal{E}^x$ of $\tau'(S, g')$ by 1
9:             if $\mathcal{E}^x$ in $\tau'(S, g')$ was decreased from 1 to 0 then
10:                Unset the flag $\mathcal{F}$ in $\tau'(S, g')$
11:                Update the flags $\mathcal{F}$ in the nodes along the path from $\tau'(S, g')$ to the root of $\tau'(S)$
12:            end if
13:        end for
14:    else if $\mathcal{C}^x = 0$ and $\mathcal{C}^{\bar{x}} \geq 1$ then
15:        for all $g'$ that is an $\epsilon$-neighbor of $g(\mathrm{id}(p_i))$ where the local counter $\mathcal{C}^x \geq 1$ in $\tau'(S, g')$ do
16:            Increase $\mathcal{E}^{\bar{x}}$ of $\tau'(S, g')$ by 1
17:            Increase $\mathcal{E}^x$ of $\tau'(S, g(\mathrm{id}(p_i)))$ by 1
18:            if $\mathcal{E}^{\bar{x}}$ in $\tau'(S, g')$ was increased from 0 to 1 then
19:                Set the flag $\mathcal{F}$ in $\tau'(S, g')$
20:                Update the flags $\mathcal{F}$ in the nodes along the path from $\tau'(S, g')$ to the root of $\tau'(S)$
21:            end if
22:        end for
23:        if $\mathcal{E}^x = 0$ in $\tau'(S, g(\mathrm{id}(p_i)))$ then
24:            Unset the flag $\mathcal{F}$ in $\tau'(S, g(\mathrm{id}(p_i)))$
25:            Update the flags $\mathcal{F}$ in the nodes along the path from $\tau'(S, \mathrm{id}(p_i))$ to the root of $\tau'(S)$
26:        end if
27:    end if
28: end procedure
```

Chapter 8: Quantum Meta-Complexity

8.1 Introduction

The Minimum Circuit Size Problem (MCSP) is one of the central computational problems in complexity theory. Given the truth table of a Boolean function $f : \{0,1\}^n \to \{0,1\}$ and a size parameter s (in unary) as inputs, MCSP asks whether there exists a circuit of size at most s for f. While MCSP has been studied as early as the 1950s in the Russian cybernetics program [Tra84], its complexity remains mysterious: we do not know whether it is in P or NP-hard. Meanwhile, besides being a natural computational problem, in recent years, researchers have discovered many surprising connections of MCSP to other areas such as cryptography [RR97], learning theory [CIKK16], circuit complexity [KC00], average-case complexity [Hir18], and others.

Quantum computing is of growing interest, with applications to cryptography [Sho94], machine learning [BWP+17], and complexity theory [JNV+20], etc. Inspired by the great success of MCSP in classical computation and the flourishing of quantum computers, we propose a new research program of studying quantum computation through the lens of MCSP. We envision MCSP as a central problem that connects different quantum computation applications and provides deeper insights into the complexity-theoretic foundation of quantum circuits.

8.1.1 The classical MCSP and its connections to other problems

It is immediate that MCSP $\in$ NP because the input size is 2^n so one can verify if a circuit (given as the certificate/proof) computes the input truth table in time $2^{O(n)}$. However, there is no consensus on the complexity status of this problem – MCSP could be in P, NP-complete, or NP-intermediate. Several works [MW17, KC00] showed negative evidence for proving the NP-hardness of MCSP using standard reduction techniques. We also do not know whether there is an algorithm better than brute force search (see Perebor conjecture for MCSP [Tra84]) or whether there is a search-to-decision reduction or a self-reduction[1] for MCSP[2]. On the other hand, several variants of MCSP are NP-hard under either deterministic reductions [Mas79, HOS18] or randomized reductions [Ila19, ILO20].

Researchers have discovered many surprising connections of MCSP to other fields in Theoretical Computer Science including cryptography, learning theory, and circuit lower bounds. To name a few, Razborov and Rudich [RR97] related natural properties against P/poly with circuit lower bounds and pseudorandomness. Kabanets and Cai [KC00] showed that MCSP $\in$ P implies new circuit lower bounds, and that MCSP $\in$ BPP implies that any one-way function can be inverted. Allender and Das [AD14a] related the complexity class SZK (Statistical Zero Knowledge) to MCSP. Carmosino et al. [CIKK16] showed that MCSP $\in$ BPP gives efficient PAC-learning algorithms. Impagliazzo et al. [IKV18] showed that the existence of indistinguishable obfuscation implies that SAT reduces to MCSP under a randomized reduction. Hirahara [Hir18] showed that if an approximation version of MCSP is NP-hard, then the average-case and worst-case hardness of NP are equivalent. Arunachalam et al. [AGG$^+$20] proved that MCSP $\in$ BQP implies new circuit lower bounds. All these results indicate that the MCSP serves as a "hub" that connects many fundamental problems in different fields. Therefore, a deeper understanding of this problem could lead to significant progress in Theoretical Computer Science.

8.1.2 Main results and technical overview

In this chapter, we consider three different natural objects that a quantum circuit can compute: Boolean functions, unitaries, and quantum states. We start

[1]Roughly, a problem is self-reducible if one can solve the problem with size n by algorithms for smaller size.

[2]It is worth noting that every NP-complete problem has search-to-decision reductions and self-reductions.

with giving the informal definitions of the minimum circuit size problem for each of them. See Section 8.3 and Section 8.5 for the formal definitions.

Definition 8.1 (MQCSP, informal)**.** Given the truth table of a Boolean function f and a size parameter s in unary, decide if there exists a quantum circuit C which has size at most s and uses at most s ancilla qubits such that C computes f with high probability.

Definition 8.2 (UMCSP, informal)**.** Given the full description of a 2^n-dimensional unitary matrix U and a size parameter s in unary, decide if there exists a quantum circuit C which has size at most s and uses at most s ancilla qubits such that C and U are close[3].

Definition 8.3 (SMCSP, informal)**.** Let $|\psi\rangle$ be an n-qubit state. Given size parameters s and n in unary and access to arbitrarily many copies of $|\psi\rangle$ (or the classical description of $|\psi\rangle$), decide if there exists a quantum circuit C which has size at most s using at most s ancilla qubits such that $C|0^n\rangle$ and $|\psi\rangle$ are close in terms of fidelity.

In the rest of this subsection, we first discuss several challenges and difficulties we encountered in the study of MCSP when moving from the classical setting to the quantum setting. Next, we give an overview of all the results and techniques. In particular, we focus on both interpreting the new connections we establish as well as the technical subtleties when quantizing the previous works in the classical setting. For a quick summary of the results, please take a look at Table 8.1.

8.1.2.1 Challenges and difficulties when moving to the quantum setting

In the following, we summarize several fundamental properties of quantum circuits, unitaries, and quantum states that induce problems and difficulties that would not appear in the classical setting.

[3]We say C and U are close if $|(\langle\psi|\otimes I)U^\dagger C(|\psi\rangle|0\rangle)|$ is large for all $|\psi\rangle$.

Quantum computation is generally random and erroneous. It is natural to consider quantum circuits that approximate (rather than exactly computing) the desired unitary. One immediate consequence is that we have to define the quantum MCSPs as promise problems (with respect to the error)[4], which is more challenging to deal with. Moreover, since unitaries and quantum states are specified by complex numbers, we also need to properly tackle the precision issue. These quantum properties make generalizing classical results to the quantum setting non-trivial. For instance, some classical analyses (see [AGG+20] for an example) rely on the fact that the classical circuits are deterministic after the random string is made public, while any intermediate computation of a quantum circuit is inherently not deterministic.

Quantum circuits are reversible. This follows from the fact that every quantum gate is reversible. While this seems to be a restriction for quantum circuits, we observe that this enables search-to-decision reductions for UMCSP and SMCSP. Note that the existence of such reduction is a longstanding open question for classical MCSP. This suggests that quantum MCSPs can provide a new angle to leverage the reversibility of quantum circuits.

The introduction of ancilla qubits. As quantum circuits are reversible, every intermediate computation has to happen on the input qubits. Thus, it is very common to introduce *ancilla qubits* which are extra qubits initialized to all zero and can be regarded as additional registers for intermediate computation. Ancilla qubits introduce complications in quantum MCSPs. First, the quantum circuit complexity of an object could be very different when the allowed number of ancilla qubits is different. Second, the classical simulation time of a quantum circuit scales exponentially in the number of input qubits plus the number of ancilla qubits. Namely, when the number of ancilla

[4]The definitions above are not promise problems for simplicity. Check Section 8.3 and 8.5 for formal definitions.

qubits is super-linear, classical simulations would require super-polynomial time[5]. An immediate consequence is that, unlike classical MCSP, MQCSP is not trivially in NP when allowing a super-linear number of ancilla qubits. In addition, the output of quantum circuits on ancilla qubits can be arbitrary quantum states in general. This property makes certain reductions for quantum MCSPs fail when considering many ancilla qubits.

Various universal quantum gate sets. The choice of the gate set affects the circuit complexity of the given Boolean functions (and unitaries and states). There are various universal quantum gate sets, and transforming from one to the other results in additional polylogarithmic overhead to the circuit complexity by the Solovay-Kitaev Theorem. We note that when considering certain hardness results, the choice of the gate set might matter. Take the approximate self-reduction for SMCSP (in Theorem 8.9) as an example, we start from constructing such reductions for a particular gate set. We then generalize the result to an arbitrary gate set via the Solovay-Kitaev Theorem; however, it introduces additional overhead to the approximation ratio. Another example is proving NP-hardness for multi-output MQCSP, where we show that the problem is NP-hard when considering particular gate sets, and it is still open whether the problem is NP-hard for all universal gate sets.

8.1.2.2 The Hardness of MQCSP and cryptography

We start with stating the hardness results of MQCSP and its implications in cryptography.

Theorem 8.1 (Informal)**.**

1. MQCSP *is in* QCMA $\subseteq$ QMA*.*

[5]The running time is measured with respect to the size of the truth table or the size of the unitary/quantum state.

2. *If* MQCSP *can be solved in quantum polynomial time, then quantum-secure one-way function (*qOWF*) does not exist.*

3. *If one can solve* MQCSP *efficiently, then all problems in* SZK *have efficient algorithms.*

4. *Suppose that quantum-secure indistinguishability obfuscator ($i\mathcal{O}$) for polynomial-size circuits exists. Then,* MQCSP $\in$ BQP *implies* NP $\subseteq$ coRQP[6].

5. *Multiple-output* MQCSP *(under a gate set with some natural properties) is* NP*-hard under randomized reductions.*

We have discussed why MQCSP is not trivially in NP earlier. So, it is natural to wonder what can be a tighter upper bound for MQCSP. Instead of considering classical verifier, we allow the verifier to check the given witness circuit quantumly and thus are able to prove that MQCSP is in QCMA (which is a quantum analogue of MA allowing efficient quantum verifiers but classical witness).

For item 2 – 5, we study whether some hard problems reduce to MQCSP. Classically, many results use the fact that an MCSP oracle can break certain *pseudo-random generators* to show reductions from hard problems to MCSP. A distinguisher can break a pseudorandom generator by viewing that the string is a truth table of some Boolean function and using the MCSP oracle to decide if the function has small circuit complexity[7]. We generalize this idea to the quantum setting by observing that if the Boolean function has small classical circuit complexity, then its quantum circuit complexity is also small. It is worth noting that the second result implies efficient algorithms for some lattice problems if MQCSP is in BQP.

[6]coRQP is a complexity class of quantumly solvable problems with perfect soundness and bounded-error completeness.

[7]If the truth table is truly random, it corresponds to a random function and must have large circuit complexity with high probability.

For item 5, we generalize the recent breakthrough of Ilango et al. [ILO20] on the NP-hardness of MCSP. We note that the formal theorem statement depends on the gate set choices of MQCSP. To prove this theorem, we follow the proof ideas in [ILO20] and overcome some additional obstacles that appear in the quantum world. The new obstacle comes from (i) the quantum gate set is different from the one in the classical case; (ii) in the quantum world, we need to deal with error terms. We carefully handle these issues and extend the proof to the quantum setting.

8.1.2.3 MQCSP and learning theory

A central learning theory setting is (approximately) reconstructing a circuit for an unknown function given a limited number of samples. Learning Boolean functions in the classical setting was extensively studied (see, for example, a survey by Hellerstein and Servedio [HS07]); however, relatively few explorations have been made under the quantum setting. There are two natural quantum extensions: (i) learning a quantum circuit and (ii) adding quantumness in the learning algorithm. We study both scenarios and provide generic connections between MQCSP and the two settings

PAC learning for quantum circuits. Probabilistic approximately correct (PAC) learning [Val84] is a standard theoretical framework in learning theory. There are several variants, but for simplicity, we focus on the query model where a classical learning algorithm can query an unknown n-bit Boolean function f on inputs $x_1, \ldots, x_m \in \{0,1\}^n$ and aim to output a circuit approximating f with high probability. To have efficient PAC learning algorithms for polynomial-size quantum circuits, we show that it is necessary and sufficient to have efficient algorithms for MQCSP or its variants.

Theorem 8.2 (Informal)**.** *The existence of an efficient PAC learning algorithm for* BQP/poly *is equivalent to the existence of an efficient randomized algorithm for* MQCSP*.*

Quantum learning. In the past two decades, there has been increased interest in quantum learning (see a survey by Arunachalam and de Wolf [AdW17]) due to the success of machine learning and quantum computing. While there have been interesting quantum speed-ups for specific learning problems such as Principal Component Analysis [LMR14] and quantum recommendation system [KP17], it is unclear whether the quantumness can provide a generic speed-up in learning theory. A recent result of Arunachalam et al. [AGG$^+$20] suggested that this might be difficult by showing that the existence of efficient quantum learning algorithms for a circuit class would imply a breakthrough circuit lower bound. We further generalize their result by showing the equivalence of efficient quantum PAC learning and the non-trivial upper bound for MQCSP.

Theorem 8.3 (Informal). *The existence of efficient quantum learning algorithms for PAC learning a circuit class* C *is equivalent to the existence of efficient quantum algorithms for* C-MQCSP[8].

The proof idea is to quantize the "learning from a natural property" paradigm of [CIKK16]. Briefly speaking, the converse direction "algorithms for MQCSP imply learning algorithms" follows from the idea that one can use the Boolean function (the object to be learned) to construct a PRG with the property that breaking the PRG implies a reconstructing algorithm for f. Then, since an algorithm for MQCSP can break PRG, we obtain an algorithm for f. Another direction follows from the observation that we can still apply the learning algorithm given the truth table of the function. Specifically, for Theorem 8.2, it turns out that the converse direction is straightforward because $\mathsf{P/poly} \subset \mathsf{BQP/poly}$ while the forward direction requires the number of ancilla bits to be $O(n)$ due to the overhead from a classical simulation for quantum circuits. For Theorem 8.3, the difficulty lies in the fact that a quantum circuit is *inherently random* and one cannot arbitrarily compose quantum circuits as

[8] C-MQCSP is MQCSP with respect to circuit class C.

their wishes. To circumvent these issues, we invoke the techniques in [AGG$^+$20] which built up composable tools for *reconstructing* a circuit from a quantum distinguisher. See Theorem 8.23, Theorem 8.22, and Section 8.4.2 for more details.

8.1.2.4 MQCSP and quantum circuit lower bounds

The classical MCSP is tightly connected to circuit lower bounds. We generalize the results of Oliveira and Santhanam [OS16], Arunachalam et al. [AGG$^+$20], and Kabanets and Cai [KC00] to MQCSP.

Theorem 8.4 (Informal). *Suppose that* $\mathsf{MQCSP} \in \mathsf{BQP}$. *Then*

1. $\mathsf{BQE} \not\subset \mathsf{BQC}[n^k]$ *for any constant* $k \in \mathbb{N}$[9]*; and*

2. $\mathsf{BQP}^{\mathsf{QCMA}} \not\subset \mathsf{BQC}[n^k]$ *for any constant* $k \in \mathbb{N}$.

For item 1, we use MQCSP to construct a BQP-natural property against quantum circuit classes. Then, with a quantum-secure pseudorandom generator, we can use a "win-win argument" to show that $\mathsf{BQE} \not\subset \mathsf{BQC}[n^k]$ for any $k > 0$. The proof mainly follows from [AGG$^+$20, OS16]. However, we extend their proofs to the quantum natural properties against *quantum* circuit classes. One technical contribution is a diagonalization lemma for quantum circuits.

For item 2, we follow the idea in [KC00] to show that the maximum quantum circuit complexity problem[10] can be solved in exponential time with a QCMA oracle. The main difference from the classical case is that we require a QCMA oracle instead

[9] $\mathsf{BQC}[n^k]$ is the complexity class for problems that can be solved by $O(n^k)$-size quantum circuits with bounded fan-in, and BQE in the set of problems that can be solved in $2^{O(n)}$ time by quantum computers. Previously, Aaronson [Aar06] showed that $\mathsf{P}^{\mathsf{PP}} \not\subset \mathsf{BQC}[n^k]$ unconditionally. However, the relations between P^{PP}, BQE, and $\mathsf{BQP}^{\mathsf{QCMA}}$ are still unclear.

[10] The problem is, given 1^n, ask for a Boolean function $f : \{0,1\}^n \to \{0,1\}$ that has the maximum complexity.

of an NP one, which follows from the fact that we assume MQCSP is in BQP[11]. Then, the statement follows from the standard padding argument.

Another aspect of quantum circuit complexity is *hardness amplification*. Kabanets and Cai [KC00] showed that MCSP can be used as an amplifier to generate many hard Boolean functions. In this part, we show that with an MQCSP oracle, given one quantum extremely hard Boolean function, there is an efficient quantum algorithm that outputs many quantum-hard functions.

Theorem 8.5 (Hardness amplification by MQCSP, informal)**.** *Assume* $\mathsf{MQCSP} \in \mathsf{BQP}$*. There exists a* BQP *algorithm that, given the truth table of a Boolean function with quantum circuit complexity* $2^{\Omega(n)}$*, outputs* $2^{\Omega(n)}$ *Boolean functions with* $m = \Omega(n)$ *variables such that each function has quantum circuit complexity greater than* $2^m/(c+1)m$ *for* c *some constant.*

The proof of Theorem 8.5 closely follows the proof in [KC00]. The key ingredient is a quantum Impagliazzo-Wigderson generator, which "quantizes" the construction in [IW97]. The quantum Impagliazzo-Wigderson generator can transform the given quantum extremely hard function to a quantum pseudorandom generator that fools quantum circuits of size $2^{O(n)}$. Since we assume $\mathsf{MQCSP} \in \mathsf{BQP}$, it means that we can construct a small quantum distinguishing circuit to accept the truth tables of hard functions. And we can show that our quantum Impagliazzo-Wigderson generator can fool the distinguishing circuit. Hence, most of the outputs of the quantum pseudorandom generator will have high quantum circuit complexity.

To quantize the Impagliazzo-Wigderson generator, we construct a quantum-secure direct-product generator, and also use the quantum Goldreich-Levin Theorem and quantum-secure Nisan-Wigderson generator developed in [AGG+20].

[11]Along this line, the result still holds if we consider $\mathsf{MCSP} \in \mathsf{BQP}$ and maximum classical circuit complexity.

Hardness magnification is an interesting phenomenon in classical circuit complexity defined by [OS18]. It shows that a weak worst-case lower bound can be "magnified" into a strong worst-case lower bound for another problem. (See a recent talk by Oliveira [Oli19].) In this part, we show that MQCSP also has a quantum hardness magnification.

Theorem 8.6 (Hardness magnification for MQCSP, informal)**.** *If a gap version of* MQCSP *does not have nearly-linear size quantum circuit, then* QCMA *cannot be computed by polynomial size quantum circuits.*

We note that this is a nontrivial theorem because even if we assume $\mathsf{QCMA} \subseteq \mathsf{BQC}[\mathsf{poly}(n)]$, we can only show $\mathsf{MQCSP} \in \mathsf{BQC}[\mathsf{poly}(2^n)]$, i.e., MQCSP has a polynomial-size quantum circuit by the fact that $\mathsf{MQCSP} \in \mathsf{QCMA}$. But the theorem implies that some gap-version of MQCSP has nearly-linear size circuit!

We prove the above theorem via a quantum antichecker lemma, whose classical version was given by [OPS19, CHO+20]. And we observe that the two key ingredients: a delicate design of a Boolean circuit and a counting argument can be quantized.

8.1.2.5 MQCSP and quantum fine-grained complexity

Fine-grained complexity theory aims to study the *exact* lower/upper bounds of some problems. For example, most theorists believe 3-SAT is not in P, but we do not know if it can be solved in $2^{o(n)}$ time. Exponential Time Hypothesis (ETH) is a commonly used conjecture in this area which rules out this possibility (see a survey by Williams [Wil18]). Very recently, [Ila20b] showed the fine-grained hardness of MCSP for partial function based on ETH. In the quantum setting, [ACL+20, BPS21] proposed quantum fine-grained reductions and quantum strong exponential time hypothesis (QSETH) to study the quantum hardness of problems in BQP. In this part, we follow the works of [Ila20b, ACL+20] and prove the quantum hardness of MQCSP for partial functions based on the quantum ETH conjecture,which conjectures that there does not exist a $2^{o(n)}$-time quantum algorithm for solving 3-SAT[12].

Theorem 8.7 (Fine-grained hardness of $\mathsf{MQCSP}^{\star}$, informal)**.** *Quantum* ETH *implies* $N^{o(\log\log N)}$*-quantum hardness of* MQCSP *for partial functions.*

To prove the above theorem, we basically follow the reduction path in [Ila20b], which gave a reduction from a fine-grained problem studied by [LMS11] to MQCSP for partial functions. But we need to bypass two subtleties:

- The proof of [Ila20b] relies on the structure of the classical read-once formula, but there is no direct correspondence with quantum;
- [LMS11] only proved the classical hardness of the bipartite permutation independent set problem, but we need quantum hardness result.

For the first issue, we prove an unconditional quantum circuit lower bound for that function in the reduction. More specifically, we first show that if a small quantum circuit can compute the partial function γ in the reduction, then that circuit is a quantum read-once formula (defined by [Yao93]); and vice versa. And then, we apply a "dequantization" result by [CKP13] to show that the quantum read-once formula can be converted to a classical read-once formula with the same size. Then, by the structure of the "dequantized" read-once formula, we finally conclude that deciding MQCSP for γ is equivalent to solving the bipartite permutation independent set problem.

For the second issue, we use the quantum fine-grained reduction framework and give a reduction from 3-SAT to the bipartite permutation independent set problem. Therefore, the quantum hardness of MQCSP for partial function follows from the quantum hardness of deciding 3-SAT conjectured by the quantum ETH.

[12]Existing quantum SAT solvers are not much faster than Grover's search; they need $2^{\Omega(n)}$-time even for 3-SAT.

8.1.2.6 Quantum circuit complexity for states and unitaries

In this section, we study UMCSP and SMCSP. For SMCSP in Definition 8.3, we consider two types of inputs: quantum states and the classical description of the state. We consider the inputs as quantum states since we generally cannot have the classical description of the quantum state in the real world, and many related problems (such as shadow tomography [Aar18], quantum gravity [BFV20], and quantum pseudorandom state [JLS18]) have multiple copies of states as inputs. Although this input format makes SMCSP harder, we are able to show that SMCSP has a QCMA protocol[13]. Furthermore, the search-to-decision reduction and the self-reduction in Theorem 8.9 hold for both versions of SMCSP. We first show hardness upper bounds for UMCSP and SMCSP.

Theorem 8.8 (Informal)**.** (1) UMCSP $\in$ QCMA. (2) SMCSP *can be verified by* QCMA *protocols.*

To prove Theorem 8.8, we use the *swap test* to test whether the witness circuit C outputs the correct states. This suffices to show that SMCSP has a QCMA protocol. To show that UMCSP is in QCMA, checking if the circuit C and U agree on all inputs by using swap test is infeasible since there are infinitely many quantum states in the 2^n-dimensional Hilbert space. If one only checked all the computational basis states (i.e., $\{|x\rangle : x \in \{0,1\}^n\}$), it is possible that the circuit C and the given unitary U are not close on inputs in the form of superposition states. This can come from the following two sources. (a) C can introduce different phases on different computational basis states; (b) using ancilla qubits to implement U results in entanglement between the output qubits and ancilla qubits, which may fail the swap test.

To deal with these difficulties, we introduce an additional step in the test called "coherency test". This step tests the circuit output on all the initial states in the form

[13]Note that since SMCSP has quantum inputs, the problem is not in QCMA under the standard definition.

of $|a\rangle + |b\rangle$, where $|a\rangle, |b\rangle$ are different computational basis states. We can prove that it forces the behavior of C to be coherent on all the computational basis states, and forces the phases to be roughly the same.

Reductions for UMCSP and SMCSP that are unknown to the classical MCSP. In addition to the upper bounds, we also show interesting reductions for UMCSP and SMCSP.

Theorem 8.9 (Informal)**.**

- ***Search-to-decision reductions:*** *There exist search-to-decision reductions for* UMCSP *and* SMCSP *when no ancilla qubits are allowed.*
- ***Self-reduction:*** SMCSP *is approximately self-reducible.*
- *A gap version of* MQCSP *reduces to* UMCSP*.*

Classically, it is unknown whether MCSP is self-reducible or has search-to-decision reductions. Ilango [Ila20a] proved that some variants of MCSP have search-to-decision reductions. Recently, Ren and Santhanam [RS21] showed that a relativization barrier applies to the deterministic search-to-decision reduction and self-reduction of MCSP. We prove the existence of search-to-decision reductions by using the property that "*quantum circuits are reversible*". In particular, we guess the i-th gate, uncompute the gate from the state or the unitary, and use the decision oracles to check whether the complexity of the new state or the new unitary reduces. By repeating this process for all gates, we can find the desired circuits. This approach suffices for the case where the quantum circuits use no ancilla qubits. On the other hand, when the quantum circuits use ancilla qubits and are not forced to turn ancilla qubits back to the all-zero state, this approach does not work. Consider UMCSP. The quantum circuit may implement a unitary $U \otimes V$. To find the circuit, the approach above needs to start from $U \otimes V$ and do the uncomputation iteratively. However, V is unknown. SMCSP has the similar issues.

For the self-reducibility of SMCSP, we show that one can approximate the circuit complexity of an n-qubit state by computing the circuit complexities of $(n-1)$-qubit states. Roughly, we find a "win-win decomposition" of an n-qubit state such that its circuit complexity is either close to the circuit complexity of an $(n-1)$-qubit state or can be approximated by two $(n-1)$-qubit states.

Finally, we show a reduction related to MQCSP and UMCSP. The proof is by encoding a Boolean function into a particular unitary and showing that the circuit complexity of that unitary gives both upper and lower bounds for the circuit complexity of the Boolean function.

Implications of Hardness of SMCSP and UMCSP For UMCSP, one application is related to a question Aaronson asked in [Aar16]: does there exist an efficient quantum process that generates a family of unitaries that are indistinguishable from random unitaries given the full description of the unitary? If there is an efficient algorithm for UMCSP, then there is no efficient quantum process that generates unitaries indistinguishable from random unitaries given the full unitary.

Moreover, several implications of MCSP carry to UMCSP by Theorem 8.9. This follows from the fact that the gap version of MQCSP suffices to break certain pseudorandom generators.

For SMCSP, we focus on the version where the inputs are copies of quantum states and present its relationships to quantum cryptography, tomography, and quantum gravity.

Theorem 8.10 (Informal)**.**

1. *If* SMCSP *has quantum polynomial-time algorithms, then there are no pseudorandom states, and thus no quantum-secure one-way functions.*

2. *Assuming additional conjectures from physics and complexity theory, the existence of an efficient algorithm for* SMCSP *implies the existence of an efficient*

algorithm for estimating the wormhole's volume.

3. *If* SMCSP *can be solved efficiently, then one can solve the succinct state tomography problem*[14] *in quantum polynomial time.*

The first result in Theorem 8.10 follows from the observation that we can use SMCSP algorithms to distinguish whether the given states have large circuit complexities. This results in algorithms for breaking pseudorandom states, and thus algorithms for inverting quantum-secure one-way functions by [JLS18]. It is worth noting that a recent work by Kretschmer [Kre21] showed some relativized results for the problem of breaking pseudorandom states. Since that problem reduces to SMCSP, his results would provide another angle for understanding the hardness of SMCSP. We show the second result under the model and assumptions considered in [BFV20]. Roughly speaking, the volumes of wormholes correspond to circuit complexities of particular quantum states. Thus efficient algorithms for one implies solving the other one efficiently if the correspondence can be computed efficiently. The third result mainly uses the *search-to-decision reduction* in Theorem 8.9 to find the circuit that computes the state.

8.1.3 Discussion and open questions

We lay out the following three-aspect road map for the quantum MCSP program. For each aspect, we present several results and also propose many open directions to explore. We have also summarized all results in this chapter in Table 8.1.

First, we define the Minimum Quantum Circuit Size Problem (MQCSP) and study upper bounds and lower bounds for its complexity. Furthermore, we explore the connections between MQCSP and other areas of quantum computing such as quantum

[14]The succinct state tomography problem is that given many copies of a state with the promise that its circuit complexity is at most certain s, the problem is to find a circuit that computes the state.

cryptography, quantum learning, quantum circuit lower bounds, and quantum fine-grained complexity.

Then, we further extend MQCSP to study the quantum circuit complexities for quantum objects, including unitaries and states.[15] We want to investigate their hardness and connections to other areas in TCS. In this chapter, we show upper bounds and lower bounds for their complexities, search-to-decision reductions (for UMCSP and SMCSP), a self-reduction (for SMCSP), and reductions from MQCSP to UMCSP. In addition to connections generalized from classical analogues (such as cryptography, learning, and circuit lower bounds), we also find connections that might be unique in the quantum setting, such as tomography and quantum gravity.

For the last part, we want to turn around and ask what could happen when considering quantum algorithms or quantum reductions for MCSP (and also for MQCSP, UMCSP, and SMCSP)? In the previous two parts, we have already observed that efficient quantum algorithms for these problems result in surprising implications to other fields. One can further consider other influences of quantum algorithms to study quantum and classical MCSPs. For example, can SAT reduce to MCSP under quantum reductions?

Following the three-aspect road map for the quantum MCSP program, there are many open directions to explore. In particular, we are interested to understand the hardness of these problems, the relationships between them, and their connections to other fields in computer science.

8.1.3.1 Open problems: the complexity of quantum circuits

We start with open problems related to the hardness and relationships between quantum MCSPs. The most basic questions are to understand the complexity of

[15]Aaronson has raised questions about quantum circuit complexity for unitaries or states in [Aar16].

different quantum MCSPs. As we have already seen, it is unclear if quantum MCSPs are in NP. Besides, we do not know if NP- or QCMA-hard problems reduce to them.

Open Problem 8.1. Are UMCSP, MQCSP, and SMCSP in NP? Are these problems NP-hard, QCMA-hard, or C-hard for some complexity class C that is between QCMA and SZK?

We note that the case that makes these problems not known to be in NP is when there are more than linearly many ancilla qubits. Therefore, if one can show that adding superpolynomially many ancilla qubits does not lead to significant improvement on quantum circuit complexity, then we are likely to put these problems in NP directly. Along this line, we pose the following open question:

Open Problem 8.2. For every $n, s, t \in \mathbb{N}$ with $t \leq s \leq 2^{O(n)}$, is $\mathsf{BQC}(s,t) \subset \mathsf{BQC}(\mathsf{poly}(s,t), O(n))$?

For the hardness of UMCSP and SMCSP, One potential approach for proving NP-hardness of UMCSP is as follows: Prove the NP-hardness of the gap version of certain variants of MQCSP (such as sparse MQCSP or multiMQCSP), and then reduce it to UMCSP via the last reduction in Theorem 8.9. The hardness of SMCSP seems to be slightly more mysterious than UMCSP. One reason for this is that we do not know any relationship between SMCSP and other quantum MCSPs, and thus the approach of reducing particular variants of quantum MCSP to SMCSP does not directly work. This leads to another important open question:

Open Problem 8.3. What are the relationships between UMCSP, MQCSP, and SMCSP?

To answer whether quantum MCSPs are NP-complete, we can also study these problems from another angle, that is, check if quantum MCSPs have particular reductions that all NP-complete problems have. In the previous section, we observed that quantum circuits have some properties leading to search-to-decision reductions for UMCSP and SMCSP without ancilla qubits and an approximate self-reduction for SMCSP. Therefore, we ask whether we can have search-to-decision reductions and self-reductions for these quantum MCSPs.

Open Problem 8.4. Are there search-to-decision reductions and self-reductions for quantum MCSPs?

It is worth noting that our search-to-decision reductions fail when ancilla qubits are allowed. This mainly follows from the fact that the circuit of the solution can be an non-identity operator on the ancilla qubits in general. This could possibly be addressed by iterating all possible unitaries or states on an ϵ-net when the number of ancilla qubits are not large (e.g., at most $\log\log n$). However, we need new ideas when considering more ancilla qubits.

Moreover, it would be interesting to investigate the applications of these reductions. For instance, we have seen that the search-to-decision reductions give algorithms with UMCSP or SMCSP oracle additional power to obtain the circuits. This power may lead to interesting applications.

Open Problem 8.5. Is there any application of search-to-decision reductions or self-reductions for quantum MCSPs?

The hardness of average-case quantum MCSPs (which inputs are given randomly) is another interesting topic to explore. Hirahara [Hir18] showed that there is a worst-case to average-case reduction for the (gap version of) classical MCSP. We wonder if we can prove that quantum MCSPs have worst-case to average-case reductions.

Open Problem 8.6. Are there worst-case to average-case reductions for quantum MCSPs?

Note that there is negative evidence [BT06] showing that such classical reductions might not exist for NP-complete problems[16]. The existence of such reduction could result in important applications in cryptography, which we will discuss later.

[16]However, there is no evidence for the existence of quantum worst-case to average-case reductions for NP-complete since the analysis in [BT06] fails in the quantum setting. See [CHS20] for related discussion.

Finally, we can also try to prove the hardness of quantum MCSPs under stronger assumptions or more powerful reductions.

Open Problem 8.7. Assuming QETH or QSETH, is MQCSP, UMCSP, or SMCSP quantumly hard?

Open Problem 8.8. Does quantum reduction provide more power to show the hardness of MCSP? Specifically, is $\mathsf{NP} \subseteq \mathsf{BQP}^{\mathsf{MCSP}}$ or $\mathsf{NP} \subseteq \mathsf{BQP}^{\mathsf{MQCSP}}$?

8.1.3.2 Open problems: potential connections to other areas

In this chapter, in addition to generalizing several known connections for MCSP to quantum MCSPs, we have also discovered several connections which could be unique for quantum MCSPs. There are still many classically existing or unknown connections that we can explore. One fascinating question is whether we can base the security of one-way functions on any of these problems.

Open Problem 8.9. Can we base the security of cryptographic primitives on MQCSP, UMCSP, SMCSP, or some variants of these problems?

Note that since quantum MCSPs considered in this chapter are all worst-case problems, to answer Problem 8.9, we probably need worst-case to average-case reductions discussed in Problem 8.6. Moreover, Liu and Pass [LP20b] recently showed that the existence of classical one-way function is equivalent to the average-case hardness of a type of Kolmogorov complexity on uniform distribution. However, the average-case hardness of MCSP on uniform distribution is not known to imply one-wayness even classically, and the quantum version faces a similar obstacle. Very recently, Ilango, Ren, and Santhanam [IRS21] showed that the average-case hardness of Gap-MCSP on a locally samplable distribution is equivalent to the existence of one-way function. Liu and Pass [LP21] further generalized this result to show equivalence between the existence of one-way functions and the existence of sparse languages that are hard-on-average (including Kolmogorov complexity, k-SAT, and t-Clique). It is natural to ask whether their results can be generalized to quantum MCSPs. In addition to one-way functions, We are interested in connections between quantum MCSPs and "quantum-only" primitives, e.g., quantum $i\mathcal{O}$, copy protection, quantum process learning, etc.

Along this line, as many quantum problems have quantum inputs, it is natural to consider quantum MCSPs with quantum inputs. We have shown how SMCSP connects to problems in quantum cryptography, quantum gravity, and tomography given quantum states as inputs. This fact gives the possibility that MQCSP, UMCSP, and SMCSP with "succinct" quantum or classical inputs may have surprising connections to other problems in quantum computing. For instance, one can consider inputs which are quantum circuits that encode some objects (e.g., unitaries). Then, the problem is to find another significantly smaller circuit. In [CCCW21], Chakrabarti et al. have studied this problem and show applications to quantum supremacy.

8.2 Preliminaries

The basics of quantum computing are deferred to Appendix B. In this subsection, we will introduce some quantum complexity classes and non-uniform quantum circuit classes.

8.2.1 Quantum complexity classes

We introduce quantum complexity classes that are related to our study on the quantum MCSP. The classes we define in below are actually PromiseBQP and PromiseQCMA. To avoid abuse of notation, we just denote them as BQP and QCMA.

We first give the definition of the quantum analogue of BPP and P.

Definition 8.4 (BQP)**.** A promise problem $P = (P_Y, P_N)$ is in BQP if there exists a polynomial-time classical Turing Machine that on input 1^n for any $n \in \mathbb{N}$ outputs the description of a quantum circuit $\mathcal{C}_n$ with $\mathsf{poly}(n)$ gates and $\mathsf{poly}(n)$ ancilla qubits such that for $x \in \{0,1\}^n$ the following holds:

1. if $x \in P_Y$, $\Pr[M_1 \circ \mathcal{C}_n|x, 0^t\rangle = 1] \geq 2/3$;

2. if $x \in P_N$, $\Pr[M_1 \circ \mathcal{C}_n|x, 0^t\rangle = 1] \leq 1/3$,

where M_1 is the computational-basis measurement on the first qubit of the given state.

We also consider the quantum analogue of NP and MA in this chapter.

Definition 8.5 (QCMA)**.** A promise problem $P = (P_Y, P_N)$ is in QCMA if there exists a quantum polynomial-time (QPT) algorithm V such that

1. for $x \in P_Y$, there exists $w \in \{0,1\}^{\mathsf{poly}(n)}$ such that $\Pr[V(x,w) = 1] \geq 2/3$;
2. for $x \in P_N$, for all $w \in \{0,1\}^{\mathsf{poly}(n)}$, $\Pr[V(x,w) = 1] \leq 1/3$.

Another quantum analogue of MA and NP is called QMA. The difference between QMA and QCMA is that QMA allows the certificates to be quantum states. This difference makes $\mathsf{QCMA} \subseteq \mathsf{QMA}$[17].

We also consider the class RQP, which is the one-sided error version of BQP:

Definition 8.6 (RQP)**.** A promise problem $P = (P_Y, P_N)$ is in RQP if there exists a QPT algorithm $\mathcal{A}$ such that

1. for $x \in P_Y$, then $\Pr[\mathcal{A}(x) = 1] \geq \frac{1}{2}$;
2. for $x \in P_N$, then $\Pr[\mathcal{A}(x) = 1] = 0$.

8.2.2 Nonuniform quantum circuit complexity classes

With the mathematical background of quantum computing, we can define nonuniform quantum circuit complexity classes. We define the quantum analogues

[17]One may expect that the quantum certificate gives the malicious prover more power to cheat in the soundness case. However, it can be shown that the existence of such a cheating prover in QMA would also imply a cheating prover in QCMA by the convexity of quantum states.

of MCSP as promise problems. (We will justify the reason later in Section 8.3.) Therefore, we also define complexity classes for promise problems. A promise problem is defined as $P = \{P^n\}$, where $P^n = (P_Y^n, P_N^n)$ satisfying $P_Y^n \cap P_N^n = \emptyset$ and $P_Y^n \cup P_N^n \subseteq \{0,1\}^n$. We say a promise problem P is in some class C if there exists a language $L \in \mathsf{C}$ such that $P_Y \subseteq L$ and $P_N \subseteq \{0,1\}^* \backslash L$. In other words, for $x \in \{0,1\}^* \backslash P$, the answer could be arbitrary. Note that promise problems are naturally considered in quantum computing; for example, the local Hamiltonian problem [KSV02] (which is QMA-complete) and Identity check on basis states [WJB03] (which is QCMA-complete.)

Definition 8.7 ($\mathsf{BQC}(s,t,\mathcal{G})$)**.** Let $s,t : \mathbb{N} \to \mathbb{N}$ and $\mathcal{G}$ be a quantum gate set. $\mathsf{BQC}(s,t,\mathcal{G})$ is the set of promise problems $P = \{P^n : n > 0\}$ for which there exists a circuit family $\{\mathcal{C}_n : n > 0\} \in \mathsf{QC}(s,t,\mathcal{G})$ such that for $n > 0$, for any x where $|x| = n$,

- if $x \in P_Y^n$, then $\Pr[M_1 \circ \mathcal{C}_n|x, 0^t\rangle = 1] \geq 2/3$;
- if $x \in P_N^n$, $\Pr[M_1 \circ \mathcal{C}_n|x, 0^t\rangle = 1] \leq 1/3$.

Here, M_1 is the computational-basis measurement on the first qubit.

In the rest of the chapter, we will write $\mathsf{BQC}(s,t,\mathcal{G})$ as $\mathsf{BQC}(s)$ for simplicity if the number of ancilla qubits is at most $O(s)$.

In addition to BQC, we will also consider quantum complexity classes such as QMCA and BQP. For the same reason, the classes we consider are actually PromiseBQP and PromiseQCMA. To avoid abuse of notation, we just denote them as BQP and QCMA. Also, when NP is mentioned, we are actually considering PromiseNP. The formal definitions of these classes are given in Appendix 8.2.1.

8.3 Minimum Quantum Circuit Size Problems

We start off the quantum MCSP program by giving the definitions of various quantum analogs of the classical MCSP in Section 8.3.1 and investigating some basic complexity-theoretic results in Section 8.3.2 and Section 8.3.3.

8.3.1 Problem definitions

While classical computation works on Boolean strings, quantum computation works on unit complex vectors. Thus, there are multiple natural notions of MCSP that can be defined and studied in the quantum realm. But first let us formally define the classical MCSP as follows.

Definition 8.8 (Classical MCSP)**.** Let $n, s \in \mathbb{N}$[18]. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. The problem is, given the truth table $\mathsf{tt}(f)$ of f and the size parameter s in unary, decide if there exists a classical Boolean circuit C of size at most s such that $C(x) = f(x)$ for all $x \in \{0,1\}^n$.

Note that MCSP $\in$ NP because given a truth table $\mathsf{tt}(f)$ a circuit C, we can verify whether $C(x) = f(x)$ for all $x \in \{0,1\}^n$ in $\mathsf{poly}(|\mathsf{tt}(f)|, 1^s)$ time. On the other hand, when $s = \Omega(n)$, the number of circuits of size at most s is $2^{\Theta(s \log s)}$, which is $2^{\omega(n)}$ by the counting argument. Besides, for every Boolean function, there exists a circuit with size at most $O(2^n/n)$ [Lup58]; therefore, we can suppose the $s = O(2^n/n)$, which implies that brute-force search takes $2^{O(2^n)}$ time to solve MCSP in the worst case and it is the best known algorithm for MCSP.

As quantum computation is generally believed to be more powerful than classical computation, it is likely that the quantum circuit complexities for some Boolean functions are much different from their classical circuit complexities. Specifically, quantum circuits can create quantum entanglement between qubits that cannot be simulated classically. Therefore, we define the following problem for studying the quantum circuit complexity of the given Boolean function.

[18]For every Boolean function, there is a circuit with size at most $O(2^n/n)$. Therefore, one can suppose s is at most $O(2^n/n)$. Besides, one can also consider s is given in unary, such that the problem is still well-defined in the sense that it is trivially in NP.

Definition 8.9 ($\mathsf{MQCSP}_{\alpha,\beta}$)**.** Fix a universal gate set $\mathcal{G}$. Let $n, s, t \in \mathbb{N}$ and $t \leq s$. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. Let $\alpha, \beta \in (1/2, 1)$ such that $\alpha - \beta \geq \frac{1}{\mathsf{poly}(2^n)}$. MQCSP is a promise problem defined as follows.

- **Inputs:** the truth table $\mathsf{tt}(f)$ of f, the size parameter s in unary representation, and the ancilla parameter t.
- **Yes instance:** there exists a quantum circuit $\mathcal{C}$ using at most s gates and operating on at most $n + t$ qubits such that for all $x \in \{0,1\}^n$, $\|(\langle f(x)| \otimes I_{n+t-1})\mathcal{C}|x, 0^t\rangle\|^2 \geq \alpha$.
- **No instance:** for every quantum circuit $\mathcal{C}$ using at most s gates and operating on at most $n + t$ qubits, there exists $x \in \{0,1\}^n$ such that $\|(\langle f(x)| \otimes I_{n+t-1})\mathcal{C}|x, 0^t\rangle\|^2 \leq \beta$.

With the promise that the input must be either a yes instance or a no instance, the problem is to decide whether the input is a yes instance or not.

Remark 8.1. Here, we set the thresholds for the yes and no instances to be α, β such that $1/2 < \beta < \alpha < 1$ and $\alpha - \beta > \frac{1}{\mathsf{poly}(2^n)}$. We require α and β to be greater than $1/2$ because a quantum circuit that outputs a uniformly random bit (e.g., measure $|+\rangle$ in the computational basis) can compute $f(x)$ with $1/2$ probability for all x. For simplicity, in the rest of the work, we will ignore the subscription α, β and will specify them when it is necessary.

For MQCSP, which gate set $\mathcal{G}$ is used is another important parameter to be considered. One may ask if circuit complexity can significantly change when considering different $\mathcal{G}$. Fortunately, according to the Solovay-Kitaev Theorem in Theorem B.1, we can conclude that any s-gate circuit using gates from $\mathcal{G}$ can be ϵ-approximated by an $(s \cdot \mathsf{polylog}\,\frac{s}{\epsilon})$-gate circuit from another universal gate set. Hence, the circuit complexity only modestly changes when considering different gate sets.

Claim 8.11. *Fix two universal gate sets $\mathcal{G}$ and $\mathcal{G}'$. Suppose that there exists a s-gate circuit $\mathcal{C}$ that uses gates from $\mathcal{G}$ such that for all x, $\|(\langle f(x)| \otimes I_{n+t-1})\mathcal{C}|x, 0^t\rangle\| \geq$*

$1-\delta$. *Then, there exists another circuit* $\mathcal{C}'$ *that uses* $s \cdot \mathsf{polylog}\, \frac{s}{\epsilon}$ *gates in* $\mathcal{G}'$ *such that* $\|(\langle f(x)| \otimes I_{n+t-1})\mathcal{C}|x, 0^t\rangle\| \geq 1 - \delta - \epsilon^2/2$.

Proof. The proof simply follows from the Solovay-Kitaev Theorem in Theorem B.1. The only subtlety is that the distance measure in Theorem B.1 is L_2 norm distance. However, for any two states $|\psi\rangle$ and $|\phi\rangle$, we have $|\langle\psi|\phi\rangle| \geq 1 - \frac{1}{2}\||\psi\rangle - |\phi\rangle\|^2$. Thus, we can obtain the lower bound for $\|(\langle f(x)| \otimes I_{n+t-1})\mathcal{C}|x, 0^t\rangle\|$ by using the L_2 norm between $\mathcal{C}$ and $\mathcal{C}'$. $\square$

In this chapter, we mainly focus on arbitrary gate sets containing one- and two-gates and $|\mathcal{G}| = O(1)$. However, for some applications, we may require a particular gate set such as $\{\mathsf{Toffoli}, \mathsf{H}\}$. We will specify $\mathcal{G}$ when it is necessary. We assume $t \leq s$ without loss of generality since we mainly consider the gate set $\mathcal{G}$ to have one- and two-qubit gates. Specifically, if there are more than s ancilla qubits, there must be ancilla qubits that are not used by any gate.

We define the problem as a promise problem for two reasons: first, applying measurements on quantum states generally gives probabilistic outputs. Similar to many probabilistic algorithms, we say a quantum algorithm solves a problem if it outputs the answer with high probability in general. Check the definition of BQP for an example. Along this line, we expect a quantum circuit $\mathcal{C}$ to implement the given Boolean function f with high probability, i.e., for each input x, the circuit outputs $f(x)$ with high probability. The second reason is about verifying the circuit. Consider the case where $\mathcal{C}$ only fails on one x with success probability $2/3-\epsilon$, where ϵ is some extremely small number. In this case, it is hard to verify the circuit efficiently. Therefore, we require a gap for efficient verification and say that $\mathcal{C}$ does not implement f if it can only output $f(x)$ with probability with small probability for some x.

Other variants. In many applications, the *gap-version* of MCSP is much easier and more flexible to work with. Below we define the gap-version of MQCSP and the multi-output MQCSP.

Definition 8.10 ($\mathsf{MQCSP}_{a,b}[s, s', t]$)**.** Let $n, s, s', t \in \mathbb{N}$ such that $t \leq s < s' \leq 2^{O(n)}$. Let $a - b \geq 1/\mathsf{poly}(2^n, 1^{|s|})$. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. $\mathsf{MQCSP}[s, s']$ is a promise problem defined as follows.

- Input: the truth table $\mathsf{tt}(f)$ of f, the size parameter s in unary, and the ancilla parameter t.
- Yes instance: there exists a quantum circuit $\mathcal{C}$ using at most s gates and operating on at most $n + t$ qubits such that for all $x \in \{0,1\}^n$,

$$\|(\langle f(x)| \otimes I_{n+t-1})\mathcal{C}|x, 0^t\rangle\|^2 \geq \frac{2}{3}.$$

- No instance: for every quantum circuit $\mathcal{C}$ using at most s' gates and operating on at most $n + t$ qubits, there exists $x \in \{0,1\}^n$ such that

$$\|(\langle f(x)| \otimes I_{n+t-1})\mathcal{C}|x, 0^t\rangle\|^2 \leq \frac{1}{2}.$$

With the promise that the input must be either a yes instance or a no instance, the problem is to decide whether the input is a yes instance or not.

When it is clear from the context, we may use $\mathsf{MQCSP}^\star$ to denote $\mathsf{MQCSP}_{a,b}[s, s', t]$.

Definition 8.11 ($\mathcal{G}$-$\mathsf{multiMQCSP}_{\alpha,\beta}(s, t)$)**.** Let m, s, t be functions of n such that $t \leq s \leq 2^{o(n)}$ and $m \leq n + t$. Let $\alpha, \beta \in [2^{-m}, 1]$ such that $\alpha - \beta > \frac{1}{\mathsf{poly}(2^n)}$. Let $f : \{0,1\}^n \to \{0,1\}^m$ be a multioutput function. $\mathcal{G}-\mathsf{multiMQCSP}_{\alpha,\beta}(s, t)$ is a promise problem that

1. Input: the truth table $\mathsf{tt}(f)$ of f.
2. Yes instance: there exists a quantum circuit $\mathcal{C}$ using at most s gates from $\mathcal{G}$ and operating on at most $n + t$ qubits such that for all $x \in \{0,1\}^n$,

$$\|(\langle f(x)| \otimes I_{n+t-m})\mathcal{C}|x, 0^t\rangle\|^2 \geq \alpha,$$

3. No instance: for any quantum circuit $\mathcal{C}$ using at most s gates from $\mathcal{G}$ and operating on at most $n+t$ qubits, there exists $x \in \{0,1\}^n$ such that

$$\|(\langle f(x)| \otimes I_{n+t-m})\mathcal{C}|x, 0^t\rangle\|^2 \leq \beta.$$

With the promise that the input must be either a yes instance or a no instance, the problem is to decide whether the input is a yes instance or not.

Natural property. It is worth noting that we can view an efficient quantum algorithm for MQCSP as *quantum natural property against quantum circuit classes*. Natural properties against circuit classes were first defined by Razborov and Rudich [RR97], and recently, Arunachalam et al. [AGG+20] further considered quantum natural properties against circuit classes.

Definition 8.12 (Natural Property [RR97])**.** Let C be a uniform complexity class and C' be a circuit class. We say that a property $\Gamma = \{\Gamma_n : n \in \mathbb{N}\}$ is C-natural against C' if the following holds.

1. **Constructivity:** for all $L \in \Gamma$, $L \in C$.
2. **Largeness:** There exists $n_0 \in \mathbb{N}$, for $n \geq n_0$, $|\Gamma_n|/|\mathcal{F}_n| \geq \frac{1}{2}$, where $\mathcal{F}_n$ is the set of all Boolean functions with input length n.
3. **Usefulness:** There exists $n_0 \in \mathbb{N}$, for $n \geq n_0$, $\Gamma_n \cap C'_n = \emptyset$, where C'_n is the set of circuits in C' on n (qu)bits.

Note that an MQCSP oracle can be used to construct natural properties against quantum circuit classes BQC$[s]$ for any s. Therefore, if we suppose that MQCSP is in BQP, then we can have properties that are BQP-natural against quantum circuit classes. For simplicity, we call properties that are BQP-natural as *quantum natural properties*. Arunachalam et al. [AGG+20] first considered quantum natural properties against circuit classes, and proved circuit lower bounds for quantum complexity classes. Our work can also be viewed as a study of quantum natural properties against quantum circuit classes. The formal definition of BQP-natural property is in below:

Definition 8.13 (BQP-Natural Property [AGG+20])**.** We say that a combinatorial property Γ is C-natural against polynomial-size quantum circuits ($\mathsf{BQC}[\mathsf{poly}]$) if the following holds.

1. **Constructivity:** for any string $L \in \Gamma$, L can be accepted by a BQP algorithm.
2. **Largeness:** There exists $n_0 \in \mathbb{N}$, for $n \geq n_0$, $\frac{|\Gamma_n|}{|\mathcal{F}_n|} \geq \frac{1}{2}$.
3. **Usefulness:** There exists $n_0 \in \mathbb{N}$, for $n \geq n_0$, any string accepted by $\mathsf{BQC}[\mathsf{poly}]$ is not in Γ_n.

Then, our observation on the connection between MQCSP and quantum natural property is formally stated as follows:

Observation 8.12. *If* $\mathsf{MQCSP} \in \mathsf{BQP}$*, then there exists a* BQP*-natural property against quantum circuits* $\mathsf{QC}[n^k]$ *for any* $k \in \mathbb{N}_+$*.*

8.3.2 Upper bounds for MQCSP

It turns out that, unlike the classical MCSP, MQCSP is not trivially in NP. The best upper bound we are able to get for MQCSP is QCMA, the quantum analogue of NP (or MA). Before showing that MQCSP is in QCMA, we first discuss why it is not trivially in NP like the classical MCSP. One obvious reason is that MQCSP is a promise problem. Therefore, we consider PromiseNP, which definition is the same as NP except that PromiseNP relax the definition of NP to contain promise problems that have NP certificates. For the ease of presentation, we will use NP for both NP and PromiseNP. Then, when the number of ancilla qubits is linear, one can verify the given circuit by simply writing down the corresponding unitary.

Theorem 8.13. MQCSP *is in* NP *when only a linear number of ancilla qubits are allowed.*

However, when the number of ancilla qubits is superlinear, e.g., n^2, the quantum circuit $\mathcal{C}$ operates on $2^{O(n^2)}$ qubits, and thus the corresponding unitary $U_{\mathcal{C}}$ has dimension $2^{O(n^2)}$ which is superpolynomial in 2^n. In this case, the verifier cannot compute $U_{\mathcal{C}}$ classically in time $\mathsf{poly}(2^n)$. Therefore, the trivial approach does not work.

Note that although the trivial approach fails to show that MQCSP is in NP, it does not rule out the possibility that MQCSP can be efficiently verified via other approaches. In the following theorem, we show that a quantum verifier can efficiently verify the given quantum circuit, and thus MQCSP is in QCMA.

Theorem 8.14. MQCSP $\in$ QCMA.

We leave the proof to Appendix 8.6 for completeness.

8.3.3 Hardness of quantum MCSP

It is a major open problem in complexity theory to understand the hardness of classical MCSP. Here, we show that the state-of-the-art hardness results on MCSP (and its variants) can be extended to MQCSP. We remark that this is actually not straightforward to see because the classical MCSP is incomparable with MQCSP.

First, we show that the SZK-hardness result of MCSP by Allender and Das [AD14a] can be extended to MQCSP. Here, SZK stands for the complexity class *Statistical Zero Knowledge* that lies between P and NP. We first define SZK and the statistical distance as follows.

Definition 8.14 (Statistical Distance $SD(X, Y)$)**.** Let X and Y be two probability distributions, the statistical distance between X and Y can be defined as follows:

$$\max_{S \subseteq \{0,1\}^{m'}} |\Pr[X \in S] - \Pr[Y \in S]|$$

Definition 8.15 (SZK)**.** A promise problem $P = (P_Y, P_N)$ is in SZK if there exists a PPT verifier V and an interactive proof system (P, V) satisfying the following properties:

1. **Completeness:** For $x \in P_Y$, there exists P such that $\Pr[\langle P, V\rangle(x) = 1] \geq \frac{2}{3}$.

2. **Soundness:** For $x \in P_N$, for all P, $\Pr[\langle P, V\rangle(x) = 1] \leq \frac{1}{3}$.

3. **Statistical zero-knowledge:** There exists a PPT simulator S, for all PPT verifier V^*, for all $x \in P_Y$,

$$\mathsf{SD}(S(V^*)(x), \langle P, V^*\rangle(x)) \leq \mathsf{negl}(n).$$

We introduce an SZK-complete problem by Ben-Or and Gutfreund [BOG08].

Definition 8.16 (Polarized Image Intersection Density (PIID), [BOG08])**.** Given two circuits $C_0, C_1 : \{0,1\}^m \to \{0,1\}^{m'}$ of size n^k with the promise that either

1. $\max_{S\subseteq\{0,1\}^{m'}} |\Pr_x[C_0(x) \in S] - \Pr_x[C_1(x) \in S]| \leq \frac{1}{2^n}$, or

2. $\Pr_{x\in\{0,1\}^{m'}}[\exists y \in \mathsf{Im}(C_0) \text{ such that } C_1(x) = y] \leq \frac{1}{2^n}$,

where $n = \mathsf{poly}(m)$ and $\mathsf{Im}(C) := \{C(x) : x \in \{0,1\}^m\}$. The problem is to decide which case is true.

Theorem 8.15. $\mathsf{SZK} \subseteq \mathsf{BPP}^{\mathsf{MQCSP}}$

To prove Theorem 8.15, we first observe that the existence of small classical circuit implies the existence small quantum circuits and an MQCSP oracle can invert one-way functions (which we will prove in Section 8.4.1.1). Then, we can show that PIID is in $\mathsf{BPP}^{\mathsf{MQCSP}}$ following the framework of [AD14a]. We leave the proof to Appendix 8.6 for completeness.

Next, we quantize the recent breakthrough of Ilango et al. [ILO20] on the NP-hardness of classical MCSP. There are two main differences between the classical and quantum settings: (i) the circuit model is different and hence makes the combinatorics different, and (ii) the quantum setting allows the output to have some errors. We partially overcome these two difficulties and prove the following theorem.

Theorem 8.16. *Suppose* $\mathsf{CNOT} \circ (\mathsf{I} \otimes \mathsf{X}), \mathsf{Toffoli} \in \mathcal{G}$. *Every multi-bit gate in* $\mathcal{G}$ *behaves classically on classical inputs and has at most 1 target wire and at most 2 control wire. (That is, except 1 wire, the outputs of the other wires, at most 2, are the same as their corresponding classical inputs. For example,* CNOT *gate.) Then* $\mathcal{G}$-$\mathsf{multiMQCSP}$ *is* NP*-hard under randomized reduction.*

$\mathsf{CNOT} \circ (\mathsf{I} \otimes \mathsf{X})$ is the following operation on two input wires, denoted as control wire and target wire: first do a X on the target wire, and do a CNOT from the control wire to the target wire. We consider it as a single gate, as the analog of the classical NOT gate.

Here the choice of gate set matter: we need the quantum gate set to contain the analog of the usual classical gate set. $\mathsf{CNOT} \circ (\mathsf{I} \otimes \mathsf{X})$ is the analog of classical single-bit NOT operation, and $\mathsf{Toffoli}$ is the analog of classical AND operation. Here the correspondence has two properties: (1) if the target wire is in the zero state and the control wire is classical, the output of the target wire will be the corresponding classical logical computation result; (2) if the input of the control wire is classical, the output of the control wire will remain the same. Since in the quantum world data copy is not for free, the second property is important for deriving our result.

The proof follows the outline of the proof in [ILO20]. We note there are two differences during the proof in the quantum case compared to the classical case:

- The circuit model is different. In the classical world the gates are single-output and we assume free-copy. And the basic gate set contains AND, OR, NOT gates. In quantum world, data copy is not for free and we need to use the $\mathsf{Toffoli}$ gate to implement the AND/OR gates.

  Remark 8.2. One idea might be to use the Solovay-Kitaev theorem to switch the gate set and make the theorem general. But this does not work here in an immediate way. Our proof does not imply the problem is also NP-hard to approximate multiplicatively. On the other hand, the classical result [ILO20] is not known to be general on different gate set either.

- In the definition of multi-output minimum quantum circuit size problem, we allow the output to have some errors, which is not considered in the classical world.

Proof of Theorem 8.16. We consider the same construction as [ILO20]. Let's restate it here for completeness.

1. Choose a large enough constant r so that 20-approximating r-bounded set cover problem is NP-hard. Consider an instance $(1^n, \mathcal{S})$ of this problem.

2. m is the least power of 2 that is greater than n^3. Sample the truth table T representing a function on $\{0,1\}^{\log m} \to \{0,1\}$ uniformly at random. Construct $g := \bullet_{S\in\mathcal{S}}\text{Eval-DNF}_{T_{\langle S^m\rangle}}$ where:

   - To define DNF_f that encode the truth table f, we first repeat the construction in [ILO20] for completeness:

     $$\text{DNF}_f := ((x_1 = y_1^1) \wedge \cdots \wedge (x_n = y_n^1)) \vee \cdots \vee ((x_1 = y_1^t) \wedge \cdots \wedge (x_n = y_n^t))$$

     where $y^1, \cdots y^t$ are YES inputs of f in lexicographical order, $x_1, \cdots x_n$ index the bits of the input string x, $y_1^j, \cdots y_n^j$ index the bits of y^j, and $(x_i = y_i^t)$ denotes $(x_i \oplus (1 \oplus y_i^j))$.

     We use the same construction with one difference: here $\vee$ is further decomposed to $\neg$ and $\wedge$.

   - $T_{\langle S\rangle}$ is the truth table that is equal to T for input in S and 0 everywhere else.

   - $S^m := \cup_{i\in S} P_i^{m,n}$ where $P_i^{m,n} := \{j \in [m] : j \equiv i \mod n\}$. This step closes the gap between $[m]$ (the MCSP size) and $[n]$ (the set cover size).

   - "$\bullet$" is used on two functions that have the same input domain, and it concatenates the outputs of these functions to get a new function.

  - To define Eval-C, we first consider $x_1 \bullet x_2 \bullet \cdots x_n \bullet g_1 \bullet g_2 \bullet \cdots g_s$ where $g_1, \cdots g_s$ are the output of each gate in circuit C. Then we remove the gate output that are the same on all the inputs.

3. As in [ILO20], define k as the number of distinct components of g that are not directly a function identical to an input. Note that this can be efficiently computed.

   Take $\alpha = 1, \beta = 0.99, t = 10s$ (s is the output number of our construction).

   Define $CC_{\alpha,\beta}(t, \mathsf{tt}(f))$ as the subroutine that uses binary search to find the minimum s such that $\mathcal{G}-\mathsf{multiMQCSP}_{\alpha,\beta}(s,t)(\mathsf{tt}(f)) = \text{true}$.[19] Use the $\mathsf{multiMQCSP}$ oracle and compute

$$\Delta := CC_{\alpha,\beta}(t, \mathsf{tt}(T \bullet g)) - k$$

   as the approximation of the set cover instance $(1^n, \mathcal{S})$.

To analyze this reduction, we need to prove the followings steps:

1. $CC_{\alpha,\beta}(t, \mathsf{tt}(g)) = k$

2. $\Delta \leq 3 \cdot \text{cover}([n], \mathcal{S}) + 1$ where $\text{cover}([n], \mathcal{S})$ is the size of the minimum set cover solution for $\mathcal{S}$.

3. $\Delta \geq \text{cover}([n], \mathcal{S})/6 - 6$ with probability $1 - 2^{-\Omega(m)}$.

Then we get an approximation to the set cover problem.

Let us prove the three statements step-by-step.

[19]Since multiMQCSP is a promise problem this routine does not necesarrily find the minimum s but should return a value that there exists a circuit of this size that approximate the function everywhere with correct probability β. This is sufficient for later proof.

Step 1: The $\leq$ part is proved by the function construction itself. We implement $\neg$ with the $\mathsf{CNOT} \circ (\mathsf{I} \otimes \mathsf{X})$ gate (and write the output on an empty ancilla system) and implement $\wedge$ with the $\mathsf{Toffoli}$ gate.

The $\geq$ part is slightly different since in quantum case the gate model is different. In classical world all the gates are single-output, while in quantum world there are multi-output gates. However, for the multi-output gates like CNOT and $\mathsf{Toffoli}$, there is only one target wire, and the other wires are control wire. Thus for each output component, we can always find the nearest gate that does not use it as a control wire (if there is such a gate along the way, ignore it). In this way each different output component corresponds to a different gate in the circuit, which completes the proof.

Step 2: As [ILO20], when $\text{cover}([n], \mathcal{S}) = \ell$, without loss of generality assume $S_1, \cdots S_\ell$ are a set cover. Then $T = T_{\langle S_1^m \rangle} \vee \cdots \vee T_{\langle S_\ell^m \rangle}$. This can be computed using $3\ell + 1$ extra gates on the minimum circuit of Eval-g. (Note that in the quantum world we need slightly more gates than the classical world. And we need to evaluate the OR gate by NOT-AND-NOT gates to get T.)

Step 3: Denote $\ell = \lfloor \text{cover}([n], \mathcal{S})/6 \rfloor$. The goal is to show that the probability that $\Delta \leq \ell$ is small by showing that T satisfying $\Delta \leq \ell$ must have a short description. Suppose T is a truth table such that the condition $\Delta > \ell$ does not hold. We need to find a circuit of gate number $\leq 2\ell$ where:

- The inputs are: the bits of x; and the output of g.
- It encodes the output of T.

We use the similar idea to [ILO20] but we need to address the two problems discussed before this proof.

As what we did in Step 1, we can associate each output component ($g_i(x)$, for example) to a unique gate in the circuit. As [ILO20], we remove these gates from the circuit. There might be some gates between this gate and the output $g_i(x)$ that use the wire as control wires. For these gates, simply use $g_i(x)$ as the control value.

As in [ILO20] we have $CC_{\alpha,\beta}(t, \mathsf{tt}(T \bullet g)) \leq \ell + k$. And since for each g_i at least one gate is removed, the remaining circuit is a circuit D that takes $\log(m) + k$ inputs and has at most ℓ gates such that

$$D(x, g_1(x), \cdots g_k(x)) \text{ encodes } T(x)$$

Then since each gate has fan-in at most 3 the circuit uses at most 3ℓ components of g. Then after a possible relabling of $g_1 \cdots g_k$ we can assume D takes $\log(m) + 3\ell$ inputs such that

$$D(x, g_1(x), \cdots g_{3\ell}(x)) \text{ encodes } T(x)$$

The new circuit does not necessarily behave the same as the original circuit, but they do behave the same (up to a global phase) on the subspace that all the outputs are computed correctly. By the definition of $\mathsf{multiMQCSP}$ and the choices of parameters this is true with norm ≥ 0.99. Thus we can view the shrinked circuit as an encoding of T by focusing on the most-possible outputs of this circuit. Then by the same argument as [ILO20] such a shrinked circuit has a description of $(1 - \Omega(1))m$ bits, which implies such T has at most $2^{(1-\Omega(1))m}$ choices thus a random T falls into this case with exponentially small probability. $\square$

However, we don't know whether this problem is NP-complete, since it's not known to be in NP. With a proof similar to that of Theorem 8.14, we only know $\mathsf{multiMQCSP} \in \mathsf{QCMA}$. Namely, there remains a gap between our understandings of the upper bound and hardness of $\mathsf{multiMQCSP}$. We pose it as an open problem to settle the complexity of $\mathsf{multiMQCSP}$.

8.4 Connections Between MQCSP and Other Problems

8.4.1 Cryptography and MQCSP

Classically, we have already known connections between MCSP and one-way functions [KC00, RR97] and indistinguishable obfuscation [IKV18]. In this section, we show the quantum analogies of these results.

8.4.1.1 Quantum cryptographic primitives

We first introduce relevant primitives in cryptography.

Definition 8.17 (Pseudorandom Generator (PRG))**.** Let $G : \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time computable function. Let $\ell : \mathbb{N} \to \mathbb{N}$ be a polynomial-time computable function such that $\ell(n) > n$ for all n. G is a pseudorandom generator of stretch $\ell(n)$ if it satisfies:

1. $|G(x)| = \ell(|x|)$ for all $x \in \{0,1\}^*$, and
2. for all Probabilistic polynomial-time (PPT) algorithm $\mathcal{A}$, there exists a negligible function $\epsilon : \mathbb{N} \to [0,1]$ such that for all $n \in \mathbb{N}$

$$\left| \Pr_{x \sim \{0,1\}^n}[\mathcal{A}(G(x)) = 1] - \Pr_{y \sim \{0,1\}^{\ell(n)}}[\mathcal{A}(y) = 1] \right| \leq \epsilon(n).$$

We say that a PRG is *local* if every output bit of the PRG can be computed in time $\mathsf{poly}(n)$. In the following, we define PRG secure against any quantum polynomial-time adversary.

Definition 8.18 (Quantum-Secure Pseudorandom Generator (qPRG))**.** Let $G : \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time computable function[20]. Let $\ell : \mathbb{N} \to \mathbb{N}$ be a polynomial-time computable function such that $\ell(n) > n$ for all n. G is a pseudorandom generator secure against quantum adversaries of stretch $\ell(n)$ if it satisfies:

[20]It is worth noting that G can be any function that is efficiently computable in either quantum or classical polynomial time.

1. $|G(x)| = \ell(|x|)$ for all $x \in \{0,1\}^*$, and

2. for all quantum polynomial-time (QPT) algorithm $\mathcal{A}$, there exists a negligible function $\epsilon : \mathbb{N} \to [0,1]$ such that for all $n \in \mathbb{N}$

$$\left| \Pr_{x \sim \{0,1\}^n}[\mathcal{A}(G(x)) = 1] - \Pr_{y \sim \{0,1\}^{\ell(n)}}[\mathcal{A}(y) = 1] \right| \leq \epsilon(n).$$

In this chapter, we consider two ways of constructing quantum-secure PRGs based on different cryptographic primitives. One is based on the quantum-secure one-way functions and the other one is based on the hard function.

Definition 8.19 (Quantum-Secure One-Way function (qOWF))**.** A function $f : \{0,1\}^* \to \{0,1\}^*$ is a quantum-secure one-way function, if the following conditions hold: For every $n \in \mathbb{N}$, for any $x \in \{0,1\}^n$ picked uniformly at random,

1. There exists a $\mathsf{poly}(n)$-time deterministic algorithm for computing f.

2. For any $\mathsf{poly}(n)$-time quantum algorithm $\mathcal{A}'$, $\Pr_x[\mathcal{A}'(f(x)) \in f^{-1}(f(x))] = \mathsf{negl}(n)$.

Definition 8.20 (GGM Construction [GGM86])**.** Let $G : \{0,1\}^n \to \{0,1\}^{2n}$ be a (q)PRG. For every $z \in \{0,1\}^m$, the GGM construction of a pseudorandom function family $\{h_z : \{0,1\}^n \to \{0,1\}^n\}_{z \in \{0,1\}^m}$ is defined as follows:

$$f_z(x) = G_{z_m} \circ G_{z_{m-1}} \circ \cdots \circ G_{z_1}(x),$$

where we denote by $G_0(x)$ the first n bits of G, and by $G_1(x)$ the last n qubits.

Lemma 8.17 ([HILL99])**.** *If* OWF*s exist, then for every $c \in \mathbb{N}$, there exists a secure* PRG *with stretch $\ell(n) = n^c$.*

Since the security proof of Lemma 8.17 is black-box, the analysis carries over to the quantum setting directly if the one-way function is secure against quantum adversaries. Therefore, we can obtain Lemma 8.18.

Lemma 8.18 (Folklore)**.** *If* qOWF*s exist, then for every* $c \in \mathbb{N}$*, there exist* qPRG*s with stretch* $\ell(n) = n^c$*.*

Lemma 8.19. *Suppose that there exists a* qPRG $G : \{0,1\}^n \to \{0,1\}^{2n}$*. Then, for* $m = O(\log n)$*, there exists a local* qPRG $\hat{G} : \{0,1\}^n \to \{0,1\}^{2^m}$*.*

Proof. We first give the construction of $\hat{G}$. Follow the GGM construction in Definition 8.20, we let

$$h'_x(z) = G_{z_m} \circ G_{z_{m-1}} \circ \cdots \circ G_{z_1}(x)$$

where $z \in \{0,1\}^m$, $x \in \{0,1\}^n$. We let $h_x(z)$ be the first output bit of $h'_x(z)$ and define the qPRG as

$$\hat{G}(x) = h_x(0) \mid h_x(1) \mid \cdots \mid h_x(2^m - 1).$$

It is obvious that each bit of $\hat{G}(x)$ can be computed in time m times the runtime of G.

We then prove that $\hat{G}(x)$ is indistinguishable from a truly random string by the standard hybrid approach. For $i \in [m]$, we define

$$H^i(z) = (G_{z_m} \circ G_{z_{m-1}} \circ \cdots \circ G_{z_i}(y_{z,i}))_1,$$

where $y_{z,i}$ is drawn independently and uniformly randomly from $\{0,1\}^n$. Note that $H^1(z) = h_z(x)$ and $H^m(z)$ is a random bit. Let

$$\hat{G}^i = H^i(0) \mid H^i(1) \mid \cdots \mid H^i(2^m - 1) \quad \forall i \in [m].$$

Suppose that there exists a QPT algorithm $\mathcal{A}$ such that

$$\left| \Pr_{x \sim \{0,1\}^n}[\mathcal{A}(\hat{G}(x)) = 1] - \Pr_{u \sim \{0,1\}^{2^m}}(\mathcal{A}(u)) \right| \geq 1/\mathsf{poly}(n).$$

Then, by the triangular inequality,

$$\sum_{i=1}^{m-1} \left| \Pr[\mathcal{A}(\hat{G}^i) = 1] - \Pr[\mathcal{A}(\hat{G}^{i+1}) = 1] \right| \geq 1/\mathsf{poly}(n)$$

which implies that there exists i^* such that $|\Pr[\mathcal{A}(\hat{G}^{i^*}) = 1] - \Pr[\mathcal{A}(\hat{G}^{i^*+1}) = 1]| \geq 1/\mathsf{poly}(n)$. Since distinguishing $\hat{G}^{i^*}$ and $\hat{G}^{i^*+1}$ implies that one can distinguish $G(x)$ from a random string, G is not a qPRG. This completes the proof.

□

8.4.1.2 Implications for quantum-secure one-way functions (qOWF)

Here, we show a quantum analogous result for [KC00, RR97] by considering the implication of the existence of efficient quantum algorithms for either classical or quantum MCSP.

Theorem 8.20. *If* $\mathsf{MQCSP} \in \mathsf{BQP}$*, then there is no quantum-secure one-way function (*qOWF*).*

Proof. Let $f : \{0,1\}^* \to \{0,1\}^*$ be any function. By Lemma 8.18, we construct $G_f : \{0,1\}^n \to \{0,1\}^{n^a}$ that is a qPRG if f is a qOWF. We denote the runtime for G_f as $O(n^b)$ for some constant b.

Given G_f, we construct a qPRG $\hat{G} : \{0,1\}^n \to \{0,1\}^{2^m}$ where $m = O(\log n)$ by Lemma 8.19. Then, we view the outputs of $\hat{G}(x)$ as a truth table of some Boolean function $g_x : \{0,1\}^m \to \{0,1\}$. Note that according to the construction in Lemma 8.19, the time for evaluating g_x on $z \in \{0,1\}^m$ is $O(m \cdot n^b) = \tilde{O}(n^b)$. On the other hand, for a random Boolean function from $\{0,1\}^m$ to $\{0,1\}$, we know from Claim 8.74 that its circuit complexity is greater than $\frac{2^m}{(c+1)m}$ with high probability. Therefore, by setting $m = d \log n$ for some constant $d \gg b$, the circuit complexity of the random function is $\tilde{O}(n^d) \gg \tilde{O}(n^b)$ with high probability.

Algorithm 30 A quantum algorithm for breaking qPRG

```
Input: Given $\mathsf{tt}(h)$ for $h : \{0,1\}^m \to \{0,1\}$ constructed from $\hat{G}$ in Lemma 8.19.
1: Runs the quantum algorithm for $\mathsf{MQCSP}$ with $s = \frac{2^m}{(c+1)m}$
2: return "Yes" if the algorithm in previous step outputs yes.
3: return "No", otherwise
```

Since we assume $\mathsf{MQCSP} \in \mathsf{BQP}$, we obtain a quantum polynomial-time algorithm $\mathcal{A}$ for distinguishing $\{g_x\}_{x\in\{0,1\}^n}$ and the random function family $\mathcal{F}_m$ as in Algorithm 30. The circuit complexity for g_x is at most $\tilde{O}(n^b)$ and the for a random function h is greater than $\frac{2^m}{(c+1)m} = \tilde{O}(n^d)$ for $d \gg b$. thus, we obtain

$$\left| \Pr_{x\sim\{0,1\}^n}[\mathcal{A}(\mathsf{tt}(g_x)) = 1] - \Pr_{h\sim\mathcal{F}_m}[\mathcal{A}(\mathsf{tt}(h)) = 1] \right| \geq 1/\,\mathsf{poly}(n).$$

This implies that we can use $\mathcal{A}$ to break G in quantum polynomial time by Lemma 8.19. Finally, by Lemma 8.18, we obtain a quantum polynomial-time algorithm $\mathcal{A}_{inv}$ for inverting any f. □

8.4.1.3 Implication for quantum-secure $i\mathcal{O}$

In this section, we use Theorem 8.20 and quantum-secure $i\mathcal{O}$ to show that if MQCSP can be solved by a BQP algorithm, then $\mathsf{NP} \subset \mathsf{coRQP}$, which is the class of one-sided error quantum polynomial-time algorithms such that a "Yes" instance will always be accepted while a "NO" instance will be rejected with high probability.

We define the quantum-secure $i\mathcal{O}$ as follows:

Definition 8.21 (Quantum-secure indistinguishability obfuscation, $i\mathcal{O}$)**.** A probabilistic polynomial-time machine iO is an indistinguishability obfuscator for a circuit class $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ if the following conditions are satisfied for all $\lambda \in \mathbb{N}$:

- **Functionality:** For any $C \in \mathcal{C}_\lambda$, for all inputs x, $i\mathcal{O}(C)(x) = C(x)$.
- **Indistinguishability:** For any $C_1, C_2 \in \mathcal{C}_\lambda$ such that $|C_1| = |C_2|$ and $C_1(x) = C_2(x)$ for all inputs x, any quantum polynomial-time distinguisher $\mathcal{A}$ cannot distinguish the distributions $i\mathcal{O}(C_1)$ and $i\mathcal{O}(C_2)$ with noticeable probability, i.e., $\left|\Pr[\mathcal{A}(i\mathcal{O}(C_1)) = 1] - \Pr[\mathcal{A}(i\mathcal{O}(C_2)) = 1]\right| \leq \mathsf{negl}(\lambda)$.

Remark 8.3. We note that there are some (candidate) constructions of post-quantum $i\mathcal{O}$, based on different assumptions. For example, [BDGM20] constructed $i\mathcal{O}$ based on the circular security of LWE-based encryption schemes, which is conjectured to be quantum-secure. [WW20] showed a construction of $i\mathcal{O}$ based on the indistinguishability of two distributions which is also arguably quantum-secure.

Theorem 8.20 implies the following result for quantum-secure $i\mathcal{O}$:

Theorem 8.21. *Suppose that quantum-secure $i\mathcal{O}$ for polynomial-size circuits exists. Then,* $\mathsf{MQCSP} \in \mathsf{BQP}$ *implies* $\mathsf{NP} \subseteq \mathsf{coRQP}$.

Proof. Let $f_C(r) := i\mathcal{O}(C, r)$, where r is the random string. Then, by Theorem 8.20, we know that there exists a quantum polynomial-time algorithm $\mathcal{A}_{inv}$ with access to an MQCSP oracle and a non-negligible function p such that for any circuit C,

$$\Pr_r \left[f_C(\mathcal{A}_{inv}^{\mathsf{MQCSP}}(C, i\mathcal{O}(C, r))) = f_C(r) \right] \geq p(|r|). \tag{8.1}$$

Then, we can use $\mathcal{A}_{inv}$ to solve the $\mathsf{Circuit\text{-}SAT}$ problem. The algorithm is as follows:

Algorithm 31 A quantum algorithm for $\mathsf{Circuit\text{-}SAT}$

```
Input: The description of a circuit $C : \{0,1\}^n \to \{0,1\}$.
1: $s \leftarrow |C|$.
2: Compute $\bot_s$.                                   ▷ A canonical unsatisfiable circuit
3: $\hat{C} \leftarrow i\mathcal{O}(C, r)$.
4: $r' \leftarrow \mathcal{A}_{inv}^{\mathsf{MQCSP}}(\bot_s, \hat{C})$.
5: return "No" if $\hat{C} = i\mathcal{O}(\bot_s, r')$.
```

We assume that for any $s \geq 0$, we can compute a canonical unsatisfiable circuit of size s in $\mathsf{poly}(s)$ time.

If $C \in \mathsf{UNSAT}$, then $C \equiv \bot_s$. If $C = \bot_s$, by Eq. (8.1), $\mathcal{A}_{inv}^{\mathsf{MQCSP}}$ finds r with probability at least $p(|r|)$. Otherwise, by the indistinguishability of $i\mathcal{O}$ and $\mathsf{MQCSP} \in \mathsf{BQP}$, $\mathcal{A}_{inv}^{\mathsf{MQCSP}}$ is a quantum polynomial-time algorithm and hence cannot distinguish $C \in \mathsf{UNSAT} \backslash \{\bot_s\}$ and $\bot_s$ with more than $\mathsf{negl}(|r|)$ probability. Therefore, Algorithm 31 will reject C with probability $O(p(|r|))$.

If $C \in \mathsf{SAT}$, then $C \not\equiv \bot_s$. By the functionality of $i\mathcal{O}$, for any r, r', $i\mathcal{O}(C, r) \neq i\mathcal{O}(\bot_s, r')$. Hence, Algorithm 31 will always accept C.

Hence, by repeatedly running Algorithm 31 many times, we get that $\mathsf{NP} \subset \mathsf{coRQP}$, the one-sided error analog of BQP $\square$

Remark 8.4. It is worth noting that in the classical setting, the existence of $i\mathcal{O}$ implies that NP and MCSP are equivalent under randomized reductions; the other direction directly follows from the fact that $\mathsf{MCSP} \in \mathsf{NP}$. However, since it is unclear if $\mathsf{MQCSP} \in \mathsf{NP}$, we can only conclude that $\mathsf{NP} \subseteq \mathsf{RQP}^{\mathsf{MQCSP}}$ assuming the existence of quantum-secure $i\mathcal{O}$.

8.4.2 Learning theory

In this section, we discuss connections between MQCSP and learning theory. We consider two standard settings: probably approximately correct (PAC) learning and quantum learning. We postpone the details to Appendix 8.7.

PAC learning. Let C be a circuit class. We are interested in how to efficiently learn a function in C. PAC learning is a theoretical framework to evaluate how well a learning algorithm is. Here we focus on a special setting of PAC learning where the algorithm is able to query any input to the unknown function. In the following, we denote C-MCSP as the classical MCSP problem with respect to the circuit class C.

Definition 8.22 (PAC learning over the uniform distribution with membership queries)**.** Let C be a circuit class and let $\epsilon, \delta > 0$. We say an algorithm (ϵ, δ)-PAC-learns C over the uniform distribution with membership queries if the following hold. For every $n \in \mathbb{N}$ and n-variate $f \in \mathsf{C}$, given membership query access to f, the algorithm outputs a circuits C such that with probability at least $1 - \delta$ over its internal randomness, we have $\Pr_{x \in \{0,1\}^n}[f(x) \neq C(x)] < \epsilon$. The running time of the learning algorithm is measured as a function of $n, 1/\epsilon, 1/\delta$ and, $\mathsf{size}(f)$.

The seminal paper of Carmosino et al. [CIKK16] showed that efficient PAC learning for a (classical) circuit class C is *equivalent* to the corresponding MCSP being easy. Here, we quantize this connection and show in the following theorem that efficient PAC-learning for $\mathsf{BQP/poly}$ is equivalent to efficient algorithm for MQCSP. Here, $\mathsf{BQP/poly}$ is defined as $\bigcup_{s\leq \mathsf{poly}(n)} \mathsf{BQC}(s)$.

For technical reason, we need to work on a gap version of MQCSP in one direction of the equivalence. Let $\tau : \mathbb{N} \to (0, 1/2)$, $\mathsf{MQCSP}[s, s', t, \tau]$ is defined as the gap problem where the No instances in Definition 8.10 becomes "for every quantum circuit $\mathcal{C}$ using at most s' gates and operating on at most $n + t$ qubits, there are at least τ fraction of $x \in \{0, 1\}^n$ such that $\|(\langle f(x)| \otimes I_{n+t-1})\mathcal{C}|x, 0^t\rangle\|^2 \leq \frac{1}{2}$".

Theorem 8.22 (Equivalence of efficient PAC learning for $\mathsf{BQP/poly}$ and efficient randomized algorithm for MQCSP)**.**

- *If* $\mathsf{MQCSP} \in \mathsf{BPP}$*, then there is a randomized algorithm that* $(1/\mathsf{poly}(n), \delta)$*-PAC learns* $f \in \mathsf{BQP/poly}$ *under the uniform distribution with membership queries for every* $\delta > 0$*. Specifically, the algorithm runs in quasi-polynomial time.*
- *If there is a randomized algorithm that* $(1/\mathsf{poly}(n), \delta)$*-PAC learns* $f \in \mathsf{BQP/poly}$ *under the uniform distribution with membership queries for some* $\delta > 0$ *in* $2^{O(n)}$ *time, then we have* $\mathsf{MQCSP}[\mathsf{poly}(n), \omega(\mathsf{poly}(n)), \mathsf{poly}(n), \tau] \in \mathsf{BQP}$ *and* $\mathsf{MQCSP}[\mathsf{poly}(n), \omega(\mathsf{poly}(n)), O(n), \tau] \in \mathsf{BPP}$ *for every* $\tau > 0$*.*

Similarly, the positive resolution of Open Problem 8.2 would strengthen the conclusion of the second item in Theorem 8.22 to $\mathsf{MQCSP}[\mathsf{poly}(n), \omega(\mathsf{poly}(n)), \mathsf{poly}(n), \tau] \in \mathsf{BPP}$.

Quantum learning. As it could be the case that MQCSP might have non-trivial quantum algorithm, it is also of interest to study the connection to quantum learning [AGG+20].

Definition 8.23 (Quantum learning)**.** Let C be a circuit class of boolean functions and let $\epsilon, \delta > 0$. We say a quantum algorithm (ϵ, δ)-learns C if the following hold. For every $n \in \mathbb{N}$ and n-variate $f \in \mathsf{C}$, given quantum oracle access to f, the algorithm outputs a polynomial-size quantum circuit U such that with probability at least $1-\delta$, we have $\mathbb{E}_{x\in\{0,1\}^n}[|(\langle f(x)| \otimes I)U|x, 0^m\rangle|^2] > 1 - \epsilon$. The running time of the learning algorithm is measured as a function of $n, 1/\epsilon, 1/\delta$ and, $\mathsf{size}(f)$.

It turns out that efficient quantum learning for a circuit class C (could be either a classical circuit class or a quantum circuit class) is equivalent to efficient quantum algorithm for C-MCSP. Similarly, C-$\mathsf{MCSP}[s, s', \tau]$ is defined as the gap problem with the No instances being the truth tables where every circuit $\mathcal{C}$ of size s' errs on τ fraction of the inputs.

Theorem 8.23 (Equivalence of efficient quantum learning and efficient quantum algorithm for C-MCSP)**.** *Let C be a circuit class.*

- *If C-$\mathsf{MCSP} \in \mathsf{BQP}$, then there exists a quantum algorithm that $(1/\mathsf{poly}(n), \delta)$-learns C for every $\delta > 0$. Specifically, the algorithm runs in polynomial time.*
- *If there exists a quantum algorithm that (ϵ, δ)-learns C in time $2^{O(n)}$ for some constants $\epsilon, \delta \in (0, 1/2)$, then we have C-$\mathsf{MCSP}[\mathsf{poly}(n), \omega(\mathsf{poly}(n)), \tau] \in \mathsf{BQP}$ for every $\tau > 0$.*

8.4.3 Circuit lower bounds

The classical MCSP is tightly connected to circuit lower bounds. Many results show that a fast algorithm for MCSP will lead to breakthrough in circuit lower bounds, which on the other hand indicates that MCSP might be very difficult to solve. In this section, we "quantize" four results relating MQCSP and quantum circuit lower bounds.

Quantum circuit lower bound via quantum natural proof By Observation 8.12, we know that MQCSP gives a BQP-quantum natural property. Then, we follow a recent work by Arunachalam et al. [AGG+20] and prove the following theorem:

Theorem 8.24. *If* $\mathsf{MQCSP} \in \mathsf{BQP}$*, then* $\mathsf{BQE} \not\subset \mathsf{BQC}[n^k]$ *for any constant* $k \in \mathbb{N}_+$*, where* $\mathsf{BQE} = \mathsf{BQTIME}[2^{O(n)}]$.

Remark 8.5. A key difference between Theorem 8.24 and [AGG+20] is that their circuit lower bound for BQE is against *classical* circuits, while ours is against *quantum* circuits by proving a diagonalization lemma for quantum circuits.

An ingredient of our proof is a conditional pesudorandom generator against uniform quantum computation. We first recall the definition of PRG against uniform quantum circuits given by [AGG+20].

Definition 8.24 (Pesudorandom generator against uniform quantum circuit, [AGG+20])**.** A family of functions $\{G_n\}_{n\geq 1}$ is an infinitely often (ℓ, m, s, ϵ)-generator against uniform quantum circuits if the following properties holds:

1. Stretch: $G_n : \{0,1\}^{\ell(n)} \to \{0,1\}^{m(n)}$.
2. Uniformity and efficiency: There is a deterministic algorithm A that when given 1^n and $x \in \{0,1\}^{\ell(n)}$ runs in time $O(2^{\ell(n)})$ and outputs $G_n(x)$.
3. Pseudorandomness: For every deterministic algorithm A such that when given $1^{m(n)}$ runs in time $s(m)$ and outputs a quantum circuit C_m of size at most $s(m)$ computing a m-input Boolean function, for infinitely many $n \geq 1$,
$$\left| \Pr_{x\sim\{0,1\}^{\ell(n)}, C_m}[C_m(G_n(x)) = 1] - \Pr_{y\sim\{0,1\}^{m(n)}, C_m}[C_m(y) = 1] \right| \leq \epsilon(m).$$

[AGG+20] constructed the following infinitely often PRG based on the assumption $\mathsf{PSPACE} \not\subseteq \mathsf{BQSUBEXP}$.

Theorem 8.25 (Conditional PRG against uniform quantum computations, [AGG+20])**.** *Suppose that* $\mathsf{PSPACE} \not\subseteq \mathsf{BQSUBEXP}$*. Then, for some choice of constants* $\alpha \geq 1$

and $\lambda \in (0, 1/5)$, there is an infinitely often $(\ell, m, s, \varepsilon)$-generator $G = \{G_n\}_{n\geq 1}$, where $\ell(n) \leq n^{\alpha}$, $m(n) = \lfloor 2^{n^{\lambda}} \rfloor$, $s(m) = 2^{n^{2\lambda}} \geq \mathsf{poly}(m)$ (for any polynomial), and $\varepsilon(m) = 1/m$.

Now, we are ready to prove the lower bound for BQE based on the conditional PRG and a diagonalization theorem for quantum circuits.

Proof of Theorem 8.24. We use a win-win argument to prove the circuit lower bound.

Case 1: Suppose $\mathsf{PSPACE} \subseteq \mathsf{BQSUBEXP}$, i.e., for every $\gamma \in (0, 1]$, $\mathsf{PSPACE} \subseteq \mathsf{BQTIME}[2^{n^{\gamma}}]$. Then, for a fixed $k \in \mathbb{N}$, by a diagonalization lemma for quantum circuits (Claim 8.76), we know that there exists a language $L \in \mathsf{PSPACE}$ such that $L \notin \mathsf{BQC}[n^k]$. However, by the assumption, $L \in \mathsf{BQE}$, which implies that $\mathsf{BQE} \not\subset \mathsf{BQC}[n^k]$.

Case 2: $\mathsf{PSPACE} \not\subseteq \mathsf{BQSUBEXP}$, that is, there exists a language $L \in \mathsf{PSPACE}$ and $\gamma > 0$ such that $L \notin \mathsf{BQTIME}[2^{n^{\gamma}}]$. By Theorem 8.25, for some $\alpha \geq 1, \lambda \in (0, 1/5)$, there exists an infinitely often (ℓ, m, s, ϵ)-PRG $\mathcal{G} = \{G_n\}_{n\geq 1}$, where $\ell(n) = n^{\alpha}$, $m(n) = \lfloor 2^{n^{\lambda}} \rfloor$, $s(m) = \lfloor 2^{n^{2\lambda}} \rfloor$, $\epsilon(m) = 1/m$.

For each $w \in \{0, 1\}^{n^{\alpha}}$, we consider $G_n(w)$ as the truth table of Boolean function $\mathsf{fnc}(G_n(w)) : \{0, 1\}^d \to \{0, 1\}$, where $d := \log(m(n))$ is the input length of the function. We will show that $\mathsf{fnc}(G_n(w))$ is a hard function for $\mathsf{BQC}[d^{O(1)}]$ for most $w \in \{0, 1\}^{\ell(n)}$.

Suppose that this is not true, i.e., there exists a $k > 0$ such that for almost every $n > 0$, $\mathsf{fnc}(G_n(w)) \in \mathsf{BQC}[d(n)^k]$ for a constant fraction of seeds $w \in \{0, 1\}^{\ell(n)}$. Then, consider a quantum circuit C_m^{MQCSP} which takes a m-bit string s and accepts it if and only if $\mathsf{MQCSP}(s, 1^{d^k}) = 1$, where s is the truth table and d^k is the size parameter. Since we assume $\mathsf{MQCSP} \in \mathsf{BQP}$, the quantum circuit C_m^{MQCSP} can be generated by a deterministic algorithm in time $\mathsf{poly}(m) \leq s(m)$[21]. This implies that

$$\Pr_{w \sim \{0,1\}^{\ell(n)}, C_m^{\mathsf{MQCSP}}} \left[C_m^{\mathsf{MQCSP}}(G_n(w)) = 1 \right] \geq \delta$$

for some constant $\delta \in (0,1)$. On the other hand, by the pseudorandomness property of G_n (part 3 in Definition 8.24), for infinitely many n, we have

$$\left| \Pr_{w \sim \{0,1\}^{\ell(n)}, C_m^{\mathsf{MQCSP}}} \left[C_m^{\mathsf{MQCSP}}(G_n(w)) = 1 \right] - \Pr_{y \sim \{0,1\}^{m(n)}, C_m^{\mathsf{MQCSP}}} \left[C_m^{\mathsf{MQCSP}}(y) = 1 \right] \right| \leq \frac{1}{m}. \tag{8.2}$$

However, only $o(1)$-fraction of random functions have polynomial-size quantum circuits, i.e.,

$$\Pr_{y \sim \{0,1\}^{m(n)}, C_m^{\mathsf{MQCSP}}} \left[C_m^{\mathsf{MQCSP}}(y) = 1 \right] \leq o(1),$$

which means Eq. (8.2) cannot hold. Therefore, for infinitely many n, and almost all w, the function $\mathsf{fnc}(G_n(w)) \notin \mathsf{BQC}[n^k]$ for every $k \in \mathbb{N}_+$.

Therefore, we can construct a hard language $L^{\mathcal{G}}$ as follows:

- For any $n > 0$ and every $x \in \{0,1\}^n$, check if x can be written as (w, y), where $|w| = \ell(t)$ and $|y| = \lceil \log m(t) \rceil$ for some $t \in \mathbb{N}$.
- If not, then $L^{\mathcal{G}}(x) := 0$.
- Otherwise, $L^{\mathcal{G}}(x) := \mathsf{fnc}(G_t(w))(y)$.

We first show that $L^{\mathcal{G}} \in \mathsf{BQE}$. By the running time property of G_n (part 2 in Definition 8.24), $G_n(w)$ can be computed in deterministic time $O(2^{\ell(t)}) \leq O(2^n)$. Hence, $L^{\mathcal{G}} \in \mathsf{E} \subset \mathsf{BQE}$.

Then, we show that $L^{\mathcal{G}} \notin \mathsf{BQC}[n^k]$ for every $k \in \mathbb{N}_+$. Fix $k > 0$. Suppose there exists a quantum circuit family $\{C_n\}_{n \geq 1}$ that computes $L^{\mathcal{G}}$ and C_n has size n^k

[21]For all problems in BQP, there exists a classical Turing machine that can efficiently uniformly generate the quantum circuits.

for every $n \geq 1$. However, we already proved that there exists an infinite-size subset $\{\mathcal{S} \subset \mathbb{N}\}$ such that for $n \in \mathcal{S}$, there exists many "hard seed" w_n such that

$$\mathsf{fnc}(G_t(w_n)) \notin \mathsf{BQC}[t^{2\alpha k}]. \tag{8.3}$$

Then, for any $n \in S$ and any w_n that makes Eq. (8.3) hold, define a new quantum circuit family $\{C \downharpoonright_{w_n}\}_{n\geq 1}$ such that $C \downharpoonright_{w_n} (y) := C(w_n, y)$, i.e., $C \downharpoonright_{w_n}$ computes the hard function $\mathsf{fnc}(G_t(w_n))$. Hence, $C \downharpoonright_{w_n}$ must have size larger than $t^{2\alpha k}$. Since $n = \ell(t) + \log m(t) = t^\alpha + t^\lambda \leq t^{2\alpha}$, and the size of C_n should be least the size of its restriction $C \downharpoonright_{w_n}$, we conclude that C_n has size larger than n^k for these infinitely many $n \in \mathcal{S}$. Therefore, the BQE language $L^{\mathcal{G}} \notin \mathsf{BQC}[n^k]$, which implies $\mathsf{BQE} \not\subset \mathsf{BQC}[n^k]$.

Combining Case 1 and 2 completes the proof of the theorem. □

Circuit lower bound for $\mathsf{BQP}^{\mathsf{QCMA}}$ Our second result shows that if $\mathsf{MQCSP} \in \mathsf{BQP}$, then $\mathsf{BQP}^{\mathsf{QCMA}}$ cannot be computed by polynomial-size quantum circuits. Our result follows the seminal work of Kabanets and Cai [KC00], which showed a circuit lower bound for P^{NP} based on MCSP is easy. More specifically, we consider the following "hard problem":

Definition 8.25 (Maximum quantum circuit complexity problem)**.** The input of this problem is 1^n for $n \in \mathbb{N}_+$. The output is the truth table of a function $f : \{0,1\}^n \to \{0,1\}$ such that for any $f' : \{0,1\}^n \to \{0,1\}$, the quantum circuit complexity $\mathrm{qCC}(f) \geq \mathrm{qCC}(f')$.

We first prove that $\mathsf{BPE}^{\mathsf{QCMA}}$ can solve the maximum quantum circuit complexity problem, which implies that $\mathsf{BPE}^{\mathsf{QCMA}}$ contains the hardest Boolean function. Then, by the standard padding argument, we can show quantum circuit lower bound for $\mathsf{BQP}^{\mathsf{QCMA}}$.

Theorem 8.26. *If $\mathsf{MQCSP} \in \mathsf{BQP}$, then $\mathsf{BPE}^{\mathsf{QCMA}}$ contains a function with maximum quantum circuit complexity. Furthermore, $\mathsf{BQP}^{\mathsf{QCMA}} \not\subset \mathsf{BQC}[n^k]$ for any constant $k > 0$.*

We note that there are two subtle differences between Theorem 8.26 and [KC00]'s result:

- We need a QCMA oracle while [KC00] used an NP oracle. This is because we assume that MQCSP $\in$ BQP. In order to decide the maximum quantum circuit complexity, we can non-deterministically guess a truth table and use the BQP algorithm to verify its quantum circuit complexity. This process can be achieved by an QCMA oracle.
- Another difference is that we consider the BPE class while [KC00] considered the E class. This is because our QCMA oracle can only output correct answers with high probability. Thus, the whole algorithm will be a randomized algorithm.

The formal proof is deferred to Section 8.8.1.

Hardness amplification using MQCSP [KC00] showed that the classical MCSP can be used for hardness amplification, i.e., given one very hard Boolean function, there exists an efficient algorithm to find many hard functions via an MCSP oracle. We show that it also holds for quantum circuits:

Theorem 8.27. *Assume* MQCSP $\in$ BQP*. Then, there exists a* BQP *algorithm that, given the truth table of an n-variable Boolean function of quantum circuit complexity $2^{\Omega(n)}$, outputs $2^{\Omega(n)}$ Boolean functions on $m = \Omega(n)$ variables each, such that all of the output functions have quantum circuit complexity greater than $\frac{2^m}{(c+1)m}$ for any $c > 0$.*

In order to prove Theorem 8.27, we first construct a "quantum version" of the Impagliazzo-Wigderson generator [IW97]. We note that the construction in the following lemma is stronger than the Definition 8.24, based on the truth table of a very hard function.

Lemma 8.28 (Quantum Impagliazzo-Wigderson generator)**.** *For every $\epsilon > 0$, there exist $c, d \in \mathbb{N}$ such that the truth table of a Boolean function $f : \{0,1\}^{cn} \to \{0,1\}$ of* *quantum circuit complexity $2^{\epsilon cn}$ can be transformed in time $O(2^n)$ into a pseudorandom generator $G : \{0,1\}^{dn} \to \{0,1\}^{2^n}$ running in time $O(2^n)$ that can fool quantum circuits of size $2^{O(n)}$, i.e., for any $p > 0$, any quantum circuit $\mathcal{C}$ of size at most 2^{pn},*

$$\left| \Pr_{x \sim \{0,1\}^{dn}, \mathcal{C}}[\mathcal{C}(G(x)) = 1] - \Pr_{y \sim \{0,1\}^{2^n}, \mathcal{C}}[\mathcal{C}(y) = 1] \right| \leq 2^{-n}.$$

Proof of Theorem 8.27. Let $c > 0$ and $s(n) = \frac{2^n}{(c+1)n}$. Assuming that $\mathsf{MQCSP} \in \mathsf{BQP}$, we get a polynomial-size quantum circuit family $\{\mathcal{D}_n\}$ that only accept n-variable Boolean functions of quantum circuit complexity greater than $s(n)$. By Claim 8.74, the acceptance probability is close to one.

However, the size of $\mathcal{D}_n$ is bounded by a fixed polynomial in the input size, by Lemma 8.28, the quantum Impagliazzo-Wigderson generator G will fool $\mathcal{D}_n$. That is, almost all 2^n-bit strings output by G will have quantum circuit complexity greater than $s(n)$. We can then use the MQCSP circuit to decide the quantum circuit complexity of these strings and only output hard functions. $\square$

The proof of Lemma 8.28 relies on a quantum-secure direct product generator and several hardness amplification steps. It is deferred to Section 8.8.3.

Hardness magnification for MQCSP. Hardness magnification refers to a transformation of a weak circuit lower bound (e.g., linear size lower bound) to a stronger circuit lower bound (e.g., polynomial size lower bound). Note that a magnification theorem for a circuit class is highly dependent on the structure of the circuits. Specifically, it is not immediately clear that every circuit class is *magnifiable.* Here, we show that there exists hardness magnification for quantum circuits when it comes to MQCSP.

Theorem 8.29. *If* $\mathsf{MQCSP}\left[2^{n^{1/2}}/2n, 2^{n^{1/2}}\right]$ *is hard for* $\mathsf{BQC}\left[2^{n+O(n^{1/2})}\right]$, *then* $\mathsf{QCMA} \not\subseteq \mathsf{BQC}[\mathsf{poly}(n)]$.

The proof of Theorem 8.29 is via antichecker lemma, which was first given by [OPS19, CHO+20] for proving hardness magnification for MCSP.

Lemma 8.30 (Antichecker lemma for quantum circuits)**.** *Assume* $\mathsf{QCMA} \subseteq \mathsf{BQC}[\mathsf{poly}]$*. Then for any* $\lambda \in (0,1)$ *there are circuits* $\{C_{2^n}\}_{n=1}^{\infty}$ *of size* $2^{n+O(n^\lambda)}$ *which given the truth table* $\mathsf{tt}(f) \in \{0,1\}^{2^n}$ *, outputs* $2^{O(n^\lambda)}$ *n-bit strings* $y_1, \dots, y_{2^{O(n^\lambda)}}$ *together with bits* $f(y_1), \dots, f(y_{2^{O(n^\lambda)}})$ *forming a set of anticheckers for* f*, i.e. if* f *is hard for quantum circuits of size* 2^{n^λ} *then every quantum circuit of size* $2^{n^\lambda}/2n$ *fails to compute* f *on one of the inputs* $y_1, \dots, y_{2^{O(n^\lambda)}}$*.*

With Lemma 8.30, we can prove Theorem 8.29 by using a small quantum circuit to verify the given circuits only on the anticheckers.

Proof of Theorem 8.29. Suppose $\mathsf{QCMA} \subseteq \mathsf{BQC}[\mathsf{poly}]$. Let $\mathsf{tt}(f)$ be the input of $\mathsf{MQCSP}[2^{n^{1/2}}/2n, 2^{n^{1/2}}]$. By Lemma 8.30, we can find a set of anticheckers $y_1, \dots, y_{2^{O(n^{1/2})}}$ by a quantum circuit of size $2^{n+O(n^{1/2})}$. Then, we use a QCMA algorithm to decide if there exists a quantum circuit of size $2^{n^\lambda}/2n$ that computes f correctly on $\{(y_1, f(y_1)), \dots, (y_{2^{O(n^\lambda)}}, f(y_{2^{O(n^\lambda)}}))\}$. By the assumption, it can be done by a $2^{O(n^\lambda)}$ size quantum circuit. Then, there are two cases:

- If the QCMA algorithm returns "Yes", it means that $y_1, \dots, y_{2^{O(n^{1/2})}}$ are not anticheckers. By Lemma 8.30, f is *not* hard for $2^{n^{1/2}}$ size quantum circuit.
- If the QCMA algorithm returns "No", then no $2^{n^{1/2}}/2n$ size quantum circuit can compute f on $y_1, \dots, y_{2^{O(n^{1/2})}}$. So, f is hard for $2^{n^{1/2}}/2n$ size quantum circuit.

Hence, $\mathsf{MQCSP}[2^{n^{1/2}}/2n, 2^{n^{1/2}}] \in \mathsf{BQC}[2^{n+O(n^{1/2})}]$. □

The proof of Lemma 8.30 is deferred to Section 8.8.2.

8.4.4 Fine-grained complexity

It is a long-standing open problem to show the hardness of MCSP based on some fine-grained complexity hypotheses, like the Exponential-Time Hypothesis (ETH), which was conjectured by Impagliazzo, Paturi, and Zane [IPZ01] and becomes a widely used assumption in fine-grained complexity area.

Definition 8.26 (Exponential Time Hypothesis (ETH))**.** There exists $\delta > 0$ such that 3-SAT with n variables cannot be solved in time $2^{\delta n}$.

Very recently, a breakthrough result by Ilango [Ila20b] proved the ETH-hardness of MCSP for partial Boolean functions. On the other hand, Quantum fine-grained complexity was studied very recently by [ACL+20, BPS21, AL20b, GS20]. Motivated by the fact that currently there is no quantum algorithm for 3-SAT that is significantly faster than Grover's search, we conjecture that 3-SAT with n variables cannot be solved in $2^{o(n)}$ quantum time (QETH). And based on QETH, we want show that MQCSP for partial Boolean function is also hard.

We first formally define QETH and MQCSP for partial functions (MQCSP*).

Definition 8.27 (Quantum Exponential Time Hypothesis (QETH))**.** There exists $\delta' > 0$ such that 3-SAT with n variables cannot be solved in time $2^{\delta' n}$ in quantum.

Definition 8.28 (MQCSP for partial functions (MQCSP*))**.** The input is the truth table $\{0,1,\star\}^{2^n}$ of a partial function $f : \{0,1\}^n \to \{0,1,\star\}$ and an integer parameter s. The goal is to decide whether there exists a quantum circuit C of size at most s (using single-qubit and 2-qubit gates) that computes f. That is, for all $x \in \{0,1\}^n$ such that $f(x) \neq \star$, we have

$$\Pr[C(x) = f(x)] \geq \frac{2}{3}.$$

Our main result of this section is as follows:

Theorem 8.31 (QETH-hardness of $\mathsf{MQCSP}^\star$)**.** $\mathsf{MQCSP}^\star$ *cannot be solved in* $N^{o(\log\log N)}$*-time quantumly on truth tables of length* N *assuming* QETH*.*

Our reduction reveals the connections between $\mathsf{MQCSP}^\star$, quantum read-once formula and classical read-once formula. The proof is given in Section 8.9.

Classical reduction for $\mathsf{MCSP}^\star$. We first give a brief overview of the classical reduction for $\mathsf{MCSP}^\star$ in [Ila20b]. They reduced $\mathsf{MCSP}^\star$ to a fine-grained problem: $2n \times 2n$ *Bipartite Permutation Independent Set problem*, which is defined as follows:

Definition 8.29 (Bipartite Permutation Independent Set problem)**.** A $2n \times 2n$ bipartite permutation independent set problem is defined on a directed graph G with vertex set $[n] \times [n]$ and edge set E. The goal is to decide whether there exists a permutation $\pi \in \mathcal{S}_{2n}$ such that

- $\pi([n]) = [n]$,
- $\pi(\{n+i : i \in [n]\}) = \{n+i : i \in [n]\}$,
- if $((j,k),(j',k')) \in E$, then either $\pi(j) \neq k$ or $\pi(n+j') \neq (n+k')$.

Lokshtanov, Marx, and Saurabh [LMS11] proved that this problem is $2^{o(n\log n)}$-hard under ETH, which implies the ETH-hardness of $\mathsf{MCSP}^\star$.

The reduction from $2n \times 2n$ bipartite permutation independent set problem to $\mathsf{MCSP}^\star$ is via the following partial function γ. Consider an instance $G = ([n]\times[n], E)$ of $2n \times 2n$ bipartite permutation independent set problem. The reduction outputs the truth table of a partial Boolean function $\gamma : \{0,1\}^{2n} \times \{0,1\}^{2n} \times \{0,1\}^{2n} \to \{0,1,\star\}$

such that

$$\gamma(x,y,z) := \begin{cases} \bigvee_{i\in[2n]}(y_i \wedge z_i) & \text{if } x = 0^{2n}, \\ \bigvee_{i\in[2n]} z_i & \text{if } x = 1^{2n}, \\ \bigvee_{i\in[2n]}(x_i \vee y_i) & \text{if } z = 1^{2n}, \\ 0 & \text{if } z = 0^{2n}, \\ \bigvee_{i\in[n]} x_i & \text{if } z = 1^n 0^n \text{ and } y = 0^{2n}, \\ \bigvee_{i\in\{n+1,\cdots,2n\}} x_i & \text{if } z = 0^n 1^n \text{ and } y = 0^{2n}, \\ 1 & \text{if } \exists((j,k),(j',k')) \in E \text{ s.t. } (x,y,z) = (\overline{e_k e_{k'}}, 0^{2n}, e_j e_{j'}), \\ \star & \text{otherwise.} \end{cases} \tag{8.4}$$

In particular, the small circuit size of γ implies that G is a "Yes" instance of the bipartite permutation independent set problem:

Lemma 8.32 ([Ila20b]). *Each of the following are equivalent:*

1. $\mathsf{MCSP}^\star(\gamma, 6n-1) = 1$;
2. *γ can be computed by a read-once formula;*
3. *there exists a $\pi \in \mathcal{S}_{2n}$ such that $\bigvee_{i\in[2n]}((x_{\pi(i)} \vee y_i) \wedge z_i)$ computes γ;*
4. *there exists a $\pi \in \mathcal{S}_{2n}$ that satisfies the instance of bipartite permutation independent set problem given by G.*

Quantum reduction for $\mathsf{MQCSP}^\star$ We follow the proof in [Ila20b] but adapt it to quantum circuits. More specifically, we want to show that for the partial function γ defined by Eq. (8.4), $\mathsf{MQCSP}^\star(\gamma, 6n-1) = 1$ is equivalent to the case that γ can be computed by a read-once formula.

The reverse direction is easy:

Claim 8.33. *If γ can be computed by a read-once formula, then* $\mathsf{MQCSP}^\star(\gamma, 6n-1) = 1$.

Proof. It is easy to see that a read-once formula on $6n$ input variables has at most $6n-1$ Boolean gates. Hence, it implies that $\mathsf{MCSP}^{\star}(\gamma, 6n-1) = 1$. Then, we have $\mathsf{MQCSP}^{\star}(\gamma, 6n-1) = 1$ because we can use a quantum circuit with all 2-qubit gates to simulate a Boolean circuit without increasing the circuit size. □

For the forward direction, we consider an intermediate model: *read-once quantum formula.* The quantum formula was defined by Yao [Yao93] as follows:

Definition 8.30. A quantum formula is a single-output quantum circuit such that every gate has at most one output that is used as an input to a subsequent one.

If a quantum formula only uses every input qubit at most once, then we say it is a read-once quantum formula.

We first prove the forward direction for the quantum read-once formula:

Claim 8.34. *If* $\mathsf{MQCSP}^{\star}(\gamma, 6n-1) = 1$*, then* γ *can be computed by a read-once quantum formula. Here, we assume that the quantum circuits only use single-qubit and 2-qubit gates.*

Proof. It is easy to verify that γ depends on all of the $6n$ input variables. Hence, by a light-cone argument, the topology of the quantum circuit that computes γ using $6n-1$ 2-qubit gates must be a full binary tree with $6n$ leaves. Hence, that circuit is a read-once quantum formula. □

Cosentino, Kothari, and Paetznick [CKP13] proved that any read-once quantum formula can be "dequantized" to the classical read-once quantum formula:

Theorem 8.35 ([CKP13])**.** *If a language is accepted by a bounded-error read-once quantum formula over single-qubit and 2-qubit gates, then it is also accepted by an exact read-once classical formula with the same size, using* NOT *and all 2-bit Boolean gates.*

Hence, we can apply Theorem 8.35 to dequantize Claim 8.34:

Claim 8.36. *If* $\mathsf{MQCSP}^\star(\gamma, 6n-1) = 1$*, then* γ *can be computed by a classical read-once formula with* $6n-1$ *2-bit gates. In particular, all the* NOT *gates can be pushed to the leaf level and the high level gates are* $\{\mathsf{AND}, \mathsf{OR}, \mathsf{XOR}\}$*.*

Proof. By Theorem 8.35, there is a read-once classical formula that computes γ using $6n-1$ 2-bit logical gates. We can enumerate all of the 2-bit Boolean function and check that they can be expressed by one of $\mathsf{AND}, \mathsf{OR}, \mathsf{XOR}$ gate with some NOT gates on the input wire. Then, by De Morgan's laws, we can push the NOT gate to the bottom level. Note that these transformations will preserve the read-once property. □

The next claim shows that NOT and XOR gates do not help computing γ:

Claim 8.37. *The classical read-once formula computing* γ *only uses* AND *and* OR *gates.*

Proof. The proof is similar to the proof of Claim 13 in [Ila20b].

We first note that the XOR gate is not monotone. Then, by setting $x = 0^{2n}$, we have $\gamma(0^{2n}, y, z) = \bigvee_{i\in[2n]}(y_i \wedge z_i)$, which is a monotone function in y and z. Hence, the XOR gates in the formula cannot depend on the all the y and z variables. Similarly, by setting $z = 1^{2n}$, we have $\gamma(x, y, 1^{2n}) = \bigvee_{i\in[2n]}(x_i \vee y_i)$, which is monotone in x and y. It implies that the XOR gates cannot depend on all the x variables. Hence, the formula will not use the XOR gate.

For the NOT gate, since the function is monotone in the positive input variables after some restrictions, and the formula is read-once, the NOT gate will also not be used. □

By Claim 8.33, 8.36 and 8.37, we get that $\mathsf{MQCSP}^\star(\gamma, 6n-1) = 1$ is equivalent to the case that γ can be computed by a read-once formula using AND and OR gates. This statement corresponds to showing that (1) $\Leftrightarrow$ (2) in Lemma 8.32 for $\mathsf{MCSP}^\star$. Then, by (2) $\Leftrightarrow$ (4) in Lemma 8.32, we prove the following reduction for $\mathsf{MQCSP}^\star$:

Lemma 8.38. $\mathsf{MQCSP}^\star(\gamma, 6n-1) = 1$ *is equivalent to the existence of* $\pi \in \mathcal{S}_{2n}$ *that satisfies the instance of bipartite permutation independent set problem given by* G*.*

The remaining thing is to prove the quantum hardness of the $2n \times 2n$ Bipartite Permutation Independent Set problem. We follow the quantum fine-grained reduction framework by [ACL+20] and show the following QETH-hardness result. The proof is given in Section 8.9.

Lemma 8.39. *Assuming QETH, there is no* $2^{o(n \log n)}$*-time quantum algorithm that solves* $2n \times 2n$ *Bipartite Permutation Independent Set problem.*

Now, we can prove the QETH-hardness of $\mathsf{MQCSP}^\star$:

Proof of Theorem 8.31. By Lemma 8.38, $\mathsf{MQCSP}^\star$ can be reduced to $2n \times 2n$ Bipartite Permutation Independent Set problem and the hardness follows from Lemma 8.39. □

8.5 MCSP for Quantum Objects

In this section, we generalize the problem to considering circuit complexities of quantum objects, including unitaries and quantum states. In particular, we study their hardness, related reductions, and their implications to other subjects in quantum computer science. We start by defining the two problems.

Definition 8.31 ($\mathsf{UMCSP}_{\alpha,\beta}$). Let $n, s, t \in \mathbb{N}$ and $t \leq s$. Let $\alpha, \beta \in (0, 1]$. Let $U \in \mathbb{C}^{2^n \times 2^n}$ be a unitary. UMCSP is a promise problem defined as follows.

- **Inputs:** the unitary matrix U, the size parameter s in unary representation, and the ancilla parameter t.
- **Yes instance:** there exists a quantum circuit $\mathcal{C}$ using at most s gates and operating on at most $n + t$ qubits such that for all $|\psi\rangle \in \mathbb{C}^{2^n}$,

$$\|(\langle\psi| \otimes I_t)(U^\dagger \otimes I_t)\mathcal{C}|\psi, 0^t\rangle\|^2 \geq \alpha, \tag{8.5}$$

- **No instance:** for every quantum circuit $\mathcal{C}$ using at most s gates and operating on at most $n+t$ qubits, there exists some $|\psi\rangle \in \mathbb{C}^{2^n}$ such that

$$\|(\langle\psi| \otimes I_t)(U^\dagger \otimes I_t)\mathcal{C}|\psi, 0^t\rangle\|^2 \leq \beta. \tag{8.6}$$

With the promise that the input must be either a yes instance or a no instance, the problem is to decide whether the input is a yes instance or not.

Remark 8.6. Since the input to UMCSP is a unitary matrix U and each entry is a complex number, we cannot fully describe U and hence need to specify a precision parameter. Moreover, the precision issue is subtle in the search-to-decision reduction. For a gate set $\mathcal{G}$, we denote $\ell_{\mathcal{G}} \in \mathbb{N}$ as the maximum number of bits used to encode an entry of a gate. Note that if a circuit uses s gates from $\mathcal{G}$, then each entry in the resulting unitary can be written down with at most $s \cdot \ell_{\mathcal{G}}$ bits. Thus, by the triangle inequality for the distance between unitaries, it suffices to use $s \cdot \ell_{\mathcal{G}}$ bits to encode each entry of the input unitary. Also, note that when $\alpha - \beta < 2^{-s \cdot \ell_G}$, $\mathsf{UMCSP}_{\alpha,\beta}$ becomes a non-promise problem since effectively the gap between Yes and No instances does not matter. In the definition of UMCSP, we hide the introduction of precision parameter for simplicity. Note that from the above reasoning and the fact that the input unitary is $2^n \times 2^n$, it would not affect the complexity of the problem even one chooses the bit complexity to be $2^{O(n)}$, which is more than enough for most interesting situations.

Definition 8.32 ($\mathsf{SMCSP}_{\alpha,\beta}$)**.** Let $n, s, t \in \mathbb{N}$, where $t \leq s$. Let $\alpha, \beta \in (0, 1]$. Let $|\phi\rangle \in \mathbb{C}^{2^n}$ be a quantum state. SMCSP is a promise problem defined as follows.

- **Inputs:** size parameters s and n in unary, access to arbitrary many copies of $|\psi\rangle$, and the ancilla parameter t.
- **Yes instance:** there exists a quantum circuit $\mathcal{C}$ using at most s gates and operating on at most $n+t$ qubits such that

$$\|(\langle\phi| \otimes I_{n+t-1})\mathcal{C}|0^{n+t}\rangle\|^2 \geq \alpha,$$

- **No instance:** for every quantum circuit $\mathcal{C}$ using at most s gates and operating on at most $n+t$ qubits,

$$\|(\langle\phi| \otimes I_{n+t-1})\mathcal{C}|0^{n+t}\rangle\|^2 \leq \beta.$$

With the promise that the input must be either a yes instance or a no instance, the problem is to decide whether the input is a yes instance or not.

Remark 8.7. Similarly, the precision of the input parameters α, β of SMCSP has to depend on the bit complexity of the gate set. See Remark 8.6 for more discussion.

Remark 8.8. For the thresholds α, β, it is worth noting that a quantum circuit that outputs a mixed state can always have nonzero inner product with an arbitrary state. Therefore, we cannot set β to be arbitrarily small; otherwise, there will not be any U or $|\phi\rangle$ satisfying the no instance.

For SMCSP, we focus on the version where the inputs are multiple quantum states. The input format is quite different from UMCSP and MQCSP; instead of having the full classical description, SMCSP is given access to many copies of the quantum state. Hence, we say an algorithm for SMCSP is efficient if it runs in time $\mathsf{poly}(n,t,s)$, i.e., an efficient algorithm can use at most $\mathsf{poly}(n,t,s)$ copies of $|\psi\rangle$. We choose this input format because in the quantum setting, we generally cannot have the classical description of the quantum state. For instance, in shadow tomography[Aar18], quantum gravity[BFV20], and quantum pseudorandom states[JLS18], the problem is given many copies of a quantum state, identify some properties of the state. Furthermore, although this problem seems to be much harder than having the full description or a succinct description (e.g, a circuit that generates the state) of the state, we will see that this problem has a QCMA protocol. [22]

[22]Since SMCSP takes quantum inputs, the problem is not in QCMA under the standard definition. However, problems with quantum inputs in quantum computing is natural, so, it is also reasonable to study the complexity classes that allow quantum inputs.

Remark 8.9. On the other hand, the hardness results including the problem is in QCMA (Theorem 8.46), the search-to-decision reduction (Theorem 8.50), and the approximate self-reduction (Theorem 8.52) all hold for the version where the input is a classical description for the state.

Before proving the main theorems in this section, we introduce some notations and the swap test. Swap test [BCWdW01] is a quantum subroutine for testing whether two pure quantum states are close to each other.

Notation. We write $a \approx_\epsilon b$ for $a, b \in \mathbb{R}$ to mean $\|a - b\| \leq \epsilon$.

Notation. We write $|\varphi\rangle \approx_\epsilon |\phi\rangle$ to mean $\||\varphi\rangle - |\phi\rangle\| \leq \epsilon$.

Lemma 8.40 (Correctness of Swap Test). *For any two states $|\phi\rangle, |\psi\rangle$, consider the following state*

$$(\mathsf{H} \otimes \mathsf{I})(c\text{-}\mathsf{SWAP})(\mathsf{H} \otimes \mathsf{I})|0\rangle|\phi\rangle|\psi\rangle$$

Measuring the first qubit gives outcome 1 with probability $\frac{1}{2} - \frac{1}{2}|\langle\phi|\psi\rangle|^2$.

Claim 8.41. *Let $|\phi\rangle, |\psi\rangle \in \mathbb{C}^{2^n}$ be two quantum states such that $|\phi\rangle \approx_\epsilon |\phi\rangle$. Then, for any $|\psi'\rangle$ which is a state on at most n qubits,*

$$\|(\langle\psi'| \otimes I)|\phi\rangle\| - \epsilon \leq \|(\langle\psi'| \otimes I)|\psi\rangle\| \leq \|(\langle\psi'| \otimes I)|\phi\rangle\| + \epsilon.$$

Proof. Without loss of generality, we can write $|\psi\rangle = |\phi\rangle + |\epsilon\rangle$, where $\||\epsilon\rangle\| \leq \epsilon$. Then, $\|(\langle\psi'| \otimes I)|\psi\rangle\| = \|(\langle\psi'| \otimes I)|\phi\rangle + (\langle\psi'| \otimes I)|\epsilon\rangle\|$. By using triangular inequality, we obtain the following two inequalities:

$$\|(\langle\psi'| \otimes I)|\psi\rangle\| \leq \|(\langle\psi'| \otimes I)|\phi\rangle\| + \|(\langle\psi'| \otimes I)|\epsilon\rangle\|, \text{ and}$$
$$\|(\langle\psi'| \otimes I)|\psi\rangle\| \geq \|(\langle\psi'| \otimes I)|\phi\rangle\| - \|(\langle\psi'| \otimes I)|\epsilon\rangle\|.$$

Since $\||\epsilon\rangle\| \leq \epsilon$, $\|(\langle\psi'| \otimes I)|\epsilon\rangle\| \leq \epsilon$. This completes the proof. □

Theorem 8.42. $\mathsf{UMCSP}_{\alpha,\beta}$ *where $\beta \leq 1 - \mathsf{poly}(1/2^n)$ and $\alpha > 1 - 2^{-2n-20}(1-\beta)^4$ (for example, $\alpha = 1 - \exp(-2^n), \beta = 1 - \mathsf{poly}(1/2^n)$) is in* QCMA.

To design the verifier (that verifies a quantum circuit $\mathcal{C}$ really implements U as we want), what we will do is the following checking:

1. Standard basis check: check whether Eq. (8.5) is satisfied on standard basis states.
2. Coherency check: Check Eq. (8.5) on superposition states in the form of $|a\rangle+|b\rangle$. This step has two goals: (1) checking whether the operation does behave similar to a unitary (instead of, for example, a collapsing measurement). (2) the unitary does not introduce different phases on different basis states.

Proof. Our checking algorithm follows the two steps above. The certificate is the circuit that implements the unitary such that Eq. (8.5) is satisfied. The following algorithm verifies it (assuming the promise):

1. (Standard basis check) For each $i \in [2^n]$, evaluate $(U^\dagger \otimes I_t)\mathcal{C}(|i\rangle \otimes |0^t\rangle)$ for $\mathsf{poly}_1(2^n)$ times. Store the output state (which requires only polynomial memory); denote the j-th sample on input i as $|\varphi_i^j\rangle$. Measure each of the states and check whether the output for $|\varphi_i^j\rangle$ is i. If not, mark it as a negative sample.

   If for any i, the ratio of negative samples is $\geq 2^{-2n-18}(1-\beta)^4$, reject.

2. (Coherency check) Do the following for each $i, j \in [2^n], i \neq j$ for $\mathsf{poly}_2(2^n)$ times:

   Apply $(U^\dagger \otimes I_t)\mathcal{C}$ on $\frac{1}{\sqrt{2}}(|i\rangle+|j\rangle)\otimes|0^t\rangle$. Project the output system on $\frac{1}{\sqrt{2}}(|i\rangle+|j\rangle)$. If the projection does not succeed, consider it as a negative sample.

   If for any of i, the ratio of negative samples is $\geq 2^{-2n-18}(1-\beta)^4$, reject.

We will show, when $\mathsf{poly}_1, \mathsf{poly}_2$ are all chosen to be some sufficiently big polynomials, this test can be used as the QCMA-verifier we need.

First, if a circuit satisfies Eq. (8.5), we can prove the verifier succeeds with probability $1 - 2^{-O(\mathsf{poly}(2^n))}$.

1. First, in the standard basis check, by Eq. (8.5), the expected ratio of negative sample is at most $1-\alpha \leq \frac{1}{4} \cdot \text{threshold}$ (threshold $:= 2^{-2n-18}(1-\beta)^4$). By the Chernoff bound we have, $\forall a \in [2^n]$,

   $$\begin{aligned} & \Pr[\text{negative ratio} \geq \text{threshold}] \\ = \; & \Pr[\text{negative samples} \geq \text{threshold} \cdot \mathsf{poly}_1(2^n)] \\ \leq \; & 2^{-O(\mathbb{E}[\text{negative samples}])} && \text{(Chernoff bound)} \\ \leq \; & 2^{-O(\mathsf{poly}_1(2^n) \cdot 2^{-2n-20}(1-\beta)^4)} && (\text{threshold} \cdot \mathsf{poly}_1(2^n) \geq 4 \cdot \mathbb{E}[\text{negative samples}]) \end{aligned}$$

   which is $2^{-O(\mathsf{poly}(2^n))}$ when poly_1 is taken to be big enough. (Since $1-\beta = \mathsf{poly}(1/2^n)$)

   Summing this failure probability for all $a \in [2^n]$ altogether we know with probability is at most

   $$2^n \cdot 2^{-O(\mathsf{poly}(2^n))} = 2^{-O(\mathsf{poly}(2^n))},$$

   which means it could not pass the first step.

2. For the coherency check we can apply Eq. (8.5) directly again and know for each a, b, the expected error ratio is $\leq 1-\alpha \leq \frac{1}{4} \cdot \text{threshold}$. (Similarly threshold $:= 2^{-2n-18}(1-\beta)^4$). Thus by the Chernoff bound and similar arguments

   $$\forall a, b \in [2^n], \quad \Pr[\text{error ratio} \geq \text{threshold}] \leq 2^{-O(\mathsf{poly}_2(2^n) \cdot 2^{-2n-20}(1-\beta)^4)}$$

   which is $2^{-O(\mathsf{poly}(2^n))}$ when poly_2 is taken to be big enough. (Since $1-\beta = \mathsf{poly}(1/2^n)$)

   Thus summing this failure probability for all $a, b \in [2^n]$ we know this step fails with probability at most

   $$2^n \cdot 2^n \cdot 2^{-O(\mathsf{poly}(2^n))} = 2^{-O(\mathsf{poly}(2^n))}.$$

Thus we get the completeness.

Then we prove a circuit that satisfies Eq. (8.6) will be rejected with probability $1 - 2^{-O(\mathsf{poly}(2^n))}$. To prove that, we need to understand how the coherency check help us control the form of the states. We will prove the following lemmas step by step.

First, we show the success of coherency check implies the ancilla states have to be close to each other:

Lemma 8.43. *Suppose for some $a, b \in [2^n], a \neq b$, the following equations hold:*

$$\begin{aligned}
&\|(\langle a| \otimes I_t)(U^\dagger \otimes I_t)\mathcal{C}(|a\rangle \otimes |0^t\rangle)\|^2 \geq 1 - \delta, \\
&\|(\langle b| \otimes I_t)(U^\dagger \otimes I_t)\mathcal{C}(|b\rangle \otimes |0^t\rangle)\|^2 \geq 1 - \delta, \\
&\left\|\left(\frac{\langle a| + \langle b|}{\sqrt{2}} \otimes I_t\right)(U^\dagger \otimes I)\mathcal{C}\left(\frac{|a\rangle + |b\rangle}{\sqrt{2}} \otimes |0^t\rangle\right)\right\|^2 \geq 1 - \delta. \qquad (8.7)
\end{aligned}$$

Define the ancilla states $|\chi_a\rangle$, $|\chi_b\rangle$ via

$$(U^\dagger \otimes I_t)\mathcal{C}(|a\rangle \otimes |0^t\rangle) \approx_{\sqrt{\delta}} |a\rangle \otimes |\chi_a\rangle \qquad (8.8)$$

$$(U^\dagger \otimes I_t)\mathcal{C}(|b\rangle \otimes |0^t\rangle) \approx_{\sqrt{\delta}} |b\rangle \otimes |\chi_b\rangle \qquad (8.9)$$

where the right hand sides are the states from projecting $(U^\dagger \otimes I_t)\mathcal{C}(|a\rangle \otimes |0^t\rangle)$, and projecting $(U^\dagger \otimes I_t)\mathcal{C}(|b\rangle \otimes |0^t\rangle)$ on to $|a\rangle, |b\rangle$ respectively.

Then we have

$$|\chi_a\rangle \approx_{4\delta^{1/4}} |\chi_b\rangle \qquad (8.10)$$

Proof. We can evaluate the left hand side of Eq. (8.7) and get

$$\begin{aligned}
&\left\|\frac{1}{\sqrt{2}}((\langle a| + \langle b|) \otimes I_t)((U^\dagger \otimes I)\mathcal{C})\frac{1}{\sqrt{2}}((|a\rangle + |b\rangle) \otimes |0^t\rangle)\right\| \\
\approx_{\sqrt{2\delta}} \; & \frac{1}{2}\|((\langle a| + \langle b|) \otimes I_t)(|a\rangle \otimes |\chi_a\rangle + |b\rangle \otimes |\chi_b\rangle)\| \qquad \text{(By Eqs. (8.8),(8.9))} \\
= \; & \frac{1}{2}\||\chi_a\rangle + |\chi_b\rangle\| \\
= \; & \sqrt{1 - \frac{1}{4}\||\chi_a\rangle - |\chi_b\rangle\|^2}
\end{aligned}$$

Substitute Eq. (8.7), we know

$$\sqrt{1-\frac{1}{4}\||\chi_a\rangle-|\chi_b\rangle\|^2} \geq \sqrt{1-\delta}-\sqrt{2\delta},$$

$$\||\chi_a\rangle-|\chi_b\rangle\| \leq 2\sqrt{2\sqrt{2\delta(1-\delta)}-\delta} \leq 4\delta^{1/4}.$$

The lemma is then proved. □

Furthermore, we can show, when Eq. (8.10) holds for all pairs (a,b), the operation $(U^\dagger \otimes I)\mathcal{C}$ is indeed close to identity:

Lemma 8.44. *Suppose for all $a,b \in [2^n], a \neq b$, Eqs. (8.8),(8.9),(8.10) holds. Then for all $|\psi\rangle \in \mathbb{C}^{2^n}$,*

$$\|(\langle\psi| \otimes I_t)(U^\dagger \otimes I_t)\mathcal{C}|\psi, 0^t\rangle\|^2 \geq 1 - 10\cdot 2^{n/2}\delta^{1/4}$$

Proof. Decompose $|\psi\rangle = \sum_{i\in[2^n]} c_i|e_i\rangle$. Take $|\text{aux}\rangle = |\chi_0\rangle$. Then

$$\begin{aligned}(U^\dagger \otimes I_t)\mathcal{C}(|\psi\rangle \otimes |0^t\rangle) &= \sum_{i\in[2^n]} c_i(U^\dagger \otimes I_t)\mathcal{C}(|e_i\rangle \otimes |0^t\rangle) \\ &\approx_{\sum_i c_i\sqrt{\delta}} \sum_{i\in[2^n]} c_i|e_i\rangle \otimes |\chi_i\rangle \qquad \text{(By Eqs. (8.8),(8.9))} \\ &\approx_{\sum_i 4c_i\delta^{1/4}} \sum_{i\in[2^n]} c_i|e_i\rangle \otimes |\text{aux}\rangle \qquad \text{(By Eq. (8.10))} \\ &= |\psi\rangle \otimes |\text{aux}\rangle,\end{aligned}$$

which implies

$$\begin{aligned}\|(\langle\psi| \otimes I_t)(U^\dagger \otimes I_t)\mathcal{C}|\psi, 0^t\rangle\|^2 &\geq (1-5\delta^{1/4}\sum_i c_i)^2 \\ &\geq (1-5\cdot 2^{n/2}\delta^{1/4})^2 \\ &\geq 1-10\cdot 2^{n/2}\delta^{1/4}.\end{aligned}$$

And the proof is completed. □

Then we prove a circuit that satisfies Eq. (8.6) will be rejected with probability $1-2^{-O(\mathsf{poly}(2^n))}$.

1. After the standard basis check, $\mathcal{C}$ has to satisfy the following property, otherwise the verifier will reject with probability $1 - 2^{-O(\mathsf{poly}(2^n))}$:

$$\forall a \in [2^n], \quad \|((\langle a| \otimes I_t)(U^\dagger \otimes I_t)\mathcal{C}(|a\rangle \otimes |0^t\rangle)\|^2 \geq 1 - 2^{-2n}(\frac{1}{11}(1-\beta))^4 \quad (8.11)$$

   That's because otherwise the standard basis test for some $a \in [2^n]$ will have an expected negative ratio $\geq 2^{-2n}(\frac{1}{11}(1-\beta))^4 \geq 4 \cdot \text{threshold}$ (recall threshold $:= 2^{-2n-18}(1-\beta)^4$).

   A more detailed calculation is as follows.

$$\begin{aligned}\Pr[\text{negative ratio} < \text{threshold}] =\ & \Pr[\text{negative samples} < \text{threshold} \cdot \mathsf{poly}_1(2^n)] \\ \leq\ & \exp(-O(\mathbb{E}[\text{negative samples}])) \\ \leq\ & \exp\left(-O(\mathsf{poly}_1(2^n) \cdot 2^{-2n}((1-\beta)/11)^4)\right),\end{aligned}$$

   where the second step follows from the Chernoff bound, and the last step follows from

$$\text{threshold} \cdot \mathsf{poly}_1(2^n) < \frac{1}{4}\mathbb{E}[\text{negative samples}].$$

   Thus

$$\Pr[\text{negative ratio} \geq \text{threshold}] \geq 1 - 2^{-O(\mathsf{poly}_1(2^n) \cdot 2^{-2n}((1-\beta)/11)^4)}$$

2. After the coherency check, $\mathcal{C}$ has to satisfy the following property, otherwise the verifier will reject with probability $1 - 2^{-O(\mathsf{poly}(2^n))}$: for all $a, b \in [2^n]$, $a \neq b$,

$$\|\frac{1}{\sqrt{2}}(((\langle a| + \langle b|) \otimes I_t)(U^\dagger \otimes I)\mathcal{C}(\frac{1}{\sqrt{2}}(|a\rangle + |b\rangle) \otimes |0^t\rangle)\|^2 \geq 1 - 2^{-2n}(\frac{1}{11}(1-\beta))^4 \quad (8.12)$$

   The calculation is similar as the first step.

3. And from Eqs. (8.11) and (8.12), Lemma 8.44 implies that for all $|\psi\rangle \in \mathbb{C}^{2^n}$,

$$\|((\langle\psi| \otimes I_t)(U^\dagger \otimes I_t)\mathcal{C}|\psi, 0^t\rangle\|^2 \geq 1 - 10 \cdot 2^{n/2}(2^{-2n}(\frac{1}{11}(1-\beta))^4)^{1/4} > \beta.$$

   However, by the promise this is not possible to be in the no instance.

This completes the proof. □

Claim 8.45. $\mathsf{UMCSP}_{\alpha,\beta}$ *is in* NP *when only linear ancilla qubits are allowed and* $1-\alpha < 2^{-2n-20}(1-\beta)^4$ *and* $1-\beta \geq \mathsf{poly}(1/2^n)$ *(for example,* $1-\alpha = \mathsf{exp}(-2^n), 1-\beta = \mathsf{poly}(1/2^n)$*). However,* $\mathsf{UMCSP}_{\alpha,\beta}$ *is not trivially in* NP *in general.*

Proof. The certificate is the circuit implementation $\mathcal{C}$ that achieves Eq. (8.5). Now since the circuit only operates on a polynomial-dimension system, the unitary transformation of the whole circuit can be computed and written down using only a polynomial-time classical computer.

The subtlety is to verify whether the unitary computed here satisfies Eq. (8.5). We can prove it following the same way as the proof of Theorem 8.42. Here the quantum space is always polynomially bounded and a classical polynomial time verifier can simulate the protocol in the proof of Theorem 8.42 classically. (One note is the quantum output samples there can be lazy-sampled.) This completes the proof. □

Next, we showed that SMCSP has a QCMA protocol. Note that since SMCSP is given access to quantum states, it is even not a promise problem under the standard definition. Therefore, we can only say there is a QCMA protocol for this problem.

Theorem 8.46. $\mathsf{SMCSP}_{\alpha,\beta}$ *with gap* $|\alpha - \beta| \geq \mathsf{poly}(s)$ *has a* QCMA *protocol.*

Proof. We use the swap test to check whether the given states and the state generated from the certificate circuit are close. The verifier's algorithm is as follows:

Algorithm 32 The efficient verifier for SMCSP.

```
Input: $s, t \in \mathbb{N}$, $\mathsf{poly}(s)$ copies of $|\psi\rangle$, and quantum circuit $\mathcal{C}$.
1: Generate $\mathsf{poly}(s)$ $|\phi\rangle = \mathcal{C}|0\rangle$.
2: Apply swap test to $|\psi\rangle$ and $|\phi\rangle$.
3: return "Yes" if there are at least $\frac{a+b}{2}$ trials outputs 0.
4: return "No", otherwise.
```

Given $s, t \in \mathbb{N}$ and $\mathsf{poly}(s)$ copies of $|\psi\rangle$, we first consider the case where there exists a circuit $\mathcal{C}$ such that $\|(\langle\psi| \otimes I_t)\mathcal{C}|0^{n+t}\rangle\|^2 \geq \alpha$. Let $\mathcal{C}$ be the certificate. Then, by applying the swap test to $|\psi\rangle$ and $\mathcal{C}|0\rangle$, the probability that we get 0 (which means identical) is $\frac{1}{2} + \frac{|\langle\psi|\mathcal{C}|0\rangle|^2}{2}$, which is at least $\frac{1+\alpha}{2}$ in this case. We denote the probability of outputs 0 at the i-th trial as X_i. Then, By the Chernoff inequality,

$$\Pr\left[\sum_{i=1}^{\ell} X_i \geq (\frac{1}{2} + \frac{\alpha+\beta}{4})\ell\right] \leq \exp\left(-\frac{(\alpha-\beta)^2\ell}{16}\right).$$

Since $|\alpha - \beta| \geq \frac{1}{\mathsf{poly}(s)}$, the success probability of Algorithm 32 in this case is at least 2/3 by having $\ell = \mathsf{poly}(s)$ trials. Similarly, we can prove the case when there exists no circuit $\mathcal{C}$ such that $\|(\langle\psi| \otimes I_t)\mathcal{C}|0^{n+t}\rangle\|^2 > \beta$. This completes the proof.

□

Given Theorem 8.46, we can also obtain the following result when given classical descriptions of quantum states.

Corollary 8.47. SMCSP *with classical descriptions of quantum states as inputs is in* QCMA.

The subtlety is that the verifier needs to construct the state $|\psi\rangle$ given the classical description of $|\psi\rangle$. If the verifier can do this efficiently (in time $\mathsf{poly}(2^n)$), then the rest of the analysis follows the proof for Theorem 8.46. We leave the proof to Appendix 8.10.

For the ease of notation, we will simply denote $\mathsf{UMCSP}_{\alpha,\beta}$ and $\mathsf{SMCSP}_{\alpha,\beta}$ as UMCSP and SMCSP and will specify α and β when it is necessary in the rest of the section.

8.5.1 Reductions for UMCSP and SMCSP

In this section, we will show search-to-decision reductions for UMCSP and SMCSP. To prove the above results, it is easier for us to consider UMCSP and SMCSP as problems for computing the circuit complexity of given unitaries and states.

We first give formal definitions of approximating functions, unitaries, and states and the corresponding quantum circuit complexities.

Definition 8.33 (Approximating f with precision δ)**.** We say that a quantum circuit $\mathcal{C}$ that approximates a function $f : \mathbb{Z}^n \to \mathbb{Z}^m$ with precision δ if for all $x \in \mathbb{Z}^n$, there exists $\epsilon' \leq \epsilon$ such that

$$\mathcal{C}_{f,\delta}|x\rangle|0^t\rangle = \sqrt{1-\epsilon'}|f(x)\rangle|\psi_{f(x)}\rangle + \sqrt{\epsilon'}|\phi_x\rangle. \tag{8.13}$$

Definition 8.34 (Approximating U with precision δ)**.** Let U be as a $2^n \times 2^n$ unitary. We define $\mathcal{C}_{U,\epsilon}$ as the circuit that approximates U with precision δ such that for all $|\psi\rangle \in \mathbb{C}^{2^n}$ there exists $\delta' \leq \delta$

$$\mathcal{C}_{U,\delta}|\psi\rangle|0^t\rangle = \sqrt{1-\delta'}(U|\psi\rangle) \otimes |\psi'\rangle + \sqrt{\delta'}|\phi'\rangle.$$

Here, the additional t qubits for $\mathcal{C}_{U,\delta}$ are ancilla qubits.

Definition 8.35 (Approximating $|\psi\rangle$ with precision δ)**.** Let $|\psi\rangle \in \mathbb{C}^{2^n}$ be a quantum state. We define $\mathcal{C}_{|\psi\rangle,\epsilon}$ as the circuit that approximates $|\psi\rangle$ with precision δ

$$\mathcal{C}_{|\psi\rangle,\delta}|0^{n+t}\rangle = \sqrt{1-\delta'}|\psi\rangle|\psi'\rangle + \sqrt{\delta'}|\phi'\rangle$$

Here, $\delta' \leq \delta$ and the additional t qubits are ancilla qubits.

We use $CC(\cdot,\epsilon)$ to denote the quantum circuit complexity of the minimum quantum circuit that approximates the given Boolean functions, states, or unitaries with precision ϵ.

Remark 8.10 (Upper bounds on $CC(\cdot,\epsilon)$). For any universal gate set, any unitary U in $\mathbb{C}^{2^n \times 2^n}$ can be ϵ-approximated by a circuit with size at most $\widetilde{O}(n^2 2^{2n} \log \frac{1}{\epsilon})$ [NC11]. The same upper bound also holds for states. The existence of $2^{O(n)}$ upper bounds implies that $CC(\cdot,\epsilon)$ can be computed efficiently given efficient algorithms for SMCSP and UMCSP.

8.5.1.1 Search-to-decision reductions

In the following, we prove search-to-decision reductions for UMCSP and SMCSP. The main intuition for these reductions is that quantum circuits are reversible, which gives us the ability to do some "rewinding tricks". We define the search versions of UMCSP and SMCSP as follows:

Definition 8.36 ($\mathsf{SearchUMCSP}_\epsilon$)**.** Let $n, t \in \mathbb{N}$. Let $U \in \mathbb{C}^{2^n \times 2^n}$ be a unitary matrix and $\epsilon \in (0, 1)$. Let s be the smallest integer such that there exists a quantum circuit $\mathcal{C}$ of size s that uses at most t ancilla bits and for all $|\psi\rangle \in \mathbb{C}^{2^n}$,

$$\|(\langle\psi| \otimes I_t)(U^\dagger \otimes I_t)\mathcal{C}|\psi, 0^t\rangle\|^2 \geq 1 - \epsilon.$$

Given U, t, and ϵ, the problem is to output a circuit $\mathcal{C}'$ of size at most s that uses at most t ancilla bits and for all $|\psi\rangle \in \mathbb{C}^{2^n}$, $\|(\langle\psi| \otimes I_t)(U^\dagger \otimes I_t)\mathcal{C}'|\psi, 0^t\rangle\|^2 \geq 1 - \epsilon - 2^{-cn}$ for every constant $c > 0$.

Definition 8.37 ($\mathsf{SearchSMCSP}_{\epsilon,s}$)**.** Let $n, s, t \in \mathbb{N}$ and $\epsilon \in (0, 1)$. Let $|\psi\rangle \in \mathbb{C}^{2^n}$ be a quantum state with the promise that there exists a circuit $\mathcal{C}$ of size at most s and t ancilla bits such that

$$\|(\langle\psi| \otimes I_{n+t-1})\mathcal{C}|0^{n+t}\rangle\|^2 \geq 1 - \epsilon.$$

Given (n, s, t) in unary, ϵ, and access to arbitrary many copies of $|\psi\rangle$, the problem is to find a circuit $\mathcal{C}'$ of size at most s and t ancilla bits such that $\|(\langle\psi| \otimes I_{n+t-1})\mathcal{C}'|0^{n+t}\rangle\|^2 \geq 1 - \epsilon - 2^{-cn}$ for every constant $c > 0$.

Remark 8.11. In Definition 8.37, we have included the upper bound 1^s (the unary representation) as part of the inputs. This mainly follows from the fact that we are considering problems with copies of quantum states. One may expect that we can find s by using binary search with an efficient algorithm for SMCSP. However, efficient algorithms for SMCSP with $s = 2^n$ can run in time $\mathsf{poly}(2^n)$, and efficient algorithms for $\mathsf{SearchSMCSP}$ without 1^s as part of the inputs need to run in time

$\mathsf{poly}(n)$. Hence, this prevents us from finding s efficiently (in time $\mathsf{poly}(n)$) with an efficient algorithm for SMCSP (in time $\mathsf{poly}(s)$). On the other hand, if we consider the case where SMCSP and SearchSMCSP have the classical description of the state (instead of copies of the quantum state) as part of the inputs, then there is no need to have 1^s in the inputs of SearchSMCSP since we can find s via binary search with efficient algorithms for SMCSP.

In the following, we show search-to-decision reductions for UMCSP and SMCSP when $t = 0$ (i.e., no ancilla qubits)[23].

Theorem 8.48. *There exists a search-to-decision reduction for* UMCSP *for $t = 0$ (i.e., no ancilla qubits). In particular, if there is a time $T(n)$ algorithm for* $\mathsf{UMCSP}_{\alpha,\beta}$ *where $\alpha > 1 - 2^{-c_1 n}$ and $\alpha - \beta \geq 2^{-c_2 n}$ for every constants $c_1, c_2 > 0$, then there is a time* $\mathsf{poly}(T(n), 2^n)$ *algorithm for* $\mathsf{SearchUMCSP}_{\epsilon}$ *where $\epsilon \geq 2^{-c_3 n}$ for every constant $c_3 > 0$ and $t = 0$.*

Remark 8.12. Here we require the gap $\alpha - \beta$ in the decision UMCSP oracle to be at least $\mathsf{poly}(2^{-n})$ because our QCMA upper bound for UMCSP (see Theorem 8.42) only works in this regime.

Proof. Let us first state the reduction in the form of an algorithm with oracle queries to UMCSP as follows.

[23]In general, search-to-decision reductions for SMCSP and UMCSP mean that SearchSMCSP reduces to SMCSP and SearchUMCSP reduces to UMCSP for any $n, s, t \in \mathbb{N}$.

Algorithm 33 Search-to-decision reduction for UMCSP.

```
Input: $\epsilon \in (0,1)$, $U \in \mathbb{C}^{2^n \times 2^n}$, and a constant $c_3 > 0$.
 1: Let $U_0 = U$, $\Delta = 2^{-2c_3 n}$, $\epsilon_0 = \epsilon$ , and $\epsilon_i = \epsilon_0 + i \cdot \Delta$ for all $i \in \mathbb{N}$.
 2: Use the oracle $\mathsf{UMCSP}_{1-\epsilon_0, 1-\epsilon_0-\Delta}$ to binary-search $s$, the minimum circuit size of
    $U$.
 3: Set $i = 1$.
 4: while $i < s$ do
 5:     for all gates $h_i$ in $\mathcal{G}$ on all $n$ qubits do
 6:         if $\mathsf{UMCSP}_{1-\epsilon_i, 1-\epsilon_i-\Delta}(U_{i-1} h_i^\dagger, s-i) =$ Yes then
 7:             Set $g_i = h_i$.
 8:             Let $U_i = g_i^\dagger U_{i-1}$.
 9:             Set $i = i + 1$.
10:             Break.
11:         end if
12:     end for
13: end while
14: return $g_1, \ldots, g_s$.
```

We inductively prove the following claim.

Claim 8.49. *For every $0 < i < s$, at the i-th iteration in line 5, we know that there exists a circuit $\mathcal{C}$ of size at most $s - i + 1$ such that $\min_{|\psi\rangle} |\langle\psi|U_{i-1}^\dagger \mathcal{C}|\psi\rangle|^2 \geq 1 - \epsilon_i$.*

Proof. For the base case we consider $i = 1$ and note that after line 2 in Algorithm 33, we know that there exists a circuit $\mathcal{C}$ of size at most s such that $\min_{|\psi\rangle} |\langle\psi|U_0^\dagger \mathcal{C}|\psi\rangle|^2 \geq 1 - \epsilon - \Delta = 1 - \epsilon_1$. This proves the base case.

Now, suppose the induction statement holds for some i, we first claim that the algorithm must go into the if-loop in line 6. Note that by induction hypothesis there exists a circuit $\mathcal{C}$ of size at most $s-i+1$ such that $\min_{|\psi\rangle} |\langle\psi|U_{i-1}^\dagger \mathcal{C}|\psi\rangle|^2 \geq 1-\epsilon_i$. Let g_i be the last gate in $\mathcal{C}$, we know that $\min_{|\psi\rangle} |\langle\psi|(U_{i-1}^\dagger g_i)(g_i^\dagger \mathcal{C})|\psi\rangle\|^2 \geq 1 - \epsilon_i$ and $g_i^\dagger \mathcal{C}$ is a circuit of size at most $s-i+1-1 = s-i$. This shows that the algorithm will go into the if-loop in line 6 in the i-th iteration. Next, after the algorithm goes into line 6 in the i-th iteration, by the correctness of $\mathsf{UMCSP}_{1-\epsilon_i, 1-\epsilon_i-\Delta}$, we know that there is a circuit

$\mathcal{C}'$ $(= \mathcal{C}g_i^\dagger)$ of size at most $s - i$ such that $\min_{|\psi\rangle} |\langle\psi|U_i^\dagger \mathcal{C}'|\psi\rangle|^2 \geq 1 - \epsilon_i - \Delta = 1 - \epsilon_{i+1}$. This completes the induction step and hence proves Claim 8.49. □

Finally, with the same argument in the proof of Claim 8.49, we know that

$$\min_{|\psi\rangle} |\langle\psi|U^\dagger g_1 \cdots g_s|\psi\rangle|^2 \geq 1 - \epsilon_s = 1 - \epsilon - s \cdot 2^{-2c_3 n} \geq 1 - \epsilon - 2^{-c_3 n}$$

as desired. Also, notice that the algorithm only queries the UMCSP oracle at most 2^n times and hence the running time is $\mathsf{poly}(T(n), 2^n)$ where $T(n)$ is the running time of the UMCSP oracle. □

Theorem 8.50. *There exists a search-to-decision reduction for* SMCSP *for* $t = 0$. *In particular, if there is a time* $T(n)$ *algorithm for* $\mathsf{SMCSP}_{\alpha,\beta}$ *where* $\alpha > 1 - 2^{-c_1 n}$ *and* $\alpha - \beta \geq 2^{-c_2 n}$ *for every constants* $c_1, c_2 > 0$, *then there is a time* $\mathsf{poly}(T(n), s)$ *quantum algorithm for* $\mathsf{SearchSMCSP}_{\epsilon,s}$ *where* $\epsilon \geq 2^{-c_3 n}$ *for every constant* $c_3 > 0$ *and* $t = 0$.

Proof. The proof is similar to the proof for Theorem 8.48. We describe the reduction as follows:

Algorithm 34 Search-to-decision reduction for SMCSP.

```
Input: $s \in \mathbb{N}$, $\epsilon \in (0,1)$, access to copies of $|\psi\rangle$, and a constant $c_3 > 0$.
1: Let $|\psi_0\rangle = |\psi\rangle$, $\Delta = 2^{-2c_3 n}$, $\epsilon_0 = \epsilon$, and $\epsilon_i = \epsilon_0 + i \cdot \Delta$ for all $i \in \mathbb{N}$.
2: Use the oracle $\mathsf{SMCSP}_{1-\epsilon_0, 1-\epsilon-\Delta}$ to binary-search $s^* \leq s$, the minimum circuit
   size of $|\psi\rangle$.
3: Set $i = 1$
4: while $i < s^*$ do
5:     for all gates $h_i$ in $\mathcal{G}$ on all $n + t$ qubits do
6:         if $\mathsf{SMCSP}_{1-\epsilon_i, 1-\epsilon_i-\Delta}(|\psi_i\rangle, s^* - i)$ = Yes then
7:             Set $g_i = h_i$.
8:             Let $|\psi_i\rangle = g_i^\dagger |\psi_{i-1}\rangle$.
9:             Set $i = i + 1$.
10:            Break.
11:        end if
12:    end for
13: end while
14: return $g_1, \ldots, g_{s^*}$.
```

The analysis is similar to the proof of Theorem 8.48. Notice that given access to the quantum state $|\psi\rangle$, we can uncompute the gates using a quantum computer. Therefore, the search-to-decision reduction still holds. $\square$

Regarding SMCSP and SearchSMCSP which have the classical description of $|\psi\rangle$ as part of the inputs (instead of copies $|\psi\rangle$), we can also obtain the search-to-decision reduction following the same framework. The only difference is that the algorithm uncomputes the gates from the states by matrix-vector multiplication instead of applying the inverse of the gates on the states. The runtime of the matrix-vector multiplication is $\mathsf{poly}(2^n)$. Note that, as we have mentioned in Remark 8.11, SearchSMCSP in this case does not need to have the upper bound s in the inputs.

Corollary 8.51. *There exists a search-to-decision reduction for* SMCSP*, where the search and the decision problems are given the classical descriptions of the states in inputs.*

It is worth noting that Algorithm 34 and Algorithm 33 do not directly work when considering quantum circuits that are allowed to use ancilla qubits (i.e., $t > 0$). This follows from the fact that, based on definitions of UMCSP and SMCSP, a quantum circuit $\mathcal{C}$ that implements the target unitary or state can apply an arbitrary operator on the ancilla qubits, i.e., $C^\dagger(U \otimes I) \neq I$. In this case, we do not know the unitary of $\mathcal{C}$ or the state of $\mathcal{C}|0\rangle$, and thus we cannot run Algorithm 34 and Algorithm 33.

8.5.1.2 Self-reduction for SMCSP

In this section, we show that SMCSP is approximately self-reducible. In other words, one can approximate the circuit complexity of an n-qubit state by computing the circuit complexity of an $(n-1)$-qubit state.

Theorem 8.52. *Let $\mathcal{A}_\delta$ be an efficient algorithm for computing $CC(|\phi\rangle, \delta)$ for any $(n-1)$-qubit state $|\phi\rangle$. Let $|\psi\rangle$ be any n-qubit state. Given (n, s) in unary, $\epsilon \in (0, 1)$, and access to copies of $|\psi\rangle$, $CC(|\psi\rangle, \epsilon)$ can be approximated efficiently using $\mathcal{A}_\delta$.*

Recall that $CC(\cdot, \epsilon)$ denotes the quantum circuit complexity of the minimum quantum circuit that approximates the given states with precision ϵ.

Proof. We first fix the gate set to be $CNOT$ and all single-qubit rotations and prove the theorem under this particular gate set. Then, we generalize the theorem to all gate sets by the Solovay-Kitaev Theorem in Theorem B.1.

Let $|\psi\rangle \in \mathbb{C}^{2^n}$ be an arbitrary n-qubit quantum state. Without loss of generality, we can represent $|\psi\rangle$ as

$$c_0|0\rangle|\psi_0\rangle + c_1|1\rangle|\psi_1\rangle,$$

where $c_0, c_1 \in \mathbb{C}$ and $|c_0|^2 + |c_1|^2 = 1$. $|1\rangle$ and $|0\rangle$ are single-qubit states, and $|\psi_0\rangle$ and $|\psi_1\rangle$ are states on $n-1$ qubits and are not orthogonal in general. Our goal is show upper and lower bounds for $CC(|\psi\rangle, \epsilon)$ from $CC(|\psi_0\rangle, \delta)$ and $CC(|\psi_1\rangle, \delta)$.

To prove the upper and the lower bounds, we first estimate $|c_0|^2$ and $|c_1|^2$ to precision $\epsilon/4$ by using quantum amplitude estimation. We denote the estimated values as $|c_0'|^2$ and $|c_1'|^2$ and consider the following two cases.

1. $|c_0'|^2$ or $|c_1'|^2 < \epsilon/2$; and
2. $|c_0'|^2, |c_1'|^2 \geq \epsilon/2$.

Upper bound In case that $|c_0'|^2$ (or $|c_1'|^2$) is less than $\frac{\epsilon}{2}$, $|c_1|^2$ (or $|c_0|^2$) must be greater than $1 - \frac{3\epsilon}{4}$, which implies that the square of the inner product of $|\psi\rangle$ and $|1\rangle|\psi_1\rangle$ (or $|0\rangle|\psi_0\rangle$) is at least $1 - \frac{3\epsilon}{4}$. Therefore,

$$CC(|\psi\rangle, \epsilon) \leq CC(|\psi_1\rangle, \epsilon/4) \text{ or } CC(|\psi_0\rangle, \epsilon/4).$$

In case that both $|c_0'|^2$ and $|c_1'|^2$ are at least $\frac{\epsilon}{2}$, Let $\mathcal{C}_0 = \mathcal{C}_{|\psi_0\rangle,\epsilon}$ and $\mathcal{C}_1 = \mathcal{C}_{|\psi_1\rangle,\epsilon}$. Then, there exists $\mathcal{C}^*$ that approximates $|\psi\rangle$ with precision ϵ as follows:

$$\begin{aligned}
|0^n\rangle &\xrightarrow{R\otimes I_{n-1}} c_0|0\rangle|0^{n-1}\rangle + c_1|1\rangle|0^{n-1}\rangle \\
&\xrightarrow{control-\mathcal{C}_1} c_0|0\rangle|0^{n-1}\rangle + c_1|1\rangle\mathcal{C}_1|0^{n-1}\rangle \\
&\xrightarrow{X\otimes I_{n-1}} c_0|1\rangle|0^{n-1}\rangle + c_1|0\rangle\mathcal{C}_1|0^{n-1}\rangle \\
&\xrightarrow{control-\mathcal{C}_0} c_0|1\rangle\mathcal{C}_0|0^{n-1}\rangle + c_1|0\rangle\mathcal{C}_1|0^{n-1}\rangle \\
&\xrightarrow{X\otimes I_{n-1}} c_0|0\rangle\mathcal{C}_0|0^{n-1}\rangle + c_1|1\rangle\mathcal{C}_1|0^{n-1}\rangle
\end{aligned}$$

Here R is a single-qubit rotation gate that rotates $|0\rangle$ to $c_0|0\rangle + c_1|1\rangle$. Since our gate set includes all single-qubit rotations, the cost of R is just 1. For $control-\mathcal{C}_0$ and $control-\mathcal{C}_1$, we can think of it as every gate in $\mathcal{C}_i$ is controlled by an additional qubit, i.e., R becomes $control-R$ and CNOT becomes Toffoli gate. By the composition methods in [NC11], we can implement these control gates with only constant multiplicative overhead. Hence, $|\mathcal{C}^*| \leq k\cdot(|\mathcal{C}_0| + |\mathcal{C}_1|) + 3$ for some constant k, and we can conclude that

$$CC(|\psi\rangle,\epsilon) \leq k\cdot(CC(|\psi_0\rangle,\epsilon) + CC(|\psi_1\rangle,\epsilon)) + 3.$$

Lower bound Let $\mathcal{C}$ be the minimum quantum circuit that approximates $|\psi\rangle$ with precision ϵ.

When $|c_0'|^2$ and $|c_1'|^2$ are both at least $\epsilon/2$, $|c_0|^2$ and $|c_1|^2$ are at least $\epsilon/4$ where $|c_0'|^2$ is the estimated value of $|c_0|^2$. Intuitively, we can obtain $|\psi_0\rangle$ or $|\psi_1\rangle$ by parallelly applying $\mathcal{C}$ on $O(\frac{1}{\epsilon})$-many $|0^n\rangle$ states and measuring the first qubits of all the outputs states in the computational basis. By deferring all these measurements toward the end of the computation, we obtain

$$CC(|\psi_i\rangle,\epsilon') \leq k^*(CC(|\psi\rangle,\epsilon) + h)$$

for $i = 0, 1$, $h = O(1)$, and $k^* = O(1/\epsilon)$. Here $\epsilon \leq \epsilon' \leq (1 - \frac{\epsilon}{4})^{k^*} + \epsilon$. The additional constant cost h is from the overhead of deferring measurements.

When $|c_0'|^2$ or $|c_0'|^2$ is at least $1 - \epsilon/2$, the circuit for $|\psi\rangle$ is already a good approximation for $|\psi_1\rangle$ following the same reason for proving the upper bound in the same case. This implies that

$$CC(|\psi_i\rangle, 4\epsilon) \leq CC(|\psi\rangle, \epsilon).$$

The reduction The algorithm is as follows:

1. Estimating $|c_0|$ and $|c_1|$ with precision $\epsilon/4$.
2. Approximate $CC(|\psi\rangle, \epsilon)$ according to $|c_0'|$ and $|c_1'|$.
   - When $|c_0'|^2$ or $|c_1'|^2 \leq \frac{\epsilon}{2}$, compute $CC(|\psi_i\rangle, \epsilon/4)$ and $CC(|\psi_i\rangle, 4\epsilon)$ for $i = 0, 1$. Then,
   $$CC(|\psi_i\rangle, 4\epsilon) \leq CC(|\psi\rangle, \epsilon) \leq CC(|\psi_i\rangle, \epsilon/4).$$
   - When $|c_0'|^2, |c_1'|^2 \geq \epsilon/2$, compute $CC(|\psi_i\rangle, \epsilon')$ and $CC(|\psi_i\rangle, \epsilon)$ for $i = 0, 1$. Then,
   $$\frac{1}{k^*} \cdot \max_{i=0,1} \left(CC(|\psi_i\rangle, \epsilon')\right) - h \leq CC(|\psi\rangle, \epsilon) \leq k \cdot (CC(|\psi_0\rangle, \epsilon) + CC(|\psi_1\rangle, \epsilon)) + 3$$

For the running time of the reduction, we can estimate $|c_0|^2$ and $|c_1|^2$ with precision $\epsilon/4$ in time $\mathsf{poly}(1/\epsilon)$ using quantum amplitude estimation. In case that $|c_0'|^2$ (or $|c_1'|^2$) is less than $\frac{\epsilon}{2}$, we only need to compute $CC(|\psi_1\rangle, \epsilon/4)$ by having many enough copies of $|\psi_1\rangle$, which can be efficiently obtained by measuring $|\psi\rangle$. In case that both $|c_0'|^2$ and $|c_1'|^2$ are at least $\frac{\epsilon}{2}$, $|c_0|$ and $|c_1|$ must be at least $\frac{\epsilon}{4}$. Then, we can still obtain sufficiently many copies of $|\psi_0\rangle$ and $|\psi_1\rangle$ in time $\mathsf{poly}(\frac{1}{\epsilon})$ to compute $CC(\psi_0, \epsilon)$ and $CC(\psi_1, \epsilon)$.

Finally, we generalize the results above to arbitrary universal gate set by applying the Solovay-Kitaev Theorem. This gives upper bounds multiplicative overhead $\mathsf{polylog}\,\frac{CC(|\psi_i\rangle,\delta)}{\epsilon}$ and lower bounds multiplicative overhead $\mathsf{polylog}^{-1}\,\frac{CC(|\psi_i\rangle,\delta)}{\epsilon}$, where the choices of i and δ depend on the cases.

□

Remark 8.13. Theorem 8.52 also holds when the problem is given the classical description of the quantum state. When considering the version with classical descriptions of states, the reduction becomes even simpler since c_0 and c_1 can be easily computed from the input.

8.5.1.3 Reducing MQCSP to UMCSP

In the following, we present a reduction from MQCSP to UMCSP. We first introduce a unitary that trivially encode a given Boolean function.

Definition 8.38 (Trivial unitary encoding of Boolean functions (U_f))**.** Let $f : \mathbb{Z}^n \to \mathbb{Z}^m$. We define U_f as a $2^{n+m} \times 2^{n+m}$ unitary such that for all $x \in \mathbb{Z}^n$

$$U_f|x\rangle|0\rangle = |x\rangle|f(x)\rangle$$

Obviously, given the truth table of a function $f : \{0,1\}^n \to \{0,1\}^m$, one can compute U_f in time $\mathsf{poly}(2^n)$. Then, one might expect that the circuit complexity of f is equal to of U_f (in Definition 8.38). However, this is not the case in general since there are many unitaries that can compute f without the form of U_f. In the following lemma, we show that one can give both upper and lower bounds for $CC(f)$ by the quantities $CC(U_f, \epsilon)$ and $CC(U_f, 2\epsilon)$

Lemma 8.53.

$$\frac{CC(U_f, 2\epsilon)}{2} - m \leq CC(f, \epsilon) \leq CC(U_f, \epsilon)$$

Proof. It is easy to see that given $\mathtt{tt}(f)$, one can compute U_f in time $2^{O(n+m)}$ which is polynomial in $|T(f)| = 2^{n+m}$.

We first consider the case where $CC(f)$ and $CC(U_f)$ can be computed with probability 1. We can prove the first inequality as follows:

$$\begin{aligned} |x\rangle|0\rangle &\xrightarrow{\mathcal{C}_f} e^{-i\theta_x}|f(x)\rangle|\psi_x\rangle \\ &\xrightarrow{copy} e^{-i\theta_x}|f(x)\rangle|f(x)\rangle|\psi_x\rangle \\ &\xrightarrow{\mathcal{C}_f^\dagger} |f(x)\rangle|x\rangle|0\rangle, \end{aligned} \tag{8.14}$$

where $e^{-i\theta}$ are the global coefficient that C_f might have for each θ_x. $C_f^\dagger(copy)C_f$ perfectly computes U_f on all $x \in \{0,1\}^n$ without any global coefficient. This implies that for all $|\psi\rangle \in \mathbb{C}^{2^n}$, $C_f^\dagger(copy)C_f$ computes $U_f|\psi\rangle$ perfectly. The cost for applying this circuit is $2CC(f)+m$. Therefore, we can conclude that $CC(U_f) \leq 2CC(f)+m$. The second inequality is true since a circuit for implementing U_f is also a circuit for f by definition. Note that the global phase in Eq. (8.14) can be absorbed into the second register; however, we write it down here to help explain why $C_f^\dagger(copy)C_f$ implements U_f not just only on the computational basis, but on all the states.

In the following, we consider the case where we allow U_f and f to be computed with probability at least some thresholds.

$$\begin{aligned} |x\rangle|0\rangle &\xrightarrow{\mathcal{C}_{f,\epsilon}} \sqrt{1-\epsilon}|f(x)\rangle|\psi_{f(x)}\rangle + \sqrt{\epsilon}(\sum_{y\neq f(x)} c_y|y\rangle|\phi'_{x,y}\rangle) \\ &\xrightarrow{Copy} \sqrt{1-\epsilon}|f(x)\rangle|f(x)\rangle|\psi_{f(x)}\rangle + \sqrt{\epsilon}(\sum_{y\neq f(x)} c_y|y\rangle|y\rangle|\phi'_{x,y}\rangle) \\ &\quad = |f(x)\rangle(\sqrt{1-\epsilon}|f(x)\rangle|\psi_{f(x)}\rangle + \sqrt{\epsilon}(\sum_{y\neq f(x)} c_y|y\rangle|\phi'_{x,y}\rangle)) \\ &\qquad + \sqrt{\epsilon}(\sum_{y\neq f(x)} c_y|y\rangle|y\rangle|\phi'_{x,y}\rangle - \sum_{y\neq f(x)} c_y|f(x)\rangle|y\rangle|\phi'_{x,y}\rangle) \\ &\xrightarrow{\mathcal{C}_{f,\epsilon}^\dagger} |f(x)\rangle|x\rangle|0\rangle + |\psi'_x\rangle. \end{aligned} \tag{8.15}$$

Since $\langle f(x), x, 0|\psi'_x\rangle = -\epsilon$ and $\langle \psi'_x|\psi'_x\rangle = 2\epsilon$, we have that

$$|\psi'_x\rangle = -\epsilon|f(x), x, 0\rangle + \sqrt{2\epsilon - \epsilon^2}|\psi''_x\rangle.$$

Therefore, we can rewrite Eq. (8.15) as

$$(1-\epsilon)|f(x), x, 0\rangle + \sqrt{2\epsilon - \epsilon^2}|\psi''_x\rangle,$$

which implies that the circuit $C^\dagger_{f,\epsilon}(Copy)C_{f,\epsilon}$ can compute U_f with probability $(1-\epsilon)^2 < 1 - 2\epsilon$, i.e., $CC(U_f, 2\epsilon) \leq 2CC(f, \epsilon) + m$. $CC(f, \epsilon) \leq CC(U_f, \epsilon)$ is also trivial by the definition.

□

We describe an algorithm to approximate $CC(f)$ given an oracle to UMCSP.

Algorithm 35 A reduction from MQCSP to UMCSP

```
Input: Given $\mathsf{tt}(f)$ for $f : \{0,1\}^n \to \{0,1\}^m$
1: Construct $U_f$.
2: Use UMCSP oracle to compute $s = CC(U_f)$.
3: return $(\frac{s}{2} - m, s)$.
```

Theorem 8.54. $\mathsf{MQCSP}[s/2 - 1, s] \leq \mathsf{UMCSP}$.

Proof. By Lemma 8.53, $CC(f, \epsilon)$ is between $\frac{CC(U_f, 2\epsilon)}{2} - 1$ and $CC(U_f, \epsilon)$ when f is a Boolean function. To compute $CC(U_f, \epsilon)$, we can use the oracle for $\mathsf{UMCSP}_{1-\epsilon,\beta}$, where $\beta \leq 1 - \epsilon - \frac{1}{\mathsf{poly}}$. For $CC(U_f, 2\epsilon)$, we use the oracle for $\mathsf{UMCSP}_{1-2\epsilon,\beta'}$, where $\beta' \leq 1 - 2\epsilon - \frac{1}{\mathsf{poly}}$. This completes the proof. □

Remark 8.14. One may expect that we can use Algorithm 35 and NP-hardness result about $\mathsf{multiMCSP}$ to prove NP-hardness of UMCSP. However, since the reduction for the multioutput MCSP generates functions with exponential-size output string, it make the first inequality in Lemma 8.53 fail. Therefore, whether UMCSP is NP-hard or not is still open.

8.5.2 Applications of SMCSP and UMCSP

In this part, we give applications of UMCSP and SMCSP to other fields in computer science and physics. For SMCSP, we focus on the version with multiple quantum states as inputs.

8.5.2.1 Applications of UMCSP

A question Aaronson raised in [Aar16] is whether there exists an efficient quantum process that generates a family of unitaries that are indistinguishable from random unitaries given the full description of the unitary. Obviously, if we can solve UMCSP efficiently, we can distinguish truly random unitaries from unitaries generated from efficient quantum process.

Theorem 8.55. *If* UMCSP *has efficient (quantum) algorithms, then there is no efficient quantum process that generates a family of unitaries indistinguishable from random unitaries given the full description of the unitary.*

Besides, some results about MQCSP in Section 8.4 also hold for UMCSP by Theorem 8.54 and Algorithm 35. In the following, we list some results that trivially holds.

Corollary 8.56. *If* UMCSP $\in$ BQP*, then there is no* qOWF*.*

Corollary 8.57. *If there exists a quantum-secure* $i\mathcal{O}$*, then* UMCSP $\in$ BQP *implies* NP $\subseteq$ coRQP*.*

Corollary 8.58. *Assume* UMCSP $\in$ BQP*. Then, there exists a* BQP *algorithm that, given the truth-table of an n-variable Boolean function of quantum circuit complexity $2^{\Omega(n)}$, output $2^{\Omega(n)}$ Boolean functions on $m = \Omega(n)$ variables each, such that all of the output functions have quantum circuit complexity greater than $\frac{2^m}{(c+2)m}$ for any $c > 0$.*

Corollary 8.56, Corollary 8.57 and Corollary 8.58 hold since we use the MQCSP oracle as a distinguisher to distinguish functions whose sizes have a large gap, i.e., functions with quantum circuit complexity $\mathsf{poly}(n)$ from functions with quantum circuit complexity $2^{\Omega(n)}$. As the UMCSP oracle can solve $\mathsf{MQCSP}[\frac{s}{2}-1, s]$, the existence of efficient algorithms for UMCSP also implies the same results.

Corollary 8.59. *If* $\mathsf{UMCSP} \in \mathsf{BQP}$*, then* $\mathsf{BQE} \not\subset \mathsf{BQC}[n^k]$ *for all constant* $k \in \mathbb{N}$*.*

Corollary 8.59 holds because for the gap version of MQCSP with a constant gap, it gives a *promise* BQP-natural property, which is defined in [AGG+20]. Suppose we have an efficient quantum algorithm for solving $\mathsf{MQCSP}[2^{\epsilon n}/2-1, 2^{\epsilon n}]$ for small constant ϵ, then it will reject any function with quantum circuit complexity less than $2^{\epsilon n}/2$ and will accept another large subset of functions with quantum circuit complexity larger than $2^{\epsilon n}$. Then, we can use the technique in [AGG+20] to construct the hard language $\mathcal{L}$ from the quantum PRG (Theorem 8.25) and promise quantum natural property. The remaining proof of Theorem 8.24 will work after this adaptation.

8.5.2.2 Pseudorandom states

An efficient algorithm for SMCSP gives an efficient distinguisher for separating states with large circuit complexity from states with small circuit complexity given many copies of the state. Obviously, this gives us a way to distinguish random states from states that are generated from some efficient process.

Definition 8.39 (Pseudorandom states (PRS) ([JLS18]))**.** Let κ be the security parameter. Let K be the key space and $\mathcal{H}$ be the state space both parameterized by κ. A family of quantum states $\{|\psi_k\rangle\}_{k\in K} \subset \mathcal{H}$ is pseudorandom if the following properties hold.

1. **Efficiency:** There is a quantum polynomial-time algorithm G that given $k \in K$, can generate $|\psi_k\rangle$.

2. **Indistinguishability:** For all quantum polynomial-time algorithm $\mathcal{A}$ and any $m = \mathsf{poly}(\kappa)$

$$|\Pr_k[\mathcal{A}(|\psi_k\rangle) = 1] - \Pr_{|\psi\rangle \leftarrow \mu}[\mathcal{A}(|\psi\rangle) = 1]| \leq \mathsf{negl}(\kappa),$$

where μ is the Haar measure on $\mathcal{H}$.

Theorem 8.60. *If* $\mathsf{SMCSP} \in \mathsf{BQP}$*, then there is no* PRS *and* qOWF*.*

Proof. Let $|\psi\rangle$ be the state and $\mathcal{A}$ be the algorithm to distinguish whether $|\psi\rangle$ is a truely random state or from a particular efficient algorithm. In the definition of PRS, $\mathcal{A}$ knows the algorithm for constructing the PRS (but it does not know the key.) Therefore, $\mathcal{A}$ also knows the circuit complexity s for generating the PRS $|\psi\rangle$. Suppose $|\psi\rangle$ is an n-qubit PRS generated by a quantum circuit with size s, by solving SMCSP with size parameter s and $\mathsf{poly}(s)$ copies of $|\psi\rangle$, the adversary can distinguish $|\psi\rangle$ from a Haar random state with high probability since a Haar random state has complexity exponential in n.

Finally, by [JLS18], there exist PRS assuming the existence of qOWF. Since we can break any PRS scheme by solving SMCSP, we can also invert any qOWF by solving SMCSP.

□

8.5.2.3 Estimating the wormhole volume

Integrating general relativity and quantum mechanics into a comprehensive theorem for quantum gravity is one of the most challenging physics problems. The AdS/CFT correspondence plays an important role in this line of research. The AdS/CFT correspondence conjectures the duality between the Anti-de Sitter space (i.e., the bulk) and a conformal field theory (i.e., the boundary). In particular, it conjectures the dictionary maps from wormholes and operators in the bulk to quantum states and operators on the boundary. One fascinating puzzle in Ads/CFT correspondence is about the volume of the wormhole. The volume of the wormhole grows steadily with time; what is the quantity of the corresponding quantum state on the boundary that has this feature? Susskind proposed the *Complexity=Volume Conjecture* [Sus16]. It states that the wormhole volume equals the quantum circuit complexity of the corresponding quantum state times some constant c. In the following, we give a brief description of the Complexity=Volume Conjecture and related backgrounds. One can see [Sus16, BFV20] for detailed discussions.

AdS/CFT Correspondence AdS/CFT correspondence conjectures a dual map Φ between wormholes (AdS side) and quantum systems (CFT side). The setting we consider here is wormholes with two-sided blackholes. Under this setting, the CFT side is divided into left and right systems denoted by Hamiltonians H_L and H_R, where the left and right CFT systems are on n qubits (compatible with the entropy of the wormhole 2^n). We denote the whole system (with both left and right systems) as $H = H_L + H_R$. An early model of AdS/CFT goes under the ER=EPR slogan: the wormhole (Einstein-Rosen Bridge) is dual to maximally entangled (EPR) pair. The corresponding state is usually called the thermal field double (TFD) state $|TFD\rangle$ [MS13]

$$|TFD\rangle = \frac{1}{\sqrt{2^n}} \sum_i e^{-E_i/\beta} |i\rangle_L |i\rangle_R, \tag{8.16}$$

where $|i\rangle_L$ and $|i\rangle_R$ are energy eigenstates of H_L and H_R.

The quantum state after time-t evolution is

$$|TFD(t)\rangle = e^{-iHt}|TFD\rangle.$$

Recall the dual map Φ between a wormhole (AdS side) and a quantum system (CFT side), one can represent the wormhole after time t as $\Phi(e^{-iHt}|TFD\rangle)$ (and view $\Phi(|TFD\rangle)$ as the wormhole at time 0).

The statement of Complexity=Volume Conjecture can be stated as follows:

Conjecture 8.61 (Complexity=Volume Conjecture [Sus16])**.** Consider a wormhole and its corresponding CFT system H, for some suitable ϵ, c, and $0 \leq t \leq O(2^n)$,

$$CC_\epsilon(|TFD\rangle, |TFD(t)\rangle) = c \cdot Volume(\Phi(e^{-iHt}|TFD\rangle)),$$

where $CC_\epsilon(|TFD\rangle, |TFD(t)\rangle)$ is the circuit complexity for constructing $|TFD(t)\rangle$ from $|TFD\rangle$ with at most ϵ error.

The SMCSP oracle gives a way to identify the quantum circuit complexity of the given state. This implies that if the dictionary map between the wormhole and the quantum state is efficient, one can estimate the wormhole volume in two ways. 1) Apply the dictionary map to transfer the wormhole to the corresponding state and then apply the SMCSP oracle for the circuit complexity, which gives the wormhole volume. 2) As it is hard to imagine mapping wormholes to states, one can view the SMCSP oracle as a POVM and then uses the dictionary map to transfer the POVM to the corresponding operators in the bulk to measure the volume. This gives the following lemma.

Theorem 8.62. *Assuming the Volume=Complexity Conjecture, if the dictionary map can be computed in quantum polynomial time and* $\mathsf{SMCSP} \in \mathsf{BQP}$*, then one can estimate the wormhole volume in quantum polynomial time when the volume is at most polynomially large.*

Here, we require the volume is at most polynomially large. This follows from the fact that we need a upper bound polynomial in n for doing binary search to find the circuit complexity with an efficient SMCSP algorithm. If the upper bound is $2^{O(n)}$, the running time of the SMCSP algorithm can be $\mathsf{poly}(2^n)$. Therefore a quantum polynomial-time algorithm for SMCSP in this case would not imply a quantum polynomial-time algorithm for estimating the wormhole's volume.

Besides, recall that the wormhole is initially described by $|TFD\rangle$. So, we also need to modify the definition of SMCSP to allow such an initial state.

Bouland et al. in [BFV20] used this correspondence in a reverse way. In particular, they showed that if the dictionary map and simulating the state in the bulk are efficient (i.e., the quantum Extended Church-Turing thesis holds for quantum gravity), then one can efficiently distinguish certain PRS from Haar random state by mapping the state to the wormhole in the bulk and do the simulation in the bulk to estimate the volume. Following this idea, we can also conclude that if there is a quantum polynomial time algorithm for estimating the wormhole's volume, then one can compute the circuit complexity of the corresponding quantum state efficiently assuming the the Volume=Complexity Conjecture and that the dictionary map is efficient[24].

8.5.2.4 Succinct state tomography

In the following, we show that solving SMCSP can help to have a succinct answer to state tomography for states which are generated from a polynomial-size circuit without any measurement.

Definition 8.40 (Succinct state tomography). Let $|\psi\rangle$ be an n-qubit quantum state that is generated from a quantum circuit $\mathcal{C}$ of size s without using measurement and ancilla qubits. Given $\mathsf{poly}(n)$ copies of $|\psi\rangle$ and an upper bound s' where $s \leq s' \leq \mathsf{poly}(n)$, the problem is to output a succinct description (e.g., $\mathcal{C}$) of $|\psi\rangle$.

Theorem 8.63. *Succinct state tomography in Def. 8.40 reduces to* SMCSP.

Proof. Obviously, succinct state tomography reduces to the search version of SMCSP. By the search-to-decision reduction in Theorem 8.50, we can solve succinct state tomography by solving SMCSP. □

[24]Note that this does not give an efficient algorithm for solving SMCSP in general since it can only solve SMCSP for CFT states.

8.6 Proof for the Hardness of MQCSP

Theorem 8.14. $\mathsf{MQCSP} \in \mathsf{QCMA}$.

Proof. The certificate is still the classical description of a quantum circuit $\mathcal{C}$ that has size at most s and operates on at most $n+t$ qubits. The verifier first implements $\mathcal{C}$. Then, the verifier repeats evaluating $\mathcal{C}|x, 0^t\rangle$ and measuring the first qubit $\ell = \mathsf{poly}(2^n)$ times. We denote the measurement outcomes of the ℓ trials as binary random variables $X_1, \ldots, X_\ell$ which are all independent. Finally, the verifier checks if for all $x \in \{0,1\}^n$, there are at least $\frac{\alpha+\beta}{2}$ of the outcomes are consistent with $f(x)$.

For the yes instance, we have the promise that $\|(\langle f(x)| \otimes I_{n+t-1})\mathcal{C}|x, 0^t\rangle\|^2 \geq \alpha$ for all $x \in \{0,1\}^n$. Let $X = \sum_{i=1}^n X_i$. By using the second statement of Chernoff inequality, we have that $\Pr[X \leq \frac{(\alpha+\beta)\ell}{2}] \leq \exp\left(-\frac{(\alpha+\beta)^2\ell}{8\alpha}\right)$. By setting $\ell = \mathsf{poly}(2^n)$, we obtain $\Pr[X \leq \frac{(\alpha+\beta)\ell}{2}] \leq e^{-\mathsf{poly}(2^n)}$. This implies that $\Pr[X \geq \frac{(\alpha+\beta)\ell}{2}$ for all $x \in \{0,1\}^n] \geq 1 - e^{-\mathsf{poly}(2^n)}$. For the no instance, we can do the similar analysis using Chernoff bound and show that there exists $x \in \{0,1\}^n$ such that $\Pr[X \geq \frac{(\alpha+\beta)\ell}{2}]$ is negligible. $\square$

Theorem 8.15. $\mathsf{SZK} \subseteq \mathsf{BPP}^{\mathsf{MQCSP}}$

Proof of Theorem 8.15. Let (n, C_0, C_1) be a PIID instance, where $C_0, C_1 : \{0,1\}^m \to \{0,1\}^{m'}$ of size n^k. For $b = 0,1$ and $x \in \{0,1\}^m$, we let $f_b(x) = C_b(x)$. Then, similar to the proof for Theorem 8.20, the idea is using f_b to construct a pseudorandom generator $\hat{G}$ and break $\hat{G}$ by applying the MQCSP oracle. Specifically, the algorithm is as follows:

Algorithm 36 A PPT algorithm $\mathcal{A}$ for PIID with MQCSP oracle

```
Input: $C_0, C_1$ of size $n^k$ and $m$-qubit input.
1: Pick $x$ uniformly randomly from $\{0,1\}^m$.
2: Compute $f_0(x)$.
3: Use $f_0(x)$ to generate a pseudorandom string $G_{f_0(x)}(r)$ as in Lemma 8.18.
4: Use $G_{f_0(x)}(r)$ to generate the truth table $\mathsf{tt}(g) = \hat{G}(r)$ as in Lemma 8.19.
5: Apply the inverting algorithm $\mathcal{A}_{inv}^{\mathsf{MQCSP}}$ with access to function $f_1$ in Theorem 8.20
   to invert $f_1$ for $x'$. Note that the function used in the inverting algorithm is $f_1$
   instead of $f_0$.
6: return "Yes" if $C_0(x) = C_1(x')$; "No" if $C_0(x) \neq C_1(x')$.
```

In Algorithm 36, we do not explicitly describe the inverting algorithms $\mathcal{A}_{inv}$. However, based on Theorem 8.20, such algorithms must exist.

Then, when (C_0, C_1) is a no instance, i.e., $\Pr_{x\in\{0,1\}^m}[\exists y \in \mathsf{Im}(C_0)$ such that $C_1(x) = y] \leq \frac{1}{2^n}$, the probability that there exists x' such that $C_1(x') = C_0(x)$ over x is at most $1/2^n$. In this case, Algorithm 36 outputs "Yes" with probability at most $1/2^n$.

When (C_0, C_1) is a yes instance, C_0 and C_1 has statistical distance $1/2^n$ over $x \in \{0,1\}^m$. Then, the success probability of the algorithm $\mathcal{A}$ in Algorithm 36 is

$$\begin{aligned}\Pr[\mathcal{A}(C_0, C_1) = \text{“Yes”}] &= \Pr_x[f_1(\mathcal{A}_{inv}^{\mathsf{MQCSP}}(f_1, f_0(x))) = f_0(x)] \\ &= \sum_{y\in\{0,1\}^{m'}} \Pr_x[f_0(x) = y] \Pr_x[f_1(\mathcal{A}_{inv}^{\mathsf{MQCSP}}(f_1, y)) = y | y]\end{aligned}$$

Note that if we compute $f_1(x)$ (instead of $f_0(x)$) at step 2 in Algorithm 36, then the success probability of $\mathcal{A}$ is

$$\begin{aligned}\Pr_x[f_1(\mathcal{A}_{inv}^{\mathsf{MQCSP}}(f_1, f_1(x))) = f_1(x)] &= \sum_{y\in\{0,1\}^{m'}} \Pr_x[f_1(x) = y] \Pr_x[f_1(\mathcal{A}_{inv}^{\mathsf{MQCSP}}(f_1, y)) = y | f_1(x) = y] \\ &\geq 1/\mathsf{poly}(n).\end{aligned}$$

The last inequality follows from Theorem 8.20. The MQCSP oracle can break $\hat{G}$ due to the fact that the construction of $\hat{G}$ is a small classical circuit and thus also a small quantum circuit. Therefore, we can use the MQCSP oracle to distinguish it from a truely random string.

The difference between these two probabilities above is

$$\begin{aligned}
&\Pr_x[f_1(\mathcal{A}_{inv}^{\mathsf{MQCSP}}(f_1, f_0(x))) = f_0(x)] - \Pr_x[f_1(\mathcal{A}_{inv}^{\mathsf{MQCSP}}(f_1, f_1(x))) = f_1(x)] \\
&= \sum_y \Pr_x[f_1(\mathcal{A}_{inv}^{\mathsf{MQCSP}}(f_1, y)) = y | f_1(x) = y](\Pr_x[f_0(x) = y] - \Pr_x[f_1(x) = y]) \\
&\leq \sum_y (\Pr_x[f_0(x) = y] - \Pr_x[f_1(x) = y]) \leq \frac{1}{2^n}.
\end{aligned}$$

The last inequality follows from the definition of statistical distance. Therefore, Algorithm 36 succeeds with probability at least $1/\mathsf{poly}(n) - 2^{-n}$ for a "Yes" instance. Finally, we can amplify the success probability for the yes instance to 2/3 by repetition. Thus, $\mathsf{PIID} \in \mathsf{BPP}^{\mathsf{MQCSP}}$.

□

8.7 Learning Theory

In this section, we provide the details of Section 8.4.2 on the connection between learning theory and MQCSP.

8.7.1 PAC learning

Let us recall the definition of PAC learning.

Definition 8.22 (PAC learning over the uniform distribution with membership queries)**.** Let C be a circuit class and let $\epsilon, \delta > 0$. We say an algorithm (ϵ, δ)-PAC-learns C over the uniform distribution with membership queries if the following hold. For every $n \in \mathbb{N}$ and n-variate $f \in \mathsf{C}$, given membership query access to f, the algorithm outputs a circuits C such that with probability at least $1 - \delta$ over its internal randomness, we have $\Pr_{x \in \{0,1\}^n}[f(x) \neq C(x)] < \epsilon$. The running time of the learning algorithm is measured as a function of $n, 1/\epsilon, 1/\delta$ and, $\mathsf{size}(f)$.

The following theorem shows that efficient PAC-learning for $\mathsf{BQP/poly}$ is equivalent to efficient algorithms for MQCSP. Here, $\mathsf{BQP/poly}$ is defined as $\bigcup_{s \leq \mathsf{poly}(n)} \mathsf{BQC}(s)$

Theorem 8.22 (Equivalence of efficient PAC learning for $\mathsf{BQP/poly}$ and efficient randomized algorithm for MQCSP)**.**

- *If* $\mathsf{MQCSP} \in \mathsf{BPP}$*, then there is a randomized algorithm that* $(1/\mathsf{poly}(n), \delta)$*-PAC learns* $f \in \mathsf{BQP/poly}$ *under the uniform distribution with membership queries for every* $\delta > 0$*. Specifically, the algorithm runs in quasi-polynomial time.*
- *If there is a randomized algorithm that* $(1/\mathsf{poly}(n), \delta)$*-PAC learns* $f \in \mathsf{BQP/poly}$ *under the uniform distribution with membership queries for some* $\delta > 0$ *in* $2^{O(n)}$ *time, then we have* $\mathsf{MQCSP}[\mathsf{poly}(n), \omega(\mathsf{poly}(n)), \mathsf{poly}(n), \tau] \in \mathsf{BQP}$ *and* $\mathsf{MQCSP}[\mathsf{poly}(n), \omega(\mathsf{poly}(n)), O(n), \tau] \in \mathsf{BPP}$ *for every* $\tau > 0$*.*

Proof.

- The key ingredient to show $\mathsf{MQCSP} \in \mathsf{BPP}$ implies efficient PAC learning for $\mathsf{BQP/poly}$ is the "learning from a natural property" framework by [CIKK16]. First, note that $\mathsf{BQP/poly}$ is a circuit class that contains $\mathsf{P/poly}$ and hence can implement both the *Nisan Wigderson generator* and the *Direct Product + Goldreich-Levin amplification.* Second, $\mathsf{MQCSP} \in \mathsf{BPP}$ implies there is a BPP-natural property against $\mathsf{BQP/poly}$. Finally, by Theorem 5.1 of [CIKK16], there is a randomized algorithm that $(1/\mathsf{poly}(n), \delta)$-PAC learns $f \in \mathsf{BQP/poly}$ under the uniform distribution with membership queries for every $\delta > 0$ in quasipolynomial time.

- Let ALG be a randomized algorithm that $(1/\mathsf{poly}(n), \delta)$-PAC learns $f \in \mathsf{BQP/poly}$ under the uniform distribution with membership queries for some $\delta > 0$. We design the following randomized algorithm for $\mathsf{MQCSP}[\mathsf{poly}(n), \omega(\mathsf{poly}(n)), t(n), \tau]$ where $t(n)$ is the number of ancilla bits that will be determined later. For every $\tau > 0$, let $\epsilon = \tau/2$.

Algorithm 37 A quantum algorithm for $\mathsf{MQCSP}[\mathsf{poly}(n), \omega(\mathsf{poly}(n)), t(n), \tau]$

```
Input: The truth table $T$ of a $n$-variate Boolean function $f$.
1: for $i = 1, \dots, 10\lceil \log 1/\delta \rceil$ do
2:     Run $ALG$ and supply the membership query with the truth table $T$. Let $C_i$ be the output of $ALG$.
3:     Uniformly and independently sample $x_1, \dots, x_\ell \in \{0,1\}^n$ where $\ell = \lceil 100 \log(1/\delta)/\epsilon^2 \rceil$.
4:     if $|\{j \in [\ell] : C_i(x_j) \neq f(x_j)\}| < \frac{\epsilon}{10} \cdot \ell$ then
5:         Break and output "Yes".
6:     end if
7: end for
8: Output "No".
```

Let us analyze the correctness of the above algorithm. First, if f is an Yes instance, i.e., there exists a polynomial size quantum circuit C that computes f, then due to the correctness of ALG, $\Pr_{C_i}[|\{x \in \{0,1\}^n : C_i(x) \neq f(x)\}| < 2^n/\mathsf{poly}(n)] > \delta$ for each i. Namely, with probability at least $9/10$, there exists an $i \in [10\lceil \log 1/\delta \rceil]$ such that $|\{x \in \{0,1\}^n : C_i(x) \neq f(x)\}| < 2^n/\mathsf{poly}(n)$. For this specific i, by Chernoff bound, with probability at least $9/10$ the algorithm will go to line 5 and output "Yes". That is, the above algorithm accepts an Yes instance with probability at least $2/3$ as desired.

Next, if f is a No instance, i.e., for every polynomial size quantum circuit C, we have $|\{x \in \{0,1\}^n : C(x) \neq f(x)\}| \geq \tau \cdot 2^n > \epsilon \cdot 2^n$. For each $i \in [10\lceil \log 1/\delta \rceil]$, C_i is a polynomial size circuit and hence by Chernoff bound, the algorithm goes to line 5 with probability at most $2^{-\Omega(\epsilon^2 \ell)}$. Due to the choice of ℓ, we know that the algorithm will output "No" with probability at least $2/3$. That is, the above algorithm rejects an No instance with probability at least $2/3$ as desired.

Finally, the running time of the algorithm is $\mathsf{poly}(\text{Time}(ALG), 1/\delta, 1/\epsilon, n, m)$ where the dependency on $\mathsf{poly}(n, m)$ is for calculating $C_i(x_j)$ using the quantumness. Note that this running time is $\mathsf{poly}(2^n)$ and hence we conclude that $\mathsf{MQCSP}[\mathsf{poly}(n), \omega(\mathsf{poly}(n)), t(n)] \in \mathsf{BQP}$.

When the number of ancilla bits is $O(n)$, note that we can calculate $C_i(x_j)$ in $\mathsf{poly}(2^n)$ time and hence $\mathsf{MQCSP}[\mathsf{poly}(n), \omega(\mathsf{poly}(n)), t(n)] \in \mathsf{BPP}$

$\square$

8.7.2 Quantum learning

As it could be the case that MQCSP might have non-trivial quantum algorithm, it is also of interest to study the connection to quantum learning.

Definition 8.23 (Quantum learning)**.** Let C be a circuit class of boolean functions and let $\epsilon, \delta > 0$. We say a quantum algorithm (ϵ, δ)-learns C if the following hold. For every $n \in \mathbb{N}$ and n-variate $f \in \mathsf{C}$, given quantum oracle access to f, the algorithm outputs a polynomial-size quantum circuit U such that with probability at least $1-\delta$, we have $\mathbb{E}_{x\in\{0,1\}^n}[|(\langle f(x)| \otimes I)U|x, 0^m\rangle|^2] > 1 - \epsilon$. The running time of the learning algorithm is measured as a function of $n, 1/\epsilon, 1/\delta$ and, $\mathsf{size}(f)$.

It turns out that efficient quantum learning for a circuit class C is equivalent to efficient quantum algorithm for its corresponding MCSP, i.e., C-MCSP.

Theorem 8.23 (Equivalence of efficient quantum learning and efficient quantum algorithm for C-MCSP)**.** *Let* C *be a circuit class.*

- *If* C-$\mathsf{MCSP} \in \mathsf{BQP}$*, then there exists a quantum algorithm that* $(1/\mathsf{poly}(n), \delta)$*-learns* C *for every* $\delta > 0$*. Specifically, the algorithm runs in polynomial time.*
- *If there exists a quantum algorithm that* (ϵ, δ)*-learns* C *in time* $2^{O(n)}$ *for some constants* $\epsilon, \delta \in (0, 1/2)$*, then we have* C-$\mathsf{MCSP}[\mathsf{poly}(n), \omega(\mathsf{poly}(n)), \tau] \in \mathsf{BQP}$ *for every* $\tau > 0$*.*

Proof.

- The key idea is to quantize the "learning from a natural property" framework [CIKK16]. Let us start with three important lemmas from [AGG+20].

Lemma 8.64 (Corollary of Lemma 4.3 and Lemma 4.4 in [AGG+20])**.** *Let $L, s_D : \mathbb{N} \to \mathbb{N}$ be constructive functions and $\gamma \in (0,1)$ with $1 \leq L(n) \leq 2^n$ for every $n \in \mathbb{N}$. There exists an algorithm A_{NW} on input 1^n and 1^L outputs $\mathsf{code}(C_{NW})$ for a quantum circuit C_{NW} in time $S(n) = \mathsf{poly}(n, L(n), s_D(n))$ with the following properties. In the following, we abbreviate $L = L(n)$ and $s_D = s_D(n)$.*

There exists a constant $c > 0$ and an oracle function $NW^{\mathcal{O}} : \{0,1\}^m \to \{h : \{0,1\}^{\log L} \to \{0,1\}\}$ where $m = cn^2$ and size$(NW^{\mathcal{O}}(z)) = \mathsf{poly}(n, \text{size}(\mathcal{O}))$ for all $z \in \{0,1\}^m$. Let $g : \{0,1\}^n \to \{0,1\}$. Suppose there is a quantum circuit D of size at most s_D with

$$\left| \Pr_{z \in \{0,1\}^m, D}[D(NW^g(z)) = 1] - \Pr_{y \in \{0,1\}^L}[D(y) = 1] \right| \geq \gamma\,.$$

Then C_{NW} on input $\mathsf{code}(D)$ and with oracle access to g, outputs $\mathsf{code}(C)$ for a quantum circuit C of size $O(L^2 \cdot s_D)$. With probability $\Omega(\gamma/L^2)$ over the output measurement of C_{NW}, we have

$$\Pr_{x \in \{0,1\}^n, C}[C(x) = g(x)] \geq \frac{1}{2} + \frac{\gamma}{2L}\,.$$

Lemma 8.65 (Lemma 4.5 in [AGG+20])**.** *Let $k, s : \mathbb{N} \to \mathbb{N}$ be constructive functions and $\gamma > 0$. There exists an algorithm A_{GL} such that on input 1^n and $1^{k(n)}$ outputs a circuit C_{GL} of size $\mathsf{poly}(n, k(n), s(n))$ in time $\mathsf{poly}(n, k(n), s(n))$ with the following properties. In the following, we abbreviate $k = k(n)$ and $s = s(n)$.*

Let $f : \{0,1\}^{kn} \to \{0,1\}^k$. Suppose there is a quantum circuit C of size at most s satisfying

$$\mathop{\mathbb{E}}_{x \in \{0,1\}^{kn}} \mathop{\mathbb{E}}_{r \in \{0,1\}^k}[|(\langle f(x) \cdot r| \otimes I)C|x, r, 0^m\rangle|^2] \geq \frac{1}{2} + \gamma\,.$$

Then C_{GL} on input $\mathsf{code}(C)$ outputs $\mathsf{code}(G^{\mathcal{O}})$ for a quantum oracle circuit $G^{\mathcal{O}}$ of size $O(kn)$ such that

$$\mathop{\mathbb{E}}_{x, G^C}[|(\langle f(x)| \otimes I)G^C|x, 0^{k+m+1}\rangle|^2] \geq \frac{\gamma^3}{2}\,.$$

Lemma 8.66 (Theorem in 4.28 [AGG$^+$20])**.** *Let $k, s : \mathbb{N} \to \mathbb{N}$ be constructive functions and $\epsilon, \delta \in (0,1)$. There exists a constant $c \geq 1$ and an algorithm A_{IJKW} such that on input 1^n and $1^{k(n)}$ outputs a circuit C_{IJKW} of size $\mathsf{poly}(n, k(n), s(n), \log 1/\delta, 1/\epsilon)$ in time $\mathsf{poly}(n, k(n), s(n), \log 1/\delta, 1/\epsilon)$ with the following properties. In the following, we abbreviate $k = k(n)$ and $s = s(n)$.*

Let $g : \{0,1\}^n \to \{0,1\}$. Suppose k is an even integer with

$$k \geq c \cdot \frac{1}{\delta}\left[\log\frac{1}{\delta} + \log\frac{1}{\epsilon}\right],$$

and suppose G is a quantum circuit of size at most s defined over $S_{n,k} := \{S \subset \{0,1\}^n : |S| = k\}$ with k output bits with

$$\mathop{\mathbb{E}}_{B \sim S_{n,k}, G}[G(B) = g^k(B)] \geq \epsilon\,.$$

Then C_{IJKW} on input $\mathsf{code}(G)$ outputs $\mathsf{code}(C)$ for a quantum circuit C of size $\mathsf{poly}(n, k, s, \log(1/\delta), 1/\epsilon)$ such that

$$\mathop{\mathbb{E}}_{x \sim \{0,1\}^n, C}[C(x) = g(x)] \geq 1 - \delta\,.$$

Now, we are ready to describe our quantum learning algorithm for C.

Algorithm 38 A quantum learning algorithm for C

```
Input: $1^n$, quantum oracle access to $n$-variate $f \in \mathsf{C}$, and parameters $\delta \in (0,1)$.
1: Let $L = \mathsf{poly}(n)$, $\epsilon = 1/\mathsf{poly}(n)$, and $k = \lceil c \cdot \frac{1}{\delta}(\log\frac{1}{\delta} + \log\frac{1}{\epsilon})\rceil$.
2: $C_{NW} \leftarrow A_{NW}(1^{kn+k})$; $C_{GL} \leftarrow A_{GL}(1^n, 1^k)$; $C_{IJKW} \leftarrow A_{IJKW}(1^n, 1^k)$.
3: Let $\mathsf{code}(D)$ be the description of a quantum circuit solving $\mathsf{C}$-$\mathsf{MCSP}$ with truth
   table size $L$.
4: Use the oracle access to $f$ to build an oracle access to $NW^g$ where $g$ :
   $\{0,1\}^{kn} \times \{0,1\}^k \to \{0,1\}$ with $g(x_1, \ldots, x_k, r_1, \ldots, r_k) = \oplus_{i=1}^k (r_i \cdot f(x_i))$ for
   every $x_1, \ldots, x_k \in \{0,1\}^n$ and $r_1, \ldots, r_k \in \{0,1\}$.
5: $\mathsf{code}(\tilde{C}) \leftarrow C^g_{NW}(\mathsf{code}(D))$
6: $\mathsf{code}(G^{\mathcal{O}}) \leftarrow C_{GL}(\mathsf{code}(\tilde{C}))$.
7: $C \leftarrow C_{IJKW}(\mathsf{code}(G^{\tilde{C}}))$.
8: Output $C$.
```

Let us analyze the correctness and running time of Algorithm 38 simultaneously. Let $f : \{0,1\}^n \to \{0,1\} \in \mathsf{C}$ be the function we want to learn. Let $g : \{0,1\}^{kn} \times \{0,1\}^k \to \{0,1\}$ be $g(x_1, \ldots, x_k, r_1, \ldots, r_k) = \oplus_{i=1}^{k}(r_i \cdot f(x_i))$ for every $x_1, \ldots, x_k \in \{0,1\}^n$ and $r_1, \ldots, r_k \in \{0,1\}$. Observe that if $\mathrm{size}(f) = \mathsf{poly}(n)$, then $\mathrm{size}(NW^g) = \mathsf{poly}(n) = \mathsf{poly}(\log L)$.

Next, if $\mathsf{C}\text{-}\mathsf{MCSP} \in \mathsf{BQP}$, then there exists a quantum algorithm D running in time $\mathsf{poly}(L)$ with

$$\left| \Pr_{z\in\{0,1\}^m, D}[D(NW^g(z)) = 1] - \Pr_{y\in\{0,1\}^L}[D(y) = 1] \right| \geq \frac{1}{3}\,.$$

By Lemma 8.64, $C^g_{NW}(\mathsf{code}(D))$ outputs the description of a quantum circuit C of size $O(L^2 \cdot \mathrm{size}(D)) = \mathsf{poly}(n)$ in time $\mathsf{poly}(L, \mathsf{size}(D))$ such that with probability $\Omega(1/L^2)$,

$$\Pr_{\substack{x_1,\ldots,x_r\in\{0,1\}^n \\ r_1,\ldots,r_k\in\{0,1\}, C}}[C(x_1, \ldots, x_k, r_1, \ldots, r_k) = g(x_1, \ldots, x_k, r_1, \ldots, r_k)] \geq \frac{1}{2} + \frac{1}{6L}\,.$$

Next, by Lemma 8.65, $C_{GL}(\mathsf{code}(C))$ outputs the description of an oracle quantum circuit $G^{\mathcal{O}}$ of size $O(kn \cdot \mathrm{size}(C)) = \mathsf{poly}(n)$ in time $\mathsf{poly}(n, k)$ such that

$$\mathop{\mathbb{E}}_{x_1,\ldots,x_k, G^C}\left[|(\langle f^k(x_1, \ldots, x_k)| \otimes I)G^C|x, 0^{k+m+1}\rangle|^2\right] \geq \Omega\left(\frac{1}{L^3}\right) = \frac{1}{\mathsf{poly}(n)}\,.$$

Finally, by Lemma 8.66, $C_{IKJW}(\mathsf{code}(G))$ outputs the description of a quantum circuit C of size $\mathsf{poly}(n, k, \mathrm{size}(G), \log(1/\delta), 1/\epsilon) = \mathsf{poly}(n, 1/\delta, 1/\epsilon) = \mathsf{poly}(n)$ in time $\mathsf{poly}(n)$ such that

$$\mathop{\mathbb{E}}_{x\sim\{0,1\}^n, C}[C(x) = g(x)] \geq 1 - \delta\,.$$

We conclude that there is a polynomial time $(1/3, \delta)$-quantum learning algorithm for C.

- Let ALG be a (ϵ, δ)-quantum learning algorithm for C for some $\epsilon, \delta \in (0, 1/2)$. We design the following quantum algorithm for $\mathsf{C}\text{-}\mathsf{MCSP}[\mathsf{poly}(n), \omega(\mathsf{poly}(n)), \tau]$. For every $\tau > 0$, let $\epsilon = \tau/4$ and $\epsilon' = \tau/2$.

Algorithm 39 A quantum algorithm for $\mathsf{C}\text{-}\mathsf{MCSP}[\mathsf{poly}(n), \omega(\mathsf{poly}(n)), \tau]$

```
Input: The truth table $T$ of a $n$-variate Boolean function $f$.
1: for $i = 1, \dots, 10\lceil \log 1/\delta \rceil$ do
2:     Run $ALG$ and supply quantum oracle access to $f$ using the truth table $T$. Let $C_i$ be the output of $ALG$. Let $U_i$ be the unitary corresponding to $C_i$.
3:     Uniformly and independently sample $x_1, \dots, x_\ell \in \{0,1\}^n$ where $\ell = \lceil 100 \log(1/\delta)/\epsilon^2 \rceil$.
4:     if $\sum_{j \in [\ell]} |(\langle f(x_j)| \otimes I)U_i|x, 0^m\rangle|^2 \geq (1 - \frac{\epsilon+\epsilon'}{2}) \cdot \ell$ then
5:         Break and output "Yes".
6:     end if
7: end for
8: Output "No".
```

Let us analyze the correctness of the above algorithm. First, if f is an Yes instance, i.e., there exists a polynomial size circuit C (from the circuit class C) that computes f, then due to the correctness of ALG, $\Pr_{C_i}[\mathbb{E}_{x \in \{0,1\}^n}[|(\langle f(x)| \otimes I)U_i|x, 0^m\rangle|^2] > 1 - \epsilon] > \delta$ for each i. Namely, with probability at least 9/10, there exists an $i \in [10\lceil \log 1/\delta \rceil]$ such that $\mathbb{E}_{x \in \{0,1\}^n}[|\{x \in \{0,1\}^n : |(\langle f(x)| \otimes I)U_i|x, 0^m\rangle|^2\}|] \geq 1 - \epsilon$. For this specific i, by Chernoff bound, with probability at least 9/10 the algorithm will go to line 5 and output "Yes". That is, the above algorithm accepts an Yes instance with probability at least 2/3 as desired.

Next, if f is an No instance, i.e., for every polynomial size circuit C (from the circuit class C), at least τ fraction of $x \in \{0,1\}^n$ has $|(\langle f(x)| \otimes I)U_i|x, 0^m\rangle|^2 \leq 1/2$. Hence, by the choice of ϵ', we have $\mathbb{E}_{x \in \{0,1\}^n}[|(\langle f(x)| \otimes I)U_i|x, 0^m\rangle|^2] < (1 - \epsilon')$. For each $i \in [10\lceil \log 1/\delta \rceil]$, C_i is a polynomial size circuit and hence by Chernoff bound, the algorithm goes to line 5 with probability at most $2^{-\Omega(\epsilon^2 m)}$. Due to the choice of m, we know that the algorithm will output "No" with probability at least 2/3. That is, the above algorithm rejects an No instance with probability at least 2/3 as desired.

Finally, the running time of the algorithm is $\mathsf{poly}(\text{Time}(ALG), 1/\delta, 1/\epsilon, n, m)$ where the dependency on $\mathsf{poly}(n, m)$ is for calculating $C_i(x_j)$ using the quantumness. Note that this running time is polynomial in the size of the truth table and hence we conclude that $\mathsf{C\text{-}MQCSP}[\mathsf{poly}(n), \omega(\mathsf{poly}(n)), \tau] \in \mathsf{BQP}$.

$\square$

8.8 Proofs in Section 8.4.3

In this section, we provide some missing proofs in Section 8.4.3.

8.8.1 Proof for Theorem 8.26

The goal of this section is to prove Theorem 8.26.

Theorem 8.26. *If* $\mathsf{MQCSP} \in \mathsf{BQP}$*, then* $\mathsf{BPE}^{\mathsf{QCMA}}$ *contains a function with maximum quantum circuit complexity. Furthermore,* $\mathsf{BQP}^{\mathsf{QCMA}} \not\subset \mathsf{BQC}[n^k]$ *for any constant* $k > 0$*.*

Proof. We follow the proof of a classical result in [KC00, Theorem 10].

We first determine the maximum quantum circuit complexity for all Boolean functions using an MQCSP oracle. For each $s = 2^{O(n)}, 2^{O(n)} - 1, \cdots$, decide if there exists a function f_s such that $\mathrm{qCC}(f_s) \geq s$. The first s we meet such that f_s exists is the maximum quantum circuit complexity. It can be achieved by a QCMA algorithm with input 1^s, by the assumption $\mathsf{MQCSP} \in \mathsf{BQP}$. Hence, in classical $2^{O(n)}$ time with query access to a QCMA oracle, we can find the maximum quantum circuit complexity $s_\star$ with high probability.

Then, we can construct the truth table by guessing bit-by-bit. We start from the empty truth table $T = \emptyset$. We first try to choose the first bit $T_1 = 0$ and decide if T can be extended to a truth table with quantum circuit complexity $s_\star$, which can be done by a QCMA oracle query. If the answer is "No", we set $T_1 = 1$. Then, we iterate over all bits of T. It is easy to see that in $O(2^n)$ time we can construct T with high probability.

Therefore, we get a $\mathsf{BPE}^{\mathsf{QCMA}}$ algorithm for the maximum quantum circuit complexity problem, which immediately gives a $\mathsf{BPE}^{\mathsf{QCMA}}$ algorithm for computing such hard functions. By Claim 8.74, this function has quantum circuit complexity at least $\Omega(2^n/n)$. Hence, by a padding argument for quantum circuits, we obtain a polynomial lower bound for $\mathsf{BQP}^{\mathsf{QCMA}}$. $\square$

8.8.2 Proof of Quantum Antichecker Lemma

The goal of this section is to prove Lemma 8.30.

Lemma 8.30 (Antichecker lemma for quantum circuits). *Assume* $\mathsf{QCMA} \subseteq \mathsf{BQC}[\mathsf{poly}]$. *Then for any $\lambda \in (0,1)$ there are circuits $\{C_{2^n}\}_{n=1}^{\infty}$ of size $2^{n+O(n^\lambda)}$ which given the truth table $\mathsf{tt}(f) \in \{0,1\}^{2^n}$, outputs $2^{O(n^\lambda)}$ n-bit strings $y_1, \dots, y_{2^{O(n^\lambda)}}$ together with bits $f(y_1), \dots, f(y_{2^{O(n^\lambda)}})$ forming a set of anticheckers for f, i.e. if f is hard for quantum circuits of size 2^{n^λ} then every quantum circuit of size $2^{n^\lambda}/2n$ fails to compute f on one of the inputs $y_1, \dots, y_{2^{O(n^\lambda)}}$.*

Proof. The proof follows [CHO+20].

Let $\lambda \in (0,1)$ and f be a Boolean function with n input bits that is hard for 2^{n^λ}-size quantum circuits.

For $i \geq 0$ and $s \in [0,1]$, define the predicate:

$$P_f(y_1, \dots, y_i)[s] = 1 \iff$$
$$\leq s \text{ fraction of all quantum circuits of size } 2^{n^\lambda}/2n \text{ compute } f \text{ correctly on } y_1, \dots, y_i.$$

We also define the function:

$$R_f(y_1, \dots, y_i) := \#\left\{\text{quantum circuits of size } 2^{n^\lambda}/2n \text{ compute } f \text{ correctly on } y_1, \dots, y_i\right\}.$$

Then, we construct $y_1, \dots, y_{2^{O(n^\lambda)}}$ iteratively. It is easy to see that $P_f(\cdot)[1] = 1$. Suppose we already have $y_1, \dots, y_{i-1}$ such that $P_f(y_1, \dots, y_{i-1})[(1 - 1/4n)^{i-1}] = 1$

holds. We want to find y_i such that $P_f(y_1, \ldots, y_i)[(1-1/4n)^i] = 1$. We will construct a formula F of size $2^{O(n^\lambda)}$ such that if $R_f(y_1, \ldots, y_{i-1}) \geq 2n^2$, then

$$\begin{aligned} & P_f(y_1, \ldots, y_{i-1}) \left[(1-1/4n)^{i-1}\right] = 1 \\ \Rightarrow\ & \exists y_i\ F(y_1, \ldots, y_i, f(y_1), \ldots, f(y_i)) = 1 \\ \Rightarrow\ & P_f(y_1, \ldots, y_i) \left[(1-1/4n)^{i}\right] = 1. \end{aligned}$$

We first show how to find y_i given this formula F. The idea is to use Valiant-Vazirani Isolation Lemma. Let r be uniformly chosen from $\{2, n+1\}$ and let $h : \{0,1\}^n \to \{0,1\}^r$ be uniformly chosen from a pairwise independent hash family $\mathcal{H}_{n,r}$. Consider the following predicate

$$\begin{aligned} & F^{r,h}(y_1, \ldots, y_{i-1}, z, f(y_1), \ldots, f(y_{i-1}), f(z)) := \\ & F(y_1, \ldots, y_{i-1}, z, f(y_1), \ldots, f(y_{i-1}), f(z)) \wedge h(z) = 0^r. \end{aligned}$$

The quantum circuit size of $F^{r,h}$ is $2^{O(n^\lambda)}$.

By the Isolation Lemma, for fixed $y_1, \ldots, y_{i-1}$, with probability at least $1/8n$, there is a unique z such that

$$F^{r,h}(y_1, \ldots, y_{i-1}, z, f(y_1), \ldots, f(y_{i-1}), f(z)) = 1.$$

If we sample $2^{O(n^\lambda)}$ many tuples of (r, h), then the probability that none of those (r, h) will lead to unique solution of $F^{r,h}$ is less than $2^{-2^{O(n^\lambda)}/8n} \leq 2^{-2^{O(n^\lambda)}}$ by choosing proper constant. On the other hand, the total number of all possible $y_1, \ldots, y_{i-1}, f(y_1), \ldots, f(y_{i-1})$ is at most $2^{2^{O(n^\lambda)}}$. It means that there exists a set $\mathcal{R}$ of $2^{O(n^\lambda)}$ tuples of (r, h) such that for any $y_1, \ldots, y_{i-1}, f(y_1), \ldots, f(y_{i-1})$, there exists an $(r, h) \in \mathcal{R}$ that makes $F^{r,h}$ have unique solution. Note that $\mathcal{R}$ can be hard-wired into the circuit C_n. Hence, the j-th bit of the antichecker y_i can be computed by the following formula of size $2^{n+O(n^\lambda)}$:

$$\bigvee_{z \in \{0,1\}^n} z_j \wedge F^{r,h}(y_1, \ldots, y_{i-1}, z, f(y_1), \ldots, f(y_{i-1}), f(z)). \tag{8.17}$$

Then, we need to select an (r, h) from $\mathcal{R}$ that gives the unique y_i. This task is in QCMA, and by assumption, $\mathsf{QCMA} \subseteq \mathsf{BQC}[\mathsf{poly}]$. So, we just need to apply a $2^{O(n^\lambda)}$-size quantum circuit. Once we have y_i, $f(y_i)$ can be obtained from $\mathsf{tt}(f)$ via an Address function, which can be implemented by a circuit of size $2^{n+O(\log n)}$.

By repeating this process, we can get $y_1, \ldots, y_{2^{O(n^\lambda)}}$ and $f(y_1), \ldots, f(y_{2^{O(n^\lambda)}})$ by a $2^{n+O(n^\lambda)}$ circuit. Then, we need to check $R_f(y_1, \ldots, y_{2^{O(n^\lambda)}}) \geq 2n^2$. Deciding whether $R_f(y_1, \ldots, y_i) \geq 2n^2$ is in $\mathsf{QCMA} \subseteq \mathsf{BQC}[\mathsf{poly}]$ with input $(y_1, \ldots, y_i, f(y_1), \ldots, f(y_i), 1^{2^{O(n^\lambda)}})$ since the witness is $2n^2$ quantum circuits each of size $2^{n^\lambda}/2n$, which can be represented by a $2^{O(n^\lambda)}$ binary string. The witness can be checked by simulating the quantum circuits. Therefore, there exists a $2^{O(n^\lambda)}$ quantum circuit for it. When $R_f(y_1, \ldots, y_i) \leq 2n^2$, the $2n^2$ circuits of size $2^{n^\lambda}/2n$ can be generated by an $\mathsf{QCMA}^{\mathsf{coQCMA}}$ algorithm. And since $\mathsf{QCMA} \subseteq \mathsf{BQC}[\mathsf{poly}]$, by uncomputing the garbage, we can show that $\mathsf{QCMA}^{\mathsf{coQCMA}} \subseteq \mathsf{BQC}[\mathsf{poly}]$ and this step can be done by a $2^{O(n^\lambda)}$ quantum circuit. For each circuit, by exhaustively searching, we can find an n-bit string that witness the error. The circuit size of this step is $2^{n+O(n^\lambda)}$.

In order to construct F, we use a result in [OPS19] (Lemma 23) showing that if $P_f(y_1, \ldots, y_{i-1})[(1 - 1/4n)^{i-1}] = 1$ and $R_f(y_1, \ldots, y_{i-1}) \geq 2n^2$, then

$$\exists y_i \ P_f(y_1, \ldots, y_i) \left[(1 - 1/4n)^{i-1}(1 - 1/2n)\right] = 1. \tag{8.18}$$

The proof is by a standard counting argument, and by examining the proof, we find that it also holds for quantum circuits.

By Eq. (8.18), we know that there exists a y_i such that $\leq (1 - 1/4n)^{i-1}(1 - 1/2n) < (1 - 1/4n)^i$ fraction of circuits of size $2^{n^\lambda}/2n$ that can compute f on $y_1, \ldots, y_i$. The remaining task is to find a witness (which is F) that can certify $P_f(y_1, \ldots, y_i)\left[(1 - 1/4n)^i\right] = 1$. We can use an approximate counting with linear hash functions to construct F. More specifically, by [Jeř09], the witness is a set of matrices $A_1, \ldots, A_{2^{O(n^\lambda)}}$ defining an injective map from the Cartesian power of the set of all circuits of size $2^{n^\lambda}/2n$ that compute f on $y_1, \ldots, y_i$ to the same Cartesian power of $(1 - 1/4n)^i$ fraction of the set of all circuits of size $2^{n^\lambda}/2n$. The existence of these matrices can be decided by an $\mathsf{QCMA}^{\mathsf{coQCMA}}$ algorithm, which can also be decided by a $2^{O(n^\lambda)}$ quantum circuit, by our assumption. $\square$

8.8.3 Quantum Impagliazzo-Wigderson generator

The goal of this section is to prove Lemma 8.28.

Lemma 8.28 (Quantum Impagliazzo-Wigderson generator)**.** *For every $\epsilon > 0$, there exist $c, d \in \mathbb{N}$ such that the truth table of a Boolean function $f : \{0,1\}^{cn} \to \{0,1\}$ of quantum circuit complexity $2^{\epsilon cn}$ can be transformed in time $O(2^n)$ into a pseudorandom generator $G : \{0,1\}^{dn} \to \{0,1\}^{2^n}$ running in time $O(2^n)$ that can fool quantum circuits of size $2^{O(n)}$, i.e., for any $p > 0$, any quantum circuit $\mathcal{C}$ of size at most 2^{pn},*

$$\left| \Pr_{x\sim\{0,1\}^{dn},\mathcal{C}}[\mathcal{C}(G(x)) = 1] - \Pr_{y\sim\{0,1\}^{2^n},\mathcal{C}}[\mathcal{C}(y) = 1] \right| \le 2^{-n}.$$

Before giving the proof, we first recall some necessary definitions and lemmas in the previous work.

Lemma 8.67 (A variant of Lemma 4.29 in [AGG+20])**.** *Let $L : \{0,1\}^* \to \{0,1\}$ be a language that is randomly reducible to the language L'. For every n, suppose we have the description of a quantum circuit U such that*

$$\mathop{\mathbb{E}}_{x\in\{0,1\}^n} \left[\|\Pi_{L'(x)} U|x, 0^q\rangle\|^2\right] \ge 1 - \frac{1}{n^k},$$

for some $k \ge 2b + a$.

There is a $O(|U| \cdot \mathsf{poly}(n))$-size quantum circuit $\widetilde{U}$ that satisfies

$$\|\widetilde{\Pi}_x \widetilde{U}|0, x, 0^{q^*}\rangle\|^2 \ge 1 - 2^{-2n+1} \qquad \textit{for every } x \in \{0,1\}^n,$$

where $\widetilde{\Pi}_x = |L(x)\rangle\langle L(x)| \otimes |x\rangle\langle x| \otimes |0^{q^*}\rangle\langle 0^{q^*}|$ and $q^* = \mathsf{poly}(n)$.

Definition 8.41 (Expander walks)**.** Let $\mathcal{G}$ be a graph with vertex set $\{0,1\}^n$ and degree 16. Let the expander walk generator $\mathsf{EW} : \{0,1\}^n \times [16]^k \to \{0,1\}^{nk}$ such that $\mathsf{EW}(v, d) := (v_1, \ldots, v_k)$, where $v_1 = v$ and v_{i+1} is the d_i-th neighbor of v_i in $\mathcal{G}$.

Definition 8.42 (Nearly disjoint subsets)**.** Let $\Sigma = \{S_1, \dots, S_k\}$ be a family of subsets of $[m]$ of size n. We say Σ is γ-disjoint if $|S_i \cap S_j| \leq \gamma n$ for any $i \neq j$.

For $r \in \{0,1\}^m$, $S \subseteq [m]$, let $r|_S$ be the restriction of r to S. Then, for a γ-disjoint Σ, $\mathsf{ND}^{\Sigma} : \{0,1\}^m \to \{0,1\}^{nk}$ is defined by $\mathsf{ND}^{\Sigma}(r) := r|_{S_1}, \dots, r|_{S_k}$.

Definition 8.43 (M-restrictable)**.** We say $G_n : \{0,1\}^m \to \{0,1\}^{nk}$ is M-restrictible if there exists a polynomial-time computable function $h : [n] \times \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^m$ such that

- For any $i \in [n], x \sim \{0,1\}^n, \alpha \sim \{0,1\}^m$, $h(i, x, \alpha)$ is uniformly distributed.
- For any i, x, α, let $G(h(i, x, \alpha)) := x_1, \dots, x_k$. Then, we have $x_i = x$.
- For any $i, j \neq i$, for any α, there exists a set $S \subseteq \{0,1\}^n$, $|S| \leq M$ such that for any x, $x_j \in S$.

Definition 8.44 ((k', q, δ)-hitting)**.** We say $G_n : \{0,1\}^m \to \{0,1\}^{nk}$ is (k', q, δ)-hitting if for any sets $H_1, \dots, H_k \subseteq \{0,1\}^n$, $|H_i| \geq \delta 2^n$, we have

$$\Pr[|\{i : x_i \in H_i\}| < k'] < q.$$

Proof of Lemma 8.28. We follow the proof in [IW97]. We first assume that there exists a function $f_0 : \{0,1\}^n \to \{0,1\}$ such that the quantum circuit complexity of f_0 is $2^{\Omega(n)}$. We may assume that $f_0 \in \mathsf{BQE}$. Then, encoding the truth table of f_0 by a locally list-decodable code, we obtain a function $f_1 : \{0,1\}^{O(n)} \to \{0,1\}$ such that $f_1 \in \mathsf{BQE}$, and for any quantum circuit $\mathcal{B}_1$ of size less than $2^{\Omega(n)}$,

$$\mathop{\mathbb{E}}_{x \sim \{0,1\}^{O(n)}, \mathcal{B}_1} [B_1(x) = f_1(x)] := \mathop{\mathbb{E}}_{x \sim \{0,1\}^{O(n)}, \mathcal{B}_1} [\|\Pi_{f_1(x)} \mathcal{B}_1 |x, 0\rangle\|] \leq 1 - n^{-O(1)}.$$

The properties of f_1 can be proved by Lemma 8.67.

Then, by Lemma 8.66 with $k = \mathsf{poly}(n), \epsilon = O(1), \delta = \frac{1}{\mathsf{poly}(n)}$, we have a function $f_2 = f_1^{\otimes k} : \{0,1\}^{kn} \to \{0,1\}^k$ such that for any quantum circuit $\mathcal{B}_2$ of size less than $2^{\Omega(n)}$,

$$\mathop{\mathbb{E}}_{x \in \{0,1\}^{nk}, \mathcal{B}_2} [\mathcal{B}_2(x) = f_2(x)] \leq O(1).$$

We can apply the quantum Goldreich-Levin Theorem (Lemma 8.65) to f_2 and get a function $f_3 : \{0,1\}^n \to \{0,1\}$ (scaling the input size) such that for any quantum circuit $\mathcal{B}_3$ of size less than $2^{\Omega(n)}$,

$$\mathop{\mathbb{E}}_{x \in \{0,1\}^{n}, \mathcal{B}_3} [\mathcal{B}_3(x) = f_2(x)] \leq \frac{2}{3}.$$

The remaining thing is to "quantize" the direct-product generator defined by [IW97] using f_3. More specifically, we say G is a $(s, s', \epsilon, \delta)$ *quantum direct-product generator* if $G : \{0,1\}^m \to \{0,1\}^{nk}$ such that for every Boolean function g that is δ-hard for any quantum circuit of size s, we have $g^{\otimes} \circ G$ is ϵ-hard for any quantum circuit of size s'. The main result of [IW97] is the construction of $(2^{\Omega(n)}, 2^{\Omega(n)}, 2^{-\Omega(n)}, \frac{1}{3})$ direct-product generator. We first briefly describe the construction and then show that it also works for quantum circuits.

The direct-product generator in [IW97] is constructed from the expander random walks (Definition 8.41) and nearly disjoint subsets (Definition 8.42). They defined the direct-product generator $\mathsf{XG}(r, r', v, d) := \mathsf{EW}(v, d) \oplus \mathsf{ND}^{\Sigma}(r')$, where $\Sigma \subseteq [m]$ is selected by r such that $|r| = O(n), |r'| = m = O(n), |v| = n, |d| = O(n)$. They proved that XG is $2^{\Omega(n)}$-restrictible and $(O(n), 2^{-\Omega(n)}, 1/3)$-hitting. It's easy to see that the restrictible and hitting properties are pure combinatorial and circuit independent, which means that they also hold for quantum circuits. Then, they proved that these combinatorial properties imply XG is also a direct product generator. This step, however, need to be reproved for quantum circuits.

Claim 8.68. *Let $s > 0$, $G(r) : \{0,1\}^m \to \{0,1\}^{nk}$ be a $(\rho k, q, \delta)$-hitting, M-restrictible pseudo-random generator, where $q > 2^{-\rho k/3}, s > 2Mnk$. Then, G is a $(s, \Omega(sq^2 n^{-O(1)}), O(q), \delta)$-quantum direct product generator.*

Proof. Let $\epsilon = (4\delta/\rho + 1)q$. Suppose there is a quantum circuit $\mathcal{C}$ such that

$$\mathop{\mathbb{E}}_{x\sim\{0,1\}^m,\mathcal{C}}\left[\mathcal{C}(x) = g^{\otimes k} \circ G(x)\right] \geq \epsilon.$$

Then, we construct a quantum circuit $\mathcal{F}$ of size $O(|C| + kMn)$ such that for any $H \subseteq \{0,1\}^n, |H| \geq \delta 2^n$,

$$\mathop{\mathbb{E}}_{y\sim H,\mathcal{F}}[\mathcal{F}(y) = g(y)] \geq \frac{1}{2} + \frac{q}{2}.$$

We use the same construction as [IW97]. Let $i \sim [k], \alpha_0 \sim \{0,1\}^m$. Let $x_1, \ldots, x_k$ be the output of $G(h(i, x, \alpha_0))$. For each $j \neq i$, we non-uniformly construct a table of $g(x_j)$ for any x_j that is a possible output of $G(h(i, x, \alpha_0))$ for different x. Since G is M-restrictible, each table has at most M values. Then, on input $y \in \{0,1\}^n$, the circuit $\mathcal{F}$ simulates $\mathcal{C}$ on $h(i, y, \alpha_0)$ and let $c_1, \ldots, c_k$ be the output. Then, for $y_1, \ldots, y_k := G(h(i, y, \alpha))$, $\mathcal{F}$ counts the number of indices $j \neq i$ such that $c_i \neq g(y_i)$ using the tables. Let t be the number. Then, with probability 2^{-t}, $\mathcal{F}$ outputs c_i; otherwise, $\mathcal{F}$ outputs a random bit.

For analysis of quantum circuits, as in [AGG+20], we first consider $\mathcal{C}$ being an inherently probabilistic circuit. For any $H \subseteq \{0,1\}^n$, let $y \sim H$ uniformly at random. Then, for any $y_1, \ldots, y_k \in (\{0,1\}^n)^k$,

$$\Pr_{y\sim H}[y_1, \ldots, y_k \text{ generated by } \mathcal{F}] = \frac{u}{\delta k} \cdot \Pr_{r\sim\{0,1\}^m}[y_1, \ldots, y_k \text{ generated by } G(r)]. \quad (8.19)$$

where u is the number of $y_i \in H$. Since $\mathbb{E}_{r,\mathcal{C}}[\mathcal{C}(r) = g^{\otimes k}(G(r))] \geq \epsilon$, for a random r, the probability that $u \geq \rho k$ and $\mathcal{C}(r) = g^{\otimes k}(G(r))$ is at least $\epsilon - q$, by the hitting property of G. Hence, the probability that $u \geq \rho k$ and $\mathcal{C}$ succeeds for $y_1, \ldots, y_k$ generated by F on a random $x \in H$ is $(\epsilon - q) \cdot \rho/\delta = 4q$, since each $(y_1, \ldots, y_k)$ has at least ρ/δ of its probability under $G(r)$ by Eq. (8.19). Then, we can compute the expected success probability of $\mathcal{F}$ on $y \in H$ given $u \geq \rho k$ by Theorem 3.2 in [IW97], which is

$$\mathop{\mathbb{E}}_{y\sim H}[\mathcal{F}(y) = g(y) \mid u \geq \rho k] \geq \frac{1}{2} + q.$$

Since $u \geq \rho k$ has probability at least $1-q$, the overall success probability is at least $(1+q)/2$. Finally, by Lemma 2.7 in [AGG+20], we can change the inherently probabilistic circuit by a quantum circuit and the result still holds.

Hence, $\mathcal{F}$ has expected probability $(1+q)/2$ on $1-\delta$ fraction of inputs. Then, we can take $O(n/q^2)$ copies and take the majority of them, which gives a circuit of size $O((|\mathcal{C}|+kM)n/q^2) \leq s$ if $|\mathcal{C}| = \Omega(sq^2n^{-O(1)})$, and has success probability at least $1-\delta$. The Claim is then proved. □

By Claim 8.68, we know that $\mathsf{XG} : \{0,1\}^{O(n)} \to \{0,1\}^{n^2}$ is a $(2^{\Omega(n)}, 2^{\Omega(n)}, 2^{-\Omega(n)}, 1/3)$-quantum direct product generator.

Finally, feeding the output of XG to the quantum Nisan-Wigderson generator (Lemma 8.64) C_{NW} gives the desired quantum pseudo-random generator, which completes the proof of the lemma. □

8.9 Quantum Fine-Grained Hardness Based on QETH

In this section, we will show that $2n \times 2n$ bipartite permutation independent set problem is hard under QETH.

Lemma 8.39. *Assuming QETH, there is no $2^{o(n \log n)}$-time quantum algorithm that solves $2n \times 2n$ Bipartite Permutation Independent Set problem.*

More specifically, We "quantize" the fine-grained reduction in [LMS11]. The reduction chain is as follows:

3-SAT $\leq_{FG}$ 3-Coloring $\leq_{FG}$ $n \times n$ Clique $\leq_{FG}$ $n \times n$ Permutation Clique

$\leq_{FG}$ $n \times n$ Permutation Independent Set

$\leq_{FG}$ $2n \times 2n$ Bipartite Permutation Independent Set

We first define some intermediate fine-grained problems.

Definition 8.45 ($n \times n$ Clique problem)**.** Given a graph on the vertex set $[n] \times [n]$, decide if there exists $i_1, \ldots, i_n \in [n]$ such that the subgraph on $(1, i_1), \ldots, (n, i_n)$ forms an n-clique.

Definition 8.46 ($n \times n$ Permutation Clique/Independent Set problem)**.** Given a graph on the vertex set $[n] \times [n]$, decide if there exists a permutation $\pi \in \mathcal{S}_n$ such that the subgraph on $(1, \pi(1)), \ldots, (n, \pi(n))$ forms an n-clique/independent set.

The following claims shows that the aforementioned reductions work for quantum lower bounds.

Claim 8.69. *Under QETH, there is no $2^{o(n)}$-time quantum algorithm for 3-Coloring, where n is the number of vertices in the input graph.*

Proof. By the NP-complete proof of 3-Coloring, we know that a 3-CNF formula with n variables and m clauses can be reduced to a 3-Coloring instance in time $O(n+m)$. Hence, a $2^{o(n)}$-time quantum algorithm for 3-Coloring implies a $2^{o(n)}$-time quantum algorithm for 3-SAT, which implies that QETH fails. □

Claim 8.70. *If $n \times n$ Clique can be solved in $2^{o(n \log n)}$ time quantumly, then 3-Coloring can be solved in $2^{o(n)}$ time quantumly.*

Proof. We use the reduction given by [LMS11]. Let G be an instance of 3-Coloring with n vertices. The reduction can produce a graph H with vertices $[k] \times [k]$ such that $n \le k \log_3 k - k$. Then, G is 3-colorable if and only if H is a "Yes" instance of $k \times k$ Clique. The reduction takes $\mathsf{poly}(k)$-time classically.

Hence, if there exists a quantum algorithm for $k \times k$ Clique in time $2^{o(k \log k)}$, then it gives a quantum algorithm for 3-Coloring that runs in time $2^{o(n)}$. □

Claim 8.71. *If $n \times n$ Permutation Clique/Independent Set can be solved in $2^{o(n \log n)}$ time quantumly, then $n \times n$ Clique can also be solved in $2^{o(n \log n)}$ time quantumly.*

Proof. By [LMS11], there is a reduction from $n \times n$ Clique to $n \times n$ Permutation Clique that takes $2^{O(n \log \log n)} = 2^{o(n \log n)}$ time classically. Hence, the reduction also works for quantum $2^{o(n \log n)}$-time lower bound.

Note that $n \times n$ Permutation Clique and $n \times n$ Permutation Independent Set are equivalent problem, since we can reduce them by taking the complement graph. □

Claim 8.72. *If $2n \times 2n$ Bipartite Permutation Independent Set can be solved in $2^{o(n \log n)}$ quantumly, then $n \times n$ Permutation Independent Set can be solved in $2^{o(n \log n)}$ time quantumly.*

Proof. By [LMS11], the classical reduction takes time $O(n^2)$. Hence, it also works for quantum algorithms. □

Finally, we can prove the QETH-hardness of $2n \times 2n$ bipartite permutation independent set problem:

Proof of Lemma 8.39. It follows from Claim 8.69, 8.70, 8.71 and 8.72. □

8.10 Proofs for Corollary 8.47

Corollary 8.47. SMCSP *with classical descriptions of quantum states as inputs is in* QCMA*.*

Lemma 8.73. *Given $v = [v_0, \ldots, v_{2^n-1}]$ for $v_i \in \mathbb{C}$ for $i = 0, \ldots, 2^n - 1$, there exists a quantum circuit such that the state $|v\rangle$ can be computed in time $\mathsf{poly}(2^n)$ with $\langle i|v\rangle = v_i$.*

Proof. We show that one can use single-qubit rotations to construct $|v\rangle$.

We first prepare $|0^{n+1}\rangle$. Then, we do a single-qubit rotation on the first qubit such that

$$|0^{n+1}\rangle \to \sqrt{\frac{\sum_{i=0}^{2^{n-1}-1} |v_i|^2}{\sum_{i=0}^{2^n-1} |v_i|^2}}|0\rangle|0^n\rangle + \sqrt{\frac{\sum_{i=2^{n-1}}^{2^n-1} |v_i|^2}{\sum_{i=0}^{2^n-1} |v_i|^2}}|1\rangle|0^n\rangle.$$

Then, let the first qubit be the control qubit and apply the controlled rotation to rotate the second qubit to be

$$\sqrt{\frac{\sum_{i=0}^{2^{n-2}-1}|v_i|^2}{\sum_{i=0}^{2^{n-1}-1}|v_i|^2}}|0\rangle + \sqrt{\frac{\sum_{i=2^{n-2}}^{2^{n-1}-1}|v_i|^2}{\sum_{i=0}^{2^{n-1}-1}|v_i|^2}}|1\rangle, \text{if the first qubit is } |0\rangle,$$
$$\sqrt{\frac{\sum_{i=2^{n-1}}^{2^{n-1}+2^{n-2}-1}|v_i|^2}{\sum_{i=2^{n-1}}^{2^n-1}|v_i|^2}}|0\rangle + \sqrt{\frac{\sum_{i=2^{n-1}+2^{n-2}}^{2^n-1}|v_i|^2}{\sum_{i=2^{n-1}}^{2^n-1}|v_i|^2}}|1\rangle, \text{if the first qubit is } |1\rangle.$$

By doing these controlled rotations in sequence, we can obtain $||v|\rangle$ where $\langle v|i\rangle = |v_i|$ for all i. Let $v_j = e^{-i\theta_j}|v_j|$ without loss of generality. Then, condition on j, we do the following rotation on the $(n+1)$-th qubit:

$$|0\rangle \to e^{-i\theta_j}|0\rangle$$

for all j. This gives $|v\rangle$.

Finally, we use at most $2^{O(n)}$ (control) rotations. By Remark 8.10, each controlled rotation can be implemented with at most $2^{O(n)}$ overhead. Hence, the verifier can construct $|v\rangle$ in time $\mathsf{poly}(2^n)$.

$\square$

Proof of Corollary 8.47. Following Lemma 8.73, we can make $\mathsf{poly}(n, s, t)$ copies of the state in polynomial time. Then, following the proof for Theorem 8.46, the problem is in QCMA.

$\square$

8.11 Quantum Circuit Class

In this section, we will show some properties of the quantum circuit $\mathsf{QC}(s, \mathcal{G})$. Note that $\mathcal{G}$ considered in this chapter are universal gate set with constant fan-in. So, the results here are also for constant fan-in universal gate sets.

Claim 8.74. *For $n \in \mathbb{N}$, there exists a constant c such that a random Boolean function $f : \{0,1\}^n \to \{0,1\}$ has quantum circuit complexity greater $\frac{2^n}{(c+1)n}$ with probability at least $1 - 2^{\frac{2^n}{c+1}}$.*

Proof. For any s-gate and $(n+t)$-qubit quantum circuit (where $n+t \leq s$), there are at most

$$\binom{n+qs+t}{q}^s |\mathcal{G}|^s \leq 2^{cs\log s}$$

possible circuits for some constant c large enough, where $\mathcal{G}$ is the quantum gate set, and q is the maximum number of qubits for any gate in $\mathcal{G}$ can operate on. Let $s = \frac{2^n}{(c+1)n}$. Then the number of circuits of size s is at most $2^{cs\log s} < 2^{\frac{c}{c+1}\cdot 2^n}$.

There are 2^{2^n} Boolean functions from $\{0,1\}^n$ to $\{0,1\}$. Suppose we pick one function uniformly randomly, then for every fixed quantum circuit $\mathcal{C}$ and input $x \in \{0,1\}^n$, the probability that $\|(\langle f(x)| \otimes I_{n+t-1})\mathcal{C}|x, 0^t\rangle\| \geq \frac{1}{2}$ is $\frac{1}{2}$. Therefore, the probability that a fixed quantum circuit can compute $f(x)$ for all $x \in \{0,1\}^n$ is at most $\frac{1}{2^{2^n}}$. By using union bound, the probability that there exists $\mathcal{C}$ of size $\frac{2^n}{(c+1)n}$ that can compute f is at most $\frac{2^{\frac{c}{c+1}\cdot 2^n}}{2^{2^n}} = 2^{\frac{2^n}{c+1}}$. □

Claim 8.75. *For $s = \mathsf{poly}(n)$ and $\mathcal{G}$ a gate set that contains only constant fan-in gates, $\mathsf{BQC}(s, \mathcal{G})$ is in $\mathsf{DSPACE}(O(s^2))/O(s^2)$.*

Proof. The proof follows from the idea of showing $\mathsf{BQP} \subset \mathsf{PSPACE}$. Let $L \in \mathsf{BQC}(s, \mathcal{G})$ and $\{\mathcal{C}_n\}$ be the quantum circuit family in $\mathsf{QC}(s, \mathcal{G})$ that can solve L. Then, we show that there is a $O(s^2)$-space TM T with $O(s\log s)$-bit advice that can simulates $\mathcal{C}_n$.

Let $\mathcal{C}_n$ be the advice to T. We first calculate the number of bits needed to represent s-gate circuit. For each gate, we need $O(\log s)$ to specify its wires and 2^a register to record the corresponding unitary, where a is the maximum fan-in of gates in $\mathcal{G}$. Note that a unitary U may has entries that cannot be written down in bounded bits. Therefore, we let the precision to every entry in U be $\epsilon = \frac{1}{c2^s}$ for some constant

c large enough, which requires number of bits $\log \frac{1}{\epsilon} = O(s)$. The total number of bits required for each gate is $O(s)$. and thus the number bits for the circuit is $O(s^2)$.

Now, suppose $\mathcal{C}_n = U_s U_{s-1} \cdots U_1$. For any $x \in \{0,1\}^n$ the probability that $\mathcal{C}_n$ accepts is

$$\sum_{y \in A} |\langle y|U_s U_{s-1} \cdots U_1|x\rangle|^2,$$

where $A := \{y : y \text{ has the first bit as } 1\}$. Then, the TM T computes each branch one-by-one. for any $y \in A$

$$\langle y|U_s U_{s-1} \cdots U_1|x\rangle = \sum_{z_1,\ldots,z_{s-1} \in \{0,1\}} \langle y|U_s|z_{s-1}\rangle\langle z_{s-1}|U_{s-1}|z_{s-2}\rangle\langle z_{s-2}| \cdots |z_1\rangle\langle z_1|U_1|x\rangle. \tag{8.20}$$

Note that U_i is a constant-dimensional unitary and x and z_j's are vectors with exactly one non-zero entry. So, computing $\langle z_j|U_j|z_{j-1}\rangle$ only requires $O(s)$ (since the entries in U takes $O(s)$ space for the precision). Then, since we can also compute $\langle z_j|U_j|z_{j-1}\rangle$ one by one, the space required for each branch in Eq. (8.20) is just $O(s)$. Therefore, the space we need is at most $O(s^2)$ (including the space for the advice).

Note that our calculation in Eq. (8.20) will have error since our precision to each entry in the unitary is $\epsilon = \frac{1}{c2^s}$. Let $\tilde{U}_s\tilde{U}_{s-1} \cdots \tilde{U}_1$ be what we really compute. Then,

$$\sum_{y \in A} |\langle y|U_s U_{s-1} \cdots U_1|x\rangle|^2 - \sum_{y \in A} |\langle y|\tilde{U}_s\tilde{U}_{s-1} \cdots \tilde{U}_1|x\rangle|^2 \leq O(2^{s+n}\epsilon).$$

By setting $\epsilon = \frac{1}{c2^s}$ for some constant c large enough, T can solve L with probability at least $2/3$ by having an amplified version of $\mathcal{C}_n$ at first (e.g., parallel repetition).

$\square$

Claim 8.76 (Diagonalization for quantum circuits)**.** *For every $k \in \mathbb{N}_+$, there exists a language $L_k \in \mathsf{PSPACE}$ but $L_k \notin \mathsf{BQC}[n^k]$ for sufficiently large n.*

Proof. By Claim 8.75, we know that $\mathsf{BQC}[n^k]$ is contained in $\mathsf{DSPACE}[n^{2k}]/n^{2k}$. By a nonuniform almost everywhere hierarchy for space complexity (Lemma 11 in [OS16]), we know that $\mathsf{DSPACE}[n^{3k}] \not\subset \mathsf{DSPACE}[n^{2k}]/n^{2k}$ for sufficiently large n. Hence, we can find a language $L_k \notin \mathsf{BQC}[n^k]$. □

Claim 8.77 (BQC size hierarchy)**.** *For $n > 0$, let $s(n) = o(\frac{2^n}{n})$. Then, there exists a Boolean function f in $\mathsf{BQC}[s(n)] \backslash \mathsf{BQC}[s(n) - O(n)]$, i.e., f can be computed by an $s(n)$-size quantum circuit but not computed by any $(s(n) - O(n))$-size quantum circuit.*

Proof. The proof is very similar to the argument for classical circuits. By Claim 8.74, we can find a function g that requires quantum circuit of size $2^n/cn$ for some $c > 1$. Suppose there are t inputs $x_1, \dots, x_t$ such that $g(x_i) = 1$ for $i \in [t]$. Then, we construct a series of functions g_i for $i = 0, 1, \cdots, t$ such that $g_i(x) = 1$ if and only if $x \in \{x_1, \dots, x_i\}$. It's easy to see that the following properties are satisfied:

- $g_0 \in \mathsf{BQC}[0]$ and $g_t \in \mathsf{BQC}[2^n/cn]$.
- For $0 \le i < t$, the difference of the quantum circuits size of g_i and g_{i+1} is at most $O(n)$. It follows since g_i and g_{i+1} are only different at x_i.

Hence, there exists an $i > 0$ such that the quantum circuit size of g_i is at most $s(n)$ but lager than $s(n) - O(n)$, since $s(n) = o(2^n/cn)$.

□

8.12 MQCSP and prBQP

In this section, we consider the MQCSP in the oracle setting and show some circuit lower bounds for $\mathsf{BQP}_{/1}$ (BQP with one classical bit advice) and promise-BQP in the relativized world. Our results are quantum versions of the [IKV18]'s results.

Theorem 8.78. *For any $k \in \mathbb{N}$, ${\mathsf{BQP}^{\mathsf{MQCSP}}}_{/1} \not\subseteq \mathsf{BQC}[n^k]$.*

Proof. It is well-known that exists a PSPACE-complete language L that is downward self-reducible and self-correctable [TV07]. Since $\mathsf{PSPACE} \subseteq \mathsf{BQEXP}$, there exists a $d \in \mathbb{N}$ such that $L \in \mathsf{BQC}[2^{n^d}]$. Let $\mathrm{qCC}(L,n)$ denote the size of the minimum quantum circuit for deciding L with input length n. Then, we have $\mathrm{qCC}(L,n) = O(2^{n^d})$.

Case 1: $\mathsf{PSPACE} \subseteq \mathsf{BQC}[\mathsf{poly}]$ Since L is PSPACE-complete, we have $\mathrm{qCC}(L,n) = O(n^k)$ for some $k \in \mathbb{N}$. Then, by Lemma 8.79, $L(x)$ can be computed in $\mathsf{BQP}^{\mathsf{MQCSP}}$. Hence, $\mathsf{PSPACE} \subseteq \mathsf{BQP}^{\mathsf{MQCSP}}$. By a diagonalization argument for quantum circuits (Claim 8.76), we have $\mathsf{PSPACE} \not\subseteq \mathsf{BQC}[n^k]$ for any $k \in \mathbb{N}$. Thus, $\mathsf{BQP}^{\mathsf{MQCSP}} \not\subseteq \mathsf{BQC}[n^k]$.

Case 2: $\mathsf{PSPACE} \not\subseteq \mathsf{BQC}[\mathsf{poly}]$ In this case, $L \notin \mathsf{BQC}[\mathsf{poly}]$. Suppose $\mathsf{BQP}^{\mathsf{MQCSP}}/1 \subseteq \mathsf{BQC}[n^k]$ for some $k \in \mathbb{N}$. Similar to the proof in [IKV18], we define a padding language $L'_k := \{1^m x\}$ satisfying the following conditions: (1) m is a power of 2; (2) $0 < r := |x| \leq m$; (3) $x \in L$; (4) $\mathrm{qCC}(L,r) \leq m^{2k}$. We claim that $L'_k \in \mathsf{BQP}^{\mathsf{MQCSP}}/1$. Let $y = 1^m x$ be the input of L'_k. Conditions (1), (2) are easy to check. The advice bit can be used to determine whether condition (4) holds. If it does not hold, then we reject the string. Otherwise, we know that $\mathrm{qCC}(L,|x|) \leq m^{2k}$. By Lemma 8.79, $L(|x|)$ can be computed in $\mathsf{poly}(|x|, m^{2k})$-quantum time given an MQCSP oracle, which checks condition (3). Therefore, we conclude that $L'_k \in \mathsf{BQP}^{\mathsf{MQCSP}}/1$. By our assumption, $L'_k \in \mathsf{BQC}[n^k]$. Then, it implies that $L \in \mathsf{BQC}[n^{2k}]$, which follows from Lemma 15 in [IKV18][25]. We get a contradiction. Hence, $\mathsf{BQP}^{\mathsf{MQCSP}}/1 \subseteq \mathsf{BQC}[n^k]$.

Combining Case 1 and 2 completes the proof of the theorem. $\square$

We need the following lemma giving an efficient MQCSP-oracle quantum algorithm to compute a downward-reducible and self-correctable language.

[25] The proof of this lemma can be found in [San09]. It is straightforward to verify that this lemma holds for both classical and quantum circuits.

Lemma 8.79. *Let L be a downward self-reducible and self-correctable language. Then, there is a quantum algorithm with access to an* MQCSP *oracle, such that given x and t computes $L(x)$ with high probability in* $\mathsf{poly}(|x|, t)$*-quantum time, provided that $t \geq \mathrm{qCC}(L, |x|)$.*

Proof. By Lemma 8.80, we have a PAC learner $\mathcal{A}$ for BQC. We can use the same strategy as in the proof of Lemma 35 in [IKV18] to compute $L(x)$. More specifically,

- Step 1: construct a circuit $\widetilde{C}_1 = C_1$ for $L|_1$.
- Step 2: for $i = 2 \dots n$
  - Step 2.1: run $\mathcal{A}^{\mathsf{MQCSP}}$ with probability parameter $\delta = i^{-3}$ and accuracy parameter $\epsilon = i^{-1}$ to learn a circuit $\widetilde{C}_i$ of size t for $L|_i$. For the membership query to $L|_i$, by the downward-reducibility of L, it can be computed via the circuit C_i.
  - Step 2.2: apply the self-correction algorithm for L to $\widetilde{C}$ and obtain the circuit C_i.
- Step 3: output $C_n(x)$.

□

The following lemma is a direct extension of Theorem 8.22.

Lemma 8.80 (PAC learning MQCSP-oracle quantum circuit)**.** BQC *is PAC learnable under the uniform distribution, using membership query and* MQCSP *queries with hypothesis being* MQCSP*-oracle circuits.*

Formally, there exists a randomized algorithm that makes oracle queries to MQCSP *such that, given $s \in \mathbb{N}$, oracle access to a function $f \in \mathsf{BQC}[s]$, and $\epsilon > 0$, it outputs an* MQCSP*-oracle quantum circuit C such that $\Pr[C[x] \neq f(x)] \leq \epsilon$ in* $\mathsf{poly}(n, s, \epsilon^{-1})$*-time.*

Theorem 8.78 has the following consequence:

Corollary 8.81. *For any $k \in \mathbb{N}$,* $\mathsf{prBQP}^{\mathsf{MQCSP}} \not\subseteq \mathsf{BQC}[n^k]$.

Proof. Fix $k \in \mathbb{N}$. Suppose $L \in \mathsf{BQP}^{\mathsf{MQCSP}}{}_{/1}$ and $L \notin \mathsf{BQC}[n^k]$. Let $\mathcal{C}$ be the quantum oracle circuit that takes one-bit advice to compute L. We can construct a language $L' \in \mathsf{prBQP}^{\mathsf{MQCSP}}$ as follows: for an input string (x, b) with $|b| = 1$, if $x \in L$ and $\mathcal{C}$ accepts x with advice b, then (x, b) is in the YES case of L'. If $x \notin L$ and $\mathcal{C}$ rejects x with advice b, then (x, b) is in the NO case of L'. Then, if L' can be computed by a n^k-size quantum circuit, L can also be computed by a n^k-size quantum circuit, which contradicts our assumption. Hence, by Theorem 8.78, we get that $\mathsf{prBQP}^{\mathsf{MQCSP}} \not\subseteq \mathsf{BQC}[n^k]$. □

	Results	Informal Theorem Index (Formal Theorem Index)
MQCSP (Def. 8.9)	$\mathsf{MQCSP} \in \mathsf{QCMA}$	Theorem 8.1 (Theorem 8.14)
	$\mathsf{MQCSP} \in \mathsf{BQP} \Rightarrow$ No qOWF	Theorem 8.1 (Theorem 8.20)
	$\mathsf{SZK} \leq \mathsf{MQCSP}$	Theorem 8.1 (Theorem 8.15)
	$\mathsf{multiMQCSP}$ is NP-hard under a natural gate set	Theorem 8.1 (Theorem 8.16)
	$i\mathcal{O} + \mathsf{MQCSP} \in \mathsf{BQP} \Rightarrow \mathsf{NP} \subseteq \mathsf{coRQP}$	Theorem 8.1 (Theorem 8.21)
	PAC learning for $\mathsf{BQP/poly} \Leftrightarrow \mathsf{MQCSP} \in \mathsf{BPP}$	Theorem 8.2 (Theorem 8.22)
	BQP learning $\Leftrightarrow \mathsf{MQCSP} \in \mathsf{BQP}$	Theorem 8.3 (Theorem 8.23)
	$\mathsf{MQCSP} \in \mathsf{BQP} \Rightarrow \mathsf{BQE} \not\subset \mathsf{BQC}[n^k],\ \forall k \in \mathbb{N}_+$	Theorem 8.4 (Theorem 8.26)
	$\mathsf{MQCSP} \in \mathsf{BQP} \Rightarrow \mathsf{BQP}^{\mathsf{QCMA}} \not\subset \mathsf{BQC}[n^k],\ \forall k \in \mathbb{N}_+$	Theorem 8.4 (Theorem 8.29)
	$\mathsf{MQCSP} \in \mathsf{BQP} \Rightarrow$ Hardness amplification	Theorem 8.5 (Theorem 8.27)
	Hardness magnification for MQCSP	Theorem 8.6 (Theorem 8.29)
	$\mathsf{QETH} \Rightarrow$ quantum hardness of $\mathsf{MQCSP}^{\star}$	Theorem 8.7 (Theorem 8.31)
	$\mathsf{BQP}^{\mathsf{MQCSP}}{}_{/1}$, $\mathsf{prBQP}^{\mathsf{MQCSP}} \not\subset \mathsf{BQC}[n^k], \forall k \in \mathbb{N}_+$	(Theorem 8.78, Corollary 8.81)
UMCSP (Def. 8.31)	$\mathsf{UMCSP} \in \mathsf{QCMA}$	Theorem 8.8 (Theorem 8.42)
	Search-to-decision reduction for UMCSP	Theorem 8.9 (Theorem 8.48)
	$\mathsf{gap\text{-}MQCSP} \leq \mathsf{UMCSP}$	Theorem 8.9 (Theorem 8.54)
	$\mathsf{UMCSP} \in \mathsf{BQP}$ $\Rightarrow$ No pseudorandom unitaries and no qOWF	(Theorem 8.55, Corollary 8.56)
	$i\mathcal{O} + \mathsf{UMCSP} \in \mathsf{BQP} \Rightarrow \mathsf{NP} \subseteq \mathsf{coRQP}$	(Corollary 8.57)
	$\mathsf{UMCSP} \in \mathsf{BQP} \Rightarrow$ Hardness amplification for BQP	(Corollary 8.58)
	$\mathsf{UMCSP} \in \mathsf{BQP} \Rightarrow \mathsf{BQE} \not\subset \mathsf{BQP}[n^k],\ \forall k \in \mathbb{N}$	(Corollary 8.59)
SMCSP (Def. 8.32)	SMCSP can be verified via QCMA	Theorem 8.8 (Theorem 8.46)
	Search-to-decision reduction for SMCSP	Theorem 8.9 (Theorem 8.50)
	Self-reduction for SMCSP	Theorem 8.9 (Theorem 8.52)
	$\mathsf{SMCSP} \in \mathsf{BQP}$ $\Rightarrow$ No pseudorandom states and no qOWF	Theorem 8.10 (Theorem 8.60)
	Assume conjectures from physics $\mathsf{SMCSP} \Rightarrow$ Estimating wormhole's volume	Theorem 8.10 (Theorem 8.62)
	Succinct state tomography $\leq \mathsf{SMCSP}$	Theorem 8.10 (Theorem 8.63)

Table 8.1: Summary of our results. A result with color Blue is a direct extension from its classical analog. A result with color Yellow requires additional techniques. A result with color Red is unique in the quantum setting.

Part II

Optimization

Chapter 9: Faster Classical Semi-Definite Programming Solver

9.1 Introduction

Semidefinite programming (SDP) optimizes a linear objective function over the intersection of the positive semidefinite (PSD) cone with an affine space. SDP is of great interest both in theory and in practice. Many problems in operations research, machine learning, and theoretical computer science can be modeled or approximated as semidefinite programming problems. In machine learning, SDP has applications in adversarial machine learning [RSL18], learning structured distribution [CLM20], sparse PCA [AW08, dEGJL07], robust learning [DKK$^+$16, DHL19, JLT20]. In theoretical computer science, SDP has been used in approximation algorithms for max-cut [GW94], coloring 3-colorable graphs [KMS94], and sparsest cut [ARV09], quantum complexity theory [JJUW11], robust learning and estimation [CG18, CDG19, CDGW19], graph sparsification [LS17], algorithmic discrepancy and rounding [BDG16, BG17, Ban19], sum of squares optimization [BS16, FKP19], terminal embeddings [CN21], and matrix discrepancy [HRS21].

SDP is formally defined as follows:

Definition 9.1 (Semidefinite programming)**.** Given symmetric[1] matrices $C, A_1, \cdots, A_m \in \mathbb{R}^{n\times n}$ and a vector $b \in \mathbb{R}^m$, the goal is to solve the following optimization problem:

$$\max_{X\in\mathbb{R}^{n\times n}} \ \langle C, X\rangle \text{ subject to } \langle A_i, X\rangle = b_i, \ \ \forall i \in [m], \ X \succeq 0, \tag{9.1}$$

where $\langle A, B\rangle := \sum_{i,j} A_{i,j}B_{i,j}$ is the matrix inner product.

[1]We can without loss of generality assume that $C, A_1, \cdots, A_m$ are symmetric. Given any $A \in \mathbb{R}^{n\times n}$, we have $\sum_{i,j} A_{ij}X_{ij} = \sum_{i,j} A_{ij}X_{ji} = \sum_{i,j}(A^\top)_{ij}X_{ij}$ since X is symmetric, so we can replace A with $(A + A^\top)/2$.

The input size of an SDP instance is mn^2, since there are m constraint matrices each of size $n \times n$. The well-known linear programming (LP) is a simpler case than SDP, where $X \succeq 0$ and $C, A_1, \cdots, A_m$ are restricted to be $n \times n$ diagonal matrices. The input size of an LP instance is thus mn.

Over the last many decades, there are three different lines of high accuracy SDP solvers (with logarithmic accuracy dependence in the running time). The first line of work is using the cutting plane method, such as [Sho77, YN76, Kha80, KTE88, NN89, Vai89a, BV02, KM03, LSW15, JLSW20]. This line of work uses m iterations, and each iteration uses some SDP-based oracle call. The second line of work is using interior point method (IPM) and log barrier function such as [NN92, JKL+20]. The third line of work is using interior point method and hybrid barrier function such as [NN94, Ans00].

Recently, a line of work uses robust analysis and dynamic maintenance to speedup the running time of linear programming [CLS19, Bra20, BLSS20, JSWZ21, Bra21]. One major reason made solving SDP much more harder than solving linear programming is: in LP the slack variable is a vector(can be viewed as a diagonal matrix), and in SDP *the slack variable is a positive definite matrix.* Due to that reason, the gradient/Hessian computation requires some complicated and heavy calculations based on the Kronecker product of matrices, while LP only needs the basic matrix-matrix product [Vai89b, CLS19, JSWZ21]. Therefore, handling the errors in each iteration and maintaining the slack matrices are way more harder in SDP. Thus, we want to ask the following question:

Can we efficiently solve SDP <u>without</u> computing exact gradient, Hessian, and Newton steps?

In this chapter, we will answer the above question by introducing new framework for both IPM analysis and variable maintenance. For IPM analysis, we build a robust IPM framework for arbitrary barrier functions that supports errors in computing gradient, Hessian, and Newton steps. For variable maintenance, we provide a general amortization method that gives improved guarantees on reducing the computational complexity by lazily updating the Hessian matrices.

For solving SDP using IPM with log barrier, the current best algorithm (due to Jiang, Kathuria, Lee, Padmanabhan and Song [JKL+20]) runs in $O(\sqrt{n}(mn^2 + m^\omega + n^\omega))$ time. Since the input size of SDP is mn^2, ideally we would want an SDP algorithm that runs in $O(mn^2 + m^\omega + n^\omega)$ time, which is roughly the running time to solve linear systems[2]. The current best algorithms are still at least a $\sqrt{n}$ factor away from the optimal.

Inspired by the result [CLS19] which solves LP in the current matrix multiplication time, a natural and fundamental question for SDP is

Can we solve SDP in the current matrix multiplication time?

More formally, for the above formulation of SDP (Definition 9.1), is that possible to solve it in $mn^2 + m^\omega + n^\omega$ time? In this chapter, we give a positive answer to this question by using our new techniques. For the tall dense SDP where $m = \Omega(n^2)$, our algorithm runs in $m^\omega + m^{2+1/4}$ time, which matches the current matrix multiplication time. The tall dense SDP finds many applications and is one of the two predominant cases in [JKL+20][3]. This is the first result that shows SDP can be solved as fast as solving linear systems.

Finally, we also show that our techniques and framework are quite versatile and can be used to directly speedup the SDP solver via the hybrid barrier [NN89, Ans00].

Our results. We present the simplified version of our main result in the following theorem. The formal version can be found in Theorem 9.18.

[2]We note that a recent breakthrough result by Peng and Vempala [PV21] showed that a sparse linear system can be solved faster than matrix multiplication. However, their algorithm essentially rely on the sparsity of the problems. And it is still widely believed that general linear system requires matrix multiplication time.

[3]See Table 1.2 in [JKL+20] and Section 9.6.

Theorem 9.1 (Main result, informal version of Theorem 9.18)**.** *For ϵ-accuracy, there is a classical algorithm that solves a general SDP instance with variable size $n \times n$ and m constraints in time*[4] $O^*((\sqrt{n}(m^2 + n^4) + m^\omega + n^{2\omega}) \log(1/\epsilon))$, *where ω is the exponent of matrix multiplication.*

In particular, for $m = \Omega(n^2)$, our algorithm takes matrix multiplication time m^ω for current $\omega \approx 2.373$.

Remark 9.1. For any $m \geq n^{2-0.5/\omega} \approx n^{1.79}$ with current $\omega \approx 2.37286$ [Wil12, LG14, AW21], our algorithm runs faster than [JKL+20].

Theorem 9.1 and [JKL+20] are focusing on the log barrier method for solving SDP. However, the area of speeding up the hybrid barrier-based SDP solver is quite blank. We also improve the state-of-the-art implementation of the hybrid barrier-based SDP solver [NN89, Ans00] in all parameter regimes. See Section 9.5 and Theorem 9.4 for more details.

Roadmap. In Section 9.2, we review the previous approaches for solving SDP and discuss their bottlenecks. In Section 9.3, we introduce our robust framework for IPM. In Section 9.4, we show our main techniques and sketch the proof of our main result (Theorem 9.1). In Section 9.5, we overview the approach of applying our robust framework to speedup the hybrid barrier-based SDP solver. Related works are provided in Section 9.6. We define our notations and include several useful tools in Section 9.7. In Section 9.8, we give the formal version of our algorithm and the main theorem, where the proof is given Section 9.9 and 9.10. Our general robust IPM framework is displayed in Section 9.11. Our fast implementation of the hybrid barrier-based SDP solver can be found in Section 9.12.

[4]We use $O^*(\cdot)$ to hide $n^{o(1)}$ and $\log^{O(1)}(mn/\epsilon)$ factors, and $\widetilde{O}(\cdot)$ to hide $\log^{O(1)}(mn/\epsilon)$ factors.

9.2 An Overview of Previous Techniques

Under strong duality, the primal formulation of the SDP in Eq. (9.1) is equivalent to the following dual formulation:

Definition 9.2 (Dual problem)**.** Given symmetric matrices $C, A_1, \ldots, A_m \in \mathbb{R}^{n\times n}$ and $b_i \in \mathbb{R}$ for all $i \in [m]$, the goal is to solve the following convex optimization problem:

$$\min_{y\in\mathbb{R}^m} b^\top y \quad \text{subject to } S = \sum_{i=1}^{m} y_i A_i - C, \quad S \succeq 0. \tag{9.2}$$

Interior point methods (IPM) solve the above problem by (approximately) following a central path in the feasible region $\{y \in \mathbb{R}^m : S = \sum_{i=1}^m y_i A_i - C \succeq 0\}$. As a rich subclass of IPM, barrier methods [NN92, Ans00] define a point on the central path as the solution to the following optimization problem parametrized by $\eta > 0 :\ \min_{y\in\mathbb{R}^m} f_\eta(y)$ where

$$f_\eta(y) := \eta \cdot \langle b, y\rangle + \phi(y) \tag{9.3}$$

is the augmented objective function and $\phi : \mathbb{R}^m \to \mathbb{R}$ is a barrier function[5] that restricts y to the feasible region since $\phi(y)$ increases to infinity when y approaches the boundary of the feasible region. Barrier methods usually start with an initial feasible y for a small η, and increase η in each iteration until y is close to the optimal solution of the SDP. In short-step barrier methods with log barrier, $\eta^{\text{new}} = (1 + 1/\sqrt{n})\eta$. It takes a Newton step $-H(y)^{-1}g(y,\eta)$ in each iteration to keep y in the proximity of the central path. Here $g(y,\eta)$ and $H(y)$ are the gradient and the Hessian of $f_\eta(y)$.

[5]The choice of the barrier function leads to different numbers of iterations. Nesterov and Nemirovski [NN92] utilize the log barrier function $\phi_{\log}(y) = -\log\det(S)$ which guarantees convergence in $\widetilde{O}(\sqrt{n})$ iterations. Anstreicher [Ans00] uses the Hybrid barrier $\phi_{\text{hybrid}}(y) = 225(n/m)^{1/2} \cdot (\phi_{\text{vol}}(y) + \phi_{\log}(y) \cdot (m-1)/(n-1))$ where $\phi_{\text{vol}}(y)$ is the volumetric barrier $\phi_{\text{vol}}(y) = \frac{1}{2}\log\det(\nabla^2\phi_{\log}(y))$. Hybrid barrier guarantees convergence in $\widetilde{O}((mn)^{1/4})$ iterations.

Techniques and bottlenecks of existing algorithms

Fast solvers of SDP include the cutting plane method and interior point method. The fastest known algorithms for SDP based on the cutting plane method [LSW15, JLSW20] have m iterations and run in $O^*(m(mn^2 + n^\omega + m^2))$ time. The fastest known algorithm for SDP based on the interior point method [JKL+20] has $\sqrt{n}$ iterations and runs in $O^*(\sqrt{n}(mn^2 + n^\omega + m^\omega))$ time. In most applications of SDP where $m \geq n$, interior point method of [JKL+20] runs faster. In the following we briefly discuss the techniques and bottlenecks of interior point methods.

Central path. Interior point method updates the dual variable y by Newton step $-H(y)^{-1}g(y,\eta)$ to keep it in the proximity of central path. This proximity is measured by the potential function $\|H(y)^{-1}g(y,\eta)\|_{H(y)}$.[6] In classical interior point literature, this potential function is well controlled by taking exact Newton step (see e.g. [Ren01]). [JKL+20] relaxes this guarantee and allows PSD approximation to the Hessian matrix $H(y)$. However, their convergence also relies on exact computation of slack matrix S and gradient g. This leads to a $mn^{2.5}$ term in their running time.

Amortization techniques. [JKL+20] keeps a PSD approximation $\widetilde{H}$ of the Hessian H and updates $\widetilde{H}$ by a low rank matrix in each iteration. The running time of this low rank update is then controlled by a delicate amortization technique. This technique also appears in linear programming [CLS19] and empirical risk minimization [LSZ19]. [JKL+20] brings this technique to SDP, and costs $n^{0.5}m^\omega$ time in computing the inverse of Hessian matrix. When m becomes larger, this term dominates the complexity and becomes undesirable.

Algorithm 40 The general robust barrier method framework for SDP.

```
1:  procedure GeneralRobustSDP($\mathsf{A} \in \mathbb{R}^{m \times n^2}$, $b \in \mathbb{R}^m$, $C \in \mathbb{R}^{n \times n}$)
2:      Choose $\eta$ and $T$
3:      Find initial feasible dual vector $y \in \mathbb{R}^m$            ▷ Condition 0 in Lemma 9.2
4:      for $t = 1 \to T$ do do                          ▷ Iterations of approximate barrier method
5:          $\eta^{\mathrm{new}} \leftarrow \eta \cdot (1 + \frac{\epsilon_N}{20\sqrt{\theta}})$
6:          $\widetilde{S} \leftarrow$ ApproxSlack()
7:          $\widetilde{H} \leftarrow$ ApproxHessian()                    ▷ Condition 1 in Lemma 9.2
8:          $\widetilde{g} \leftarrow$ ApproxGradient()                   ▷ Condition 2 in Lemma 9.2
9:          $\widetilde{\delta}_y \leftarrow$ ApproxDelta()                  ▷ Condition 3 in Lemma 9.2
10:         $y^{\mathrm{new}} \leftarrow y + \delta_y$
11:         $y \leftarrow y^{\mathrm{new}}$                                          ▷ Update variables
12:     end for
13: end procedure
```

9.3 The Robust SDP Framework

In section 9.3, we introduce our robust SDP framework. This framework works for general barrier functions and finds applications in both Algorithm 42 and Algorithm 45-46. We consider self-concordant barrier function ϕ with complexity θ (Definition 9.8)[7]. For the regularized objective f_η in Eq. (9.3), we define the gradient $\mathsf{g} : \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^m$ as

$$\mathsf{g}(y, \eta) = \eta \cdot b - \nabla \phi(y).$$

Interior point method takes Newton step $(\nabla^2 \phi(y))^{-1} g(y, \eta)$ and guarantees the variables in the proximity of the central path by bounding the potential function $\Phi(z, y, \eta) = \|\mathsf{g}(y, \eta)\|_{(\nabla^2 \phi(z))^{-1}}$. In practical implementations, there are perturbations in the Newton step due to errors in slack matrix S, gradient $\mathsf{g}(y, \eta)$, Hessian matrix $\nabla^2 \phi(y)$ and Newton step $(\nabla^2 \phi(y))^{-1} \cdot \mathsf{g}(y, \eta)$. Many fast algorithms maintain approximations to these quantities to reduce the running time. We propose a more general

[6]For symmetric PSD matrix A, let $\|x\|_A = \sqrt{x^\top A x}$ denote matrix norm of x.

[7]The barrier function being "self-concordant" is a key assumption in the interior-point method [Nes88a, Nes88b, NN89, Ren01]. It is also useful in many optimization tasks [Hil14, Nar16, LLV20].

robust framework (compared with [Ren01, JKL$^+$20]) which captures all these errors. We show that as long as these errors are bounded by constants in the local norm[8], the potential function stays bounded, which guarantees the closeness to central path. Therefore this analysis is currently the most robust possible. The main component of our robust analysis is the following one step error control.

Lemma 9.2 (One step error control of the robust framework, informal version of Lemma 9.40)**.** *Let the potential function of IPM defined by*

$$\Psi(z, y, \eta) := \|\mathsf{g}(y, \eta)\|_{(\nabla^2\phi(z))^{-1}}.$$

Given any parameters $\alpha_S \in [1, 1 + 10^{-4}]$, $c_H \in [10^{-1}, 1]$, $\epsilon_g, \epsilon_\delta \in [0, 10^{-4}]$, and $\epsilon_N \in (0, 10^{-1})$, $\eta > 0$. Suppose that there is

- **Condition 0.** *a feasible dual solution $y \in \mathbb{R}^m$ satisfies $\Phi(y, y, \eta) \leq \epsilon_N$,*
- **Condition 1.** *a symmetric matrix $\widetilde{H} \in \mathbb{S}^{n\times n}_{>0}$ satisfies $c_H \cdot \nabla^2\phi(y) \preceq \widetilde{H} \preceq \nabla^2\phi(y)$,*
- **Condition 2.** *a vector $\widetilde{g} \in \mathbb{R}^m$ satisfies $\|\widetilde{g} - \mathsf{g}(y, \eta^{\mathrm{new}})\|_{(\nabla^2\phi(y))^{-1}} \leq \epsilon_g \cdot \|\mathsf{g}(y, \eta^{\mathrm{new}})\|_{(\nabla^2\phi(y))^{-1}}$,*
- **Condition 3.** *a vector $\widetilde{\delta}_y \in \mathbb{R}^m$ satisfies $\|\widetilde{\delta}_y - \widetilde{H}^{-1}\widetilde{g}\|_{\nabla^2\phi(y)} \leq \epsilon_\delta \cdot \|\widetilde{H}^{-1}\widetilde{g}\|_{\nabla^2\phi(y)}$.*

Then $\eta^{\mathrm{new}} = \eta(1 + \frac{\epsilon_N}{20\sqrt{\theta}})$ and $y^{\mathrm{new}} = y - \widetilde{\delta}_y$ satisfy

$$\Psi(y^{\mathrm{new}}, y^{\mathrm{new}}, \eta^{\mathrm{new}}) \leq \epsilon_N.$$

This result suggests that as long as we find an initial dual variable y in the proximity of central path, i.e. $\Phi(y, y, \eta) \leq \epsilon_N$, Lemma 9.2 will guarantee that the invariant $\Phi(y, y, \eta) \leq \epsilon_N$ holds throughout Algorithm 40, even when there exist errors in the slack matrices, Hessian, gradient and Newton steps. As shown in Section 9.11,

[8] See condition 0-3 in Lemma 9.2 for details.

the duality gap is upper bounded by $\theta \cdot \Phi(y, y, \eta)/\eta$. In at most $O(\sqrt{\theta} \cdot \log(\theta/\epsilon))$ iterations, η will become greater than $\theta \cdot \Phi(y, y, \eta)/\epsilon$. Therefore Algorithm 40 finds ϵ-optimal solution within $O(\sqrt{\theta} \cdot \log(\theta/\epsilon))$ iterations.

We note that [Ans00] and [Ren01] only consider Condition 0 and requires the $c_H = 1, \epsilon_g = \epsilon_\delta = 0$ in Condition 1, 2, and 3. [JKL+20] considered Condition 0 and Condition 1 in Lemma 9.2 and requires the $\epsilon_g = \epsilon_\delta = 0$ in Condition 2 and 3. Moreover, the Condition 1 in [JKL+20] requires c_H to be very close to 1, and we relax this condition to support any constant in $[10^{-1}, 1]$. In addition, our framework also relaxes the computation of gradient and Newton direction to allow some approximations, which makes it possible to apply more algorithmic techniques in the interior-point method. More details are provided in Section 9.11.2.

9.4 Our Techniques

In this section, we introduce our main techniques, and provide a self-contained proof sketch of our main result Theorem 9.1. We tackle the two bottlenecks of m^ω cost per iteration in [JKL+20] by proposing two different techniques:

Bottleneck 1: Instead of inverting the Hessian matrix from scratch in each iteration, we make use of the already-computed Hessian inverse of the previous iteration. We prove that using low-rank updates, the change to the inverse of Hessian matrices (computed using Kronecker product) is low-rank, and thus we can use Woodbury identity to efficiently update the Hessian inverse. In Section 9.4.1 we introduce the low-rank update to the Hessian, and in Section 9.4.2 we describe how to compute the Hessian inverse efficiently using Woodbury identity and fast matrix rectangular multiplication.

Bottleneck 2: We propose a better amortization scheme for PSD matrices that improves upon the previous m^ω amortized cost. We give a proof sketch of our amortized analysis in Section 9.4.3.

Algorithm 41 An implementation of GeneralRobustSDP (Informal version of Alg. 42).

```
1: procedure SolveSDP( $\mathsf{A} \in \mathbb{R}^{m \times n^2}$, $b \in \mathbb{R}^m$, $C \in \mathbb{R}^{n \times n}$)
2:     for $t = 1 \to T$ do                                                    ▷ $T = \widetilde{O}(\sqrt{n})$
3:         $\eta^{\mathrm{new}} \leftarrow \eta \cdot (1 + 1/\sqrt{n})$
4:         $g_{\eta^{\mathrm{new}}}(y)_j \leftarrow \eta^{\mathrm{new}} \cdot b_j - \mathrm{tr}[S^{-1} \cdot A_j],\ \ \forall j \in m$      ▷ Gradient computation
5:         $\delta_y \leftarrow -\widetilde{H}^{-1} \cdot g_{\eta^{\mathrm{new}}}(y)$                       ▷ Compute Newton step
6:         $y^{\mathrm{new}} \leftarrow y + \delta_y$                                        ▷ Update dual variables
7:         $S^{\mathrm{new}} \leftarrow \sum_{i \in [m]} (y^{\mathrm{new}})_i A_i - C$                      ▷ Compute slack matrix
8:         Compute $V_1, V_2 \in \mathbb{R}^{n \times r_t}$ such that $\widetilde{S}^{\mathrm{new}} = \widetilde{S} + V_1 \cdot V_2^\top$      ▷ Step 1 of Sec. 9.4.1
9:         Compute $V_3, V_4 \in \mathbb{R}^{n \times r_t}$ such that $(\widetilde{S}^{\mathrm{new}})^{-1} = (\widetilde{S})^{-1} + V_3 \cdot V_4^\top$ ▷ Step 2 of Sec. 9.4.1
10:        Compute $\mathsf{A}Y_1, \mathsf{A}Y_2 \in \mathbb{R}^{m \times nr_t}$ such that $\widetilde{H}^{\mathrm{new}} = \widetilde{H} + (\mathsf{A}Y_1) \cdot (\mathsf{A}Y_2)^\top$ ▷ Step 3 of Sec. 9.4.1
11:        $(\widetilde{H}^{\mathrm{new}})^{-1} \leftarrow \widetilde{H}^{-1}$ + low-rank update                          ▷ Sec. 9.4.2
12:        $y \leftarrow y^{\mathrm{new}}$, $S \leftarrow S^{\mathrm{new}}$, $\widetilde{S} \leftarrow \widetilde{S}^{\mathrm{new}}$, $\widetilde{H}^{-1} \leftarrow (\widetilde{H}^{\mathrm{new}})^{-1}$        ▷ Update variables
13:    end for
14: end procedure
```

9.4.1 Low rank update of Hessian

Low-rank approximation of Kronecker product itself is an interesting problem and has been studied in [SWZ19]. In this section, we describe how the low-rank update of the slack matrix leads to a low-rank update of the Hessian matrix that involves Kronecker product.

The Hessian matrix is defined as $H = \mathsf{A} \cdot (S^{-1} \otimes S^{-1}) \cdot \mathsf{A}^\top$. We take the following three steps to construct the low-rank update of H.

Step 1: low-rank update of the slack matrix. We use an approximate slack matrix that yields a low-rank update. In the t-th iteration of Algorithm 41, we use $\widetilde{S}$ to denote the current approximate slack matrix, and S^{new} to denote the new exact slack matrix. We will use $\widetilde{S}$ and S^{new} to find the new approximate slack matrix $\widetilde{S}^{\mathrm{new}}$.

Define $Z = (S^{\mathrm{new}})^{-1/2} \widetilde{S} (S^{\mathrm{new}})^{-1/2} - I$ which captures the changes of the slack matrix. We compute the spectral decomposition: $Z = U \cdot \mathrm{diag}(\lambda) \cdot U^\top$. We show that

$$\sum_{i=1}^{n} \lambda_i^2 = \|S^{-1/2} S^{\mathrm{new}} S^{-1/2} - I\|_F = O(1),$$

which implies that only a few eigenvalues of Z are significant, say e.g. $\lambda_1, \ldots, \lambda_{r_t}$. We only keep these eigenvalues and set the rest to be zero. In this way we get a low-rank approximation of Z: $\widetilde{Z} = U \cdot \mathrm{diag}(\widetilde{\lambda}) \cdot U^\top$ where $\widetilde{\lambda} = [\lambda_1, \cdots, \lambda_{r_t}, 0, \ldots, 0]^\top$. Now we can use $\widetilde{Z}$ to update the approximate slack matrix by a low-rank matrix:

$$\widetilde{S}^{\mathrm{new}} = \widetilde{S} + (S^{\mathrm{new}})^{1/2} \cdot \widetilde{Z} \cdot (S^{\mathrm{new}})^{1/2} = \widetilde{S} + V_1 \cdot V_2^\top,$$

where V_1 and V_2 both have size $n \times r_t$. Since $\widetilde{Z}$ is a good approximation of Z, $\widetilde{S}^{\mathrm{new}}$ is a PSD approximation of S^{new}, which guarantees that y still lies in the proximity of the central path.

Step 2: low-rank update of inverse of slack. Using Woodbury identity, we can show that

$$(\widetilde{S}^{\mathrm{new}})^{-1} = (\widetilde{S} + V_1 \cdot V_2^\top)^{-1} = \widetilde{S}^{-1} + V_3 V_4^\top,$$

where $V_3 = -\widetilde{S}^{-1} V_1 (I + V_2^\top \widetilde{S}^{-1} V_1)^{-1}$ and $V_4 = \widetilde{S}^{-1} V_2$ both have size $n \times r_t$. Thus, this means $(\widetilde{S}^{\mathrm{new}})^{-1} - \widetilde{S}^{-1}$ has a rank r_t decomposition.

Step 3: low-rank update of Hessian. Using the linearity and the mixed product property (Part 2 of Fact 9.10) of Kronecker product, we can find a low-rank update to $(\widetilde{S}^{\mathrm{new}})^{-1} \otimes (\widetilde{S}^{\mathrm{new}})^{-1}$. More precisely, we can rewrite $(\widetilde{S}^{\mathrm{new}})^{-1} \otimes (\widetilde{S}^{\mathrm{new}})^{-1}$ as follows:

$$(\widetilde{S}^{\mathrm{new}})^{-1} \otimes (\widetilde{S}^{\mathrm{new}})^{-1} = (\widetilde{S}^{-1} + V_3 V_4^\top) \otimes (\widetilde{S}^{-1} + V_3 V_4^\top) = \widetilde{S}^{-1} \otimes \widetilde{S}^{-1} + \mathcal{S}_{\mathrm{diff}}.$$

The term $\mathcal{S}_{\mathrm{diff}}$ is the difference that we want to compute, we can show

$$\begin{aligned}
\mathcal{S}_{\mathrm{diff}} &= \widetilde{S}^{-1} \otimes (V_3 V_4^\top) + (V_3 V_4^\top) \otimes \widetilde{S}^{-1} + (V_3 V_4^\top) \otimes (V_3 V_4^\top) \\
&= (\widetilde{S}^{-1/2} \otimes V_3) \cdot (\widetilde{S}^{-1/2} \otimes V_4^\top) + (V_3 \otimes \widetilde{S}^{-1/2}) \cdot (V_4^\top \otimes \widetilde{S}^{-1/2}) + (V_3 \otimes V_3) \cdot (V_4^\top \otimes V_4^\top) \\
&= Y_1 \cdot Y_2^\top
\end{aligned}$$

where Y_1 and Y_2 both have size $n^2 \times nr_t$. In this way we get a low-rank update to the Hessian:

$$\widetilde{H}^{\mathrm{new}} = \mathsf{A} \cdot ((\widetilde{S}^{\mathrm{new}})^{-1} \otimes (\widetilde{S}^{\mathrm{new}})^{-1}) \cdot \mathsf{A}^\top = \widetilde{H} + (\mathsf{A}Y_1) \cdot (\mathsf{A}Y_2)^\top.$$

9.4.2 Computing Hessian inverse efficiently

In this section we show how to compute the Hessian inverse efficiently.

Using Woodbury identity again, we have a low rank update to $\widetilde{H}^{-1}$:

$$(\widetilde{H}^{\mathrm{new}})^{-1} = \left(\widetilde{H} + (\mathsf{A}Y_1) \cdot (\mathsf{A}Y_2)^\top\right)^{-1} = \widetilde{H}^{-1} - \widetilde{H}^{-1} \cdot \mathsf{A}Y_1 \cdot (I + Y_2^\top \mathsf{A}^\top \cdot \mathsf{A}Y_1)^{-1} \cdot Y_2^\top \mathsf{A}^\top \cdot \widetilde{H}^{-1}$$

The second term in the above equation has rank nr. Thus $(\widetilde{H}^{\mathrm{new}})^{-1} - \widetilde{H}^{-1}$ has a rank nr decomposition. To compute $(\widetilde{H}^{\mathrm{new}})^{-1}$ in each iteration, we first compute $\mathsf{A}Y_1, \mathsf{A}Y_2 \in \mathbb{R}^{m \times nr_t}$ and multiply it with $\widetilde{H}^{-1} \in \mathbb{R}^{m \times m}$ to get $\widetilde{H}^{-1} \cdot \mathsf{A}Y_1, \widetilde{H}^{-1} \cdot \mathsf{A}Y_2 \in \mathbb{R}^{m \times nr_t}$. Then we compute $I + (Y_2^\top \mathsf{A}^\top) \cdot (\mathsf{A}Y_1) \in \mathbb{R}^{nr_t \times nr_t}$ and find its inverse $(I + Y_2^\top \mathsf{A}^\top \cdot \mathsf{A}Y_1)^{-1} \in \mathbb{R}^{nr_t \times nr_t}$. Finally, we multiply $\widetilde{H}^{-1} \cdot \mathsf{A}Y_1, \widetilde{H}^{-1} \cdot \mathsf{A}Y_2 \in \mathbb{R}^{m \times nr_t}$ and $(I + Y_2^\top \mathsf{A}^\top \cdot \mathsf{A}Y_1)^{-1} \in \mathbb{R}^{nr_t \times nr_t}$ together to obtain $(\widetilde{H})^{-1}\mathsf{A}Y_1 \cdot (I + Y_2^\top \mathsf{A}^\top \cdot \mathsf{A}Y_1)^{-1} \cdot Y_2^\top \mathsf{A}^\top (\widetilde{H})^{-1} \in \mathbb{R}^{m \times m}$, as desired. Using fast matrix multiplication in each aforementioned step, the total computation cost is bounded by

$$O(\mathcal{T}_{\mathrm{mat}}(m, n^2, nr_t) + \mathcal{T}_{\mathrm{mat}}(m, m, nr_t) + (nr_t)^\omega). \tag{9.4}$$

9.4.3 General amortization method

As mentioned in the previous sections, our algorithm relies on the maintenance of the slack matrix and the inverse of the Hessian matrix via low-rank updates. In each iteration, the time to update $\widetilde{S}$ and $\widetilde{H}$ to $\widetilde{S}_{\mathrm{new}}$ and $\widetilde{H}_{\mathrm{new}}$ is proportional to the magnitude of low-rank change in $\widetilde{S}$, namely $r_t = \mathrm{rank}(\widetilde{S}_{\mathrm{new}} - \widetilde{S})$. To deal with r_t, we propose a general amortization method which extends the analysis of several previous work [CLS19, LSZ19, JKL+20]. We first prove a tool to characterize intrinsic properties of the low-rank updates, which may be of independent interest.

Theorem 9.3 (Informal version of Theorem 9.29)**.** *Given a sequence of approximate slack matrices $\widetilde{S}^{(1)}, \widetilde{S}^{(2)}, \dots, \widetilde{S}^{(T)} \in \mathbb{R}^{n \times n}$ generated by Algorithm 42, let $r_t = \mathrm{rank}(\widetilde{S}^{(t+1)} - \widetilde{S}^{(t)})$ denotes the rank of update on $\widetilde{S}^{(t)}$. Then for any non-increasing vector $g \in \mathbb{R}^n_+$, we have*

$$\sum_{t=1}^{T} r_t \cdot g_{r_t} \le \widetilde{O}(T \cdot \|g\|_2).$$

Next, we show a proof sketch of Theorem 9.3.

Proof. For any matrix Z, let $|\lambda(Z)|_{[i]}$ denotes its i-th largest absolute eigenvalue. We use the following potential function $\Phi_g(Z) := \sum_{i=1}^n g_i \cdot |\lambda(Z)|_{[i]}$. Further, for convenient, we define $\Phi_g(S_1, S_2) := \Phi_g(S_1^{-1/2} S_2 S_1^{-1/2} - I)$. Our proof consists of the following two parts (Lemma 9.31 and Lemma 9.32):

- The change of the exact slack matrix increases the potential by a small amount, specifically $\Phi_g(S^{\mathrm{new}}, \widetilde{S}) - \Phi_g(S, \widetilde{S}) \le \|g\|_2$.
- The change of the approximate slack matrix decreases the potential proportionally to the update rank, specifically $\Phi_g(S^{\mathrm{new}}, \widetilde{S}^{\mathrm{new}}) - \Phi_g(S^{\mathrm{new}}, \widetilde{S}) \le -r_t \cdot g_{r_t}$.

In each iteration, the change of potential is composed of the changes of the exact and the approximate slack matrices:

$$\Phi_g(S^{\mathrm{new}}, \widetilde{S}^{\mathrm{new}}) - \Phi_g(S, \widetilde{S}) = \Phi_g(S^{\mathrm{new}}, \widetilde{S}) - \Phi_g(S, \widetilde{S}) + \Phi_g(S^{\mathrm{new}}, \widetilde{S}^{\mathrm{new}}) - \Phi_g(S^{\mathrm{new}}, \widetilde{S}).$$

Note that $\Phi_g(S, \widetilde{S}) = 0$ holds in the beginning of our algorithm and $\Phi_g(S, \widetilde{S}) \ge 0$ holds throughout the algorithm, combining the observations above we have $T \cdot \|g\|_2 - \sum_{t=1}^T r_t \cdot g_{r_t} \ge 0$ as desired. $\square$

Amortized analysis. Next we show how to use Theorem 9.3 to prove that our algorithm has an amortized cost of $m^{\omega - 1/4} + m^2$ cost per iteration when $m = \Omega(n^2)$. Note that in this case there are $\sqrt{n} = m^{1/4}$ iterations.

When $m = \Omega(n^2)$, the dominating term in our cost per iteration (see Eq. (9.4)) is $\mathcal{T}_{\mathrm{mat}}(m, m, nr_t)$. We use fast rectangular matrix multiplication to upper bound this term by

$$\mathcal{T}_{\mathrm{mat}}(m, m, nr_t) \le m^2 + m^{2-\frac{\alpha(\omega-2)}{1-\alpha}} \cdot n^{\frac{\omega-2}{1-\alpha}} \cdot r_t^{\frac{\omega-2}{1-\alpha}}.$$

We define a non-increasing sequence $g \in \mathbb{R}^n$ as $g_i = i^{\frac{\omega-2}{1-\alpha}-1}$. This g is tailored for the above equation, and its ℓ_2 norm is bounded by $\|g\|_2 \le n^{\frac{(\omega-2)}{1-\alpha}-1/2}$. Then using Theorem 9.3 we have

$$\sum_{t=1}^{T} r_t^{\frac{\omega-2}{1-\alpha}} = \sum_{t=1}^{T} r_t \cdot r_t^{\frac{\omega-2}{1-\alpha}-1} = \sum_{t=1}^{T} r_t \cdot g_{r_t} \le T \cdot n^{\frac{(\omega-2)}{1-\alpha}-1/2}.$$

Combining this and the previous equation, and since we assume $m = \Omega(n^2)$, we have

$$\sum_{t=1}^{T} \mathcal{T}_{\mathrm{mat}}(m, m, nr_t) \le T \cdot (m^2 + m^{2-\frac{\alpha(\omega-2)}{1-\alpha}} \cdot n^{\frac{2(\omega-2)}{1-\alpha}-1/2}) = T \cdot (m^2 + m^{\omega-1/4}).$$

Since $T = \widetilde{O}(m^{1/4})$, we proved the desired computational complexity in Theorem 9.1.

9.5 Solving SDP With Hybrid Barrier

Volumetric barrier was first proposed by Vaidya [Vai89a] for the polyhedral, and was generalized to the spectrahedra $\{y \in \mathbb{R}^m : y_1 A_1 + \cdots + y_m A_m \succeq 0\}$ by Nesterov and Nemirovski [NN94]. They showed that the volumetric barrier ϕ_{vol} can make the interior point method converge in $\sqrt{m}n^{1/4}$ iterations, while the log barrier $\phi_{\log}$ need $\sqrt{n}$ iterations. By combining the volumetric barrier and the log barrier, they also showed that the hybrid barrier achieves $(mn)^{1/4}$ iterations. Anstreicher [Ans00] gave a much simplified proof of this result.

We show that the hybrid barrier also fits into our robust IPM framework. And we can apply our newly developed low-rank update and amortization techniques in the log barrier case to efficiently implement the SDP solver based on hybrid barrier. The informal version of our result is stated in below.

Theorem 9.4 (Informal version of Theorem 9.58)**.** *There is an SDP algorithm based on hybrid barrier which takes* $(mn)^{1/4}\log(1/\epsilon)$ *iterations with cost-per-iteration* $O^*(m^2n^\omega + m^4)$*.*

In particular, our algorithm improves [Ans00] in nearly all parameter regimes. For example, if $m = n^2$*, our new algorithm takes* $n^{8.75}$ *time while [Ans00] takes* $n^{10.75}$ *time. If* $m = n$*, our new algorithm takes* $n^{\omega+2.5}$ *time, while [Ans00] takes* $n^{6.5}$ *time.*

The hybrid barrier function is as follows:

$$\phi(y) := 225\sqrt{\frac{n}{m}} \cdot \left(\phi_{\mathrm{vol}}(y) + \frac{m-1}{n-1} \cdot \phi_{\log}(y)\right),$$

where $\phi_{\mathrm{vol}}(y) = \frac{1}{2}\log\det(\nabla^2\phi_{\log}(y))$. According to our general IPM framework (Algorithm 40), we need to efficiently compute the gradient and Hessian of $\phi(y)$. Recall from [Ans00] that the gradient of the volumetric barrier is:

$$(\nabla\phi_{\mathrm{vol}}(y))_i = -\mathrm{tr}[H(S)^{-1} \cdot \mathsf{A}(S^{-1}A_iS^{-1} \otimes S^{-1})\mathsf{A}^\top] \quad \forall i \in [m].$$

And the Hessian can be written as $\nabla^2\phi_{\mathrm{vol}}(y) = 2Q(S) + R(S) - 2T(S)$, where for any $i, j \in [m]$,

$$\begin{aligned} Q(S)_{i,j} &= \mathrm{tr}[H(S)^{-1}\mathsf{A}(S^{-1}A_iS^{-1}A_jS^{-1} \otimes_S S^{-1})\mathsf{A}^\top], \\ R(S)_{i,j} &= \mathrm{tr}[H(S)^{-1}\mathsf{A}(S^{-1}A_iS^{-1} \otimes_S S^{-1}A_jS^{-1})\mathsf{A}^\top], \\ T(S)_{i,j} &= \mathrm{tr}[H(S)^{-1}\mathsf{A}(S^{-1}A_iS^{-1} \otimes_S S^{-1})\mathsf{A}^\top H(S)^{-1}\mathsf{A}(S^{-1}A_jS^{-1} \otimes_S S^{-1})\mathsf{A}^\top]. \end{aligned} \tag{9.5}$$

Here, $\otimes_S$ is the symmetric Kronecker product[9].

A straight-forward implementation of the hybrid barrier-based SDP algorithm can first compute the matrices $S^{-1}A_i$ and $S^{-1}A_iS^{-1}A_j$ for all $i \in \{1, 2, \cdots, m\}$ for all $j \in \{1, 2, \cdots, m\}$ in time $O(m^2n^\omega)$. The gradient $\nabla\phi(y)$ and the Hessian of $\phi_{\log}(y)$ can be computed by taking traces of these matrices. To compute $\nabla\phi_{\mathrm{vol}}(y), Q(S), R(S), T(S)$, we observe that each entry of these matrices can be written as the inner-product between $H(S)^{-1}$ and some matrices formed in terms of

[9] $X \otimes_S Y := \frac{1}{2}(X \otimes Y + Y \otimes X)$.

$\mathrm{tr}[S^{-1}A_iS^{-1}A_jS^{-1}A_k]$ and $\mathrm{tr}[S^{-1}A_iS^{-1}A_jS^{-1}A_kS^{-1}A_l]$ for $i,j,k,l \in [m]$. Hence, we can spend $O(m^4n^2)$-time computing these traces and then get $\nabla\phi_{\mathrm{vol}}(y), Q(S), R(S), T(S)$ in $O(m^{\omega+2})$-time. After obtaining the gradient and Hessian of the hybrid barrier function, we finish the implementation of IPM SDP solver by computing the Newton direction $\delta_y = -(\nabla^2\phi(y))^{-1}(\eta b - \nabla\phi(y))$. (More details are given in Section E.3).

To speedup the straight forward implementation, we observe two bottleneck steps in each iteration:

1. Computing the traces $\mathrm{tr}[S^{-1}A_iS^{-1}A_jS^{-1}A_kS^{-1}A_l]$ for $i,j,k,l \in [m]$.
2. Computing the matrices $Q(S), R(S), T(S)$.

To handle the first issue, we use the low-rank update and amortization techniques introduced in the previous section to approximate the change of the slack matrix S by a low-rank matrix. One challenge for the volumetric barrier is that its Hessian (Eq. (9.5)) is much more complicated than the log barrier's Hessian $H(S)$. For $H(S)$, if we replace S with its approximation $\widetilde{S}$, then $H(\widetilde{S})$ will be a PSD approximation of $H(S)$. However, this may not hold for the volumetric barrier's Hessian if we simply replace all the S in $\nabla^2\phi(y)$ by its approximation $\widetilde{S}$. We can resolve this challenge by carefully choosing the approximation place: if we approximate the second S in the trace, i.e., $\mathrm{tr}[S^{-1}A_i\widetilde{S}^{-1}A_jS^{-1}A_kS^{-1}A_l]$, then the resulting matrix will be a PSD approximation of $\nabla^2\phi(y)$. In other words, the Condition 1 in our robust IPM framework (Lemma 9.2) is satisfied. Notice that in each iteration, we only need to maintain the change of $\mathrm{tr}[S^{-1}A_i\widetilde{S}^{-1}A_jS^{-1}A_kS^{-1}A_l]$, which by the low-rank guarantee, can be written as

$$\mathrm{tr}[A_lS^{-1}A_i \cdot V_3V_4^\top \cdot A_jS^{-1}A_kS^{-1}],$$

where $V_3, V_4 \in \mathbb{R}^{n\times r_t}$. Then, we can first compute the matrices

$$\left\{A_lS^{-1}A_iV_3 \in \mathbb{R}^{n\times r_t}\right\}_{i,l\in[m]} \text{ and } \left\{V_4^\top A_jS^{-1}A_kS^{-1} \in \mathbb{R}^{r_t\times n}\right\}_{j,k\in[m]}.$$

It takes $m^2 \cdot \mathcal{T}_{\mathrm{mat}}(n, n, r_t)$-time. And we can compute all the traces $\mathrm{tr}[S^{-1}A_i\widetilde{S}^{-1}A_jS^{-1}A_kS^{-1}A_l]$ simultaneously in $\mathcal{T}_{\mathrm{mat}}(m^2, nr_t, m^2)$ by batching them together and using fast matrix multiplication on a m^2-by-nr_t matrix and a nr_t-by-m^2 matrix. A similar amortized analysis in the log barrier case can also be applied here to get the amortized cost-per-iteration for the low-rank update. One difference is that the potential function $\Phi_g(Z)$ (defined in Section 9.4.3) changes more drastically in the hybrid barrier case. And we can only get $\sum_{t=1}^T r_t \cdot g_{r_t} \leq O(T \cdot (n/m)^{1/4} \cdot \|g\|_2 \cdot \log n)$.

For the second issue, we note that computing the $T(S)$ matrix is the most time-consuming step, which need $m^{\omega+2}$-time. In [Ans00], it is proved that $\frac{1}{3}Q(S) \preceq \nabla^2\phi_{\mathrm{vol}}(y) \preceq Q(S)$. With this PSD approximation, our robust IPM framework enables us to use $Q(S)$ as a "proxy Hessian" of the volumetric barrier. That is, in each iteration, we only compute $Q(S)$ and ignore $R(S)$ and $T(S)$. And computing $Q(S)$ only takes $O(m^4)$-time, which improves the $m^{\omega+2}$ term in the straight forward implementation.

Combining them together, we obtain the running time in Theorem 9.4. More details are provided in Section 9.12.

Lee-Sidford barrier for SDP? In LP, the hybrid barrier was improved by Lee and Sidford [LS19] to achieve $O^*(\sqrt{\min\{m, n\}})$ iterations. For SDP, we hope to design a barrier function with $O^*(\sqrt{m})$ iterations. However, the Lee-Sidford barrier function does not have a direct correspondence in SDP due to the following reasons. First, [LS19] defined the barrier function in the dual space of LP which is a polyhedron, while for SDP, the dual space is a spectrahedron. Thus, the geometric intuition of the Lee-Sidford barrier (John's ellipsoid) may not be helpful to design the corresponding barrier for SDP. Second, efficient implementation of Lee-Sidford barrier involves a primal-dual central path method [BLSS20]. However, the cost of following primal-dual central path in SDP is prohibitive since this involves solving Lyapunov equations in $\mathbb{R}^{n\times n}$. Third, the Lewis weights play an important role in the Lee-Sidford barrier.

Notice that in LP, the volumetric barrier can be considered as reweighing the constraints in the log barrier based on the leverage score, and the Lee-Sidford barrier uses Lewis weights for reweighing to improve the volumetric barrier. However, in SDP, we have observed that the leverage score vector becomes the leverage score matrix. Thus, we may need some matrix version of Lewis weights to define the Lee-Sidford barrier for SDP. Section E.4 studies several properties of the leverage score matrix and give an algorithm to efficiently maintain this matrix in each iteration of the IPM, which might be the first step towards improving the SDP hybrid barrier.

9.6 Related Work

Other SDP solvers. The interior point method is a second-order algorithm. Second-order algorithms usually have logarithmic dependence on the error parameter $1/\epsilon$. First-order algorithms do not need to use second-order information, but they usually have polynomial dependence on $1/\epsilon$. There is a long list of work focusing on first-order algorithms [AK07, JY11, ALO16, GH16, AZL17, CDST19, LP20a, YTF$^+$19, JLL$^+$20]. Solving SDPs has also attracted attention in the parallel setting [JY11, JY12, ALO16, JLL$^+$20].

Cutting plane method. Cutting plane method is a class of optimization algorithms that iteratively queries a separation oracle to cut the feasible set that contains the optimal solution. There has been a long line of work to obtain fast cutting plane methods [Sho77, YN76, Kha80, KTE88, NN89, Vai89a, AV95, BV02, LSW15, JLSW20].

Low-rank approximation Low-rank approximation is a well-studied topic in numerical linear algebra [Sar06, CW13, BWZ16, SWZ17, SWZ19]. Many different settings of that problem have been studied. In this chapter, we are dealing with Kronecker product type low rank approximation.

Applications of SDP. As described by [JKL+20], $m = \Omega(n^2)$ is an essential case of using SDP to solve many practical combinatorial optimization problems. Here we provide a list of examples, e.g., the sparsest cut [ARV09], the c-balanced graph separation problem [FHL08] and the minimum uncut [ACMM05] can be solved by SDP with $m = \Omega(n^3)$. The optimal experiment design [VBW98], Haplotype frequencies estimation [HH06] and embedding of finite metric spaces into ℓ^2 [LLR95] need to solve SDPs with $m = \Omega(n^2)$.

9.7 Preliminary

9.7.1 Notations

Basic matrix notations. For a square matrix X, we use $\mathrm{tr}[X]$ to denote the trace of X.

We use $\|\cdot\|_2$ and $\|\cdot\|_F$ to denote the spectral norm and Frobenious norm of a matrix. Let us use $\|\cdot\|_1$ to represent the Schatten-1 norm of a matrix, i.e., $\|A\|_1 = \mathrm{tr}[(A^* A)^{1/2}]$.

We say a symmetric matrix $A \in \mathbb{R}^{n\times n}$ is positive semi-definite (PSD, denoted as $A \succeq 0$) if for any vector $x \in \mathbb{R}^n$, $x^\top A x \geq 0$. We say a symmetric matrix $A \in \mathbb{R}^{n\times n}$ is positive definite (PD, denoted as $A \succ 0$) if for any vector $x \in \mathbb{R}^n$, $x^\top A x > 0$.

We define $\mathbb{S}^{n\times n}_{\succ 0}$ to be the set of all n-by-n symmetric positive definite matrices.

Let us define $\mathbb{S}^{n\times n}_{\succeq 0}$ to be the set of all n-by-n symmetric positive semi-definite matrices.

For a matrix $A \in \mathbb{R}^{m\times n}$, we use $\lambda(A) \in \mathbb{R}^n$ to denote the eigenvalues of A.

For any vector $v \in \mathbb{R}^n$, we use $v_{[i]}$ to denote the i-th largest entry of v.

For a matrix $A \in \mathbb{R}^{m\times n}$, and subsets $S_1 \subseteq [m], S_2 \subseteq [n]$, we define $A_{S_1,S_2} \in \mathbb{R}^{|S_1|\times|S_2|}$ to be the submatrix of A that only has rows in S_1 and columns in S_2. We also define $A_{S_1,:} \in \mathbb{R}^{|S_1|\times n}$ to be the submatrix of A that only has rows in S_1, and $A_{:,S_2} \in \mathbb{R}^{m\times|S_2|}$ to be the submatrix of A that only has columns in S_2.

For two symmetric matrices $A, B \in \mathbb{R}^{n\times n}$, we say $A \preceq B$ (or equivalently, $B \succeq A$), if $B - A$ is a PSD matrix.

Fact 9.5 (Spectral norm implies Loewner order)**.** *Let $A, B \in \mathbb{R}^{n\times n}$ be two symmetric PSD matrices. Then, for any $\epsilon \in (0,1)$,*

$$\left\|A^{-1/2} B A^{-1/2} - I\right\|_2 \leq \epsilon$$

implies

$$(1-\epsilon)A \preceq B \preceq (1+\epsilon)A.$$

Fact 9.6 (Trace property of matrix Loewner order)**.** *Given symmetric PSD matrices $A, B \in \mathbb{R}^n$. Suppose $(1+\epsilon)^{-1} \cdot A \preceq \widetilde{A} \preceq (1+\epsilon) \cdot A$, then*

$$(1+\epsilon)^{-1} \cdot \mathrm{tr}[AB] \leq \mathrm{tr}[\widetilde{A}B] \leq (1+\epsilon) \cdot \mathrm{tr}[AB].$$

Proof. Consider the spectral decomposition of B: $B = \sum_{i=1}^n \lambda_i v_i v_i^\top$ where $\lambda_i \geq 0$. Then

$$\begin{aligned}\mathrm{tr}[\widetilde{A}B] &= \mathrm{tr}[\widetilde{A} \cdot (\sum_{i=1}^n \lambda_i v_i v_i^\top)] \\ &= \sum_{i=1}^n \lambda_i v_i^\top \widetilde{A} v_i \\ &\leq (1+\epsilon) \cdot (\sum_{i=1}^n \lambda_i v_i^\top A v_i) \\ &= \mathrm{tr}[AB].\end{aligned}$$

Similarly, $\mathrm{tr}[\widetilde{A}B] \geq (1+\epsilon)^{-1} \cdot \mathrm{tr}[AB]$. □

Matrix related operations For two matrices $A, B \in \mathbb{R}^{m \times n}$, we define the matrix inner product $\langle A, B \rangle := \mathrm{tr}[A^\top B]$.

We use vec[] to denote matrix vectorization: for a matrix $A \in \mathbb{R}^{m \times n}$, $\mathrm{vec}[A] \in \mathbb{R}^{mn}$ is defined to be $\mathrm{vec}[A]_{(j-1)\cdot n+i} = A_{i,j}$ for any $i \in [m]$ and $j \in [n]$, i.e.,

$$\mathrm{vec}[A] = \begin{bmatrix} A_{:,1} \\ \vdots \\ A_{:,n} \end{bmatrix} \in \mathbb{R}^{mn}.$$

We use $\otimes$ to denote matrix Kronecker product: for matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{p \times q}$, $A \otimes B \in \mathbb{R}^{pm \times qn}$ is defined to be $(A \otimes B)_{p(i-1)+s, q(j-1)+t} = A_{i,j} \cdot B_{s,t}$ for any $i \in [m]$, $j \in [n]$, $s \in [p]$, $[t] \in [q]$, i.e.,

$$A \otimes B = \begin{bmatrix} A_{1,1} \cdot B & A_{1,2} \cdot B & \ldots & A_{1,n} \cdot B \\ A_{2,1} \cdot B & A_{2,2} \cdot B & \cdots & A_{2,n} \cdot B \\ \vdots & \vdots & \ddots & \vdots \\ A_{m,1} \cdot B & A_{m,2} \cdot B & \ldots & A_{m,n} \cdot B \end{bmatrix} \in \mathbb{R}^{pm \times qn}.$$

Definition 9.3 (Stacking matrices)**.** Let $A_1, A_2, \cdots, A_m \in \mathbb{R}^{n \times n}$ be m symmetric matrices. We use $\mathsf{A} \in \mathbb{R}^{m \times n^2}$ to denote the matrix that is constructed by stacking the m vectorizations $\mathrm{vec}[A_1], \cdots, \mathrm{vec}[A_m] \in \mathbb{R}^{n^2}$ as rows of A, i.e.,

$$\mathsf{A} := \begin{bmatrix} \mathrm{vec}[A_1]^\top \\ \vdots \\ \mathrm{vec}[A_m]^\top \end{bmatrix} \in \mathbb{R}^{m \times n^2}.$$

Fact 9.7. *For any $\epsilon_1, \epsilon_2 \in (0, 1/10)$. Let $D \in \mathbb{R}^{n \times n}$ be a diagonal matrix with non-negative entries and such that $\|D^2 - I\|_F \leq \epsilon_1$. Let $X \in \mathbb{R}^{n \times n}$ be a matrix that has bounded norm, e.g., $\|X\|_2 \leq \epsilon_2$. Then*

$$\|DXD - X\|_F \leq 3 \cdot \epsilon_1 \cdot \epsilon_2.$$

Proof. Denote $D = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$. We have

$$\begin{aligned}
\|DXD - X\|_F &\leq \|(D-I)X(D-I) + (D-I)X + X(D-I)\|_F \\
&\leq \|(D-I)X(D-I)\|_F + 2 \cdot \|(D-I)X\|_F \\
&\leq 3 \cdot \|(D-I)X\|_F \\
&\leq 3 \cdot \left(\mathrm{tr}[(D-I)^2 X^2]\right)^{1/2} \\
&\leq 3 \cdot \left(\epsilon_2^2 \cdot \mathrm{tr}[(D-I)^2]\right)^{1/2} \\
&= 3 \cdot \epsilon_2 \cdot \left(\sum_{i=1}^n (\sigma_i - 1)^2\right)^{1/2} \\
&\leq 3 \cdot \epsilon_2 \cdot \left(\sum_{i=1}^n (\sigma_i^2 - 1)^2\right)^{1/2} \\
&\leq 3 \cdot \epsilon_2 \cdot \epsilon_1
\end{aligned}$$

where the second step uses triangle inequality, the third step uses $-I \preceq D - I \preceq I$, the fifth step uses $\|X\|_2 \leq \epsilon_2$, the penultimate step uses $\sigma_i \geq 0$, and the last step uses $\|D^2 - I\|_F \leq \epsilon_1$. □

9.7.2 Tools: Woodbury identity

We state a common fact on matrix inverse update in [Woo49, Woo50].

Fact 9.8 (Woodbury matrix identity). *Given two integers n and k. Let $n \geq k$. For square matrix $A \in \mathbb{R}^{n \times n}$, tall matrix $B \in \mathbb{R}^{n \times k}$, square matrix $C \in \mathbb{R}^{k \times k}$, fat matrix $D \in \mathbb{R}^{k \times n}$,*

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}.$$

9.7.3 Tools: Properties of matrix operations

Fact 9.9 (Matrix inner product). *For two matrices $A, B \in \mathbb{R}^{m \times n}$, we have $\langle A, B \rangle = \mathrm{tr}[A^\top B] = \mathrm{vec}[A]^\top \mathrm{vec}[B]$.*

Fact 9.10 (Basic properties of Kronecker product). *The Kronecker product $\otimes$ satisfies the following properties.*

1. *For matrices $A \in \mathbb{R}^{a \times n}$ and $B \in \mathbb{R}^{b \times m}$, we have $(A \otimes B)^\top = A^\top \otimes B^\top \in \mathbb{R}^{nm \times ab}$.*
2. *For matrices $A \in \mathbb{R}^{a \times n}$, $B \in \mathbb{R}^{b \times m}$, $C \in \mathbb{R}^{n \times c}$, $D \in \mathbb{R}^{m \times d}$, we have $(A \otimes B) \cdot (C \otimes D) = (AC \otimes BD) \in \mathbb{R}^{ab \times cd}$.*

Fact 9.11 (Spectral properties of Kronecker product). *The Kronecker product satisfies the following spectral properties.*

1. *For matrices A, B, if A and B are PSD matrices, then $A \otimes B$ is also PSD.*
2. *For two PSD matrices A and $B \in \mathbb{R}^{n \times n}$, if $A \preceq B$, then $A \otimes A \preceq B \otimes B$.*

The following result is often used in SDP-related calculations.

Fact 9.12 (Kronecker product and vector multiplication). *Given $A \in \mathbb{R}^{m \times n}, B \in \mathbb{R}^{n \times k}, C \in \mathbb{R}^{k \times l}$, $D \in \mathbb{R}^{l \times m}$, we have*

1. $\mathrm{vec}[ABC] = (C^\top \otimes A) \cdot \mathrm{vec}[B]$.

   Note that $ABC \in \mathbb{R}^{m\times l}$, $C^\top \otimes A \in \mathbb{R}^{ml\times nk}$, *and* $\mathrm{vec}[B] \in \mathbb{R}^{nk}$.

2. $\mathrm{tr}[ABCD] = \mathrm{vec}[D]^\top \cdot (C^\top \otimes A) \cdot \mathrm{vec}[B]$.

   Note that $ABCD \in \mathbb{R}^{m\times m}$, $\mathrm{vec}[D] \in \mathbb{R}^{ml}$, $C^\top \otimes A \in \mathbb{R}^{ml\times nk}$, *and* $\mathrm{vec}[B] \in \mathbb{R}^{nk}$.

We state a standard fact for Kronecker product.

Fact 9.13 (Positive Semidefinite property of Kronecker product)**.** *Let m, n denote two positive integers. Given a matrix $\mathsf{A} \in \mathbb{R}^{m\times n^2}$, let $S, \widetilde{S} \in \mathbb{R}^{n\times n}$ be two PSD matrices. Define*

$$H := \mathsf{A} \cdot (S^{-1} \otimes S^{-1}) \cdot \mathsf{A}^\top \in \mathbb{R}^{m\times m}, \quad \textit{and} \quad \widetilde{H} := \mathsf{A} \cdot (\widetilde{S}^{-1} \otimes \widetilde{S}^{-1}) \cdot \mathsf{A}^\top \in \mathbb{R}^{m\times m}.$$

Then, for any accuracy parameter $\alpha \geq 1$, if $\widetilde{S}$ is an α-PSD approximation of S, i.e., $\alpha^{-1}S \preceq \widetilde{S} \preceq \alpha S$, then

$$\alpha^{-2}H \preceq \widetilde{H} \preceq \alpha^2 H.$$

Proof. Given any vector $v \in \mathbb{R}^m$, we can write $v^\top H v$ and $v^\top \widetilde{H} v$ in the following way:

$$\begin{aligned} v^\top H v = & \sum_{i=1}^m \sum_{j=1}^m v_i v_j H_{i,j} = \sum_{i=1}^m \sum_{j=1}^m v_i v_j \mathrm{tr}[S^{-1}A_i S^{-1} A_j] \\ = & \ \mathrm{tr}\Big[S^{-1/2}\Big(\sum_{i\in[m]} v_i A_i\Big) S^{-1} \Big(\sum_{i\in[m]} v_i A_i\Big) S^{-1/2}\Big] \\ = & \ \Big\|\mathrm{vec}\Big[S^{-1/2}\Big(\sum_{i\in[m]} v_i A_i\Big) S^{-1/2}\Big]\Big\|_2^2 \\ = & \ \Big\|\Big(S^{-1/2} \otimes S^{-1/2}\Big)\mathrm{vec}\Big[\sum_{i=1}^m v_i A_i\Big]\Big\|_2^2, \end{aligned} \tag{9.6}$$

where the last line follows from Fact 9.12. Similarly,

$$v^\top \widetilde{H} v = \Big\|\Big(\widetilde{S}^{-1/2} \otimes \widetilde{S}^{-1/2}\Big)\mathrm{vec}\Big[\sum_{i=1}^m v_i A_i\Big]\Big\|_2^2. \tag{9.7}$$

Since the right hand side of Eq. (9.6) and Eq. (9.7) are non-negative for any $v \in \mathbb{R}^m$, both H and $\widetilde{H}$ are PSD matrices.

Since $\alpha^{-1} S \preceq \widetilde{S} \preceq \alpha S$, we have

$$\alpha^{-1} S^{-1} \preceq \widetilde{S}^{-1} \preceq \alpha S^{-1}.$$

By Fact 9.11, it further implies that

$$\alpha^{-2} S^{-1} \otimes S^{-1} \preceq \widetilde{S}^{-1} \otimes \widetilde{S}^{-1} \preceq \alpha^2 S^{-1} \otimes S^{-1}.$$

Let $b := \mathrm{vec}[\sum_{i=1}^m v_i A_i]^\top$. We have

$$\begin{aligned}
\left\| \left(\widetilde{S}^{-1/2} \otimes \widetilde{S}^{-1/2} \right) b \right\|_2^2 &= b^\top (\widetilde{S}^{-1/2} \otimes \widetilde{S}^{-1/2}) \cdot (\widetilde{S}^{-1/2} \otimes \widetilde{S}^{-1/2}) b \\
&= b^\top (\widetilde{S}^{-1} \cdot \widetilde{S}^{-1}) b \\
&\leq \alpha^2 \cdot b^\top (S^{-1} \otimes S^{-1}) b \\
&= \alpha^2 \cdot \left\| \left(S^{-1/2} \otimes S^{-1/2} \right) b \right\|_2^2 . \qquad (9.8)
\end{aligned}$$

And

$$\left\| \left(\widetilde{S}^{-1/2} \otimes \widetilde{S}^{-1/2} \right) b \right\|_2^2 \geq \ \alpha^{-2} \cdot \left\| \left(S^{-1/2} \otimes S^{-1/2} \right) b \right\|_2^2 . \qquad (9.9)$$

Combining Eqs. (9.6)-(9.9), we come to

$$\alpha^{-2} \cdot v^\top H v \leq v^\top \widetilde{H} v \leq \alpha^2 \cdot v^\top H v.$$

Since v can be arbitrarily chosen from $\mathbb{R}^m$, we complete the proof. $\square$

We state another fact for Kronecker product in below:

Fact 9.14 (Kronecker product with equivalence for matrix norm)**.** *Given a constraint matrix* $\mathsf{A} \in \mathbb{R}^{m \times n^2}$ *and vector* $b \in \mathbb{R}^m$*. Let* $\eta > 0$ *denote a parameter. Let* $g(y, \eta) \in \mathbb{R}^m$ *be defined as*

$$g(y, \eta)_i = \eta b_i - \mathrm{tr}[S^{-1} A_i] \quad \forall i \in [m].$$

Let $X \in \mathbb{R}^{n \times n}$ denote a matrix that

$$\langle X, A_i \rangle = \eta b_i \quad \forall i \in [m].$$

Let $H := \mathsf{A}(S^{-1} \otimes S^{-1})\mathsf{A}^\top$. If matrix S is a PSD matrix, then we have

$$g(y,\eta)^\top H^{-1} g(y,\eta) = v^\top \mathsf{B}^\top (\mathsf{B}\mathsf{B}^\top)^{-1} \mathsf{B} v,$$

where $v := \mathrm{vec}[S^{1/2} X S^{1/2} - I] \in \mathbb{R}^{n^2}$ and $\mathsf{B} \in \mathbb{R}^{m \times n^2}$ is a matrix that i-th row is $B_i = \mathrm{vec}[S^{-1/2} A_i S^{-1/2}] \in \mathbb{R}^{n^2}$

Proof. We start with re-writing $g(y,\eta) \in \mathbb{R}^m$ as follows: for each $i \in [m]$

$$\begin{aligned} g(y,\eta)_i &= b_i \eta - \mathrm{tr}[S^{-1} A_i] \\ &= \mathrm{tr}[X A_i] - \mathrm{tr}[S^{-1} A_i] \\ &= \mathrm{tr}[(X - S^{-1}) A_i] \\ &= \mathrm{tr}[S^{1/2}(X - S^{-1}) S^{1/2} \cdot S^{-1/2} A_i S^{-1/2}] \\ &= \mathrm{tr}[(S^{1/2} X S^{1/2} - I) \cdot B_i]. \end{aligned}$$

Thus, using the definition of v, we have

$$g(y,\eta) = \mathsf{B} v.$$

Our next step is to rewrite H as follows: for each $i, j \in [m] \times [m]$

$$\begin{aligned} H_{i,j} &= \mathrm{tr}[A_i S^{-1} A_j S^{-1}] \\ &= \mathrm{tr}[S^{-1/2} A_i S^{-1/2} \cdot S^{-1/2} A_j S^{-1/2}] \\ &= \mathrm{tr}[B_i \cdot B_j] \end{aligned}$$

which implies that $H = \mathsf{B}\mathsf{B}^\top$.

Thus, combine all the above computations, we have

$$g(y,\eta)^\top H^{-1} g(y,\eta) = v^\top \mathsf{B}^\top (\mathsf{B}\mathsf{B}^\top)^{-1} \mathsf{B} v.$$

Therefore, we complete the proof. $\square$

9.7.4 Tools: Fast matrix multiplication

We use $\mathcal{T}_{\mathrm{mat}}(a, b, c)$ to denote the time of multiplying an $a \times b$ matrix with another $b \times c$ matrix. Fast matrix multiplication [Cop82, Wil12, LG14, GU18, CGLZ20, AW21] is a fundamental tool in theoretical computer science.

For $k \in \mathbb{R}_+$, we define $\omega(k) \in \mathbb{R}_+$ to be the value such that $\forall n \in \mathcal{N}_+$, $\mathcal{T}_{\mathrm{mat}}(n, n, n^k) = O(n^{\omega(k)})$.

For convenience we define three special values of $\omega(k)$. We define ω to be the fast matrix multiplication exponent, i.e., $\omega := \omega(1)$. We define $\alpha \in \mathbb{R}_+$ to be the dual exponent of matrix multiplication, i.e., $\omega(\alpha) = 2$. We define $\beta := \omega(2)$.

The following fact can be found in Lemma 3.6 of [JKL$^+$20], also see [BCS97].

Fact 9.15 (Convexity of $\omega(k)$)**.** *The function $\omega(k)$ is convex.*

The following fact can be found in Lemma A.5 of [CLS19].

Fact 9.16 (Fast rectangular matrix multiplication)**.** *For any two integers $r \leq n$, the time of multiplying an $n \times n$ matrix with another $n \times r$*

$$\mathcal{T}_{\mathrm{mat}}(n, n, r) \leq n^2 + r^{\frac{\omega-2}{1-\alpha}} \cdot n^{2-\frac{\alpha(\omega-2)}{(1-\alpha)}}.$$

The following fact can be found in Lemma A.4 of [CLS19].

Fact 9.17 (Relation of ω and α)**.** $\frac{\omega-2}{1-\alpha} - 1 \leq 0$*; that is,* $\omega + \alpha \leq 3$.

9.8 Our Algorithm and Result

We state our main result of Algorithm 42 as follows:

Theorem 9.18 (Main result for Algorithm 42)**.** *Given symmetric matrices $C, A_1, \cdots, A_m \in \mathbb{R}^{n \times n}$, and a vector $b \in \mathbb{R}^m$. Define matrix $\mathsf{A} \in \mathbb{R}^{m \times n^2}$ by stacking the m vectors*

$\mathrm{vec}[A_1], \cdots, \mathrm{vec}[A_m] \in \mathbb{R}^{n^2}$ *as rows. Consider the following SDP instance:*

$$\begin{aligned} \max_{X \in \mathbb{R}^{n \times n}} \quad & \langle C, X \rangle \\ \text{s.t.} \quad & \langle A_i, X \rangle = b_i, \ \forall i \in [m], \\ & X \succeq 0, \end{aligned}$$

There is a SDP algorithm (Algorithm 42) that runs in time

$$O^*\Big(\big(\sqrt{n}(m^2 + n^4) + m^\omega + n^{2\omega}\big) \cdot \log(1/\epsilon) \Big).$$

and outputs a PSD matrix $X \in \mathbb{R}^{n \times n}$ that satisfies

$$\langle C, X \rangle \geq \langle C, X^* \rangle - \epsilon \cdot \|C\|_2 \cdot R \quad \text{and} \quad \sum_{i=1}^m |\langle A_i, X \rangle - b_i| \leq 4n\epsilon \cdot \Big(R \sum_{i=1}^m \|A_i\|_1 + \|b\|_1 \Big), \tag{9.10}$$

where X^* is an optimal solution of the SDP instance, and $\|A_i\|_1$ is the Schatten 1-norm of matrix A_i.

Proof. The correctness (Eq. (9.10)) follows from Theorem 9.20. The running time follows from Theorem 9.24. □

Remark 9.2. For current matrix multiplication time $\omega \approx 2.373$ ([LG14]), the running time of our algorithm can be written as

$$O(\max\{n^{2\omega}, m^\omega\} \cdot \log(1/\epsilon)).$$

Therefore, when $m \geq n^{2-0.5/\omega} \approx n^{1.79}$, we have $\max\{n^{2\omega}, m^\omega\} \leq \sqrt{n} \cdot m^\omega$ and thus our algorithm is better than [JKL+20]. For the regimes when m is smaller, we can apply Algorithm 45-46 in Section 9.12 or Algorithm 1 in [JKL+20].

Corollary 9.19 (Tall SDPs)**.** *When $m = \Omega(n^2)$, we can solve SDP in $O(m^\omega \cdot \log(1/\epsilon))$ time for current $\omega \approx 2.373$.*

Proof. When $m = \Omega(n^2)$, the running time of Theorem 9.18 is

$$O\Big((m^2 \cdot \sqrt{n} + m^\omega + n^{4.5} + n^{2\omega}) \cdot \log(1/\epsilon)\Big) = O((m^2 \cdot \sqrt{n} + m^\omega) \cdot \log(1/\epsilon)).$$

For current $\omega \approx 2.373$, $m^2 \cdot \sqrt{n} = m^{2.25} < m^\omega$. $\square$

Note that the running time of [JKL+20] is $O(\sqrt{n} \cdot (n^\omega + m^\omega + mn^2)) = O(m^{\omega+0.25})$ when $m = \Omega(n^2)$.

9.9 Correctness

In this section we prove the correctness of our SDP solver Algorithm 42. In Section 9.9.1 we prove that $\widetilde{S} \in \mathbb{R}^{n\times n}$ updated by Algorithm 43 is a PSD approximation to the true slack matrix $S \in \mathbb{R}^{n\times n}$. In Section 9.9.2 we prove that the algorithm maintains $G = \widetilde{H}^{-1} \in \mathbb{R}^{m\times m}$, and $\widetilde{H}$ is a PSD approximation to the true Hessian matrix $H \in \mathbb{R}^{m\times m}$.

Theorem 9.20 (Correctness of Algorithm 42)**.** *Consider an SDP instance as in Definition 9.1 with no redundant constraints. Let us assume that the feasible region is bounded, i.e., any feasible solution $X \in \mathbb{R}^{n\times n}_{\succeq 0}$ satisfies $\|X\|_2 \leq R$. Then for any error parameter $0 < \epsilon \leq 0.01$ and Newton step size ϵ_N satisfying $\sqrt{\epsilon} < \epsilon_N \leq 0.1$, Algorithm 42 outputs, in $T = 40\epsilon_N^{-1}\sqrt{n}\log(n/\epsilon)$ iterations, a PSD matrix $X \in \mathbb{R}^{n\times n}_{\succeq 0}$ that satisfies*

$$\begin{aligned} &\langle C, X\rangle \geq \langle C, X^*\rangle - \epsilon \cdot \|C\|_2 \cdot R, \quad \textit{and} \\ &\textstyle\sum_{i=1}^m |\langle A_i, X\rangle - b_i| \leq 4n\epsilon \cdot \Big(R \sum_{i=1}^m \|A_i\|_1 + \|b\|_1\Big), \end{aligned} \tag{9.11}$$

where X^* is any optimal solution to the SDP instance, and $\|A_i\|_1$ is the Schatten 1-norm of matrix A_i.

Furthermore, in each iteration of Algorithm 42, the following invariant holds for $\alpha_H = 1 + 10^{-5}$:

$$\|S^{-1/2} S^{\text{new}} S^{-1/2} - I\|_F \leq \alpha_H \cdot \epsilon_N. \tag{9.12}$$

Algorithm 42 Our SDP solver with log barrier.

```
1: procedure SolveSDP($m, n, C, \{A_i\}_{i=1}^m$, $\mathsf{A} \in \mathbb{R}^{m \times n^2}$, $b \in \mathbb{R}^m$)
2:                                                                          ▷ Initialization
3:     Construct $\mathsf{A} \in \mathbb{R}^{m \times n^2}$ by stacking $m$ vectors $\mathrm{vec}[A_1], \mathrm{vec}[A_2], \cdots, \mathrm{vec}[A_m] \in \mathbb{R}^{n^2}$
4:     $\eta \leftarrow \frac{1}{n+2}$,   $T \leftarrow \frac{40}{\epsilon_N}\sqrt{n}\log(\frac{n}{\delta})$
5:     Find initial feasible dual vector $y \in \mathbb{R}^m$ according to Lemma E.1
6:     $S \leftarrow \sum_{i \in [m]} y_i \cdot A_i - C$,   $\widetilde{S} \leftarrow S$          ▷ $S, \widetilde{S} \in \mathbb{R}^{n \times n}$
7:     $G \leftarrow (\mathsf{A} \cdot (\widetilde{S}^{-1} \otimes \widetilde{S}^{-1}) \cdot \mathsf{A}^\top)^{-1}$          ▷ $G \in \mathbb{R}^{m \times m}$
8:                         ▷ Maintain $G = \widetilde{H}^{-1}$ where $\widetilde{H} := \mathsf{A} \cdot (\widetilde{S}^{-1} \otimes \widetilde{S}^{-1}) \cdot \mathsf{A}^\top$
9:     for $t = 1 \to T$ do                    ▷ Iterations of approximate barrier method
10:        $\eta^{\mathrm{new}} \leftarrow \eta \cdot (1 + \frac{\epsilon_N}{20\sqrt{n}})$
11:        for $j = 1, \cdots, m$ do
12:            $g_{\eta^{\mathrm{new}}}(y)_j \leftarrow \eta^{\mathrm{new}} \cdot b_j - \mathrm{tr}[S^{-1} \cdot A_j]$          ▷ Gradient computation, $g_{\eta^{\mathrm{new}}}(y) \in \mathbb{R}^m$
13:        end for
14:        $\delta_y \leftarrow -G \cdot g_{\eta^{\mathrm{new}}}(y)$          ▷ Update on $y \in \mathbb{R}^m$
15:        $y^{\mathrm{new}} \leftarrow y + \delta_y$
16:        $S^{\mathrm{new}} \leftarrow \sum_{i \in [m]} (y^{\mathrm{new}})_i \cdot A_i - C$
17:        $V_1, V_2 \leftarrow$ LowRankSlackUpdate($S^{\mathrm{new}}, \widetilde{S}$)          ▷ $V_1, V_2 \in \mathbb{R}^{n \times r_t}$. Algorithm 43.
18:        $\widetilde{S}^{\mathrm{new}} \leftarrow \widetilde{S} + V_1 V_2^\top$          ▷ Approximate slack computation.
19:        $V_3 \leftarrow -\widetilde{S}^{-1} V_1 (I + V_2^\top \widetilde{S}^{-1} V_1)^{-1}$          ▷ $V_3 \in \mathbb{R}^{n \times r_t}$
20:        $V_4 \leftarrow \widetilde{S}^{-1} V_2$          ▷ $V_4 \in \mathbb{R}^{n \times r_t}$
21:        $Y_1 \leftarrow [(\widetilde{S}^{-1/2} \otimes V_3), (V_3 \otimes \widetilde{S}^{-1/2}), (V_3 \otimes V_3^\top)]$          ▷ $Y_1 \in \mathbb{R}^{n^2 \times (2nr_t + r_t^2)}$
22:        $Y_2 \leftarrow [(\widetilde{S}^{-1/2} \otimes V_4), (V_4 \otimes \widetilde{S}^{-1/2}), (V_4 \otimes V_4^\top)]$          ▷ $Y_2 \in \mathbb{R}^{n^2 \times (2nr_t + r_t^2)}$
23:        $G^{\mathrm{new}} \leftarrow G - G \cdot \mathsf{A} Y_1 \cdot (I + Y_2^\top \mathsf{A}^\top \mathsf{A} Y_1)^{-1} \cdot Y_2^\top \mathsf{A}^\top \cdot G$          ▷ $G^{\mathrm{new}} \in \mathbb{R}^{m \times m}$
24:                         ▷ Hessian inverse computation using Woodbury identity.
25:        $y \leftarrow y^{\mathrm{new}}$
26:        $S \leftarrow S^{\mathrm{new}}$
27:        $\widetilde{S} \leftarrow \widetilde{S}^{\mathrm{new}}$
28:        $G \leftarrow G^{\mathrm{new}}$          ▷ Update variables
29:    end for
30: end procedure
```

Proof. Combining Lemma 9.21 and Lemma 9.22 we have that $\alpha_S^{-1} S \preceq \widetilde{S} \preceq \alpha_S S$ for $\alpha_S = 1 + 10^{-5}$. Therefore condition 1' in Lemma 9.40 is satisfied by Fact 9.13 and condition 2 & 3 holds trivially. Notice $\theta_{\phi_{\log}} = n$. Then directly applying Theorem 9.46

Algorithm 43 Low Rank Slack Update

```
1: procedure LowRankSlackUpdate($S^{\mathrm{new}}, \widetilde{S}$)
2:                                  ▷ $S^{\mathrm{new}}, \widetilde{S} \in \mathbb{S}_{\geq 0}^{n \times n}$ are positive definite matrices
3:     $\epsilon_S \leftarrow 10^{-5}$                      ▷ Spectral approximation constant
4:     $Z^{\mathrm{mid}} \leftarrow (S^{\mathrm{new}})^{-1/2} \cdot \widetilde{S} \cdot (S^{\mathrm{new}})^{-1/2} - I$
5:     Compute spectral decomposition $Z^{\mathrm{mid}} = U \cdot \mathrm{diag}(\lambda) \cdot U^\top$
6:             ▷ $\lambda = [\lambda_1, \cdots, \lambda_n]^\top \in \mathbb{R}^n$ are the eigenvalues of $Z^{\mathrm{mid}}$, and $U \in \mathbb{R}^{n \times n}$ is orthogonal
7:     Let $\pi : [n] \to [n]$ be a sorting permutation such that $|\lambda_{\pi(i)}| \geq |\lambda_{\pi(i+1)}|$
8:     if $|\lambda_{\pi(1)}| \leq \epsilon_S$ then
9:         $\widetilde{S}^{\mathrm{new}} \leftarrow \widetilde{S}$
10:    else
11:        $r \leftarrow 1$
12:        while $r \leq n/2$ and $(|\lambda_{\pi(2r)}| > \epsilon_S$ or $|\lambda_{\pi(2r)}| > (1 - 1/\log n)|\lambda_{\pi(r)}|)$ do
13:            $r \leftarrow r + 1$
14:        end while
15:        $(\lambda^{\mathrm{new}})_{\pi(i)} \leftarrow \begin{cases} 0, & \text{if } i = 1, 2, \cdots, 2r; \\ \lambda_{\pi(i)}, & \text{otherwise.} \end{cases}$
16:        $L \leftarrow \mathrm{supp}(\lambda^{\mathrm{new}} - \lambda)$                      ▷ $|L| = 2r$
17:        $V_1 \leftarrow ((S^{\mathrm{new}})^{1/2} \cdot U \cdot \mathrm{diag}(\lambda^{\mathrm{new}} - \lambda))_{*,L}$      ▷ $V_1 \in \mathbb{R}^{n \times 2r}$
18:        $V_2 \leftarrow ((S^{\mathrm{new}})^{1/2} \cdot U)_{*,L}$                      ▷ $V_2 \in \mathbb{R}^{n \times 2r}$
19:                   ▷ $V_1 \cdot V_2^\top = (S^{\mathrm{new}})^{1/2} \cdot U \cdot \mathrm{diag}(\lambda^{\mathrm{new}} - \lambda) \cdot U^\top \cdot (S^{\mathrm{new}})^{1/2}$
20:    end if
21:    return $\widetilde{S}^{\mathrm{new}}$
22: end procedure
```

completes the proof.

□

9.9.1 Approximate slack maintenance

The following lemma gives a closed-form formula for the updated $\widetilde{S}^{\mathrm{new}}$ in each iteration.

Lemma 9.21 (Closed-form formula of slack update)**.** *In each iteration of Algo-*

rithm 42, the update of the slack variable $\widetilde{S}^{\mathrm{new}}$ (on Line 18) satisfies

$$\widetilde{S}^{\mathrm{new}} = \widetilde{S} + (S^{\mathrm{new}})^{1/2} \cdot U \cdot \mathrm{diag}(\lambda^{\mathrm{new}} - \lambda) \cdot U^\top \cdot (S^{\mathrm{new}})^{1/2},$$

where $\widetilde{S}$ is the slack variable in previous iteration, and $U, \lambda, \lambda^{\mathrm{new}}$ are defined in Algorithm 43.

Moreover, it implies that $\widetilde{S}^{\mathrm{new}}$ is a symmetric matrix in each iteration.

Proof. From Line 17 and 18 of Algorithm 43 we have $V_1 \cdot V_2^\top = (S^{\mathrm{new}})^{1/2} \cdot U \cdot \mathrm{diag}(\lambda^{\mathrm{new}} - \lambda) \cdot U^\top \cdot (S^{\mathrm{new}})^{1/2}$. Therefore

$$\widetilde{S}^{\mathrm{new}} = \widetilde{S} + V_1 \cdot V_2^\top = \widetilde{S} + (S^{\mathrm{new}})^{1/2} \cdot U \cdot \mathrm{diag}(\lambda^{\mathrm{new}} - \lambda) \cdot U^\top \cdot (S^{\mathrm{new}})^{1/2}.$$

In the first iteration, we have $\widetilde{S} = S$ which is a symmetric matrix.

By the definition of V_1, V_2, we know that $V_1 \cdot V_2^\top$ is symmetric. Hence, S^{new} is also symmetric in each iteration. $\square$

The following lemma proves that we always have $\widetilde{S} \approx S$ throughout the algorithm.

Lemma 9.22 (Approximate Slack)**.** *In each iteration of Algorithm 42, the approximate slack variable $\widetilde{S}$ satisfies that $\alpha_S^{-1} S \preceq \widetilde{S} \preceq \alpha_S S$, where $\alpha_S = 1 + 10^{-5}$.*

Proof. Notice that

$$\begin{aligned}
\widetilde{S}^{\mathrm{new}} &= \widetilde{S} + (S^{\mathrm{new}})^{1/2} \cdot U \cdot \mathrm{diag}(\lambda^{\mathrm{new}} - \lambda) \cdot U^\top \cdot (S^{\mathrm{new}})^{1/2} \\
&= \big(S^{\mathrm{new}} + (S^{\mathrm{new}})^{1/2} Z^{\mathrm{mid}} (S^{\mathrm{new}})^{1/2}\big) + (S^{\mathrm{new}})^{1/2} \cdot U \cdot \mathrm{diag}(\lambda^{\mathrm{new}} - \lambda) \cdot U^\top \cdot (S^{\mathrm{new}})^{1/2} \\
&= S^{\mathrm{new}} + (S^{\mathrm{new}})^{1/2} \cdot U \cdot \mathrm{diag}(\lambda^{\mathrm{new}}) \cdot U^\top \cdot (S^{\mathrm{new}})^{1/2},
\end{aligned}$$

where the first step comes from Lemma 9.21, the second step comes from definition $Z^{\mathrm{mid}} = (S^{\mathrm{new}})^{-1/2} \cdot \widetilde{S} \cdot (S^{\mathrm{new}})^{-1/2} - I$ (Line 4 of Algorithm 43), and the final step comes from $Z^{\mathrm{mid}} = U \cdot \mathrm{diag}(\lambda_1, \cdots, \lambda_n) \cdot U^\top$ (Line 5 of Algorithm 43).

By Line 15 of Algorithm 43 we have $(\lambda^{\mathrm{new}})_i \leq \epsilon_S$ for all $i \in [n]$, so

$$\left\|(S^{\mathrm{new}})^{-1/2} \cdot \widetilde{S}^{\mathrm{new}} \cdot (S^{\mathrm{new}})^{-1/2} - I\right\|_2 = \left\|U \cdot \mathrm{diag}(\lambda^{\mathrm{new}}) \cdot U^\top\right\|_2 \leq \epsilon_S.$$

This implies that for $\alpha_S = 1 + \epsilon_S$, by Fact 9.5, in each iteration of Algorithm 42 the slack variable $\widetilde{S}$ satisfies $\alpha_S^{-1} S \preceq \widetilde{S} \preceq \alpha_S S$. $\square$

9.9.2 Approximate Hessian inverse maintenance

The following lemma shows that the maintained matrix G equals to the inverse of approximate Hessian.

Lemma 9.23 (Close-form formula for Hessian inverse)**.** *In each iteration of Algorithm 42, we have* $G = \widetilde{H}^{-1} \in \mathbb{R}^{m \times m}$*, where* $\widetilde{H} := \mathsf{A} \cdot (\widetilde{S}^{-1} \otimes \widetilde{S}^{-1}) \cdot \mathsf{A}^\top \in \mathbb{R}^{m \times m}$*.*

Proof. We prove this lemma by induction.

In the beginning of the algorithm, the initialization of G (Line 7 of Algorithm 42) satisfies the formula $G = \widetilde{H}^{-1}$.

Assume the induction hypothesis that $G = \widetilde{H}^{-1}$ in the beginning of each iteration, next we will prove that $G^{\mathrm{new}} = (\widetilde{H}^{\mathrm{new}})^{-1}$. Note that G^{new} is updated on Line 23 of Algorithm 42. And $\widetilde{H}^{\mathrm{new}} := \mathsf{A} \cdot ((\widetilde{S}^{\mathrm{new}})^{-1} \otimes (\widetilde{S}^{\mathrm{new}})^{-1}) \cdot \mathsf{A}^\top \in \mathbb{R}^{m \times m}$, where $\widetilde{S}^{\mathrm{new}}$ is updated on Line 18 of Algorithm 42.

We first compute $(\widetilde{S}^{\mathrm{new}})^{-1} \in \mathbb{R}^{n \times n}$:

$$\begin{aligned}
(\widetilde{S}^{\mathrm{new}})^{-1} &= (\widetilde{S} + V_1 V_2^\top)^{-1} \\
&= \widetilde{S}^{-1} - \widetilde{S}^{-1} V_1 \cdot (I + V_2^\top \widetilde{S}^{-1} V_1)^{-1} \cdot V_2^\top \widetilde{S}^{-1} \\
&= \widetilde{S}^{-1} + V_3 \cdot V_4^\top,
\end{aligned} \tag{9.13}$$

where the reason of the first step is $\widetilde{S}^{\mathrm{new}} = \widetilde{S} + V_1 V_2^\top$ (Line 18 of Algorithm 42), the second step follows from Woodbury identity (Fact 9.8), and the third step follows from $V_3 = -\widetilde{S}^{-1} V_1 (I + V_2^\top \widetilde{S}^{-1} V_1)^{-1} \in \mathbb{R}^{n \times r_t}$ and $V_4 = \widetilde{S}^{-1} V_2 \in \mathbb{R}^{n \times r_t}$ (Line 19 and 20 of Algorithm 42).

We then compute a close-form formula of $(\widetilde{S}^{\mathrm{new}})^{-1} \otimes (\widetilde{S}^{\mathrm{new}})^{-1} \in \mathbb{R}^{n^2 \times n^2}$:

$$\begin{aligned}
& (\widetilde{S}^{\mathrm{new}})^{-1} \otimes (\widetilde{S}^{\mathrm{new}})^{-1} \\
= & (\widetilde{S}^{-1} + V_3 V_4^\top) \otimes (\widetilde{S}^{-1} + V_3 V_4^\top) \\
= & \widetilde{S}^{-1} \otimes \widetilde{S}^{-1} + \widetilde{S}^{-1} \otimes (V_3 V_4^\top) + (V_3 V_4^\top) \otimes \widetilde{S}^{-1} + (V_3 V_4^\top) \otimes (V_3 V_4^\top) \\
= & \widetilde{S}^{-1} \otimes \widetilde{S}^{-1} + (\widetilde{S}^{-1/2} \otimes V_3) \cdot (\widetilde{S}^{-1/2} \otimes V_4^\top) + (V_3 \otimes \widetilde{S}^{-1/2}) \cdot (V_4^\top \otimes \widetilde{S}^{-1/2}) \\
& + (V_3 \otimes V_3) \cdot (V_4^\top \otimes V_4^\top) \\
= & \widetilde{S}^{-1} \otimes \widetilde{S}^{-1} + Y_1 Y_2^\top, \qquad (9.14)
\end{aligned}$$

where the first step follows from Eq. (9.13), the second step follows from linearity of Kronecker product, the third step follows from mixed product property of Kronecker product (Part 2 of Fact 9.10), the fourth step follows from $Y_1 = [(\widetilde{S}^{-1/2} \otimes V_3), (V_3 \otimes \widetilde{S}^{-1/2}), (V_3 \otimes V_3^\top)] \in \mathbb{R}^{n^2 \times (2nr_t + r_t^2)}$ and $Y_2 = [(\widetilde{S}^{-1/2} \otimes V_4), (V_4 \otimes \widetilde{S}^{-1/2}), (V_4 \otimes V_4^\top)] \in \mathbb{R}^{n^2 \times (2nr_t + r_t^2)}$ (Line 21 and 22 of Algorithm 42), and the transpose of Kronecker product (Part 1 of Fact 9.10).

Thus we can compute $(\widetilde{H}^{\mathrm{new}})^{-1} \in \mathbb{R}^{m \times m}$ as follows:

$$\begin{aligned}
(\widetilde{H}^{\mathrm{new}})^{-1} = & \Big(\mathsf{A} \cdot \big((\widetilde{S}^{\mathrm{new}})^{-1} \otimes (\widetilde{S}^{\mathrm{new}})^{-1}\big) \cdot \mathsf{A}^\top\Big)^{-1} \\
= & \Big(\mathsf{A} \cdot (\widetilde{S}^{-1} \otimes \widetilde{S}^{-1}) \cdot \mathsf{A}^\top + \mathsf{A} \cdot Y_1 Y_2^\top \cdot \mathsf{A}^\top\Big)^{-1} \\
= & G - G \cdot \mathsf{A} Y_1 \cdot (I + Y_2^\top \mathsf{A}^\top \cdot \mathsf{A} Y_1)^{-1} \cdot Y_2^\top \mathsf{A}^\top \cdot G \\
= & G^{\mathrm{new}}, \qquad (9.15)
\end{aligned}$$

where the first step follows from the definition of $\widetilde{H}^{\mathrm{new}}$, the second step follows from Eq. (9.14), the third step follows from Woodbury identity (Fact 9.8) and the induction hypothesis that $G = \widetilde{H}^{-1} = (\mathsf{A} \cdot (\widetilde{S}^{-1} \otimes \widetilde{S}^{-1}) \cdot \mathsf{A}^\top)^{-1} \in \mathbb{R}^{m \times m}$, the fourth step follows from the definition of G^{new} on Line 23 of Algorithm 42.

The proof is then completed. $\square$

9.10 Time Analysis

In this section we analyze the running time of Algorithm 42. We first present the main theorem.

Theorem 9.24 (Running time of Algorithm 42)**.** *Algorithm 42 runs in time*

$$O^*\left((m^2 \cdot \sqrt{n} + m^\omega + n^{4.5} + n^{2\omega}) \cdot \log(1/\epsilon)\right).$$

Specifically, when $m \geq n^2$ the total running time is

$$O^*\left((m^\omega + m^2 \cdot \sqrt{n}) \cdot \log(1/\epsilon)\right).$$

Proof. The running time of Algorithm 42 consists of two parts:

1. Initialization. $O(\mathcal{T}_{\mathrm{mat}}(m, n^2, n^2) + m^\omega) \leq O^*(n^{2\omega} + m^\omega)$ time from Lemma 9.25.

2. Cost of T iterations.

$$\begin{aligned}
&\sum_{t=1}^{T} \Big(\mathcal{T}_{\mathrm{mat}}(m, n^2, nr_t) + \mathcal{T}_{\mathrm{mat}}(m, m, nr_t) + (nr_t)^\omega + m^2 + mn^2\Big) \\
&\leq T \cdot O(m^2 + n^4 + n^{2\omega - 1/2} + m^{2 - \frac{\alpha(\omega-2)}{1-\alpha}} \cdot n^{\frac{2(\omega-2)}{1-\alpha} - 1/2}) \\
&= O^*\Big((m^2 \cdot \sqrt{n} + n^{4.5} + n^{2\omega} + m^{2 - \frac{\alpha(\omega-2)}{1-\alpha}} \cdot n^{\frac{2(\omega-2)}{1-\alpha}}) \cdot \log(1/\epsilon)\Big) \\
&\leq O^*\Big((m^2 \cdot \sqrt{n} + n^{4.5} + n^{2\omega} + m^\omega) \cdot \log(1/\epsilon)\Big)
\end{aligned}$$

where the first step follows from Lemma 9.26, the second step follows from Lemma 9.33 and $mn^2 \leq O(m^2 + n^4)$, the third step follows from $T = O(\sqrt{n}\log(1/\epsilon))$, the fourth step follows from the inequalities in two cases:

$$m^{2 - \frac{\alpha(\omega-2)}{1-\alpha}} \cdot n^{\frac{2(\omega-2)}{1-\alpha}} \leq \begin{cases} (n^2)^{2 - \frac{\alpha(\omega-2)}{1-\alpha}} \cdot n^{\frac{2(\omega-2)}{1-\alpha}} = n^{2\omega} & \text{when } m \leq n^2, \\ m^{2 - \frac{\alpha(\omega-2)}{1-\alpha}} \cdot (m^{1/2})^{\frac{2(\omega-2)}{1-\alpha}} = m^\omega & \text{when } m > n^2. \end{cases}$$

Adding the costs of these two parts completes the proof. □

This section is organized as follows:

- Section 9.10.1 provides the analysis of initialization cost (Lemma 9.25).
- Section 9.10.2 studies the cost per iteration of our algorithm (Lemma 9.26).
- Section 9.10.3 present the property of low rank update (Theorem 9.29).
- Section 9.10.4 use the tools of Section 9.10.3 to bound the amortized cost of our algorithm (Lemma 9.33).

Sec.	Statement	Time	Comment
9.10.1	Lem 9.25	$\mathcal{T}_{\mathrm{mat}}(m, n^2, n^2) + m^\omega$	Initialization
9.10.2	Lem 9.26	$\mathcal{T}_{\mathrm{mat}}(m, n^2, nr_t) + \mathcal{T}_{\mathrm{mat}}(m, m, nr_t) + (nr_t)^\omega + m^2 + mn^2$	Cost per iteration
9.10.4	Lem 9.33	$m^2 + n^4 + n^{2\omega-1/2} + m^{2-\frac{\alpha(\omega-2)}{1-\alpha}} \cdot n^{\frac{2(\omega-2)}{1-\alpha}}$	Amortized cost

Table 9.1: Summary of Section 9.10.

9.10.1 Initialization cost

The goal of this section is to prove Lemma 9.25.

Lemma 9.25 (Initialization cost)**.** *Initialization of Algorithm 42 (Line 3 to 7) takes time*

$$O(\mathcal{T}_{\mathrm{mat}}(m, n^2, n^2) + m^\omega).$$

Proof. We compute the cost of each line during initialization.

- **Line 3 of Algorithm 42.** Constructing $\mathsf{A} \in \mathbb{R}^{m \times n^2}$ by stacking vectors takes $O(mn^2)$ time.
- **Line 6 of Algorithm 42.** Computing $S = \sum_{i\in[m]} y_i \cdot A_i - C$ takes $O(mn^2)$ time.
- **Line 7 of Algorithm 42.** This step computes $G = (\mathsf{A} \cdot (\widetilde{S}^{-1} \otimes \widetilde{S}^{-1}) \cdot \mathsf{A}^\top)^{-1}$. We first compute $\widetilde{S}^{-1} \otimes \widetilde{S}^{-1} \in \mathbb{R}^{n^2 \times n^2}$, which takes $O(n^4)$ time. We then compute $\mathsf{A} \cdot (\widetilde{S}^{-1} \otimes \widetilde{S}^{-1}) \cdot \mathsf{A}^\top$, which takes $O(\mathcal{T}_{\mathrm{mat}}(m, n^2, n^2) + \mathcal{T}_{\mathrm{mat}}(m, n^2, m))$ time.

Finally computing the inverse takes $O(m^\omega)$ time. Since $n^4 \leq \mathcal{T}_{\mathrm{mat}}(m, n^2, n^2)$ and $\mathcal{T}_{\mathrm{mat}}(m, n^2, m) \leq \mathcal{T}_{\mathrm{mat}}(m, n^2, n^2) + m^\omega$, this step takes $O(\mathcal{T}_{\mathrm{mat}}(m, n^2, n^2) + m^\omega)$ time in total,

Combining the cost of these three steps, and since $mn^2 \leq \mathcal{T}_{\mathrm{mat}}(m, n^2, n^2)$, we have the total cost as presented in the lemma statement. □

9.10.2 Cost per iteration

The goal of this section is to prove Lemma 9.26.

Lemma 9.26 (Cost per iteration)**.** *For $t \in [T]$, let r_t be the rank of the update in the t-th iteration of Algorithm 42. The t-th iteration takes time*

$$O(\mathcal{T}_{\mathrm{mat}}(m, n^2, nr_t) + \mathcal{T}_{\mathrm{mat}}(m, m, nr_t) + (nr_t)^\omega + m^2 + mn^2).$$

Proof. We compute the cost of each line of Algorithm 42 in the t-th iteration.

Line 12 of Algorithm 42: gradient computation, $O(mn^2)$ time.

This step computes $g_{\eta^{\mathrm{new}}}(y)_j \leftarrow \eta^{\mathrm{new}} \cdot b_j - \mathrm{tr}[S^{-1} \cdot A_j]$.

Computing one $\mathrm{tr}[S^{-1} \cdot A_j]$ takes n^2 time, and computing all traces for $j \in [m]$ takes mn^2 time.

Line 14 of Algorithm 42: δ_y computation, $O(m^2)$ time.

This step computes $\delta_y \leftarrow -(\widetilde{H}^{\mathrm{new}})^{-1} \cdot g_{\eta^{\mathrm{new}}}(y)$.

Computing the matrix vector multiplication of $\widetilde{H}(y)^{-1} \in \mathbb{R}^{m \times m}$ with $g_{\eta^{\mathrm{new}}}(y) \in \mathbb{R}^m$ takes $O(m^2)$ time.

Line 16 of Algorithm 42: S^{new} computation, $O(mn^2)$ time.

This step computes $S^{\mathrm{new}} \leftarrow \sum_{i \in [m]} (y^{\mathrm{new}})_i A_i - C$.

Brute-forcely adding all $(y^{\mathrm{new}})_i A_i \in \mathbb{R}^{n \times n}$ takes mn^2 time.

Line 17-18 of Algorithm 42: $\widetilde{S}^{\mathrm{new}}$ computation, $O(n^\omega)$ time.

Line 17 makes a call to procedure LowRankSlackUpdate, and this takes $O(\omega)$ time by Lemma 9.27.

Line 18 computes $\widetilde{S}^{\mathrm{new}} \leftarrow \widetilde{S} + V_1 V_2^\top$, which takes $O(\mathcal{T}_{\mathrm{mat}}(n, r_t, n)) \leq O(n^\omega)$ time.

Line 19-23 of Algorithm 42: Hessian inverse computation, $O(\mathcal{T}_{\mathrm{mat}}(m, n^2, nr_t) + \mathcal{T}_{\mathrm{mat}}(m, m, nr_t) + (nr_t)^\omega)$ **time.**

Computing $V_3, V_4 \in \mathbb{R}^{n \times r_t}$ on Line 19 and 20 takes $O(n^\omega)$ time.

Computing $Y_1, Y_2 \in \mathbb{R}^{n^2 \times (2nr_t + r_t^2)}$ on Line 21 and 22 takes the same time as the output size: $O(n^3 \cdot r_t)$.

We compute $G^{\mathrm{new}} \leftarrow G - G \cdot \mathsf{A} Y_1 \cdot (I + Y_2^\top \mathsf{A}^\top \mathsf{A} Y_1)^{-1} \cdot Y_2^\top \mathsf{A}^\top \cdot G$ on Line 23 in the following order:

1. Compute $\mathsf{A}Y_1, \mathsf{A}Y_2 \in \mathbb{R}^{m \times (2nr_t + r_t^2)}$ in $O(\mathcal{T}_{\mathrm{mat}}(m, n^2, nr_t))$ time since $\mathsf{A} \in \mathbb{R}^{m \times n^2}$ and $Y_1, Y_2 \in \mathbb{R}^{n^2 \times (2nr_t + r_t^2)}$.

2. Compute $I + (Y_2^\top \mathsf{A}^\top) \cdot (\mathsf{A}Y_1) \in \mathbb{R}^{(2nr_t + r_t^2) \times (2nr_t + r_t^2)}$ in $O(\mathcal{T}_{\mathrm{mat}}(nr_t, m, nr_t))$ time. Then compute the inverse $(I + Y_2^\top \mathsf{A}^\top \cdot \mathsf{A}Y_1)^{-1} \in \mathbb{R}^{(2nr_t + r_t^2) \times (2nr_t + r_t^2)}$ in $O((nr_t)^\omega)$ time.

3. Compute $G \cdot \mathsf{A}Y_1 \in \mathbb{R}^{m \times (2nr_t + r_t^2)}$ in $O(\mathcal{T}_{\mathrm{mat}}(m, m, nr_t))$ time since $G \in \mathbb{R}^{m \times m}$ and $\mathsf{A}Y_1 \in \mathbb{R}^{m \times (2nr_t + r_t^2)}$.

4. Finally compute $G\mathsf{A}Y_1 \cdot (I + Y_2^\top \mathsf{A}^\top \cdot \mathsf{A}Y_1)^{-1} \cdot Y_2^\top \mathsf{A}^\top G \in \mathbb{R}^{m \times m}$ in $O(\mathcal{T}_{\mathrm{mat}}(m, nr_t, m))$ time.

Thus in total computing G^{new} takes $O(\mathcal{T}_{\mathrm{mat}}(m, n^2, nr_t) + \mathcal{T}_{\mathrm{mat}}(m, m, nr_t) + (nr_t)^\omega)$ time.

Combined. Combining the time of the four steps on Line 12, 23, 14, and 16 of Algorithm 42, it is easy to see that the t-th iteration takes time

$$\begin{aligned}
&\text{Time per iteration} \\
&= \text{Line } 12 + \text{Line } 14 + \text{Line } 16 + \text{Line } 17\text{-}18 + \text{Line } 19\text{-}23 \\
&= \underbrace{O(mn^2)}_{\text{Line } 12} + \underbrace{O(m^2)}_{\text{Line } 14} + \underbrace{O(mn^2)}_{\text{Line } 16} + \underbrace{O(n^\omega)}_{\text{Line } 17\text{-}18} + \underbrace{\mathcal{T}_{\mathrm{mat}}(m, n^2, nr_t) + \mathcal{T}_{\mathrm{mat}}(m, m, nr_t) + (nr_t)^\omega}_{\text{Line } 19\text{-}23} \\
&= O\left(\mathcal{T}_{\mathrm{mat}}(m, n^2, nr_t) + \mathcal{T}_{\mathrm{mat}}(m, m, nr_t) + (nr_t)^\omega + m^2 + mn^2\right).
\end{aligned}$$

Thus, we complete the proof.

□

Lemma 9.27 (Cost of LowRankSlackUpdate (Algorithm 43))**.** *A call to procedure* LowRankSlackUpdate *(Algorithm 43) takes $O(n^\omega)$ time.*

Proof. In procedure LowRankSlackUpdate, the most time-consuming step is to compute the spectral decomposition of $Z^{\mathrm{mid}} \in \mathbb{R}^{n\times n}$, and this takes $O(n^\omega)$ time. □

9.10.3 Property of low rank update

The goal of this section is to prove Theorem 9.29.

We first make the following definitions.

Definition 9.4 (Potential function)**.** Let $g \in \mathbb{R}^n_+$ be a non-increasing vector. For any matrix $Z \in \mathbb{R}^{n\times n}$, let $|\lambda(Z)|_{[i]}$ to denote the i-th largest absolute eigenvalue of Z. We define a potential function $\Phi_g : \mathbb{R}^{n\times n} \to \mathbb{R}_+$,

$$\Phi_g(Z) := \sum_{i=1}^n g_i \cdot |\lambda(Z)|_{[i]}.$$

In the t-th iteration of Algorithm 42, let $S, \widetilde{S} \in \mathbb{R}^{n\times n}$ be the slack matrix and the approximate slack matrix in the beginning of the iteration, and let $S^{\mathrm{new}}, \widetilde{S}^{\mathrm{new}}$ be the updated matrices. We define the following matrices to capture their differences.

Definition 9.5 (Difference matrices)**.** Define the following matrices $Z, Z^{\mathrm{mid}}, Z^{\mathrm{new}} \in \mathbb{R}^{n\times n}$:

$$\begin{aligned}
Z &:= S^{-1/2} \cdot \widetilde{S} \cdot S^{-1/2} - I, \\
Z^{\mathrm{mid}} &:= (S^{\mathrm{new}})^{-1/2} \cdot \widetilde{S} \cdot (S^{\mathrm{new}})^{-1/2} - I, \\
Z^{\mathrm{new}} &:= (S^{\mathrm{new}})^{-1/2} \cdot \widetilde{S}^{\mathrm{new}} \cdot (S^{\mathrm{new}})^{-1/2} - I.
\end{aligned}$$

Assumption 9.28 (Closeness of S^{new} and $\widetilde{S}$ from S)**.** We make the following two assumptions about $S, \widetilde{S}, S^{\mathrm{new}} \in \mathbb{R}^{n\times n}$:

1. $\|S^{-1/2} \cdot S^{\mathrm{new}} \cdot S^{-1/2} - I\|_F \leq 0.02$,
2. $\|S^{-1/2} \cdot \widetilde{S} \cdot S^{-1/2} - I\|_2 \leq 0.01$.

Next we present the main theorem of this section.

Theorem 9.29 (General amortized guarantee)**.** *Let T denote the total number of iterations in Algorithm 42. Let r_t denote the rank of the update matrices $V_1, V_2 \in \mathbb{R}^{n\times r_t}$ generated by Algorithm 43 in the t-th iteration. The ranks r_t's satisfy the following condition: for any vector $g \in \mathbb{R}^n_+$ which is non-increasing, we have*

$$\sum_{t=1}^{T} r_t \cdot g_{r_t} \leq O(T \cdot \|g\|_2 \cdot \log n).$$

Proof. Part 1 of Assumption 9.28 is proved in Theorem 9.20, and Part 2 of Assumption 9.28 is proved in Lemma 9.22. Thus we can use Lemma 9.31.

Combining Lemma 9.31 and Lemma 9.32, we have

$$\begin{aligned}
\Phi_g(Z^{\mathrm{new}}) - \Phi_g(Z) &= (\Phi_g(Z^{\mathrm{mid}}) - \Phi_g(Z)) - (\Phi_g(Z^{\mathrm{mid}}) - \Phi_g(Z^{\mathrm{new}})) \\
&\leq \|g\|_2 - \frac{\epsilon_S}{10\log n} \cdot r_t \cdot g_{r_t}.
\end{aligned}$$

With an abuse of notation, we denote the matrix Z in the t-th iteration as $Z^{(t)}$. Since in the beginning $\Phi_g(Z^{(0)}) = 0$ and $\Phi_g(Z^{(T)}) \geq 0$, we have

$$\begin{aligned} 0 &\leq \Phi_g(Z^{(T)}) - \Phi_g(Z^{(0)}) \\ &\leq \sum_{t=1}^{T} (\Phi_g(Z^{(t)}) - \Phi_g(Z^{(t-1)})) \\ &\leq T \cdot \|g\|_2 - \frac{\epsilon_S}{10 \log n} \cdot \sum_{t=1}^{T} r_t \cdot g_{r_t}. \end{aligned}$$

This completes the proof. $\square$

9.10.3.1 S move

Lemma 9.30 (Eigenvalue change)**.** *Let matrices $Z, Z^{\mathrm{mid}} \in \mathbb{R}^{n \times n}$ be defined as in Definition 9.5. Under Assumption 9.28, we have*

$$\sum_{i=1}^{n} (\lambda(Z)_{[i]} - \lambda(Z^{\mathrm{mid}})_{[i]})^2 \leq 10^{-3},$$

where $\lambda(Z)_{[i]}$ denotes the i-th largest eigenvalue of Z.

Proof. We notice

$$\|S^{1/2}(S^{\mathrm{new}})^{-1}S^{1/2} - I\|_F^2 = \|\nu^{-1} - \mathbf{1}_n\|_2^2$$

where $\{\nu_i\}_{i \in [n]}$ are the eigenvalues of $S^{-1/2}S^{\mathrm{new}}S^{-1/2}$. By Assumption 9.28, we have

$$\max_{i \in [n]} |v_i - 1| \leq 0.02,$$

which implies that $\min_{i \in [n]} v_i \geq 0.98$.

Assumption 9.28 also gives

$$\|\nu - \mathbf{1}_n\|_2^2 \leq 0.0004.$$

Then, we have

$$\|\nu^{-1} - \mathbf{1}_n\|_2^2 \leq 5 \times 10^{-4}. \tag{9.16}$$

Define $F := (S^{\mathrm{new}})^{-1/2}S^{1/2}$ and $F = UDV^\top$ be its SVD decomposition. Let $Z' := V^\top ZV$. Notice that

$$Z^{\mathrm{mid}} = FZF^\top + FF^\top - I.$$

Combining with Eq. (9.16),

$$\begin{aligned}\sum_{i=1}^n (\lambda(Z^{\mathrm{new}})_{[i]} - \lambda(FZF^\top)_{[i]})^2 &\leq \|Z^{\mathrm{mid}} - FZF^\top\|_F^2 \\ &= \|FF^\top - I\|_F^2 \\ &= \|\nu^{-1} - \mathbf{1}_n\|_2^2 \\ &\leq 5 \times 10^{-4}.\end{aligned}$$

Since $\|D^2 - I\|_F^2 = \|FF^\top - I\|_F^2 \leq 5 \cdot 10^{-4}$ and $\|Z'\|_2 = \|Z\|_2 \leq 0.01$, Fact 9.7 gives

$$\begin{aligned}\sum_{i=1}^n (\lambda(Z)_{[i]} - \lambda(FZF^\top)_{[i]})^2 = &\sum_{i=1}^n (\lambda(Z)_{[i]} - \lambda(DZ'D)_{[i]})^2 \\ \leq &\|DZ'D - Z'\|_F^2 \\ \leq &10^{-5}.\end{aligned}$$

Combining the above inequalities, we have

$$\sum_{i=1}^n (\lambda(Z)_{[i]} - \lambda(Z^{\mathrm{mid}})_{[i]})^2 \leq 10^{-3}.$$

□

The following lemma upper bounds the increase in potential when S changes to S^{new}.

Lemma 9.31 (S move)**.** *Consider the t-th iteration. Let matrices $Z, Z^{\mathrm{mid}} \in \mathbb{R}^{n\times n}$ be defined as in Definition 9.5. Let $g \in \mathbb{R}^n_+$ be a non-increasing vector, and let $\Phi_g : \mathbb{R}^{n\times n} \to \mathbb{R}_+$ be defined as in Definition 9.4.*

Under Assumption 9.28, we have

$$\Phi_g(Z^{\mathrm{mid}}) - \Phi_g(Z) \leq \|g\|_2.$$

Proof. Let $\pi : [n] \to [n]$ be a sorting permutation such that $|\lambda(Z^{\mathrm{mid}})_{\pi(1)}| \geq |\lambda(Z^{\mathrm{mid}})_{\pi(2)}| \geq \cdots \geq |\lambda(Z^{\mathrm{mid}})_{\pi(n)}|$, i.e., $|\lambda(Z^{\mathrm{mid}})_{\pi(i)}| = |\lambda(Z^{\mathrm{mid}})|_{[i]}$. Then we have

$$\begin{aligned}
\Phi_g(Z^{\mathrm{mid}}) =& \sum_{i=1}^{n} g_i \cdot |\lambda(Z^{\mathrm{mid}})_{\pi(i)}| \\
\leq& \sum_{i=1}^{n} g_i \cdot |\lambda(Z)_{\pi(i)}| + \sum_{i=1}^{n} g_i \cdot |\lambda(Z^{\mathrm{mid}})_{\pi(i)} - \lambda(Z)_{\pi(i)}| \\
\leq& \sum_{i\in[n]} g_i \cdot |\lambda(Z)|_{[i]} + \sum_{i\in[n]} g_i \cdot |\lambda(Z^{\mathrm{mid}})_{\pi(i)} - \lambda(Z)_{\pi(i)}| \\
\leq& \sum_{i\in[n]} g_i \cdot |\lambda(Z)|_{[i]} + \Big(\sum_{i\in[n]} g_i^2\Big)^{1/2} \cdot \Big(\sum_{i\in[n]} |\lambda(Z^{\mathrm{mid}})_{\pi(i)} - \lambda(Z)_{\pi(i)}|^2\Big)^{1/2} \\
\leq& \Phi_g(Z) + \|g\|_2
\end{aligned}$$

where the first step follows from the definition of Φ_g (Definition 9.4) and π, the second step follows from triangle inequality, the third step follows from $g \in \mathbb{R}_+^n$ is non-increasing and $|\lambda(Z)|_{[i]}$ is the i-th largest absolute eigenvalue of Z, the fourth step follows from Cauchy-Schwarz inequality, the last step follows from definition of Φ_g and Lemma 9.30.

Thus we have $\Phi_g(Z^{\mathrm{mid}}) - \Phi_g(Z) \leq \|g\|_2$. $\square$

9.10.3.2 $\widetilde{S}$ move

The following lemma lower bounds the decrease in potential when $\widetilde{S}$ is updated to $\widetilde{S}^{\mathrm{new}}$.

Lemma 9.32 ($\widetilde{S}$ move)**.** *Consider the t-th iteration. Let matrices $Z^{\mathrm{mid}}, Z^{\mathrm{new}} \in \mathbb{R}^{n\times n}$ be defined as in Definition 9.5. Let $g \in \mathbb{R}_+^n$ be a non-increasing vector, and let $\Phi_g : \mathbb{R}^{n\times n} \to \mathbb{R}_+$ be defined as in Definition 9.4.*

Let r_t denote the rank of the update matrices $V_1, V_2 \in \mathbb{R}^{n\times r_t}$ generated by Algorithm 43 in the t-th iteration. We have

$$\Phi_g(Z^{\mathrm{mid}}) - \Phi_g(Z^{\mathrm{new}}) \geq \frac{\epsilon_S}{10\log n} \cdot r_t \cdot g_{r_t}.$$

Proof. Note that $r_t = 2r$, where r is the variable of Algorithm 43. We define $\lambda \in \mathbb{R}^n$ and $U \in \mathbb{R}^{n \times n}$ (Line 5), $\pi : [n] \to [n]$ (Line 7) and $\lambda^{\mathrm{new}} \in \mathbb{R}^n$ (Line 15) to be the same as Algorithm 43. We extend the definition and let $\lambda_{\pi(i)} = 0$ for $i > n$. We have

$$\begin{aligned} Z^{\mathrm{new}} &= (S^{\mathrm{new}})^{-1/2} \cdot \widetilde{S}^{\mathrm{new}} \cdot (S^{\mathrm{new}})^{-1/2} - I \\ &= (S^{\mathrm{new}})^{-1/2} \cdot \Big(\widetilde{S} + (S^{\mathrm{new}})^{1/2} \cdot U \cdot \mathrm{diag}(\lambda^{\mathrm{new}} - \lambda) \cdot U^\top \cdot (S^{\mathrm{new}})^{1/2}\Big) \cdot (S^{\mathrm{new}})^{-1/2} - I \\ &= Z^{\mathrm{mid}} + U \cdot \mathrm{diag}(\lambda^{\mathrm{new}} - \lambda) \cdot U^\top \\ &= U \cdot \mathrm{diag}(\lambda^{\mathrm{new}}) \cdot U^\top \end{aligned}$$

where the first step follows from definition of Z^{new} (Definition 9.5), the second step follows from closed-form formula of $\widetilde{S}^{\mathrm{new}}$ (Lemma 9.21), the third step follows from definition of Z^{mid} (Definition 9.5), the fourth step follows from $Z^{\mathrm{mid}} = U \cdot \mathrm{diag}(\lambda) \cdot U^\top$ (Line 5 of Algorithm 43).

Thus from the definition of Φ_g (Definition 9.4), we have

$$\Phi_g(Z^{\mathrm{mid}}) = \sum_{i=1}^n g_i \cdot |\lambda_{\pi(i)}|, \qquad \Phi_g(Z^{\mathrm{new}}) = \sum_{i=1}^n g_i \cdot |\lambda_{\pi(2r+i)}|. \tag{9.17}$$

We consider two different cases of the outcome of the while-loop on Line 12 of Algorithm 43.

Case 1. No $i \leq n/2$ satisfies both $|\lambda_{\pi(2i)}| \leq \epsilon_S$ and $|\lambda_{\pi(2i)}| \leq (1 - 1/\log n)|\lambda_{\pi(i)}|$.

In this case, the while-loop exits with $r = n/2$, and hence $r_t = 2r = n$. Thus using Eq. (9.17) we have $\Phi_g(Z^{\mathrm{new}}) = 0$. We consider two sub-cases.

- **Case 1(a). For all $i \in [n]$, $|\lambda_{\pi(i)}| > \epsilon_S$.**

In this case we have

$$\begin{aligned}\Phi_g(Z^{\mathrm{mid}}) - \Phi_g(Z^{\mathrm{new}}) &= \sum_{i=1}^{n} g_i \cdot |\lambda_{\pi(i)}| - 0 \\ &\geq n \cdot g_n \cdot \epsilon_S \\ &= \epsilon_S \cdot r_t \cdot g_{r_t}.\end{aligned}$$

- **Case 1(b). There exists a minimum $i^* \leq n/2$ such that $|\lambda_{\pi(2i)}| \leq \epsilon_S$ for all $i \geq i^*$.**

  The condition of Case 1 and Case 1(b) means that for all $i \geq i^*$, we must have $|\lambda_{\pi(2i)}| > (1 - 1/\log n)|\lambda_{\pi(i)}|$. And since i^* is the minimum index such that $|\lambda_{\pi(2i^*)}| \leq \epsilon_S$, we have $|\lambda_{\pi(i^*)}| > \epsilon_S$. Thus we have

  $$|\lambda_{\pi(n)}| > (1 - 1/\log n)|\lambda_{\pi(n/2)}| > \cdots > (1 - 1/\log n)^{\log(n/i^*)}|\lambda_{\pi(i^*)}| \geq \frac{1}{e} \cdot |\lambda_{\pi(i^*)}| \geq \frac{\epsilon_S}{e}.$$

  So we have

  $$\begin{aligned}\Phi_g(Z^{\mathrm{mid}}) - \Phi_g(Z^{\mathrm{new}}) &= \sum_{i=1}^{n} g_i \cdot |\lambda_{\pi(i)}| - 0 \\ &\geq n \cdot g_n \cdot |\lambda_{\pi(n)}| \\ &= \frac{\epsilon_S}{e} \cdot r_t \cdot g_{r_t}.\end{aligned}$$

Case 2. There exists a minimum $r \leq n/2$ such that both $|\lambda_{\pi(2r)}| \leq \epsilon_S$ and $|\lambda_{\pi(2r)}| \leq (1 - 1/\log n)|\lambda_{\pi(r)}|$ are satisfied.

This r is the outcome of the while-loop in Algorithm 43. Next we prove $|\lambda_{\pi(r)}| \geq \frac{\epsilon_S}{e}$.

Let $i^* \leq n/2$ be the minimum index such that $|\lambda_{\pi(2i)}| \leq \epsilon_S$ for all $i \geq i^*$. Note that this implies $|\lambda_{\pi(i^*)}| > \epsilon_S$. We have $r \geq i^*$ since $|\lambda_{\pi(2r)}| \leq \epsilon_S$ and i^* is the minimum such index. Hence $|\lambda_{\pi(2i)}| > (1 - 1/\log n)|\lambda_{\pi(i)}|$ for all $i \in [i^*, r)$.

If $r/2 \leq i^*$, we have $|\lambda_{\pi(r)}| = |\lambda_{\pi(2(r/2))}| \geq \epsilon_S$. If $r/2 \geq i^*$, we have

$$|\lambda_{\pi(r)}| > (1 - 1/\log n)|\lambda_{\pi(r/2)}| > \cdots > (1 - 1/\log n)^{\log(r/i^*)}|\lambda_{\pi(i^*)}| \geq \frac{1}{e} \cdot |\lambda_{\pi(i^*)}| \geq \frac{\epsilon_S}{e}.$$

Thus we have $|\lambda_{\pi(r)}| \geq \frac{\epsilon_S}{e}$ in either cases.

We have

$$1.\ \forall i \leq r,\ |\lambda_{\pi(i)}| \geq |\lambda_{\pi(r)}| \geq \frac{\epsilon_S}{e}, \qquad 2.\ \forall i \geq 2r,\ |\lambda_{\pi(i)}| \leq |\lambda_{\pi(2r)}| \leq (1 - 1/\log n)|\lambda_{\pi(r)}|. \tag{9.18}$$

Thus we can bound the potential decrease as follows:

$$\begin{aligned} \Phi_g(Z^{\mathrm{mid}}) - \Phi_g(Z^{\mathrm{new}}) &\geq \sum_{i=1}^{r} g_i \cdot (|\lambda_{\pi(i)}| - |\lambda_{\pi(i+2r)}|) \\ &\geq \sum_{i=1}^{r} g_i \cdot (|\lambda_{\pi(r)}| - |\lambda_{\pi(2r)}|) \\ &\geq \sum_{i=1}^{r} g_i \cdot \frac{1}{\log n}|\lambda_{\pi(r)}| \\ &\geq r_t \cdot g_{r_t} \cdot \frac{\epsilon_S}{2e \log n}, \end{aligned}$$

where the first step follows from Eq. (9.17), the second step follows from $|\lambda_{\pi(i)}|$ is decreasing, the third follows from Part 2 of Eq. (9.18), the fourth step follows from Part 1 of Eq. (9.18), $r_t = 2r$, and that g is non-increasing. □

9.10.4 Amortized analysis

The goal of this section is to prove Lemma 9.33 using Lemma 9.34.

Lemma 9.33 (Amortization of Hessian computation)**.** *Let T denote the total number of iterations in Algorithm 42. For $t \in [T]$, the cost of Hessian computation in the t-th iteration can be amortized as follows:*

$$\begin{aligned} &\sum_{t=1}^{T} (\mathcal{T}_{\mathrm{mat}}(m, n^2, nr_t) + \mathcal{T}_{\mathrm{mat}}(m, m, nr_t) + (nr_t)^{\omega}) \\ &\leq T \cdot O^*(m^2 + n^4 + n^{2\omega - 1/2} + m^{2 - \frac{\alpha(\omega-2)}{1-\alpha}} \cdot n^{\frac{2(\omega-2)}{1-\alpha} - 1/2}). \end{aligned}$$

Proof. We have $(nr_t)^{\omega} \leq \mathcal{T}_{\mathrm{mat}}(n^2, n^2, nr_t)$, and $\mathcal{T}_{\mathrm{mat}}(m, n^2, nr_t) \leq \mathcal{T}_{\mathrm{mat}}(m, m, nr_t) +$

$\mathcal{T}_{\mathrm{mat}}(n^2, n^2, nr_t)$, thus we can bound the cost of hessian computation as

$$\mathcal{T}_{\mathrm{mat}}(m, n^2, nr_t) + \mathcal{T}_{\mathrm{mat}}(m, m, nr_t) + (nr_t)^\omega \leq \mathcal{T}_{\mathrm{mat}}(m, m, nr_t) + \mathcal{T}_{\mathrm{mat}}(n^2, n^2, nr_t). \tag{9.19}$$

From Theorem 9.29 we know that for any vector $g \in \mathbb{R}^n_+$ which is non-increasing, we have $\sum_{t=1}^T r_t \cdot g_{r_t} \leq O(T \cdot \|g\|_2 \cdot \log n)$, so we can use Lemma 9.34:

$$\sum_{t=1}^T \mathcal{T}_{\mathrm{mat}}(m, m, nr_t) \leq O^*(m^2 + m^{2-\frac{\alpha(\omega-2)}{1-\alpha}} \cdot n^{\frac{2(\omega-2)}{1-\alpha}-1/2}) \text{ when } m \geq n^2,$$
$$\sum_{t=1}^T \mathcal{T}_{\mathrm{mat}}(n^2, n^2, nr_t) \leq O^*(n^4 + n^{4-\frac{2\alpha(\omega-2)}{1-\alpha}} \cdot n^{\frac{2(\omega-2)}{1-\alpha}-1/2}) = O^*(n^4 + n^{2\omega-1/2}).$$

Combining these two inequalities and Eq. (9.19) completes the proof. □

Lemma 9.34 (Helpful lemma for amortization of Hessian computation)**.** *Let T denote the total number of iterations. Let $r_t \in [n]$ be the rank for the t-th iteration for $t \in [T]$. Assume r_t satisfies the following condition: for any vector $g \in \mathbb{R}^n_+$ which is non-increasing, we have*

$$\sum_{t=1}^T r_t \cdot g_{r_t} \leq O(T \cdot \|g\|_2).$$

If the cost in the t-th iteration is $O(\mathcal{T}_{\mathrm{mat}}(d, d, nr_t))$ where $d = \Omega(n^2)$ is an integer, then the amortized cost per iteration is

$$O^*\big(d^2 + d^{2-\frac{\alpha(\omega-2)}{1-\alpha}} \cdot n^{\frac{2(\omega-2)}{1-\alpha}-1/2}\big).$$

Proof. Let ω be the matrix multiplication exponent, let α be the dual matrix multiplication exponent, and let $\beta = \omega(2)$ (see Section 9.7.4 for more details). Since $d \geq n^2 \geq nr_t$, we have

$$\begin{aligned}\mathcal{T}_{\mathrm{mat}}(d, d, nr_t) &\leq d^2 + (nr_t)^{\frac{\omega-2}{1-\alpha}} \cdot d^{2-\frac{\alpha(\omega-2)}{1-\alpha}} \\ &= d^2 + d^{2-\frac{\alpha(\omega-2)}{1-\alpha}} \cdot n^{\frac{\omega-2}{1-\alpha}} \cdot r_t^{\frac{\omega-2}{1-\alpha}},\end{aligned} \tag{9.20}$$

where we use Fact 9.16 in the first step.

Define $g \in \mathbb{R}_+^n$ such that $\forall r \in [n]$, $g_r = r^{\frac{\omega-2}{1-\alpha}-1}$. We observe that g is a non-increasing vector because $\frac{\omega-2}{1-\alpha} - 1 \leq 0$ (Fact 9.17). Then using the condition in the lemma statement, we have

$$\begin{aligned}
\sum_{t=1}^{T} r_t^{\frac{\omega-2}{1-\alpha}} = & \sum_{t=1}^{T} r_t \cdot g_{r_t} \\
& \leq T \cdot \|g\|_2 \\
& \leq T \cdot \Big(\int_{x=1}^{n} x^{\frac{2(\omega-2)}{1-\alpha}-2} \mathrm{d}x \Big)^{1/2} \\
& = T \cdot O(n^{\frac{(\omega-2)}{1-\alpha}-1/2}), \qquad (9.21)
\end{aligned}$$

where the first step follows from the definition that $g_r = r^{\frac{\omega-2}{1-\alpha}-1}$, $\forall r \in [n]$, the second step follows from the assumption $\sum_{t=1}^{T} r_t \cdot g_{r_t} \leq T \cdot \|g\|_2$ in the lemma statement, the third step follows from upper bounding the ℓ_2 norm $\|g\|_2^2 = \sum_{r=1}^{n} g_r^2 = \sum_{r=1}^{n} r^{\frac{2(\omega-2)}{1-\alpha}-2} \leq \int_{x=1}^{n} x^{\frac{2(\omega-2)}{1-\alpha}-2}$ and the last step follows from computing the integral $\int_{x=1}^{n} x^{\frac{2(\omega-2)}{1-\alpha}-2} = c \cdot x^{\frac{2(\omega-2)}{1-\alpha}-1}\Big|_1^n = O(n^{\frac{2(\omega-2)}{1-\alpha}-1})$ where $c := 1/(\frac{2(\omega-2)}{1-\alpha} - 1)$.

Thus we have

$$\begin{aligned}
\sum_{t=1}^{T} \mathcal{T}_{\mathrm{mat}}(d, d, nr_t) \leq & \sum_{t=1}^{T} \Big(d^2 + d^{2-\frac{\alpha(\omega-2)}{1-\alpha}} \cdot n^{\frac{\omega-2}{1-\alpha}} \cdot r_t^{\frac{\omega-2}{1-\alpha}} \Big) \\
& = T \cdot d^2 + d^{2-\frac{\alpha(\omega-2)}{1-\alpha}} \cdot n^{\frac{\omega-2}{1-\alpha}} \cdot \sum_{t=1}^{T} r_t^{\frac{\omega-2}{1-\alpha}} \\
& \leq T \cdot d^2 + d^{2-\frac{\alpha(\omega-2)}{1-\alpha}} \cdot n^{\frac{\omega-2}{1-\alpha}} \cdot T \cdot O(n^{\frac{(\omega-2)}{1-\alpha}-1/2}) \\
& = T \cdot O(d^2 + d^{2-\frac{\alpha(\omega-2)}{1-\alpha}} \cdot n^{\frac{2(\omega-2)}{1-\alpha}-1/2}).
\end{aligned}$$

where the first step follows from Eq. (9.20), the second step follows from moving summation inside, the third step follows from Eq. (9.21), and the last step follows from adding the terms together.

Thus, we complete the proof. $\square$

9.11 The Robust Interior Point Method Framework For SDP

One of the contributions in this chapter is a more robust interior point method framework that allows errors in the Hessian matrices, the gradient vectors, and the Newton steps. We will first introduce the necessary backgrounds and definitions in Section 9.11.1. Then we will perform the one step error analysis based on Newton decrements in Section 9.11.2. We also list the corresponding error analysis in previous framework for comparison. In Section 9.11.3-9.11.5, we prove several supporting Lemmata that are used in the proof of Section 9.11.2. In Section 9.11.6, we include several classical results that bound the duality gap by the Newton decrements and provide the proofs. Finally, we state the main result in Section 9.11.7.

9.11.1 Definitions

We start with some definitions.

Definition 9.6. Let $C \in \mathbb{R}^{n\times n}$. Let $\mathsf{A} \in \mathbb{R}^{m\times n^2}$ denote the matrix where the i-th row matrix $A_i \in \mathbb{R}^{n\times n}$. Consider a barrier function $\phi : \mathbb{R}^m \mapsto \mathbb{R}$ defined on the dual space $\{y \in \mathbb{R}^m : \sum_{i=1}^m y_i A_i - C \succeq 0\}$.

We define function $S : \mathbb{R}^m \to \mathbb{R}^{n\times n}$ such that

$$S(y) = \sum_{i=1}^m y_i \cdot A_i - C.$$

We define function $\nabla^2\phi : \mathbb{R}^m \to \mathbb{R}^{m\times m}$ that maps the dual variable to the Hessian matrix of barrier function ϕ. Notice $\nabla^2\phi = \nabla^2 f_\eta$ for the regularized objective $f_\eta(y)$ in Eq. (9.3), since f_η adds ϕ with a linear function of y. In particular, for logarithmic barrier:

$$\nabla^2\phi(y) = \mathsf{A} \cdot (S(y)^{-1} \otimes S(y)^{-1}) \cdot \mathsf{A}^\top.$$

We abuse the notation of $\nabla^2\phi$ and also write $\nabla^2\phi : \mathbb{R}^{n\times n} \to \mathbb{R}^{m\times m}$ as a function of the slack matrix. In particular, for logarithmic barrier:

$$\nabla^2\phi(S) = \mathsf{A} \cdot (S^{-1} \otimes S^{-1}) \cdot \mathsf{A}^\top.$$

We define $\mathsf{g} : \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^m$ that maps the dual variable y and the learning rate η to the gradient of the regularized objective $f_\eta(y)$ in Eq. (9.3), i.e. $\mathsf{g}(y, \eta) = \eta \cdot b - \nabla\phi(y)$. In particular, for logarithmic barrier:

$$\mathsf{g}(y, \eta) = \eta \cdot b - \mathsf{A} \cdot \mathrm{vec}(S(y)^{-1}).$$

We abuse the notation of g and also write $\mathsf{g}(S, \eta)$ as a function of slack variable S and learning rate η. For example, in logarithmic barrier $\mathsf{g}(S, \eta)$ is given by

$$\mathsf{g}(S, \eta) = \eta \cdot b - \mathsf{A} \cdot \mathrm{vec}(S^{-1}).$$

We define function $n : \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^m$ as the Newton step taken at y with learning rate η:

$$n(y, \eta) = (\nabla^2\phi(y))^{-1} \cdot \mathsf{g}(y, \eta).$$

Definition 9.7 (Local norm)**.** We will frequently use local inner product defined w.r.t. function ϕ as follow: for $u, v \in \mathbb{R}^n$, $\langle u, v\rangle_x := \langle u, \nabla^2\phi(x)v\rangle$. It induces a local norm defined by $\|u\|_x := \langle u, u\rangle_x = \langle u, \nabla^2\phi(x)u\rangle$. It also induces an operator norm for matrices defined by $\|A\|_x := \sup_z \frac{\|Az\|_x}{\|z\|_x}$.

In the Hilbert space equipped with local norm $\langle\cdot,\cdot\rangle_x$, the gradient at z is denoted by $\nabla\phi_x(z)$ and satisfies $\nabla\phi_x(z) = (\nabla^2\phi(x))^{-1}\nabla\phi(z)$. The Hessian at z is denoted by $\nabla^2\phi_x(z)$ and satisfies $\nabla^2\phi_x(z) = (\nabla^2\phi(x))^{-1}\nabla^2\phi(z)$. Similarly, $\mathsf{g}_x(z, \eta) = (\nabla^2\phi(x))^{-1}\mathsf{g}(z, \eta)$.

Definition 9.8 (Self-concordant functional and barrier)**.** Given a function with domain D_f. Let $B_x(y, r)$ denote the open ball of radius r centered at y, where radius is measured with respect to the local norm $\|\cdot\|_x = \|\cdot\|_{\nabla^2 f(x)}$ defined w.r.t. f. The functional f is called self-concordant if for all $x \in D_f$ we have $B_x(x, 1) \subset D_f$, and if whenever $y \in B_x(x, 1)$ the following holds for all $v \neq 0$

$$1 - \|y - x\|_x \leq \frac{\|v\|_y}{\|v\|_x} \leq \frac{1}{1 - \|y - x\|_x}.$$

A self-concordant functional f is called a self-concordant barrier or barrier functional if

$$\theta_f := \sup_{x \in D_f} \|\nabla f_x(x)\|_x^2 < \infty.$$

θ_f is referred to as the complexity of self-concordant barrier f.

Remark 9.3. The complexity of logarithmic barrier $\phi_{\log}(y) = -\log\det(S(y))$ is n ([NN94]). The complexity of Hybrid barrier $225(n/m)^{1/2} \cdot (\phi_{\mathrm{vol}}(y) + \phi_{\log}(y) \cdot (m-1)/(n-1))$ is $\sqrt{mn}$ ([Ans00]). For any barrier function ϕ with complexity θ, since f_η adds ϕ with a linear function, f_η is also a self-concordant function with complexity θ.

We will use the following properties of self-concordance functions, from [Ren01].

Theorem 9.35 (Self-concordant function, [Ren01])**.** *Given a self-concordant function $f : D_f \to \mathbb{R}$. For any two points $a, b \in D_f$, if $\|a-b\|_{\nabla^2 f(a)} \le 1/4$, then we have*

$$\|(\nabla^2 f(a))^{-1}\nabla^2 f(b)\|_{\nabla^2 f(a)},\ \|(\nabla^2 f(b))^{-1}\nabla^2 f(a)\|_{\nabla^2 f(a)} \le \frac{1}{(1-\|b-a\|_{\nabla^2 f(a)})^2}. \tag{9.22}$$

Further,

$$\|I-(\nabla^2 f(a))^{-1}\nabla^2 f(b)\|_{\nabla^2 f(a)}, \|I-(\nabla^2 f(b))^{-1}\nabla^2 f(a)\|_{\nabla^2 f(a)} \le \frac{1}{(1-\|b-a\|_{\nabla^2 f(a)})^2} - 1. \tag{9.23}$$

Theorem 9.36 (Proposition 2.2.8 in [Ren01])**.** *Given a self-concordant function $f : D_f \to \mathbb{R}$. Define $n(a) := -(\nabla^2 f(a))^{-1}\nabla f(a)$. If $\|n(a)\|_{\nabla^2 f(a)} \le 1/4$ then f has a minimizer z and*

$$\|z - a^{\mathrm{new}}\|_{\nabla^2 f(a)} \le \frac{3\|n(a)\|_{\nabla^2 f(a)}}{(1-\|n(a)\|_{\nabla^2 f(x)})^3}$$

where $a^{\mathrm{new}} = a + n(a)$ and

$$\|z-a\|_{\nabla^2 f(a)} \le \|n(a)\|_{\nabla^2 f(a)} + \frac{3\|n(a)\|_{\nabla^2 f(a)}}{(1-\|n(a)\|_{\nabla^2 f(a)})^3}.$$

Theorem 9.37 (Theorem 2.3.4 in [Ren01]). *Assume f is a self-concordance barrier with domain D_f. Let θ_f denote its complexity. If $x, y \in D_f$ then*

$$\langle \nabla f(x), y - x \rangle \le \theta_f.$$

9.11.2 One step error analysis

Classical interior point literature controls the deviation from the central path in each step by bounding the Newton decrements, given by the potential function $\|\mathsf{g}(y, \eta)\|_{(\nabla^2 \phi(y))^{-1}}$. When the exact Newton step is taken, this is achieved by the following result.

Lemma 9.38 ([Ren01] exact framework). *Given $\epsilon_N \in (0, 10^{-2})$, $\eta > 0$, and $\eta^{\mathrm{new}} = \eta(1 + \frac{\epsilon_N}{20\sqrt{\theta}})$. Suppose that ϕ is a self-concordant barrier with complexity $\theta \ge 1$ and there is*

- **Condition 0.** *a feasible solution $y \in \mathbb{R}^m$ satisfies $\|\mathsf{g}(y, \eta)\|_{(\nabla^2 \phi(y))^{-1}} \le \epsilon_N$,*

Then $y^{\mathrm{new}} = y - (\nabla^2 \phi(y))^{-1} \mathsf{g}(y, \eta^{\mathrm{new}})$ satisfies

$$\|\mathsf{g}(y^{\mathrm{new}}, \eta^{\mathrm{new}})\|_{(\nabla^2 \phi(y^{\mathrm{new}}))^{-1}} \le \epsilon_N.$$

Further, [JKL+20] relaxes the exact H to a PSD approximation version. This framework allows errors in the Hessian matrices.

Lemma 9.39 ([JKL+20] semi-robust framework). *Given parameters $\epsilon_N \in (0, 10^{-2})$, $\eta > 0$, $\alpha_H \in [1, 1 + 10^{-4}]$, and $\eta^{\mathrm{new}} = \eta(1 + \frac{\epsilon_N}{20\sqrt{\theta}})$. Suppose that ϕ is a self-concordant barrier with complexity $\theta \ge 1$ and there is*

- **Condition 0.** *a feasible solution $y \in \mathbb{R}^m$ satisfies*

$$\|\mathsf{g}(y, \eta)\|_{(\nabla^2 \phi(y))^{-1}} \le \epsilon_N,$$

- **Condition 1.** *a symmetric matrix* $\widetilde{H} \in \mathbb{S}^{n\times n}_{>0}$ *has*

$$\alpha_H^{-1}\nabla^2\phi(y) \preceq \widetilde{H} \preceq \alpha_H \nabla^2\phi(y).$$

Then $y^{\mathrm{new}} = y - \widetilde{H}^{-1}\mathsf{g}(y, \eta^{\mathrm{new}})$ *satisfies*

$$\|\mathsf{g}(y^{\mathrm{new}}, \eta^{\mathrm{new}})\|_{(\nabla^2\phi(y^{\mathrm{new}}))^{-1}} \le \epsilon_N.$$

We propose a more general framework in the following Lemma and we believe it will be useful in the future optimization tasks for semi-definite programming. This framework allows errors in the Hessian matrices, the gradient vectors, and the Newton steps. Also notice that this framework allows even more errors in the Hessian matrices. In Lemma 9.39, $\alpha_H \cdot \widetilde{H}$ must satisfy $\alpha_H^{-2} \cdot \nabla^2\phi \preceq \alpha_H \cdot \widetilde{H} \preceq \nabla^2\phi$ while α_H^{-2} should be close to 1 as $\alpha_H^{-2} \in [0.99, 1]$. In Lemma 9.40, c_H can set to smaller constants that are close to 0 as $c_H \in [10^{-1}, 1]$. This fact is important to the efficient implementation of hybrid barrier in Section 9.12.

Lemma 9.40 (Our robust Newton step)**.** *Given any parameters* $\epsilon_g, \epsilon_\delta \in [0, 10^{-4}], c_H \in [10^{-1}, 1]$, $0 < \epsilon_N \le 10^{-2}$, $\eta > 0$, *and* $\eta^{\mathrm{new}} = \eta(1 + \frac{\epsilon_N}{20\sqrt{\theta}})$*. Suppose that* ϕ *is a self-concordant barrier with complexity* $\theta \ge 1$*. Consider the following conditions.*

- **Condition 0.** *a feasible dual solution* $y \in \mathbb{R}^m$ *satisfies* $\|\mathsf{g}(y, \eta)\|_{(\nabla^2\phi(y))^{-1}} \le \epsilon_N$,
- **Condition 1.** *a symmetric matrix* $\widetilde{H} \in \mathbb{S}^{n\times n}_{>0}$ *has*

$$c_H \cdot \nabla^2\phi(y) \preceq \widetilde{H} \preceq \nabla^2\phi(y).$$

- **Condition 2.** *a vector* $\widetilde{g} \in \mathbb{R}^m$ *satisfies*

$$\|\widetilde{g} - \mathsf{g}(y, \eta^{\mathrm{new}})\|_{(\nabla^2\phi(y))^{-1}} \le \epsilon_g \cdot \|\mathsf{g}(y, \eta^{\mathrm{new}})\|_{(\nabla^2\phi(y))^{-1}}.$$

- **Condition 3.** *a vector* $\widetilde{\delta}(y) \in \mathbb{R}^m$ *satisfies*

$$\|\widetilde{\delta}(y) - (-\widetilde{H}^{-1}\widetilde{g})\|_{\nabla^2\phi(y)} \le \epsilon_\delta \cdot \|\widetilde{H}^{-1}\widetilde{g}\|_{\nabla^2\phi(y)}.$$

Suppose Condition 0,1,2,3 hold. Then $y^{\mathrm{new}} = y + \widetilde{\delta}(y)$ *satisfies*

$$\|\mathsf{g}(y^{\mathrm{new}}, \eta^{\mathrm{new}})\|_{(\nabla^2\phi(y^{\mathrm{new}}))^{-1}} \leq \epsilon_N.$$

Furthermore, Condition 1 can also be replaced by the following

- **Condition 1'.** *a symmetric matrix* $\widetilde{H} \in \mathbb{S}^{n\times n}_{>0}$ *satisfies*

$$\alpha_H^{-1} \cdot \nabla^2\phi(y) \preceq \widetilde{H} \preceq \alpha_H \cdot \nabla^2\phi(y),$$

  where $\alpha_H \in [1, 1 + 10^{-4}]$.

Remark 9.4. Notice that the error parameters $\epsilon_N, \epsilon_g, \epsilon_\delta, \alpha_H, c_H$ do not depend on the dimension nor the number of iterations. The constants $10^{-4}, 10^{-2}, 1/10$ are chosen only for simplicity. In general, if one needs smaller ϵ_N, then one should inflict smaller errors in $\epsilon_g, \epsilon_\delta, \alpha_H, c_H$.

Proof. First consider the case when Condition 0, 1', 2, 3 hold. Condition 1 is a slightly different condition than Condition 1'. Then, we explain how to modify the proof from condition 1' to condition 1.

By triangle inequality of local norm we have

$$\begin{aligned}&\|\widetilde{\delta}(y) - (-n(y, \eta^{\mathrm{new}}))\|_y \\ \leq\ &\|\widetilde{\delta}(y) - (-\widetilde{H}^{-1}\widetilde{g})\|_y + \|(\widetilde{H}^{-1} - (\nabla^2\phi(y))^{-1})\widetilde{g}\|_y + \|(\nabla^2\phi(y))^{-1}\widetilde{g} - n(y, \eta^{\mathrm{new}})\|_y.\end{aligned} \tag{9.24}$$

For the second term, Condition 1 and 2 give

$$\begin{aligned}\|(\widetilde{H}^{-1} - (\nabla^2\phi(y))^{-1})\widetilde{g}\|_y^2 &= \widetilde{g}^\top(\widetilde{H}^{-1} - (\nabla^2\phi(S))^{-1})\nabla^2\phi(S)(\widetilde{H}^{-1} - (\nabla^2\phi(S))^{-1})\widetilde{g} \\ &= \widetilde{g}^\top(\widetilde{H}^{-1}\nabla^2\phi(S)\widetilde{H}^{-1} - 2\widetilde{H}^{-1} + (\nabla^2\phi(S))^{-1})\widetilde{g} \\ &\leq (\alpha_H^2 - 2\alpha_H^{-1} + 1) \cdot \widetilde{g}^\top(\nabla^2\phi(S))^{-1}\widetilde{g} \\ &= (\alpha_H^2 - 2\alpha_H^{-1} + 1) \cdot \|\widetilde{g}\|^2_{(\nabla^2\phi(y))^{-1}} \\ &\leq (\alpha_H^2 - 2\alpha_H^{-1} + 1) \cdot (1 + \epsilon_g)^2 \cdot \|\mathsf{g}(y, \eta^{\mathrm{new}})\|^2_{(\nabla^2\phi(y))^{-1}} \\ &\leq 0.001 \cdot \|n(y, \eta^{\mathrm{new}})\|_y^2.\end{aligned}$$

For the first term, Condition 1, 2, 3 give

$$\begin{aligned}\|\widetilde{\delta}(y) - (-\widetilde{H}^{-1}\widetilde{g})\|_y &\le \epsilon_\delta \cdot \|\widetilde{H}^{-1}\widetilde{g}\|_y \\ &\le \epsilon_\delta \cdot \alpha_H \cdot \|(\nabla^2\phi(S))^{-1}\widetilde{g}\|_y \\ &= \epsilon_\delta \cdot \alpha_H \cdot \|\widetilde{g}\|_{(\nabla^2\phi(y))^{-1}} \\ &\le \epsilon_\delta \cdot \alpha_H \cdot (1+\epsilon_g) \cdot \|\mathsf{g}(y,\eta^{\mathrm{new}})\|_{(\nabla^2\phi(y))^{-1}} \\ &\le 0.001 \cdot \|n(y,\eta^{\mathrm{new}})\|_y.\end{aligned}$$

For the third term, Condition 2 gives

$$\begin{aligned}\|(\nabla^2\phi(y))^{-1}\widetilde{g} - n(y,\eta^{\mathrm{new}})\|_y &= \|(\nabla^2\phi(y))^{-1}\widetilde{g} - (\nabla^2\phi(y))^{-1}\mathsf{g}(y,\eta^{\mathrm{new}})\|_y \\ &\le \epsilon_g \cdot \|(\nabla^2\phi(y))^{-1}\mathsf{g}(y,\eta^{\mathrm{new}})\|_y \\ &\le 0.001 \cdot \|n(y,\eta^{\mathrm{new}})\|_y.\end{aligned}$$

Combining the above bounds, we have

$$\|\widetilde{\delta}(y) - (-n(y,\eta^{\mathrm{new}}))\|_y \le 0.1 \cdot \|n(y,\eta^{\mathrm{new}})\|_y. \tag{9.25}$$

Combing with Lemma 9.41 and Eq. (9.25),

$$\|\widetilde{\delta}(y)\|_y \le 1.1 \cdot \|n(y,\eta^{\mathrm{new}})\|_y \le 2\epsilon_N, \text{ and } \|\widetilde{\delta}(y) - (-n(y,\eta^{\mathrm{new}}))\|_y \le 0.3 \cdot \epsilon_N \tag{9.26}$$

Using Lemma 9.42, we have

$$\begin{aligned}\|n(y^{\mathrm{new}},\eta^{\mathrm{new}})\|_{y^{\mathrm{new}}} &\le 2 \cdot (\|\widetilde{\delta}(y)\|_y^2 + \|\widetilde{\delta}(y) - n(y,\eta^{\mathrm{new}})\|_y) \\ &\le 2 \cdot (4 \cdot \epsilon_N^2 + 0.3 \cdot \epsilon_N) \\ &\le \epsilon_N\end{aligned}$$

where the second step follows from Eq. (9.26) and the last step follows from choice of ϵ_N.

Next, we consider the case when Condition 0, 1, 2, 3 hold. By triangle inequality of local norm we still have

$$\begin{aligned}&\|\widetilde{\delta}(y) - (-n(y, \eta^{\mathrm{new}}))\|_y \\ \leq\ &\|\widetilde{\delta}(y) - (-\widetilde{H}^{-1}\widetilde{g})\|_y + \|(\widetilde{H}^{-1} - (\nabla^2\phi(y))^{-1})\widetilde{g}\|_y + \|(\nabla^2\phi(y))^{-1}\widetilde{g} - n(y, \eta^{\mathrm{new}})\|_y.\end{aligned}$$

For the second term,

$$\begin{aligned}\|(\widetilde{H}^{-1} - (\nabla^2\phi(S))^{-1})\widetilde{g}\|_y &= \|(I - \nabla^2\phi(S)\widetilde{H}^{-1})\widetilde{g}\|_{(\nabla^2\phi(S))^{-1}} \\ &\leq \|I - \nabla^2\phi(S)\widetilde{H}^{-1}\|_{(\nabla^2\phi(S))^{-1}} \cdot \|\widetilde{g}\|_{(\nabla^2\phi(S))^{-1}} \\ &= \max_{v\in\mathbb{R}^m} \frac{\langle v, ((\nabla^2\phi(S))^{-1} - \widetilde{H}^{-1})v\rangle}{\langle v, (\nabla^2\phi(S))^{-1}v\rangle} \cdot \|\widetilde{g}\|_{(\nabla^2\phi(S))^{-1}} \\ &\leq (1 - c_H) \cdot \|\widetilde{g}\|_{(\nabla^2\phi(S))^{-1}} \\ &\leq (1 - c_H) \cdot (1 + \epsilon_g) \cdot \|n(y, \eta^{\mathrm{new}})\|_y\end{aligned}$$

where the second step comes from Hölder's inequality, the third step comes from the definition of matrix norm, the penultimate step comes from $0 \preceq (\nabla^2\phi(S))^{-1} - \widetilde{H}^{-1} \preceq (1 - c_H) \cdot (\nabla^2\phi(S))^{-1}$, and the final step comes from Condition 2.

For the first term,

$$\begin{aligned}\|\widetilde{\delta}(y) - (-\widetilde{H}^{-1}\widetilde{g})\|_y &\leq \epsilon_\delta \cdot \|\widetilde{H}^{-1}\widetilde{g}\|_y \\ &\leq \epsilon_\delta \cdot \|(\nabla^2\phi(S))^{-1}\widetilde{g}\|_y \\ &= \epsilon_\delta \cdot \|\widetilde{g}\|_{(\nabla^2\phi(y))^{-1}} \\ &\leq \epsilon_\delta \cdot (1 + \epsilon_g) \cdot \|\mathsf{g}(y, \eta^{\mathrm{new}})\|_{(\nabla^2\phi(y))^{-1}} \\ &\leq 0.001 \cdot \|n(y, \eta^{\mathrm{new}})\|_y\end{aligned}$$

where we use Condition 3 in the first step, we use $\widetilde{H}^{-1} \preceq (\nabla^2\phi(S))^{-1}$ in the second step, the penultimate step uses condition 2, and the final step uses the choice of $\epsilon_\delta, \epsilon_g$.

For the third term, Condition 2 gives

$$\begin{aligned}\|(\nabla^2\phi(y))^{-1}\widetilde{g} - n(y, \eta^{\mathrm{new}})\|_y &= \|(\nabla^2\phi(y))^{-1}\widetilde{g} - (\nabla^2\phi(y))^{-1}\mathsf{g}(y, \eta^{\mathrm{new}})\|_y \\ &\leq \epsilon_g \cdot \|(\nabla^2\phi(y))^{-1}\mathsf{g}(y, \eta^{\mathrm{new}})\|_y \\ &\leq 0.001 \cdot \|n(y, \eta^{\mathrm{new}})\|_y.\end{aligned}$$

Thus we can replace the Eq. (9.25) by

$$\begin{aligned}\|\widetilde{\delta}(y) - (-n(y,\eta^{\mathrm{new}}))\|_y &\le (0.01 + 1.004 \cdot (1 - c_H)) \cdot \|n(y,\eta^{\mathrm{new}})\|_y \\ &\le 0.92 \cdot \|n(y,\eta^{\mathrm{new}})\|_y. \end{aligned} \tag{9.27}$$

Combining with Lemma 9.41 and Eq. (9.27), we have

$$\|\widetilde{\delta}(y)\|_y \le 2\epsilon_N \le 0.02.$$

Using the above inequality and Lemma 9.42, we have

$$\begin{aligned}\|n(y^{\mathrm{new}},\eta^{\mathrm{new}})\|_{y^{\mathrm{new}}} &\le 0.98^{-2} \cdot (\|\widetilde{\delta}(y)\|_y^2 + \|\widetilde{\delta}(y) - n(y,\eta^{\mathrm{new}})\|_y) \\ &\le 0.98^{-2} \cdot (4\epsilon_N^2 + 0.92\epsilon_N) \\ &\le \epsilon_N.\end{aligned}$$

Thus, we complete the proof. □

9.11.3 η move

Lemma 9.41 (η move)**.** *Let $\epsilon_N \in (0, 10^{-2})$, $\|n(y,\eta)\|_y \le \epsilon_N$ and $\eta^{\mathrm{new}} = \eta(1 + \frac{\epsilon_N}{20\sqrt{\theta}})$. Suppose ϕ is a self-concordant barrier with complexity $\theta \ge 1$. We have*

$$\|n(y,\eta^{\mathrm{new}})\|_y \le (1 + \epsilon_N/20)\|n(y,\eta)\|_y + \epsilon_N/20 \le 1.06\epsilon_N.$$

Proof. Denote the Newton step by $n(y,\eta) := (\nabla^2\phi(y))^{-1}g(y,\eta)$, thus

$$n(y,\eta) = (\nabla^2\phi(y))^{-1}(\eta b - \mathsf{g}(y)) = \eta b_y - (\nabla^2\phi(y))^{-1}\mathsf{g}(y) = \eta b_y - \mathsf{g}_y(y)$$

(here $\mathsf{g}(y)$ is the gradient of barrier function).

We also have

$$n(y,\eta^{\mathrm{new}}) = \eta^{\mathrm{new}} b_y - \mathsf{g}_y(y)$$

Combining the above two equations, we have

$$(n(y,\eta^{\mathrm{new}}) + \mathsf{g}_y(y))\eta = (n(y,\eta) + \mathsf{g}_y(y)) \cdot \eta^{\mathrm{new}}$$

which implies that

$$\begin{aligned} n(y,\eta^{\mathrm{new}}) &= \frac{\eta^{\mathrm{new}}}{\eta}(n(y,\eta) + \mathsf{g}_y(y)) - \mathsf{g}_y(y) \\ &= \frac{\eta^{\mathrm{new}}}{\eta} n(y,\eta) + (\frac{\eta^{\mathrm{new}}}{\eta} - 1)\mathsf{g}_y(y) \end{aligned}$$

Since the complexity value of barrier functional ϕ is θ,

$$\begin{aligned} \|n(y,\eta^{\mathrm{new}})\|_y &= \left\| \frac{\eta^{\mathrm{new}}}{\eta} n(y,\eta) + (\frac{\eta^{\mathrm{new}}}{\eta} - 1)\mathsf{g}_y(y) \right\|_y \\ &\leq \frac{\eta^{\mathrm{new}}}{\eta} \|n(y,\eta)\|_y + \left|\frac{\eta^{\mathrm{new}}}{\eta} - 1\right|\sqrt{\theta} \\ &\leq \frac{\eta^{\mathrm{new}}}{\eta}\epsilon_N + \left|\frac{\eta^{\mathrm{new}}}{\eta} - 1\right|\sqrt{\theta} \\ &\leq (1 + \frac{\epsilon_N}{20\sqrt{\theta}}) \cdot \epsilon_N + \frac{\epsilon_N}{20} \\ &\leq 1.06 \cdot \epsilon_N, \end{aligned}$$

where the second step follows from triangle inequality and $\|\mathsf{g}_y(y)\|_y \leq \sqrt{\theta}$, we use $\|n(y,\eta)\|_y \leq \epsilon_N$ in the third step (see Lemma statement), we use definition of η^{new} in the forth step (see Lemma statement), and we use both $\theta \geq 1$ and $\epsilon_N \in (0, 10^{-2})$ in the last step. $\square$

9.11.4 y move

Lemma 9.42 (y move)**.** *Let $y^{\mathrm{new}} = y + \delta(y)$ and $\|\delta(y)\|_y \leq 1/4$. Suppose ϕ is a self-concordant barrier with complexity $\theta \geq 1$. We have*

$$\|n(y^{\mathrm{new}},\eta^{\mathrm{new}})\|_{y^{\mathrm{new}}} \leq \left(\frac{\|\delta(y)\|_y}{1 - \|\delta(y)\|_y}\right)^2 + \frac{\|\delta(y) - (-n(y,\eta^{\mathrm{new}}))\|_y}{1 - \|\delta(y)\|_y}$$

Further, we have

$$\|n(y^{\mathrm{new}},\eta^{\mathrm{new}})\|_{y^{\mathrm{new}}} \leq 2 \cdot (\|\delta(y)\|_y^2 + \|\delta(y) - (-n(y,\eta^{\mathrm{new}}))\|_y).$$

Proof. We compute the improvement by approximate Newton step. First notice that

$$\begin{aligned}
\|n(y^{\text{new}}, \eta^{\text{new}})\|_{y^{\text{new}}}^2 &= \|\nabla^2 \phi(y)_y (y^{\text{new}})^{-1} \mathsf{g}_y(y^{\text{new}}, \eta^{\text{new}})\|_{y^{\text{new}}}^2 \\
&= \left\langle \nabla^2 \phi(y^{\text{new}}) \nabla^2 \phi_y(y^{\text{new}})^{-1} \mathsf{g}_y(y^{\text{new}}, \eta^{\text{new}}), \nabla^2 \phi_y(y^{\text{new}})^{-1} \mathsf{g}_y(y^{\text{new}}, \eta^{\text{new}}) \right\rangle \\
&= \left\langle \nabla^2 \phi(y) \mathsf{g}_y(y^{\text{new}}, \eta^{\text{new}}), \nabla^2 \phi_y(y^{\text{new}})^{-1} \mathsf{g}_y(y^{\text{new}}, \eta^{\text{new}}) \right\rangle \\
&= \left\langle \mathsf{g}_y(y^{\text{new}}, \eta^{\text{new}}), \nabla^2 \phi_y(y^{\text{new}})^{-1} \mathsf{g}_y(y^{\text{new}}, \eta^{\text{new}}) \right\rangle_y \\
&\leq \|(\nabla^2 \phi_y(y^{\text{new}}))^{-1}\|_y^2 \cdot \|\mathsf{g}_y(y^{\text{new}}, \eta^{\text{new}})\|_y^2. \qquad (9.28)
\end{aligned}$$

where we use definition of operator norm in the final step.

By Eq. (9.22) (in Theorem 9.35),

$$\|(\nabla^2 \phi_y(y^{\text{new}}))^{-1}\|_y \leq \frac{1}{(1 - \|y^{\text{new}} - y\|_y)^2} = \frac{1}{(1 - \|\delta(y)\|_y)^2}. \qquad (9.29)$$

By Lemma 9.43, we have

$$\begin{aligned}
\mathsf{g}_y(y^{\text{new}}, \eta^{\text{new}}) &= g_y(y, \eta^{\text{new}}) + \int_0^1 \nabla^2 \phi_y(y + t(y^{\text{new}} - y))(y^{\text{new}} - y) \mathrm{d}t \\
&= n(y, \eta^{\text{new}}) + \int_0^1 \nabla^2 \phi_y(y + t(y^{\text{new}} - y))(y^{\text{new}} - y) \mathrm{d}t \\
&= (n(y, \eta^{\text{new}}) + (y^{\text{new}} - y)) + \int_0^1 (\nabla^2 \phi_y(y + t(y^{\text{new}} - y)) - I)(y^{\text{new}} - y) \mathrm{d}t
\end{aligned}$$

We can upper bound the first term under local norm as follow:

$$\|n(y, \eta^{\text{new}}) + (y^{\text{new}} - y)\|_y = \| - n(y, \eta^{\text{new}}) - \delta(y)\|_y. \qquad (9.30)$$

We can upper bound the second term under local norm as follow:

$$\begin{aligned}
&\left\| \int_0^1 \left(\nabla^2 \phi_y(y + t(y^{\text{new}} - y)) - I \right) (y^{\text{new}} - y) \mathrm{d}t \right\|_y \\
&\leq \|y^{\text{new}} - y\|_y \int_0^1 \|\nabla^2 \phi_y(y + t(y^{\text{new}} - y)) - I\|_y \mathrm{d}t \\
&\leq \|y^{\text{new}} - y\|_y \int_0^1 \left(\frac{1}{(1 - t\|y^{\text{new}} - y\|_y)^2} - 1 \right) \mathrm{d}t \\
&= \frac{\|y^{\text{new}} - y\|_y^2}{1 - \|y^{\text{new}} - y\|_y} \\
&= \frac{\|\delta(y)\|_y^2}{1 - \|\delta(y)\|_y} \qquad (9.31)
\end{aligned}$$

where the first step comes from triangle inequality of local norm, the second step comes from Eq. (9.23) (property of self-concordance function, Theorem 9.35), we use simple integration in the third step comes, finally we use $y^{\mathrm{new}} = y + \delta(y)$ in the final step.

Plugging Eq. (9.29), Eq. (9.30), and Eq. (9.31) into Eq. (9.28), we have

$$\begin{aligned}
\|n(y^{\mathrm{new}}, \eta^{\mathrm{new}})\|_{y^{\mathrm{new}}} &\leq \frac{1}{1 - \|\delta(y)\|} \cdot \|\mathsf{g}_y(y^{\mathrm{new}}, \eta^{\mathrm{new}})\|_y \\
&\leq \frac{1}{1 - \|\delta(y)\|} \cdot \left(\frac{\|\delta(y)\|_y^2}{1 - \|\delta(y)\|_y} + \|\delta(y) - (-n(y, \eta^{\mathrm{new}}))\|_y \right) \\
&\leq \left(\frac{\|\delta(y)\|_y}{1 - \|\delta(y)\|_y} \right)^2 + \frac{\|\delta(y) - (-n(y, \eta^{\mathrm{new}}))\|_y}{1 - \|\delta(y)\|_y}.
\end{aligned}$$

This completes the proof. □

9.11.5 Integral under local norm

Lemma 9.43. *Let $H = g'$ and $x, y \in D_f$. It holds that*

$$(\nabla^2 \phi(x))^{-1}(\mathsf{g}(y) - \mathsf{g}(x)) = \int_0^1 (\nabla^2 \phi(x))^{-1} \nabla^2 \phi(x + t(y - x))(y - x) \mathrm{d}t$$

Proof. It is sufficient to show

$$\mathsf{g}(y) - \mathsf{g}(x) = \int_0^1 \nabla^2 \phi(x + t(y - x))(y - x) \mathrm{d}t$$

By definition of the integral, it is sufficient to prove that for all w

$$\langle \mathsf{g}(y) - \mathsf{g}(x), w \rangle = \int_0^1 \langle \nabla^2 \phi(x + t(y - x))(y - x), w \rangle \mathrm{d}t.$$

Fix arbitrary w and consider the functional

$$\psi(t) := \langle g(x + t(y - x)), w \rangle.$$

The basic Calculus gives

$$\psi(1) - \psi(0) = \int_0^1 \psi'(t) \mathrm{d}t$$

which by definition of ψ is equivalent to

$$\langle \mathsf{g}(y) - \mathsf{g}(x), w \rangle = \int_0^1 \langle \nabla^2 \phi(x + t(y-x))(y-x), w \rangle \mathrm{d}t.$$

$\square$

9.11.6 Approximate dual optimality

We make use of the following lemma that bounds the duality gap by η and the Newton decrement.

Lemma 9.44 (Approximate optimality)**.** *Suppose $0 < \epsilon_N \leq 10^{-2}$. Let $\eta \geq 1$ denote a parameter. Let $y \in \mathbb{R}^m$ be dual feasible solution. Assume*

$$\mathsf{g}(y, \eta)^\top (\nabla^2 \phi(y))^{-1} \mathsf{g}(y, \eta) \leq \epsilon_N^2.$$

Assume that y^* is an optimal solution to the Eq. (9.2). Suppose ϕ is a self-concordant barrier with complexity $\theta \geq 1$. Then we have

$$\langle b, y \rangle \leq \langle b, y^* \rangle + \frac{\theta}{\eta} \cdot (1 + 2\epsilon_N).$$

Proof. Let $y(\eta)$ denote the optimal solution to the following optimization problem:

$$\min_{y \in \mathbb{R}^m} \eta \cdot \langle b, y \rangle + \phi(y)$$

where $\phi(y) = -\log\det S(y)$ is the log barrier. Then due to optimality condition, $\eta b + \mathsf{g}(y(\eta)) = 0$. Therefore

$$\langle b, y(\eta) \rangle - \langle b, y^* \rangle = \frac{1}{\eta} \langle \mathsf{g}(y(\eta)), y^* - y(\eta)) \rangle \leq \frac{\theta}{\eta} \tag{9.32}$$

where the last step comes from Theorem 9.37.

Furthermore, we have

$$\begin{aligned} \langle b, y \rangle - \langle b, y(\eta) \rangle &= \frac{1}{\eta} \langle \mathsf{g}(y(\eta)), y(\eta) - y \rangle \\ &\leq \frac{1}{\eta} \|\mathsf{g}(y(\eta))\|_{(\nabla^2 \phi(y(\eta)))^{-1}} \cdot \|y - y(\eta)\|_{\nabla^2 \phi(y(\eta))} \\ &\leq \frac{\theta}{\eta} \cdot \|y - y(\eta)\|_{\nabla^2 \phi(y(\eta))} \end{aligned} \tag{9.33}$$

where the last step used the complexity value of barrier is θ. For $\|y-y(\eta)\|_{\nabla^2\phi(y)}$, we have

$$\begin{aligned}\|y-y(\eta)\|^2_{\nabla^2\phi(y(\eta))} &\leq \|y-y(\eta)\|^2_{\nabla^2\phi(y)} \cdot \sup_v \frac{\|v\|^2_{\nabla^2\phi(y(\eta))}}{\|v\|^2_{\nabla^2\phi(y)}} \\ &= \|y-y(\eta)\|^2_{\nabla^2\phi(y)} \cdot \|(\nabla^2\phi(y))^{-1}\nabla^2\phi(y(\eta))\|_{\nabla^2\phi(y)} \\ &\leq \|y-y(\eta)\|^2_{\nabla^2\phi(y)} \cdot (1-\|y-y(\eta)\|^2_{\nabla^2\phi(y)})^{-2} \qquad (9.34)\end{aligned}$$

where the second step comes from the definition of operator norm and the third step comes from Theorem 9.35. By Theorem 9.36,

$$\|y-y(\eta)\|_{\nabla^2\phi(y)} \leq \|n(y,\eta)\|_{\nabla^2\phi(y)} + \frac{2\|n(y,\eta)\|^2_{\nabla^2\phi(y)}}{\left(1-\|n(y,\eta)\|_{\nabla^2\phi(y)}\right)^3} \leq 1.2\cdot\epsilon_N,$$

thus back to Eq. (9.34), we have $\|y-y(\eta)\|_{\nabla^2\phi(y(\eta))} \leq 2\epsilon_N$. Therefore in Eq. (9.33), we obtain

$$\langle b, y\rangle - \langle b, y(\eta)\rangle \leq \frac{\theta}{\eta}\cdot 2\epsilon_N. \qquad (9.35)$$

Combining Eq. (9.35) and Eq. (9.32), we complete the proof. □

Theorem 9.45 (Robust barrier method)**.** *Consider a semidefinite program in Eq.* (9.2)*. Suppose in each iteration, the $\widetilde{S}, \widetilde{H}, \widetilde{g}, \widetilde{\delta}$ are computed in Line 7, Line 8, Line 9, Line 10 of Algorithm 44 such that Condition 1 or 1' & Condition 2 & Condition 3 in Lemma 9.40 hold. Suppose ϕ is a self-concordant barrier with complexity $\theta \geq 1$. Assume y^* is an optimal solution to the dual formulation Eq.* (9.2)*. Then given a feasible initial solution that satisfies the invariant $\mathsf{g}(y,\eta)^\top(\nabla^2\phi(y))^{-1}\mathsf{g}(y,\eta) \leq \epsilon_N^2$, for any error parameter $0 < \epsilon \leq 0.01$ and Newton step size ϵ_N satisfying $\sqrt{\epsilon} < \epsilon_N \leq 0.01$, Algorithm 44 outputs, in $T = 40\epsilon_N^{-1}\sqrt{\theta}\log(\theta/\epsilon)$ iterations, a vector $y \in \mathbb{R}^m$ s.t.*

$$b^\top y \leq b^\top y^* + \epsilon^2. \qquad (9.36)$$

Further, for logarithmic barrier $\phi_{\log}$, in each iteration of Algorithm 44, the following invariant holds:

$$\|S^{-1/2}S^{\text{new}}S^{-1/2} - I\|_F \leq 1.03\cdot\epsilon_N. \qquad (9.37)$$

Proof. Since the invariant $\mathsf{g}(y,\eta)^\top(\nabla^2\phi(y))^{-1}\mathsf{g}(y,\eta) \le \epsilon_N^2$ holds at initialization, by Lemma 9.40 it then holds at any iteration. After $T = 40\epsilon_N^{-1}\sqrt{\theta}\log(\theta/\epsilon)$ iterations, the step size becomes $\eta = (1+\frac{\epsilon_N}{20\sqrt{\theta}})^T/(\theta+2) \ge 2\theta/\epsilon^2$. By Lemma 9.44, we have

$$\langle b, y\rangle \le \langle b, y^*\rangle + \frac{\theta}{\eta}\cdot(1+2\epsilon_N) \le \langle b, y^*\rangle + \epsilon^2.$$

This completes the proof of Eq. (9.36).

Finally we prove Eq. (9.37) for the log-barrier $\phi_{\log}$. It gives

$$\begin{aligned}
\text{LHS in Eq. (9.37)} &= \mathrm{tr}\left[\left(S^{-1/2}(S^{\mathrm{new}}-S)S^{-1/2}\right)^2\right] \\
&= \mathrm{tr}\left[S^{-1}(\sum_{i\in[m]}\widetilde{\delta}_{y,i}A_i)S^{-1}(\sum_{i\in[m]}\widetilde{\delta}_{y,i}A_i)\right] \\
&= \sum_{i\in[m]}\sum_{j\in[m]}\widetilde{\delta}_{y,i}\widetilde{\delta}_{y,j}\mathrm{tr}[S^{-1}A_iS^{-1}A_j] \\
&= \widetilde{\delta}_y^\top\nabla^2\phi(y)\widetilde{\delta}_y \\
&= \|\widetilde{\delta}_y\|_{\nabla^2\phi(y)}^2
\end{aligned}$$

where the second step comes from $S^{\mathrm{new}} - S = \sum_{i\in[m]}\widetilde{\delta}_{y,i}A_i$. It suffices to bound $\|\widetilde{\delta}_y\|_{\nabla^2\phi(y)}$. In fact we have

$$\begin{aligned}
\|\widetilde{\delta}_y\|_{\nabla^2\phi(y)} &\le \|n(y,\eta^{\mathrm{new}})\|_{\nabla^2\phi(y)} + \|\widetilde{\delta}_y - (-n(y,\eta^{\mathrm{new}}))\|_{\nabla^2\phi(y)} \\
&\le 1.01\|n(y,\eta^{\mathrm{new}})\|_{\nabla^2\phi(y)} \\
&\le 1.01\cdot\left((1+\epsilon_N/20)\|n(y,\eta)\|_{\nabla^2\phi(y)} + \epsilon/20\right) \\
&\le 1.03\cdot\epsilon_N
\end{aligned}$$

where we use triangle inequality in the first step, we use Eq. (9.25) in the second step, we use Lemma 9.41 in the third step and the last step comes from choice of ϵ_N. This completes the proof of Eq. (9.37). □

9.11.7 Our main result

Theorem 9.46 (Robust central path)**.** *Consider an SDP instance defined in Definition 9.1 with no redundant constraints. Assume that the feasible region is bounded,*

Algorithm 44 Our robust barrier method for SDP.

```
1: procedure SolveSDP($m, n, C, \{A_i\}_{i=1}^m$, $\mathsf{A} \in \mathbb{R}^{m \times n^2}$, $b \in \mathbb{R}^m$)
2:                                                              ▷ Initialization
3:     $\eta \leftarrow \frac{1}{\theta+2}$,   $T \leftarrow \frac{40}{\epsilon_N}\sqrt{\theta}\log(\frac{\theta}{\epsilon})$
4:     Find initial feasible dual $y \in \mathbb{R}^m$ according to Lemma E.1    ▷ Condition 0 in Lemma 9.40
5:     for $t = 1 \to T$ do do                     ▷ Iterations of approximate barrier method
6:         $\eta^{\mathrm{new}} \leftarrow \eta \cdot (1 + \frac{\epsilon_N}{20\sqrt{\theta}})$
7:         $\widetilde{S} \leftarrow$ ApproxSlack()
8:         $\widetilde{H} \leftarrow$ ApproxHessian()                    ▷ Condition 1 in Lemma 9.40
9:         $\widetilde{g} \leftarrow$ ApproxGradient()                   ▷ Condition 2 in Lemma 9.40
10:        $\widetilde{\delta}(y) \leftarrow$ ApproxDelta()                  ▷ Condition 3 in Lemma 9.40
11:        $y^{\mathrm{new}} \leftarrow y + \delta(y)$
12:        $y \leftarrow y^{\mathrm{new}}$                                  ▷ We update variables
13:    end for
14:    return an approximate solution to the original problem      ▷ Lemma E.1
15: end procedure
```

i.e., $\|X\|_2 \leq R$. *Suppose in each iteration, the* $\widetilde{S}, \widetilde{H}, \widetilde{g}, \widetilde{\delta}$ *are computed in Line 7, Line 8, Line 9, Line 10 of Algorithm 44 that satisfy Condition 1 or 1' & Condition 2 & Condition 3 in Lemma 9.40. Suppose* ϕ *is a self-concordant barrier with complexity* $\theta \geq 1$. *Assume* X^* *is an optimal solution to the semidefinite program in Definition 9.1. Then for any error parameter* $0 < \epsilon \leq 0.01$ *and Newton step size* ϵ_N *satisfying* $\sqrt{\epsilon} < \epsilon_N \leq 0.01$, *Algorithm 44 outputs, in* $T = 40\epsilon_N^{-1}\sqrt{\theta}\log(\theta/\epsilon)$ *iterations, a positive semidefinite matrix* $X \in \mathbb{R}^{n \times n}_{\geq 0}$ *s.t.*

$$\begin{aligned} &\langle C, X\rangle \geq \langle C, X^*\rangle - \epsilon \cdot \|C\|_2 \cdot R, \quad \textit{and} \\ &\textstyle\sum_{i=1}^m \left|\langle A_i, \widehat{X}\rangle - b_i\right| \leq 4n\epsilon \cdot \left(R \sum_{i=1}^m \|A_i\|_1 + \|b\|_1\right), \end{aligned} \tag{9.38}$$

Furthermore, for logarithmic barrier $\phi_{\log}$, *in each iteration of Algorithm 44, the following invariant holds:*

$$\|S^{-1/2} S^{\mathrm{new}} S^{-1/2} - I\|_F \leq 1.03 \cdot \epsilon_N. \tag{9.39}$$

Proof. First, we use Lemma E.1 to rewrite the semidefinite programming and obtain an initial feasible solution near the dual central path with $\eta = 1/(\theta + 2)$. Thus, the induction hypothesis

$$\mathsf{g}(y,\eta)^\top(\nabla^2\phi(y))^{-1}\mathsf{g}(y,\eta) \leq \epsilon_N^2$$

holds at the initial of algorithm.

Say y is the modified semidefinite programming's dual solution. Theorem 9.45 shows that y has duality gap $\leq \epsilon^2$.

Finally, we use Lemma E.1, to get an approximate solution to the original semidefinite programming satisfying Eq. (9.38). $\square$

9.12 Hybrid Barrier-Based SDP Solver

The hybrid barrier [NN89, Ans00] is another useful barrier function to solve SDP and converges within a smaller number of iteration when $m \leq n$. However, it is hard to be implemented efficiently due to the complex form of Hessian matrices. In this section, we give an efficient algorithm for solving SDP using the hybrid barrier in [Ans00] that improves the naive implementation in all parameter regimes[10].

In Section 9.12.1, we review some basic facts on the hybrid barrier for SDP. In Section 9.12.2, we give the formal version of the algorithm and time complexity result. In Section 9.12.3, we show how to low-rank approximate the change of $Q(S)$. In Section 9.12.4, we prove that the slack variable S changes slowly in each iteration. Section 9.12.5 contains the amortized analysis for our hybrid barrier SDP solver. Combining them together, we prove the main theorem (Theorem 9.58) in Section 9.12.6.

[10]We also improves our straightforward algorithm in most parameter regimes. See Remark E.1 for the different implementations of [Ans00].

9.12.1 Basic facts on the hybrid barrier

The barrier function is defined as follows:

$$\phi(y) := 225\sqrt{\frac{n}{m}} \cdot \left(\phi_{\mathrm{vol}}(y) + \frac{m-1}{n-1} \cdot \phi_{\log}(y)\right),$$

with

$$\begin{aligned}\phi_{\mathrm{vol}}(y) &:= 0.5 \log\det(H(y)),\\ \phi_{\log}(y) &:= -\log\det(S(y)),\end{aligned}$$

Note $H(y)$ and $S(y)$ are defined in Definition 9.6.

According to our robust IPM framework, for every iteration, we have to calculate/estimate the gradient and Hessian of $\phi_{\mathrm{vol}}(y)$, whose closed-forms are computed in [Ans00]. More specifically,

$$(\nabla \phi_{\mathrm{vol}}(y))_i = -\mathrm{tr}[H(S)^{-1} \cdot \mathsf{A}(S^{-1}A_iS^{-1} \otimes S^{-1})\mathsf{A}^\top] \quad \forall i \in [m],$$

and

$$\nabla^2 \phi_{\mathrm{vol}}(y) = 2Q(S) + R(S) - 2T(S),$$

where for any $i, j \in [m]$,

$$\begin{aligned}Q(S)_{i,j} &= \mathrm{tr}[H(S)^{-1}\mathsf{A}(S^{-1}A_iS^{-1}A_jS^{-1} \otimes_S S^{-1})\mathsf{A}^\top],\\ R(S)_{i,j} &= \mathrm{tr}[H(S)^{-1}\mathsf{A}(S^{-1}A_iS^{-1} \otimes_S S^{-1}A_jS^{-1})\mathsf{A}^\top],\\ T(S)_{i,j} &= \mathrm{tr}[H(S)^{-1}\mathsf{A}(S^{-1}A_iS^{-1} \otimes_S S^{-1})\mathsf{A}^\top H(S)^{-1}\mathsf{A}(S^{-1}A_jS^{-1} \otimes_S S^{-1})\mathsf{A}^\top].\end{aligned}$$

The following fact in [Ans00] shows that $Q(S)$ is a good PSD approximation of the Hessian $\nabla^2\phi_{\mathrm{vol}}(y)$.

Fact 9.47 ([Ans00])**.** *For $S \succ 0$,*

$$\frac{1}{n}H(S) \preceq Q(S) \preceq \nabla^2\phi_{\mathrm{vol}}(y) \preceq 3Q(S).$$

We also need the following lower bound on the quadratic form of hybrid barrier's Hessian.

Fact 9.48 ([Ans00])**.** *Let $S \succ 0$. For any $\xi \in \mathbb{R}^m$, we have*

$$\xi^\top \left(Q(S) + \frac{m-1}{n-1} \cdot H(S) \right) \xi \geq \frac{2\sqrt{m}}{1+\sqrt{n}} \cdot \left\| S^{-1/2} \left(\sum_{i=1}^m \xi_i A_i \right) S^{-1/2} \right\|_2^2 .$$

9.12.2 Efficient implementation via robust SDP framework

Lemma 9.49. *In Algorithm 45, the (amortized) cost/time for every iteration is*

$$O^* \left(\left(\frac{n}{m} \right)^{\frac{1}{4}} \cdot (n^2 m + m^\omega n^{1/4}) + m^2 n^\omega + m^4 + m^2 \cdot n^{\omega - \frac{1}{2}} \cdot \left(\frac{n}{m} \right)^{\frac{1}{4}} \right).$$

Proof. Consider iteration t. In Line 16-21 of Algorithm 45, we update $S^{\mathrm{new}}, \widetilde{S}^{\mathrm{new}} \in \mathbb{R}^{n \times n}$ in $O(mn^2 + n^\omega)$ time.

In Line 16-17 in Algorithm 46, we can first compute $S^{-1} A_i S^{-1} A_j \in \mathbb{R}^{n \times n}$ and $S^{-1} A_i \in \mathbb{R}^{n \times n}$ for all $i, j \in [m]$, in $O(m^2 n^\omega)$ time. Notice

$$\begin{aligned} (\nabla \phi_{\mathrm{vol}}(y))_i = & -\mathrm{tr}[H(S)^{-1} \cdot \mathsf{A}(S^{-1} A_i S^{-1} \otimes S^{-1}) \mathsf{A}^\top] \\ = & \sum_{k=1}^m \sum_{l=1}^m -H(S)^{-1}_{k,l} \cdot \mathrm{tr}[A_k S^{-1} A_l S^{-1} A_i S^{-1}]. \end{aligned}$$

Then it takes $O(m^3 n^2)$ to find $\mathrm{tr}[A_k S^{-1} A_l S^{-1} A_i S^{-1}]$ for all $i, k, l \in [m]$ and subsequently $O(m^3)$-time to compute $\nabla \phi_{\mathrm{vol}}(y)$. Hence, the cost of Line 16-17 in Algorithm 46 is $O(m^2 n^\omega + m^3 n^2 + m^3)$.

For Line 24-25 of Algorithm 46, we first compute

$$A_i, S^{-1} A_j V_3,\ A_i V_3,,\ V_4^\top A_i S^{-1} A_j S^{-1},\ S^{-1} A_i V_3,\ V_4^\top A_j,\ V_4^\top A_i V_3$$

for every $i \in \{1, \cdots, m\}$, for every $j \in \{1, \cdots, m\}$, in $O(m^2 \cdot \mathcal{T}_{\mathrm{mat}}(n, n, r_t))$ time. Notice

$$\begin{aligned} \widetilde{H}_{i,j} &= \mathrm{tr}[\widetilde{S}^{-1} A_i \widetilde{S}^{-1} A_j] \\ &= H_{i,j} + \mathrm{tr}[S^{-1} A_i V_3 V_4^\top A_j] + \mathrm{tr}[S^{-1} A_j V_3 V_4^\top A_i] + \mathrm{tr}[V_3 V_4^\top A_i V_3 V_4^\top A_j]. \end{aligned}$$

Algorithm 45 Hybrid Barrier SDP solver.

```
1: members
2:     $S, \widetilde{S} \in \mathbb{R}^{n \times n}$    ▷ Slack variables
3:     $y \in \mathbb{R}^m$    ▷ Dual variable
4:     $H, Q \in \mathbb{R}^{m \times m}$    ▷ Parts of the Hessian matrices
5:     $\eta \in \mathbb{R}$    ▷ Learning rate
6:     $\mathsf{A} \in \mathbb{R}^{m \times n^2}$    ▷ Batched constraint matrix
7:     $G \in \mathbb{R}^{m \times m}$    ▷ The inverse of the Hessian matrix
8: end members

9: procedure HybridBarrier($(m, n, C, \{A_i\}_{i=1}^m,\ b \in \mathbb{R}^m)$)
10:     Initialize($m, n, C, \{A_i\}_{i=1}^m,\ b \in \mathbb{R}^m$)    ▷ Algorithm 46
11:     for $t = 1 \to T$ do    ▷ Iterations of approximate barrier method
12:         $\eta^{\mathrm{new}} \leftarrow \eta \cdot \left(1 + \frac{\epsilon_N}{20(mn)^{1/4}}\right)$
13:         $g_{\eta^{\mathrm{new}}}(y) \leftarrow$ HybridGradient($\eta^{\mathrm{new}}, b, C, \{A_i\}_{i=1}^m$)
14:         $\delta_y \leftarrow -G \cdot g_{\eta^{\mathrm{new}}}(y)$    ▷ Update on $y \in \mathbb{R}^m$
15:         $y^{\mathrm{new}} \leftarrow y + \delta_y$
16:         $S^{\mathrm{new}} \leftarrow \sum_{i \in [m]} (y^{\mathrm{new}})_i \cdot A_i - C$
17:         $V_1, V_2 \leftarrow$ LowRankSlackUpdate($S^{\mathrm{new}}, \widetilde{S}$)    ▷ $V_1, V_2 \in \mathbb{R}^{n \times r_t}$. Algorithm 43
18:         $\widetilde{S}^{\mathrm{new}} \leftarrow \widetilde{S} + V_1 V_2^\top$    ▷ Approximate slack computation
19:         $V_3 \leftarrow -\widetilde{S}^{-1} V_1 (I + V_2^\top \widetilde{S}^{-1} V_1)^{-1}$    ▷ $V_3 \in \mathbb{R}^{n \times r_t}$
20:         $V_4 \leftarrow \widetilde{S}^{-1} V_2$    ▷ $V_4 \in \mathbb{R}^{n \times r_t}$
21:         $y \leftarrow y^{\mathrm{new}}$, $S \leftarrow S^{\mathrm{new}}$, $\widetilde{S} \leftarrow \widetilde{S}^{\mathrm{new}}$, $\eta \leftarrow \eta^{\mathrm{new}}$    ▷ Update variables
22:         $(\widetilde{Q}, \widetilde{H}, G^{\mathrm{new}}) \leftarrow$ HybridHessian($V_3, V_4$)    ▷ $G^{\mathrm{new}} \in \mathbb{R}^{m \times m}$
23:             ▷ Hessian inverse computation using Woodbury identity
24:         $Q \leftarrow \widetilde{Q}$, $H \leftarrow \widetilde{H}$, $G \leftarrow G^{\mathrm{new}}$    ▷ Update matrices
25:     end for
26:     return an approximate solution to the original problem    ▷ Lemma E.1
27: end procedure
```

Hence, it takes $\mathcal{T}_{\mathrm{mat}}(m, nr_t, m)$ to compute $\mathrm{tr}[S^{-1} A_i V_3 V_4^\top A_j], \mathrm{tr}[V_3 V_4^\top A_i V_3 V_4^\top A_j]$ for all $i, j \in [m]$ (by batching them together and using fast matrix multiplication on a m-by-nr_t matrix and a nr_t-by-m matrix) and subsequently $O(m^2)$-time to compute

Algorithm 46 Hybrid Barrier SDP solver, continued.

```
1: procedure INTIALIZE($(m, n, C, \{A_i\}_{i=1}^m, \mathsf{A} \in \mathbb{R}^{m \times n^2}, b \in \mathbb{R}^m)$)
2:     Construct $\mathsf{A} \in \mathbb{R}^{m \times n^2}$ by stacking $m$ vectors $\mathrm{vec}[A_1], \mathrm{vec}[A_2], \cdots, \mathrm{vec}[A_m] \in \mathbb{R}^{n^2}$
3:     $\eta \leftarrow \frac{1}{(mn)^{1/2}+2}, \quad T \leftarrow \frac{40}{\epsilon_N}(mn)^{1/4}\log(\frac{mn}{\epsilon})$
4:     Find initial feasible dual vector $y \in \mathbb{R}^m$ according to Lemma E.1
5:     $S \leftarrow \sum_{i\in[m]} y_i \cdot A_i - C, \quad \widetilde{S} \leftarrow S$        ▷ $S, \widetilde{S} \in \mathbb{R}^{n \times n}$
6:     $H(S) \leftarrow \mathsf{A}(S^{-1} \otimes S^{-1})\mathsf{A}^\top$
7:     for $i = 1, \cdots, m$ do
8:         $Q(S)_{i,j} \leftarrow \mathrm{tr}[H^{-1}\mathsf{A}(S^{-1}A_iS^{-1}A_jS^{-1} \otimes_S S^{-1})\mathsf{A}^\top]$
9:         $R(S)_{i,j} \leftarrow \mathrm{tr}[H^{-1}\mathsf{A}(S^{-1}A_iS^{-1} \otimes_S S^{-1}A_jS^{-1})\mathsf{A}^\top]$
10:        $O(S)_{i,j} \leftarrow \mathrm{tr}[H^{-1}\mathsf{A}(S^{-1}A_iS^{-1} \otimes_S S^{-1})\mathsf{A}^\top H^{-1}\mathsf{A}(S^{-1}A_jS^{-1} \otimes_S S^{-1})\mathsf{A}^\top]$
11:    end for
12:    $G \leftarrow \left(225 \cdot \sqrt{\frac{n}{m}} \cdot \left(2Q(S) + R(S) - 2O(S) + \frac{m-1}{n-1} \cdot H(S)\right)\right)^{-1}$        ▷ $G \in \mathbb{R}^{m \times m}$
13: end procedure
14: procedure HYBRIDGRADIENT($(m, n, C, \{A_i\}_{i=1}^m, b \in \mathbb{R}^m)$)
15:    for $i = 1, \cdots, m$ do
16:        $\nabla\phi_{\log}(y)_i \leftarrow -\mathrm{tr}[S^{-1} \cdot A_j]$        ▷ Gradient of $\phi_{\log}$
17:        $\nabla\phi_{\mathrm{vol}}(y)_i \leftarrow -\mathrm{tr}[H(S)^{-1} \cdot \mathsf{A}(S^{-1}A_iS^{-1} \otimes S^{-1})\mathsf{A}^\top]$        ▷ Gradient of $\phi_{\mathrm{vol}}$
18:    end for
19:    $g_{\eta^{\mathrm{new}}}(y) \leftarrow \eta^{\mathrm{new}}b - 225\sqrt{\frac{n}{m}} \cdot \left(\nabla\phi_{\mathrm{vol}}(y) + \frac{m-1}{n-1} \cdot \nabla\phi_{\log}(y)\right)$
20:    return $g_{\eta^{\mathrm{new}}}(y)$
21: end procedure
22: procedure HYBRIDHESSIAN($(V_3, V_4)$)
23:    for $i, j = 1, \cdots, m$ do
24:        $\widetilde{H}_{i,j} \leftarrow H_{i,j} + \mathrm{tr}[S^{-1}A_iV_3V_4^\top A_j] + \mathrm{tr}[S^{-1}A_jV_3V_4^\top A_i] + \mathrm{tr}[V_3V_4^\top A_iV_3V_4^\top A_j]$
25:        $\widetilde{Q}_{i,j} \leftarrow Q_{i,j} + \mathrm{tr}[H^{-1}\mathsf{A}(S^{-1}A_iV_3V_4^\top A_jS^{-1} \otimes_S S^{-1})\mathsf{A}^\top]$
26:    end for
27:    $G^{\mathrm{new}} \leftarrow \left(225 \cdot 1.001 \cdot \sqrt{\frac{n}{m}} \cdot \left(3 \cdot \widetilde{Q} + \frac{m-1}{n-1} \cdot \widetilde{H}\right)\right)^{-1} \in \mathbb{R}^{m \times m}$        ▷ $G \in \mathbb{R}^{m \times m}$
28:    return $(\widetilde{Q}, \widetilde{H}, G^{\mathrm{new}})$
29: end procedure
```

$\widetilde{Q}$. So the cost of Line 24 is $\mathcal{T}_{\mathrm{mat}}(m, nr_t, m) + m^2$. For $\widetilde{Q}$, notice that for any $i, j \in [m]$,

$$\begin{aligned}\widetilde{Q}_{i,j} &= \mathrm{tr}[H(S)^{-1}\mathsf{A}(S^{-1}A_i\widetilde{S}^{-1}A_jS^{-1} \otimes_S S^{-1})\mathsf{A}^\top] \\ &= Q_{i,j} + \sum_{k=1}^m \sum_{l=1}^m -H(S)^{-1}_{k,l} \cdot \frac{1}{2}\left(\mathrm{tr}[A_kS^{-1}A_lV_3V_4^\top A_iS^{-1}A_jS^{-1}]\right. \\ &\quad \left. + \mathrm{tr}[A_kS^{-1}A_iV_3V_4^\top A_jS^{-1}A_\ell S^{-1}]\right).\end{aligned}$$

Then it takes $\mathcal{T}_{\mathrm{mat}}(m^2, nr_t, m^2)$ to compute $\mathrm{tr}[A_k S^{-1} A_l V_3 V_4^\top A_i S^{-1} A_j S^{-1}]$ for all $i, j, k, l \in [m]$ (by batching them together and using fast matrix multiplication on a m^2-by-nr_t matrix and a nr_t-by-m^2 matrix) and subsequently $O(m^4)$-time to compute $\widetilde{Q}$. Hence the cost of Line 25 is $\mathcal{T}_{\mathrm{mat}}(m^2, nr_t, m^2) + m^4$.

In total, Line 24-25 of Algorithm 46 takes

$$\mathcal{T}_{\mathrm{mat}}(m^2, nr_t, m^2) + m^2 n^\omega + m^3 n^2 + m^4 + m^2 \cdot \mathcal{T}_{\mathrm{mat}}(n, n, r_t).$$

Summing up, the total cost in iteration t is therefore given by

$$\mathcal{T}_{\mathrm{mat}}(m^2, nr_t, m^2) + m^2 n^\omega + m^3 n^2 + m^4 + m^2 \cdot \mathcal{T}_{\mathrm{mat}}(n, n, r_t).$$

Using Theorem 9.53 with Lemma 9.51, and Fact 9.16,

$$\begin{aligned}\sum_{t=1}^T \mathcal{T}_{\mathrm{mat}}(n, n, r_t) \leq & \sum_{t=1}^T O^*\left(n^2 + r_t^{\frac{\omega-2}{1-\alpha}} \cdot n^{2-\frac{\alpha(\omega-2)}{1-\alpha}}\right) \\ & \leq O^*\left(T \cdot n^2 + T \cdot n^{\frac{\omega-2}{1-\alpha}-1/2} \cdot n^{2-\frac{\alpha(\omega-2)}{1-\alpha}} \cdot (n/m)^{\frac{1}{4}}\right) \\ & \leq O^*\left(T \cdot (n^2 + n^{\omega-1/2} \cdot (n/m)^{\frac{1}{4}})\right).\end{aligned}$$

Using Corollary 9.57,

$$\sum_{t=1}^T \mathcal{T}_{\mathrm{mat}}(m^2, nr_t, m^2) \leq O^*\left(T \cdot (n/m)^{1/4} \cdot (n^2 m + m^\omega n^{1/4})\right).$$

Now, let us compute the (amortized) cost per iteration

$$\begin{aligned}& O^*\left(\left(\frac{n}{m}\right)^{\frac{1}{4}} \cdot (n^2 m + m^\omega n^{1/4}) + m^2 n^\omega + m^4 + m^2 \cdot \left(n^2 + n^{\omega-\frac{1}{2}} \cdot \left(\frac{n}{m}\right)^{\frac{1}{4}}\right)\right) \\ = & O^*\left(\left(\frac{n}{m}\right)^{\frac{1}{4}} \cdot (n^2 m + m^\omega n^{1/4}) + m^2 n^\omega + m^4 + m^2 \cdot n^{\omega-\frac{1}{2}} \cdot \left(\frac{n}{m}\right)^{\frac{1}{4}}\right).\end{aligned}$$

□

9.12.3 Approximation to Q

The following lemma shows that $\widetilde{Q}$ in our algorithm is a good PSD approximation of $Q(S)$.

Lemma 9.50. *Let $\widetilde{Q} \in \mathbb{R}^{m \times m}$ be given by*

$$\widetilde{Q}_{i,j} = \mathrm{tr}[H(S)^{-1}\mathsf{A}(S^{-1}A_i\widetilde{S}^{-1}A_jS^{-1} \otimes_S S^{-1})\mathsf{A}^\top].$$

Then suppose $(1+\epsilon_S)^{-1}S \preceq \widetilde{S} \preceq (1+\epsilon_S)S$ for $\epsilon_S \in (0, 0.001)$, we have $(1+\epsilon_S)^{-3}Q \preceq \widetilde{Q} \preceq (1+\epsilon_S)^3 Q$ where $Q := Q(S)$.

Proof. Fix $v \in \mathbb{R}^m$. We have

$$\begin{aligned} v^\top \widetilde{Q} v = & \sum_{i=1}^m \sum_{j=1}^m v_i v_j \mathrm{tr}[H(S)^{-1}\mathsf{A}(S^{-1}A_i\widetilde{S}^{-1}A_jS^{-1} \otimes_S S^{-1})\mathsf{A}^\top] \\ = & \ \mathrm{tr}\left[\mathsf{A}^\top H(S)^{-1}\mathsf{A} \cdot \left(S^{-1}(\sum_{i=1}^m v_iA_i)\widetilde{S}^{-1}(\sum_{j=1}^m v_jA_j)S^{-1} \otimes_S S^{-1}\right)\right] \\ = & \ \frac{1}{2} \cdot (\mathrm{tr}[\mathsf{A}^\top H^{-1}\mathsf{A} \cdot (S^{-1}B\widetilde{S}^{-1}BS^{-1} \otimes S^{-1})] + \mathrm{tr}[\mathsf{A}^\top H^{-1}\mathsf{A} \cdot (S \otimes S^{-1}B\widetilde{S}^{-1}BS^{-1})]) \end{aligned}$$

where we abbreviate $B = \sum_{i=1}^m v_iA_i$ and $H = H(S)$.

For $\mathrm{tr}[\mathsf{A}^\top H^{-1}\mathsf{A} \cdot (S^{-1}B\widetilde{S}^{-1}BS^{-1} \otimes S^{-1})]$, we have

$$\begin{aligned} & \mathrm{tr}[\mathsf{A}^\top H^{-1}\mathsf{A} \cdot (S^{-1}B\widetilde{S}^{-1}BS^{-1} \otimes S^{-1})] \\ = & \ \mathrm{tr}[(S^{-1}B\widetilde{S}^{-1/2} \otimes I)^\top \mathsf{A}^\top H^{-1}\mathsf{A}(S^{-1}B\widetilde{S}^{-1/2} \otimes I)(I \otimes S^{-1})] \\ \leq & \ (1+\epsilon_S) \cdot \mathrm{tr}[(S^{-1}B\widetilde{S}^{-1/2} \otimes I)^\top \mathsf{A}^\top H^{-1}\mathsf{A}(S^{-1}B\widetilde{S}^{-1/2} \otimes I)(I \otimes \widetilde{S}^{-1})] \\ = & \ (1+\epsilon_S) \cdot \mathrm{tr}[(S^{-1}B \otimes I)^\top \mathsf{A}^\top H^{-1}\mathsf{A}(S^{-1}B \otimes I)(\widetilde{S}^{-1} \otimes \widetilde{S}^{-1})] \\ \leq & \ (1+\epsilon_S)^3 \cdot \mathrm{tr}[(S^{-1}B \otimes I)^\top \mathsf{A}^\top H^{-1}\mathsf{A}(S^{-1}B \otimes I)(S^{-1} \otimes S^{-1})] \\ = & \ (1+\epsilon_S)^3 \cdot \mathrm{tr}[\mathsf{A}^\top H^{-1}\mathsf{A} \cdot (S^{-1}BS^{-1}BS^{-1} \otimes S^{-1})], \end{aligned}$$

where we use Fact 9.10 in first step; we use $I \otimes S^{-1} \preceq (1+\epsilon) \cdot (I \otimes \widetilde{S}^{-1})$ and Fact 9.6 in second step; the third step comes from Fact 9.10; the fourth step comes from Fact 9.11; the last step comes from Fact 9.10.

Similarly, for $\mathrm{tr}[\mathsf{A}^\top H^{-1}\mathsf{A} \cdot (S^{-1} \otimes S^{-1}B\widetilde{S}^{-1}BS^{-1})]$, we have

$$\begin{aligned}
&\mathrm{tr}[\mathsf{A}^\top H^{-1}\mathsf{A} \cdot (S^{-1} \otimes S^{-1}B\widetilde{S}^{-1}BS^{-1})]\\
= &\mathrm{tr}[(S^{-1/2} \otimes S^{-1}B)^\top \mathsf{A}^\top H^{-1}\mathsf{A}(S^{-1/2} \otimes S^{-1}B)(I \otimes \widetilde{S}^{-1})]\\
\leq &(1+\epsilon_S) \cdot \mathrm{tr}[(S^{-1/2} \otimes S^{-1}B)^\top \mathsf{A}^\top H^{-1}\mathsf{A}(S^{-1/2} \otimes S^{-1}B)(I \otimes S^{-1})]\\
= &(1+\epsilon_S) \cdot \mathrm{tr}[\mathsf{A}^\top H^{-1}\mathsf{A} \cdot (S^{-1} \otimes S^{-1}BS^{-1}BS^{-1})],
\end{aligned}$$

where we use Fact 9.10 in the first step; we use $I \otimes S^{-1} \preceq (1+\epsilon) \cdot (I \otimes \widetilde{S}^{-1})$ and Fact 9.6 in second step; the third step comes from Fact 9.10.

Summing up,

$$\begin{aligned}
v^\top \widetilde{Q} v \leq &(1+\epsilon_S)^3 \cdot \mathrm{tr}[\mathsf{A}^\top H^{-1}\mathsf{A} \cdot (S^{-1}BS^{-1}BS^{-1} \otimes S^{-1})]\\
&+ (1+\epsilon_S) \cdot \mathrm{tr}[\mathsf{A}^\top H^{-1}\mathsf{A} \cdot (S^{-1} \otimes S^{-1}BS^{-1}BS^{-1})]\\
\leq &(1+\epsilon_S)^3 \cdot \mathrm{tr}[\mathsf{A}^\top H^{-1}\mathsf{A} \cdot (S^{-1}BS^{-1}BS^{-1} \otimes_S S^{-1})]\\
= &(1+\epsilon_S)^3 \cdot v^\top Q v.
\end{aligned}$$

Similarly, $v^\top \widetilde{Q} v \geq (1+\epsilon_S)^{-3} \cdot v^\top Q v$. Therefore, $(1+\epsilon_S)^{-3} Q \preceq \widetilde{Q} \preceq (1+\epsilon_S)^3 Q$, since v can be arbitrarily chosen. □

9.12.4 S move in hybrid barrier

Lemma 9.51 (S move in hybrid barrier)**.** *Consider Algorithm 45-46, in each iteration, the following invariant holds:*

$$\begin{aligned}
&\|S^{-1/2}S^{\mathrm{new}}S^{-1/2} - I\|_F \leq 1.03 \cdot \epsilon_N \cdot (n/m)^{1/2}, &(9.40)\\
&\|S^{-1/2}S^{\mathrm{new}}S^{-1/2} - I\|_2^2 \leq 0.002 \cdot \epsilon_N.
\end{aligned}$$

Proof. We note that the robust framework and all corresponding results in Section 9.3

directly applies to hybrid barrier with $\theta = (mn)^{1/2}$. We have

$$\begin{aligned}
\|S^{-1/2}S^{\text{new}}S^{-1/2} - I\|_F^2 &= \text{tr}\left[\left(S^{-1/2}(S^{\text{new}} - S)S^{-1/2}\right)^2\right] \\
&= \text{tr}\left[S^{-1}\Big(\sum_{i\in[m]} \widetilde{\delta}_{y,i}A_i\Big)S^{-1}\Big(\sum_{i\in[m]} \widetilde{\delta}_{y,i}A_i\Big)\right] \\
&= \sum_{i\in[m]}\sum_{j\in[m]} \widetilde{\delta}_{y,i}\widetilde{\delta}_{y,j}\text{tr}[S^{-1}A_iS^{-1}A_j] \\
&= \widetilde{\delta}_y^\top H(y)\widetilde{\delta}_y \\
&= \|\widetilde{\delta}_y\|_{H(y)}^2
\end{aligned}$$

where the second step comes from $S^{\text{new}} - S = \sum_{i=1}^m \widetilde{\delta}_{y,i}A_i$ and the rest follows from algebra. It suffices to bound $\|\widetilde{\delta}_y\|_{H(y)}$.

Since $\phi(y) = \sqrt{\frac{n}{m}} \cdot \left(\phi_{\text{vol}}(y) + \frac{m-1}{n-1} \cdot \phi_{\log}(y)\right)$, we have

$$\nabla^2\phi(y) = (n/m)^{1/2}\nabla^2\phi_{\text{vol}}(y) + (m/n)^{1/2}H(y).$$

By Fact 9.47, we have

$$Q(S) \succeq \frac{1}{n}H(y).$$

And we also have

$$\begin{aligned}
\nabla^2\phi(y) &\succeq (n/m)^{1/2}Q(S) + (m/n)^{1/2}H(y) \\
&\succeq \left((n/m)^{1/2} \cdot n^{-1} + (m/n)^{1/2}\right) H(y) \\
&\succeq O((m/n)^{1/2})H(y).
\end{aligned}$$

Thus,

$$\|\widetilde{\delta}_y\|_{H(y)}^2 \le (n/m)^{1/2} \cdot \|\widetilde{\delta}_y\|_{\nabla^2\phi(y)}^2.$$

For $\|\widetilde{\delta}_y\|_{\nabla^2\phi(y)}^2$, we have

$$\begin{aligned}
\|\widetilde{\delta}_y\|_{\nabla^2\phi(y)}^2 &\le \|n(y,\eta^{\text{new}})\|_{\nabla^2\phi(y)} + \|\widetilde{\delta}_y - (-n(y,\eta^{\text{new}}))\|_{\nabla^2\phi(y)} \\
&\le 1.01\|n(y,\eta^{\text{new}})\|_{\nabla^2\phi(y)} \\
&\le 1.01 \cdot \left((1+\epsilon_N/20)\|n(y,\eta)\|_{\nabla^2\phi(y)} + \epsilon/20\right) \\
&\le 1.03 \cdot \epsilon_N
\end{aligned}$$

where the first step comes from triangle inequality, the second step comes from Eq. (9.25), the third step comes from Lemma 9.41 and the last step comes from choice of ϵ_N. Hence,

$$\|\widetilde{\delta}_y\|_{H(y)}^2 \le 1.03 \cdot \epsilon_N \cdot (n/m)^{1/2},$$

that is,

$$\|S^{-1/2} S^{\mathrm{new}} S^{-1/2} - I\|_F \le 1.03 \cdot \epsilon_N \cdot (n/m)^{1/2}.$$

Moreover, by Fact 9.48, we have

$$\begin{aligned} \|\widetilde{\delta}_y\|_{\nabla^2 \phi(y)}^2 &\ge 225 (n/m)^{1/2} \cdot \frac{2\sqrt{m}}{1+\sqrt{n}} \cdot \left\| S^{-1/2} \left(\sum_{i=1}^m \widetilde{\delta}_{y,i} A_i \right) S^{-1/2} \right\|_2^2 \\ &= O(1) \cdot \|S^{-1/2} S^{\mathrm{new}} S^{-1/2} - I\|_2^2. \end{aligned}$$

Hence, we have

$$\|S^{-1/2} S^{\mathrm{new}} S^{-1/2} - I\|_2^2 \le 0.002 \cdot \epsilon_N.$$

This completes the proof of Eq. (9.40). □

9.12.5 Property of low rank update for the hybrid barrier

Assumption 9.52 (Closeness of S^{new} and $\widetilde{S}$ from S)**.** We make the following two assumptions about $S, \widetilde{S}, S^{\mathrm{new}} \in \mathbb{R}^{n \times n}$:

1. $\|S^{-1/2} \cdot S^{\mathrm{new}} \cdot S^{-1/2} - I\|_F \le 0.02 (n/m)^{1/2}$,
2. $\|S^{-1/2} \cdot S^{\mathrm{new}} \cdot S^{-1/2} - I\|_2 \le 0.005$,
3. $\|S^{-1/2} \cdot \widetilde{S} \cdot S^{-1/2} - I\|_2 \le 0.01$.

Theorem 9.53 (General amortized guarantee for the hybrid barrier)**.** *Let T denote the total number of iterations in Algorithm 45-46. Let r_t denote the rank of the update matrices $V_1, V_2 \in \mathbb{R}^{n \times r_t}$ generated by Algorithm 43 in the t-th iteration. Suppose*

Assumption 9.52 hold. The ranks r_t's satisfy the following condition: for any vector $g \in \mathbb{R}^n_+$ which is non-increasing, we have

$$\sum_{t=1}^{T} r_t \cdot g_{r_t} \le O(T \cdot (n/m)^{1/4} \cdot \|g\|_2 \cdot \log n). \tag{9.41}$$

The proof of Theorem 9.53 relies on the following three lemmas:

Lemma 9.54 (Variant of Lemma 9.30)**.** *Let matrices $Z, Z^{\mathrm{mid}} \in \mathbb{R}^{n\times n}$ be defined as in Definition 9.5. Under Assumption 9.52, we have*

$$\sum_{i=1}^{n} (\lambda(Z)_{[i]} - \lambda(Z^{\mathrm{mid}})_{[i]})^2 \le 10^{-3}(n/m)^{1/2},$$

where $\lambda(Z)_{[i]}$ denotes the i-th largest eigenvalue of Z.

Proof sketch. The proof is very similar to Lemma 9.30 except the upper bound of the following quantity:

$$\|S^{1/2}(S^{\mathrm{new}})^{-1}S^{1/2} - I\|_F^2 = \sum_{i=1}^{n} (\nu_i^{-1} - 1)^2,$$

where $\{\nu_i\}_{i\in[n]}$ are the eigenvalues of $S^{-1/2}S^{\mathrm{new}}S^{-1/2}$. Then, by Assumption 9.52 part 2, we have

$$\max_{i\in[n]} |v_i - 1| \le 0.005,$$

which implies that $v_i \ge 0.995$ for all $i \in [n]$. By Assumption 9.52,

$$\sum_{i=1}^{n} (v_i - 1)^2 \le 0.02(n/m)^{1/2}.$$

Then, it follows that

$$\sum_{i=1}^{n} (\nu_i^{-1} - 1)^2 \le 5 \times 10^{-4} \cdot (n/m)^{1/2}.$$

The remaining part does not change. □

Lemma 9.55 (S move)**.** *Consider the t-th iteration. Let matrices $Z, Z^{\mathrm{mid}} \in \mathbb{R}^{n\times n}$ be defined as in Definition 9.5. Let $g \in \mathbb{R}^n_+$ be a non-increasing vector, and let $\Phi_g : \mathbb{R}^{n\times n} \to \mathbb{R}_+$ be defined as in Definition 9.4.*

Under Assumption 9.52, we have

$$\Phi_g(Z^{\mathrm{mid}}) - \Phi_g(Z) \leq \|g\|_2 \cdot (n/m)^{1/4}.$$

Proof sketch. The proof is basically the same as Lemma 9.31. We can upper bound the LHS as follows:

$$\begin{aligned}\Phi_g(Z^{\mathrm{mid}}) - \Phi_g(Z) &\leq \|g\|_2 \cdot \Big(\sum_{i=1}^n |\lambda(Z^{\mathrm{mid}})_{\pi(i)} - \lambda(Z)_{\pi(i)}|^2\Big)^{1/2} \\ &\leq \|g\|_2 \cdot (n/m)^{1/4},\end{aligned}$$

where the last step follows from Lemma 9.54. □

Lemma 9.56 ($\widetilde{S}$ move)**.** *Consider the t-th iteration. Let matrices $Z^{\mathrm{mid}}, Z^{\mathrm{new}} \in \mathbb{R}^{n\times n}$ be defined as in Definition 9.5. Let $g \in \mathbb{R}^n_+$ be a non-increasing vector, and let $\Phi_g : \mathbb{R}^{n\times n} \to \mathbb{R}_+$ be defined as in Definition 9.4.*

Let r_t denote the rank of the update matrices $V_1, V_2 \in \mathbb{R}^{n\times r_t}$ generated by Algorithm 43 in the t-th iteration. We have

$$\Phi_g(Z^{\mathrm{mid}}) - \Phi_g(Z^{\mathrm{new}}) \geq \frac{\epsilon_S}{10\log n} \cdot r_t \cdot g_{r_t}.$$

The proof is exactly the same as Lemma 9.32. Now, we are ready to prove Theorem 9.53.

Proof of Theorem 9.53. Combining Lemma 9.55 and Lemma 9.56, we have

$$\begin{aligned}\Phi_g(Z^{\mathrm{new}}) - \Phi_g(Z) &= (\Phi_g(Z^{\mathrm{mid}}) - \Phi_g(Z)) - (\Phi_g(Z^{\mathrm{mid}}) - \Phi_g(Z^{\mathrm{new}})) \\ &\leq (n/m)^{1/4} \cdot \|g\|_2 - \frac{\epsilon_S}{10\log n} \cdot r_t \cdot g_{r_t}.\end{aligned}$$

With an abuse of notation, we denote the matrix Z in the t-th iteration as $Z^{(t)}$. Since in the beginning $\Phi_g(Z^{(0)}) = 0$ and $\Phi_g(Z^{(T)}) \geq 0$, we have

$$\begin{aligned}
0 &\leq \Phi_g(Z^{(T)}) - \Phi_g(Z^{(0)}) \\
&\leq \sum_{t=1}^{T} (\Phi_g(Z^{(t)}) - \Phi_g(Z^{(t-1)})) \\
&\leq T \cdot (n/m)^{1/4} \cdot \|g\|_2 - \frac{\epsilon_S}{10 \log n} \cdot \sum_{t=1}^{T} r_t \cdot g_{r_t}.
\end{aligned}$$

This completes the proof. $\square$

Corollary 9.57. *Given a sequence $r_1, \ldots, r_T \in [0, n]$ that satisfies Eq. (9.41). We have*

$$\sum_{t=1}^{T} \mathcal{T}_{\mathrm{mat}}(m^2, nr_t, m^2) \leq O^* \left(T \cdot (n/m)^{1/4} \cdot (n^2 m + m^{\omega} n^{1/4}) \right).$$

Proof. Let $a_t = \log_n(r_t)$ and $b = \log_n(m^2)$. Then

$$\mathcal{T}_{\mathrm{mat}}(m^2, nr_t, m^2) = \mathcal{T}_{\mathrm{mat}}(n^b, n^{1+a_t}, n^b) = O^*(n^{b \cdot \omega((1+a_t)/b)}).$$

For each $i \in \{0, 1, \ldots, \log n)\}$, define

$$T_i = \{t \in [T] : 2^i \leq r_t \leq 2^{i+1}\}.$$

Let $g_r = r^{-1/2}$, Theorem 9.53 indicates

$$\sum_{i=1}^{\log n} |T_i| \cdot 2^{i/2} \leq \sum_{t=1}^{T} r_t^{1/2} \leq O((n/m)^{1/4} \cdot T \cdot \log^{1.5} n).$$

This implies $|T_i| \leq O((n/m)^{1/4} \cdot T \cdot \log^{1.5}(n)/2^{i/2}$. It thus follows that

$$\begin{aligned}
\sum_{t=1}^{T} \mathcal{T}_{\mathrm{mat}}(m^2, nr_t, m^2) &\leq O^* \left(\sum_{t=1}^{T} n^{b \cdot \omega((1+a_t)/b)} \right) \\
&= O^* \left(\sum_{i=1}^{\log n} \sum_{t \in T_i} n^{b \cdot \omega((1+a_t)/b)} \right) \\
&\leq O^* \left(\max_{i \in \log n} \max_{t \in T_i} \frac{(n/m)^{1/4} \cdot T}{2^{i/2}} \cdot n^{b \cdot \omega((1+a_t)/b)} \right) \\
&\leq O^* \left((n/m)^{1/4} \cdot T \cdot \max_{a_t \in [0,1]} n^{-a_t/2 + b \cdot \omega((1+a_t)/b)} \right).
\end{aligned}$$

Since $\omega(\cdot)$ is a convex function (Fact 9.15),

$$\begin{aligned}\max_{a_t \in [0,1]} -a_t/2 + b \cdot \omega((1+a_t)/b) \leq & \max_{a \in \{0,1\}} -a/2 + b \cdot \omega((1+a)/b) \\ \leq & \max\{b+2, b\omega + 0.25\}.\end{aligned}$$

Combining the above inequalities, we have

$$\begin{aligned}\sum_{t=1}^{T} \mathcal{T}_{\mathrm{mat}}(m^2, nr_t, m^2) &\leq O^*\left((n/m)^{1/4} \cdot T \cdot n^{\max\{b+2, b\omega+0.25\}}\right) \\ &\leq O^*\left(T \cdot (n/m)^{1/4} \cdot (n^2 m + m^{\omega} n^{1/4})\right).\end{aligned}$$

This completes the proof. $\square$

9.12.6 Our result

Theorem 9.58 (Main result for Algorithm 45 - 46)**.** *Given symmetric matrices $C, A_1, \cdots, A_m \in \mathbb{R}^{n \times n}$, and a vector $b \in \mathbb{R}^m$. Define matrix $\mathsf{A} \in \mathbb{R}^{m \times n^2}$ by stacking the m vectors $\mathrm{vec}[A_1], \cdots, \mathrm{vec}[A_m] \in \mathbb{R}^{n^2}$ as rows. Consider the following SDP instance:*

$$\begin{aligned}\max_{X \in \mathbb{R}^{n \times n}} \quad & \langle C, X \rangle \\ \text{s.t.} \quad & \langle A_i, X \rangle = b_i, \ \forall i \in [m], \\ & X \succeq 0,\end{aligned}$$

Let X^* be an optimal solution of the SDP instance. There is a SDP algorithm (Algorithm 45-46) that runs in time

$$O^*\left((mn)^{1/4} \cdot \left(m^2 n^{\omega} + m^4\right) \cdot \log(1/\epsilon)\right)$$

and outputs a PSD matrix $X \in \mathbb{R}^{n \times n}$ s.t.

$$\langle C, X \rangle \geq \langle C, X^* \rangle - \epsilon \cdot \|C\|_2 \cdot R \quad \text{and} \quad \sum_{i=1}^{m} |\langle A_i, X \rangle - b_i| \leq 4n\epsilon \cdot \left(R \sum_{i=1}^{m} \|A_i\|_1 + \|b\|_1\right),$$

Remark 9.5. We improve the running time of [Ans00]

$$O^*((mn)^{1/4} \cdot (m^3 n^\omega + m^4 n^2 + m^{\omega+2}) \cdot \log(1/\epsilon)))$$

for all parameters regime.

In particular, if $m = n$, the total cost of Algorithm 45-46 can be upper bounded by $n^{\omega+2.5}$. This improves the $n^{6.5}$ total cost [Ans00].

If $m = n^2$, this cost can be upper bounded by $n^{8.75}$. This improves the $n^{10.75}$ total cost of [Ans00].

Proof. We first compute the running time. From Lemma 9.49, the amortized cost per iteration is upper bounded by

$$O^*\left(\left(\frac{n}{m}\right)^{\frac{1}{4}} \cdot (n^2 m + m^\omega n^{1/4}) + m^2 n^\omega + m^4 + m^2 \cdot n^{\omega - \frac{1}{2}} \cdot \left(\frac{n}{m}\right)^{\frac{1}{4}}\right).$$

Since $T = O\left((mn)^{1/4} \cdot \log(mn/\epsilon)\right)$, the Algorithm 45-46 run in time

$$O^*\left(\left(n^{\frac{5}{2}} m + m^{2+\frac{1}{4}} n^{\omega+\frac{1}{4}} + m^{4+\frac{1}{4}} n^{\frac{1}{4}} + m^\omega n^{\frac{3}{4}}\right) \cdot \log(1/\epsilon)\right).$$

Under current matrix multiplication exponent $\omega \approx 2.373$ and $0 \leq m \leq n^2$, this simplifies to

$$\begin{aligned} & O^*\left(\left(m^{2+\frac{1}{4}} n^{\omega+\frac{1}{4}} + m^{4+\frac{1}{4}} n^{\frac{1}{4}}\right) \cdot \log(1/\epsilon)\right) \\ = & O^*\left((mn)^{1/4} \cdot \left(m^2 n^\omega + m^4\right) \cdot \log(1/\epsilon)\right) \end{aligned}$$

Now we prove the correctness of Algorithm 45-46. We invoke the robust framework. From Fact 9.47 and Lemma 9.50, $\epsilon_g = \epsilon_\delta = 0$, and $c_H = 1/3 \cdot 1.0001^{-3}$. Therefore directly applying Theorem 9.46 for $\theta = (mn)^{1/4}$ completes the proof. $\square$

Chapter 10: High-Accuracy Quantum SDP Solver

10.1 Introduction

In this previous chapter (Chapter 9), we introduced a robust interior-point method (IPM) framework that supports approximating the gradient, Hessian, and Newton step in each IPM iteration. Based on this framework, we improve the running time of the state-of-the-art second-order SDP algorithm (due to [JKL⁺20]). As the quantum era is coming, it is natural to ask if we can use quantum computers to solve SDPs faster. In this chapter, we will use our robust IPM framework to obtain a quantum SDP algorithm that is faster than the currently best classical algorithm. And more importantly, it is the first quantum algorithm that can output high-accuracy solutions.

We first recall the definition of semi-definite programming (SDP) here:

Definition 10.1 (Semidefinite programming)**.** Suppose there are $m+1$ symmetric matrices $C \in \mathbb{R}^{n\times n}$ and $A_1, \cdots, A_m \in \mathbb{R}^{n\times n}$ and a vector $b \in \mathbb{R}^m$, the goal is to solve the following optimization problem:

$$\max_{X\in\mathbb{R}^{n\times n}} \langle C, X\rangle \text{ subject to } \langle A_i, X\rangle = b_i, \quad \forall i \in [m],\ X \succeq 0,$$

where $\langle A, B\rangle$ denote the inner product bween the matrix A and matrix B.

From the above formulation, the input size of an SDP instance is mn^2. The reason is, we have m constraint matrices and each matrix has size $n \times n$. The linear programming (LP) is a simpler case than SDP, where $X \succeq 0$ and $C, A_1, \cdots, A_m$ are restricted to be $n \times n$ diagonal matrices. Since [Dan47], there is a long line of work [Dan47, Kha80, Kar84, Ren88, Vai89b, LS14, LS15, CLS19, LSZ19, Bra20, BLSS20, SY21, JSWZ21, Ye20, DLY21] about speeding up the time complexity of linear programming. Semi-definite programming can be viewed as a more general optimization problem compared to linear programming. Semi-definite programming is a more challenging problem due to a variety of complications arising from generalizing vectors to PSD matrices. After a long line of research ([Sho77, YN76, Kha80, KTE88, NN89, Vai89a, NN92, NN94, Ans00, KM03, LSW15, JLSW20, JKL+20, HJS+22])[1], the state-of-the-art classical semi-definite programming algorithms are

- Jiang, Kathuria, Lee, Padmanabhan and Song [JKL+20]'s algorithm
  - It runs in $O(\sqrt{n}(mn^2 + m^\omega + n^\omega))$ time[2], for $m = \Omega(n)$.
- Huang, Jiang, Song, Tao and Zhang [HJS+22]'s algorithm (Chapter 9)
  - It runs in $O(m^\omega + m^{2+1/4})$ time, for $m = \Omega(n^2)$.

Inspired by the quantum linear algebra results such as recommendation system [KP17, Tan19, CGL+20], linear regression [GST22, CCH+22], principal component analysis [Tan21], non-convex optimization [SYZ21], in this chapter, we study the quantum algorithm for the semi-definite programming.

It is well-known that second-order optimization method usually achieves $\log(1/\epsilon)$ dependence in running time, while the first-order optimization method has to pay $1/\epsilon$.[3] In the quantum setting, for many linear algebra and optimization tasks (e.g., solving linear systems), there exist efficient algorithms taking $\log(1/\epsilon)$ time and outputting a quantum state encoding the solution. However, extracting classical information from the quantum state usually needs to pay a $\mathrm{poly}(1/\epsilon)$ factor. Therefore, even implementing second-order algorithm in quantum, it is unclear how to obtain $\log(1/\epsilon)$ dependence in final running time. Thus, one fundamental question is

[1]For a more detailed summary, we refer the readers to Table 1.1 and 1.2 in [JKL+20], and Table 1 in [HJS+22].

[2]$\omega \approx 2.372$ is the fast matrix multiplication exponent [AW21].

[3]For example, see [AK07, GH16, AZL17, CDST19, LP20a, YTF+19, JY11, ALO16].

Is there a quantum SDP algorithm that has a logarithmic dependence on ϵ while simultaneously improving the m, n-dependence of the best classical solvers?

In this chapter, we propose a new approach to answering this question. More specifically, for the $m = \Omega(n^2)$ case, we design a quantum SDP solver running in time $m^{\omega/2+1/4} \cdot \mathrm{poly}(\kappa, \log(1/\epsilon))$ and outputting a classical solution, which is faster than the existing classical SDP algorithms. Although the time complexity dependence in m, n of our algorithm is worse than the quantum second-order SDP algorithm [KP20b, KPS21], our algorithm has better error dependence than all the existing quantum SDP algorithms on the well-conditioned instances. We also overcome the infeasibility issue in previous quantum algorithms.

We state an informal version of the main result as follows and delay the formal version into Theorem 10.17.

Theorem 10.1 (Our result)**.** *For any semidefinite programming with m constraints and variable size $n \times n$, for any accuracy parameter ϵ, there is a quantum algorithm that outputs a classical solution in time $(mn^{1.5} + n^3) \cdot \mathrm{poly}(\kappa, \log(mn/\epsilon))$, where κ includes the condition numbers of the intermediate matrices.*

Note that almost all the previous quantum SDP algorithms also depend on these parameters. We also note that a large family of SDP instances exists with $\kappa = O(1)$. On these instances, our algorithm only pays $\log(1/\epsilon)$ in the running time, while the previous quantum algorithms still need to pay $1/\epsilon$.

10.2 Quantum Barrier with Existing Algorithms

The existing quantum SDP solving algorithms can be classified into two kinds: quantum first-order methods [BS17, AGGW17, BKL+19, AG19] based on the Arora-Kale framework [AK07], and quantum second-order method [KP20b, ANTZ21, KPS21] based on the primal-dual central path framework. Although these quantum algorithms have achieved quantum speedup over the classical SDP solvers in time complexity with respect to the parameters m and n, there are still some limitations in the existing quantum SDP solving algorithms, which we discuss below.

Error dependence: polynomial or logarithmic in ϵ? The approximation error ϵ is an essential parameter for SDP solving algorithms. For the quantum first-order algorithms, they are based on the classical first-order method, which requires $\Omega(1/\epsilon)$ iterations [Bub15, CDHS19]. And these quantum algorithms cannot reduce the number of iterations. And this barrier also exists in quantum due to the $\Omega(\sqrt{m}/\epsilon)$ lower bound by [AG19]. For the quantum second-order algorithm [KP20b], they achieved $\log(1/\epsilon)$ in their running time, which is indeed the advantage of the classical second-order method over the first-order method. However, the output of their algorithm is only guaranteed to be close to the feasible region, and the running time depends polynomially on the inverse of distance to the feasible region. Hence, the error dependence of their algorithm is hard to specify[4].

Linear-algebra computation: classical or quantum? In the classical interior-point methods, each iteration employs several linear-algebra computations. Some of them, like inverting a matrix or matrix-vector multiplication, can be sped up via quantum linear-algebra techniques, e.g., the linear combination of unitaries (LCU) or the quantum singular value transformation (QSVT). However, there remain some operations that may not be computed in this way, e.g., flattening a matrix as a vector or stacking vectors as a matrix. Therefore, previous quantum algorithms [KP20b, ANTZ21, KPS21] use the state tomography to transform quantum data to classical data, incurring a $1/\epsilon$ factor to achieve an ϵ error. Even worse, although quantum algorithms can solve the Newton system in poly-logarithmic time, the classical solution obtained by tomography is only an inexact solution of the linear system. This

[4]See page 9 in [AGGW17] for more discussions.

will significantly affect the convergence of the IPM iterations, and the previous quantum second-order methods [KP20b, ANTZ21, KPS21] only output *approximately-feasible* SDP solution.

Nontrivial symmetrization. For the feasible primal-dual central path method, the matrices X and S are required to be symmetric; otherwise, the convergence analysis cannot go through. Direct application of Newton's method into the central path leads to the following linear system:

$$\begin{cases} \mathrm{d}S \cdot X + S \cdot \mathrm{d}X = \nu I - SX & \text{(complementary slackness)} \\ \langle \mathrm{d}X, A_i \rangle = 0 \quad \forall i \in [m] & \text{(primal feasibility)} \\ \mathrm{d}S \in \mathrm{span}\{A_1, \dots, A_m\} & \text{(dual feasibility)} \end{cases} . \tag{10.1}$$

However, this system, in general, yields nonsymmetric directions in the primal variable $\mathrm{d}X$ (see discussions on page 2 in [MT00]). Hence, for the classical algorithms, in addition to solving the Newton linear system (Eq. (10.1)), some nontrivial procedures should be applied to symmetrize the solution $\mathrm{d}X$. (See Section 2 in [AHO98] and Section 1 in [MT00] and Section 4.5.1 in [BTN01] for more details.)[5] However, [KP20b] does not adopt these procedures.[6] Moreover, since the tomography algorithm they used can only output some ℓ_2-approximations of the matrices $\mathrm{d}S$ and $\mathrm{d}X$, it will make both X and S nonsymmetric.

10.3 Related Work

Table 10.1 summarizes the existing quantum SDP solvers. The quantum first-order method algorithms [BS17, AGGW17, BKL$^+$19, AG19] is built on the Arora-

[5]Notice that in Linear Programming X and S are both diagonal matrices, thus the solution of Eq.(10.1) is directly feasible. The standard primal-dual central path method can be applied ([CLS19]).

[6]In their followup work [KPS21], they use a correct symmetrization in their quantum second-order cone optimization algorithm.

Kale framework [AK07] for SDP-solving via Multiplicative Weights Update (MWU) algorithm. The main observation is that the matrix

$$\rho := \frac{\exp(-\sum_{i=1}^{m} y_i A_i)}{\operatorname{tr}[\exp(-\sum_{i=1}^{m} y_i A_i)]}$$

used in the multiplicative weights update is indeed a quantum Gibbs state, and hence a quantum Gibbs sampler can give speedup in n. The remaining part of the Arora-Kale framework is to find the violated constraints and update the dual solution y. Hence, they used some Grover-like techniques to achieve further quantum speedup in m. The quantum second-order method algorithm by Kerenidis and Prakash [KP20b] is based on the primal-dual central path framework, which requires solving a Newton linear system in each iteration. [KP20b] used the quantum linear system solver of [GSLW19] to solve the Newton linear system in quantum and applied the tomography in [KP17] to obtain the classical solution.

For comparison, classical second-order SDP solvers are summarized in Table 10.2.

10.4 Technical Overview

To obtain a truly second-order quantum algorithm for solving SDP (Theorem 10.1), we overcome the barriers stated in Section 10.2 via a block encoding-based interior point method combined with our robust framework. In the following subsections, we will first describe our quantum algorithm and then demonstrate how it bypasses each barrier.

We define several notations related to the number of iterations of our algorithm.

Definition 10.2. We choose η as follows: $\eta := \frac{1}{n+2}$. We choose T as follows $T := \frac{40}{\epsilon_N}\sqrt{n}\log(\frac{n}{\epsilon})$, where $\epsilon_N \in (0, 10^{-1})$ is a constant.

References	Method	Input Model	Output	Time Complexity
[BS17]	1st. order	Sparse oracle	Feasible sol.	$\sqrt{mn}s^2\epsilon^{-32}$
[AGGW17]	1st. order	Sparse oracle		$\sqrt{mn}s^2\epsilon^{-8}$
[BKL+19]	1st. order	Sparse oracle	Decide feasibility	$\sqrt{m}s^2\epsilon^{-10} + \sqrt{n}s^2\epsilon^{-12}$
		Quantum state		$\sqrt{m}\epsilon^{-\Omega(1)}$
[AG19]	1st. order	Sparse oracle	Feasible sol.	$\sqrt{m}s\epsilon^{-4} + \sqrt{n}s\epsilon^{-5}$
		Quantum state		$\sqrt{m}\epsilon^{-4} + \epsilon^{-7.5}$
		QRAM		$\sqrt{m}\epsilon^{-4} + \sqrt{n}\epsilon^{-5}$
[KP20b, KPS21]	2nd. order	QRAM	Infeasible sol.	$n^{2.5}\mu\xi^{-2}\log(1/\epsilon)$ $(m = n^2)$
[ANTZ21]	2nd. order	QRAM	Feasible sol.	$n^{3.5}\epsilon^{-1} + n^4$ $(m = n^2)$
Ours (Thm. 10.1)	2nd. order	QRAM	Feasible sol.	$n^{3.5}\log(1/\epsilon)$ $(m = n^2)$

Table 10.1: Quantum algorithms for solving SDP. ϵ is the additive error of the output solution. s is the row-sparsity of the input matrices. ξ is the approximation feasibility. The sparse input oracle supports querying the i-th nonzero entry in the j-th row of an input matrix in superposition. The quantum state model splits each input matrix A_i as a difference of PSD matrices (quantum states) and we can access to the density matrix of the state. The QRAM model assumes that the input matrices are stored in some data structures in QRAM that support efficient quantum access to the matrix. Infeasible sol. means the algorithm only outputs a solution that can approximately satisfy the SDP constraints.

10.4.1 Block encoding-based interior point method

The framework of our quantum algorithm is the same as the classical algorithm in [HJS+22]. We notice that the most time-consuming steps involve matrix computations. Therefore, it is very natural to apply the recent block encoding and quantum linear algebra framework to speed up the interior point method.

10.4.1.1 A brief overview of quantum linear algebra

We provide the definition of the block encoding of a matrix.

Definition 10.3 (Block encoding, informal)**.** Let A be a matrix. We say a unitary matrix U is a block encoding of A if top-left block of U_A is close to A up to some

References	Method	Time Complexity
[Sho77, YN76, Kha80]	CPM	n^8
[KTE88, NN89]	CPM	n^9
[Vai89a]	CPM	$n^{6.746}$
[NN92]	IPM	$n^{6.5}$
[NN94, Ans00]	IPM	$n^{10.75}$
[KM03]	CPM	$n^{6.746}$
[LSW15]	CPM	n^6
[JLSW20]	CPM	n^6
[JKL+20]	IPM	$n^{5.246}$
[HJS+22]	IPM	$n^{4.746}$

Table 10.2: Classical algorithms for solving SDPs. Total running times are calculated for the regime $m = n^2$, where n is the size of matrices, and m is the number of constraints. CPM denotes the cutting plane method, and IPM denotes the interior point method. The running times shown in the table hide $n^{o(1)}$, $m^{o(1)}$ and $\operatorname{poly}\log(1/\epsilon)$ factors, where ϵ is the accuracy parameter. In this table, we use the current best known upper bound of $\omega \approx 2.373$. This gives $n^{4.476} = n^{2\omega} = m^{\omega}$.

Algorithm 47 Here, we briefly state an informal version of our algorithm. We put more details in later Algorithm 49. The input A has size $m \times n^2$. The input vector b has length m. And the input matrix C has size n by n.

```
1: procedure QSolveSDP($\mathsf{A}, b, C$)
2:                                   ▷ The input $\mathsf{A}, C, b$ are stored in QRAM
3:     Choose $\eta$ and $T$ according to Definition 10.2.
4:     Find $y$ and store in QRAM
5:     for $t = 1 \to T$ do
6:         $\eta^{\text{new}} \leftarrow \eta \cdot (1 + \frac{\epsilon_N}{20\sqrt{n}})$
7:         Compute $\widetilde{S}^{-1}$ using QSVT and $\ell_2$-tomography    ▷ Step 1. of Sec. 10.4.1
8:         Compute $|g_\eta\rangle$ and estimate the norm $\|g_\eta\|_2$    ▷ Step 2. of Sec. 10.4.1
9:         Compute $\widetilde{\delta}_y$ using quantum linear system solver and tomography  ▷ Step
   3. of Sec. 10.4.1
10:        $y^{\text{new}} \leftarrow y + \widetilde{\delta}_y$
11:    end for
12: end procedure
```

scaling, i.e.,

$$U_A \approx \begin{bmatrix} A/\alpha & \cdot \\ \cdot & \cdot \end{bmatrix}$$

Via block encoding, we can transform a classical matrix to a quantum operator, which enables us to obtain exponential speedup in many linear algebra tasks by the quantum singular value transformation technique [GSLW19].

Suppose we can efficiently implement the exact block encoding of a matrix A, and we can also efficiently prepare the vector state $|x\rangle$. Then, for matrix-vector multiplication, the state $|Ax\rangle = \frac{1}{\|Ax\|_2} \sum_{i=1}^{n} (Ax)_i |i\rangle$ can be ϵ-approximated in time $\widetilde{O}(\log(1/\epsilon))$.[7] For linear system solving, the state $|A^{-1}x\rangle$ can also be ϵ-approximated in time $\widetilde{O}(\log(1/\epsilon))$ [CGJ19]. In addition, their norms $\|Ax\|_2$ and $\|A^{-1}x\|_2$ could be approximated with ϵ-multiplicative error. And it takes $\widetilde{O}(1/\epsilon)$ time.

Input model Our algorithm uses the QRAM (Quantum Random Access Memory) model, which is a common input model for many quantum algorithms (e.g., [KP17, AG19, KP20b]). More specifically, a QRAM stores the classical data and supports querying in superposition. And each update operation for size S QRAM data will take $\widetilde{O}(S)$ time. Furthermore, it has been shown that a QRAM can be extended a quantum data structure for storing matrices such that the exact block encoding can be prepared $\widetilde{O}(1)$ time. For the SDP inputs A, C, we assume that they are already loaded into the QRAM such that their block encodings can be efficiently implemented.

10.4.1.2 Main steps of one iteration

We will show how to use the block encoding and quantum linear algebra tools to implement [HJS+22] general robust barrier method for SDP and achieve quantum speedup.

[7]For simplicity, we hide the dependence of $\kappa(A)$ and $\mu(A)$, which is related to the singular values of A.

Step 1: implement ApproxSlackInverse The slack matrix S is defined by

$$S(y) := \sum_{i=1}^{m} y_i A_i - C,$$

where the length-m vector y is the dual solution outputted by the previous iteration and we assume that it is stored in QRAM. Since we assume quantum access to A_i, C and y, by the linear combination of block-encodings, we can efficiently prepare the block-encoding of S. Then, for each $i \in [n]$, by the quantum linear system solver, we can prepare $|S^{-1}e_i\rangle = |(S^{-1})_i\rangle$ and also estimate the norm $\|(S^{-1})_i\|_2$ by $\overline{\|(S^{-1})_i\|_2}$. Then, we apply the ℓ_2-tomography procedure in [vACGN22] to obtain a classical vector $v_i \in \mathbb{R}^n$ that is close to $|(S^{-1})_i\rangle$. Hence, by defining $(\widetilde{S}^{-1})_i := \overline{\|(S^{-1})_i\|_2} \cdot v_i$, it holds that $\|(\widetilde{S}^{-1})_i - (S^{-1})_i\|_2 \le \epsilon_S \|(S^{-1})_i\|_2$. We repeat this procedure for all $i \in [n]$ and obtain a classical matrix $\widetilde{S}^{-1} \in \mathbb{R}^{n \times n}$ such that $\|\widetilde{S}^{-1} - S^{-1}\|_F \le \epsilon_S \|S^{-1}\|_F$. To make $\widetilde{S}$ symmetric, we further let $\widetilde{S}^{-1} \leftarrow \frac{1}{2}(\widetilde{S}^{-1} + (\widetilde{S}^{-1})^\top)$. The running time of this step is

$$\mathcal{T}_S = \widetilde{O}(n^{2.5}\epsilon_S^{-1}),$$

which is an improvement over the classical running time $O(mn^2 + n^\omega)$ for constant ϵ_S.

Step 2: implement ApproxGradient The gradient vector g is defined as

$$g_\eta := \eta b - \mathsf{A} \cdot \mathrm{vec}(S^{-1}).$$

where $\mathrm{vec}(\cdot)$ means vectorizing an n-by-n matrix into an n^2-length vector, and $\mathsf{A} \in \mathbb{R}^{m \times n^2}$. Note that the gradient is only used to compute $\delta_y := -H^{-1} g_\eta$ in Step 3. Thus, it suffices to to prepare the state $|g_\eta\rangle$ and estimate its norm $\|g_\eta\|_2$. Using QRAM, we can efficiently implement the block-encoding for $\mathsf{A}' = \begin{bmatrix} \mathsf{A} & b \end{bmatrix}$. Since we compute $\widetilde{S}^{-1}$ in the classical form in Step 1, we can also efficiently prepare the state

$|s'\rangle$ for $s' = \begin{bmatrix} -\mathrm{vec}(\widetilde{S}^{-1}) \\ \eta \end{bmatrix}$. By quantum linear algebra, we can prepare a state $|\widetilde{g}_\eta(\widetilde{S})\rangle$ that is ϵ_g-close to $|g_\eta(\widetilde{S})\rangle$ in time

$$\mathcal{T}_{g,1} = \widetilde{O}(\mu(\mathsf{A})\kappa(\mathsf{A})) \simeq \widetilde{O}(\sqrt{m}),$$

where $g_\eta(\widetilde{S}) = \eta \cdot b - \mathsf{A} \cdot \mathrm{vec}(\widetilde{S}^{-1})$. And we obtain an estimate $\overline{\|g_\eta(\widetilde{S})\|_2}$ of $\|g_\eta(\widetilde{S})\|_2$ within relative error ϵ_g in time

$$\mathcal{T}_{g,2} = \widetilde{O}(\mu(\mathsf{A})\kappa(\mathsf{A})\epsilon_g'^{-1}) \simeq \widetilde{O}(\sqrt{m}\epsilon_g'^{-1}).$$

Furthermore, if we define $\widetilde{g}_\eta(\widetilde{S}) := \overline{\|g_\eta(\widetilde{S})\|_2} \cdot |\widetilde{g}_\eta(\widetilde{S})\rangle$, then we can prove that

$$\|\widetilde{g}_\eta(\widetilde{S}) - g_\eta\|_2 \lesssim O(\epsilon_g + \epsilon_g' + \epsilon_S)\|g_\eta\|_2,$$

where we use the error guarantee of $\widetilde{S}^{-1}$.

Step 3: implement ApproxDelta In this step, we will compute $\delta_y := -H^{-1} \cdot g_\eta$, where $H = \mathsf{A} \cdot (S^{-1} \otimes S^{-1}) \cdot \mathsf{A}^\top$. Since A and $\widetilde{S}^{-1}$ are stored in QRAM, using the product and Kronecker product of block-encodings [GSLW19, CVB20], we can implement the block-encoding of $\widetilde{H} = \mathsf{A} \cdot (\widetilde{S}^{-1} \otimes \widetilde{S}^{-1}) \cdot \mathsf{A}^\top$. In order to reduce the block-encoding factor of $\widetilde{H}$, we first implement the block-encoding of $\widetilde{S}^{-1/2} \otimes \widetilde{S}^{-1/2}$, by classically compute $\widetilde{S}^{-1/2} = (\widetilde{S}^{-1})^{1/2}$. Then, we implement the block-encoding of $W = \mathsf{A} \cdot (\widetilde{S}^{-1/2} \otimes \widetilde{S}^{-1/2})$. Since $\widetilde{H} = WW^\top$, the block-encoding of $\widetilde{H}$ can be efficiently implemented. In Step 2, $|\widetilde{g}_\eta(\widetilde{S})\rangle$ are prepared. Then, we can apply the quantum linear system solver to obtain the state close to $|\widetilde{H}^{-1}\widetilde{g}_\eta(\widetilde{S})\rangle$ and also estimate the norm $\|\widetilde{H}^{-1}\widetilde{g}_\eta(\widetilde{S})\|_2$. Then, we apply the tomography to obtain the classical form of $\widetilde{H}^{-1}\widetilde{g}_\eta(\widetilde{S})$. Let $\widetilde{\delta}_y$ denote the classical vector we compute in this step. We can show that

$$\|\widetilde{\delta}_y - (-\widetilde{H}^{-1}\widetilde{g}_\eta(\widetilde{S}))\|_2 \leq \epsilon_\delta \|\widetilde{H}^{-1}\widetilde{g}_\eta(\widetilde{S}))\|_2$$

in time

$$\mathcal{T}_\delta = \widetilde{O}(m\mu(\mathsf{A})\epsilon_\delta^{-1} + m\mu(S)\epsilon_\delta^{-1}) \simeq \widetilde{O}(mn\epsilon_\delta^{-1}),$$

where we use the fact that $\mu(\mathsf{A}) \leq \sqrt{m+n^2} = O(n)$ (see Theorem 10.6).

Running time per iteration Putting three steps together, we get that the total running time per iteration of our algorithm is

$$\mathcal{T}_{\text{iter}} = \mathcal{T}_S + \mathcal{T}_{g,2} + \mathcal{T}_\delta \simeq \widetilde{O}(n^{2.5}\epsilon_S^{-1} + \sqrt{m}\epsilon_g^{-1} + mn\epsilon_\delta^{-1}).$$

10.4.2 Overcoming the quantum barriers

In this part, we discuss how our quantum algorithm bypasses each barrier in Section 10.2.

Error dependence barrier We first note that by the analysis of our robust interior-point framework (Theorem 10.12), the number of iterations will be $\widetilde{O}(\sqrt{n}\log(1/\epsilon))$ as long as the Newton step size satisfies $g_\eta^\top H^{-1} g_\eta \le \epsilon_N^2$ for some constant ϵ_N in each iteration. As shown in Section 10.4.1.2, the running time per iteration is depends linearly on $\epsilon_S^{-1}, \epsilon_g^{-1}, \epsilon_\delta^{-1}$. Fortunately, by the recently proposed robust framework (Lemma 10.11), constant accuracies[8] are enough:

Lemma 10.2 (Quantum time cost per iteration, informal)**.** *We can take some properly chosen $\epsilon_S, \epsilon_g, \epsilon_\delta$ such that **Condition 0-4** in the robust IPM framework (Lemma 10.11) are satisfied in each iteration, and the runtime per iteration is $\widetilde{O}(m^{1.5} + n^{2.5})$.*

Therefore, the total running time of our quantum SDP solving algorithm is

$$\widetilde{O}(\sqrt{n}(mn + n^{2.5})\log(1/\epsilon)).$$

It depends only logarithmically on the approximation error, which is a truly second-order algorithm.

Quantum/classical data transformation barrier We can output the classical solution by running tomography in intermediate steps. For time-consuming steps,

[8]They actually depend on the condition number of the matrices. See Section 10.7.4 for details.

we use the QSVT to speed up. And for the steps that are hard to be implemented in the QSVT (e.g., vectorizing $\widetilde{S}^{-1}$), we directly compute them classically with low-accuracy tomography procedure. In this way, the robust IPM framework can tolerate the "quantum-inherent" error, and our quantum algorithm can still output *feasible, high-accuracy* SDP solution.

Symmetrization barrier We bypass the barrier by solving SDP in the dual space instead of the primal space, where we only care about the symmetrization of $\widetilde{S}$, the approximation of the slack matrix S. It is easy to make $\widetilde{S}$ symmetric by averaging $\widetilde{S}$ and $\widetilde{S}^\top$. This is because the definition of S guarantees that the true S is symmetric. Hence, the classical interior point method does not have this problem, and our symmetrization will only make the approximation error smaller. However, we note that this symmetrization trick cannot directly apply to [KP20b]'s algorithm. Because they solved SDP also in the primal space and the true solution $\mathrm{d}X$ of the newton linear system may not be symmetric, which means averaging $\widetilde{\mathrm{d}X}$ and $(\widetilde{\mathrm{d}X})^\top$ could result in some error that is hard to control. A possible way may be to quantize the classical symmetrization procedures given in [AHO98].

10.5 Preliminary

We define several basic notations here. We say $\mathrm{tr}[\cdot]$ is the trace of a matrix.

We use $\|\cdot\|_2$ to denote spectral/operator norm of a matrix. Let $\|\cdot\|_F$ be the Frobenious norm of a matrix. We use $\|\cdot\|_1$ to represent Schatten 1-norm of matrix.

For a symmetric matrix $X \in \mathbb{R}^{n\times n}$, we say it is positive semi-definite (PSD, denoted as $X \succeq 0$) if for any vector $u \in \mathbb{R}^n$, $u^\top X u \geq 0$. Similarly, we define positive definite via $\succ$ and > 0 notations.

We say $\lambda(B)$ are the eigenvalues of B.

We use $x_{[i]}$ to denote the i-th largest entry of vector x.

We use vec[] to denote matrix vectorization.

Let $\otimes$ denote the Kronecker product.

The $\mathsf{A} \in \mathbb{R}^{m \times n^2}$ is a matrix where i-th row is $\mathrm{vec}[A_i]$, for all $i \in [m]$.

Let $\mathcal{T}_{\mathrm{mat}}(x, y, z)$ denote the time of multiplying an $x \times y$ matrix with another $y \times z$ matrix.

For $r \in \mathbb{R}_+$, let $\omega(r) \in \mathbb{R}_+$ denote the value that for all positive integer m, $\mathcal{T}_{\mathrm{mat}}(m, m, m^r) = O(m^{\omega(r)})$.

Definition 10.1 is the primal form of SDP. The dual form of SDP is defined in Definition 9.2. For convenience, we recall it here:

Definition 10.4 (The dual formulation of SDP)**.** Suppose we are given a symmetric matrix $C \in \mathbb{R}^{n \times n}$. There are also m constraints matrices $A_1, \ldots, A_m$, and each of them has size n by n. Suppose we are also given a length-m vector b. Our goal is to solve this problem:

$$\begin{aligned} \min_{y \in \mathbb{R}^m} \ & \langle b, y \rangle \\ \text{s.t. } S = & \sum_{i=1}^m y_i A_i - C, \\ S \succeq & \ 0. \end{aligned} \tag{10.2}$$

We state two tools from previous work.

Theorem 10.3 ([Wed73, MZ10])**.** *Let $A \in \mathbb{R}^{m \times n}$ and $B = A + E$. Let $A^\dagger$ and $B^\dagger$ denote the pseudo-inverse of A and B, respectively. Then*

$$\|B^\dagger - A^\dagger\| \leq \sqrt{2} \max\{\|A^\dagger\|_2^2, \|B^\dagger\|_2^2\} \cdot \|E\|.$$

Theorem 10.4 ([Wed73, MZ10])**.** *Let $A \in \mathbb{R}^{m \times n}$ and $B = A + E$. Then*

$$\|B^\dagger - A^\dagger\|_F \leq \mu \max\{\|A^\dagger\|_2^2, \|B^\dagger\|_2^2\} \cdot \|E\|_F.$$

where $\mu = 1$ if $\mathrm{rank}(A) = n = m$ and $\mu = \sqrt{2}$ otherwise.

10.5.1 Quantum linear algebra toolbox

We introduce the basics of quantum computing in Appendix B. In this section, we introduce a powerful tool in quantum algorithm design—quantum linear algebra, which stores classical information in quantum states and exponentially speeds up some fundamental linear algebra computations.

10.5.1.1 Storing data in QRAM

Quantum random access memory (QRAM) is a commonly-used model in quantum computing that assumes the quantum computer can access classical data in superposition.

Definition 10.5 (Quantum random access memory (QRAM), [GLM08])**.** A quantum random access memory is a device that stores indexed data (i, x_i) for $i \in \mathcal{N}$ and $x_i \in \mathbb{R}$ (with some bit precision). It allows to query in the form $|i\rangle|0\rangle \mapsto |i\rangle|x_i\rangle$. Each read/write/update operation has $\widetilde{O}(1)$ cost.

Furthermore, there are some ways to store classical vectors and matrices in QRAM using some quantum data structures developed in [KP17], such that for any vector or any row of a matrix, the vector state

$$|v\rangle := \frac{1}{\|v\|_2}\sum_{i=0}^{n-1} v_i|i\rangle$$

can be prepared in polylog(n) time. In addition, for a matrix A, its row norm state $\sum_{i=0}^{n} \|A_i\|_2|i\rangle$ can also be prepared in polylog(n) time.

We now introduce a commonly used matrix parameter in quantum linear algebra:

Definition 10.6 (Matrix parameter for QRAM)**.** For a matrix $A \in \mathbb{R}^{n\times m}$, for $p \in [0,1]$, let $s_1(A) := \max_{i\in[n]}\sum_{j\in[m]}|A_{i,j}|$. Then, we define a parameter $\mu(A)$ as follows:

$$\mu(A) := \|A\|^{-1} \cdot \min\left\{\|A\|_F, s_1(A)\right\}.$$

10.5.1.2 Retrieving data from quantum to classical

In order to transform a quantum vector state $|v\rangle$ to a classical vector, we need to apply the state tomography procedure [KP20b, KLP19, vACGN22]. We state a version with an ℓ_2-approximation guarantee.

Theorem 10.5 (Vector state tomography with ℓ_2 guarantee, [vACGN22, Theorem 23]). *Given access to a unitary U such that $U|0\rangle = |x\rangle = \sum_{i=0}^{n-1} x_i|i\rangle$ for some $x \in \mathbb{R}^n$, there is a tomography algorithm that outputs a vector $\widetilde{x} \in \mathbb{R}^d$ with probability $1 - 1/\mathrm{poly}(d)$ using $\widetilde{O}(d/\epsilon)$ conditional applications of U and its inverse and $\widetilde{O}(d/\epsilon)$ additional quantum gates such that $\|x - \widetilde{x}\|_2 \leq \epsilon$.*

10.5.1.3 Linear algebra operations with block encodings

Block encoding is a way to efficiently transform a classical matrix to a quantum operator that enables quantum computers to speedup several linear algebra computations.

The definition of block encoding is as follows:

Definition 10.7 (Block encoding). Let $A \in \mathbb{C}^{2^w \times 2^w}$ be a matrix. We say a unitary matrix $U \in \mathbb{C}^{2^{(w+a)} \times 2^{(w+a)}}$ is a (α, a, ϵ)block encoding of A if

$$\|A - \alpha(\langle 0|^a \otimes I)U(|0\rangle^a \otimes I)\| \leq \epsilon,$$

i.e.,

$$U_A \approx \begin{bmatrix} A/\alpha & \cdot \\ \cdot & \cdot \end{bmatrix}.$$

Kerenidis and Prakash [KP17, KP20a] showed that for a matrix stored in a quantum data structure, its block encoding can be efficiently implemented.

Theorem 10.6 (Block encoding construction from QRAM, [KP17, KP20a]). *Given quantum access to a matrix $A \in \mathbb{R}^{n\times n}$, a $(\mu(A), O(\log(n)), 0)$-block encoding of A can* *be implemented in $\widetilde{O}(1)$ time, where $\mu(A)$ is defined in Definition 10.6, and a naïve bound is*

$$\mu(A) \leq \sqrt{\operatorname{rank}(A)} \leq \sqrt{n}.$$

We state a tool from previous work [GSLW19].

Theorem 10.7 (Product of block encoded matrices, Lemma 53 in [GSLW19])**.** *We use U_A to represent an (α, a, ϵ_A)-block encoding of a matrix A which can be constructed in time T_A, and U_B be a (β, b, ϵ_B)-block encoding of a matrix that can be constructed in time T_B.*

Then, we can implement an $(\alpha\beta, a+b, \alpha\epsilon_B + \beta\epsilon_A)$-block encoding of AB in time $O(T_A + T_B)$.

Lemma 10.8 (Product of preamplified block-matrices [LC19])**.** *Let $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times k}$ such that $\|A\| \leq 1, \|B\| \leq 1$. If $\alpha \geq 1$ and U is an (α, a, δ)-block-encoding of A that can be implemented in time T_U; $\beta \geq 1$ and V is a (β, b, ϵ)-block-encoding of B that can be implemented in time T_V, then there is a $(2, a+b+2, \sqrt{2}(\delta+\epsilon+\gamma))$-block-encoding of AB that can be implemented in time $O((\alpha(T_U+a)+\beta(T_V+b))\log(1/\gamma))$.*

We state a tool from previous work [CVB20].

Theorem 10.9 (Kronecker product of block encoded matrices, Lemma 1 in [CVB20])**.** *We use U_A to represent an (α, a, ϵ_A)-block encoding of a matrix A which can be constructed in time T_A, and U_B be a (β, b, ϵ_B)-block encoding of a matrix that can be constructed in time T_B.*

Then, we can implement an $(\alpha\beta, a+b, \alpha\epsilon_B + \beta\epsilon_A + \epsilon_A\epsilon_B)$-block encoding of $A \otimes B$ in time $O(T_A + T_B + \log(n))$.

Theorem 10.10 (Quantum linear system solver, [CGJ19, GSLW19])**.** *Let $A \in \mathbb{R}^{n\times n}$ be a matrix with non-zero eigenvalues in the interval $[-1, -1/\kappa] \cup [1/\kappa, 1]$. Given an implementation of an $(\mu, O(\log n), \delta)$ block encoding for A in time T_U and a procedure for preparing state $|b\rangle$ in time T_b,*

1. *If* $\delta \leq \frac{\epsilon}{\kappa^2 \mathrm{poly}\log(\kappa/\epsilon)}$ *then a state* ϵ*-close to* $|A^{-1}b\rangle$ *can be generated in time*
$$(T_U \kappa\mu + T_b \kappa) \cdot \mathrm{poly}\log(\kappa\mu/\epsilon).$$

2. *If* $\delta \leq \frac{\epsilon}{2\kappa}$ *then a state* ϵ*-close to* $|Ab\rangle$ *can be generated in time*
$$(T_U \kappa\mu + T_b \kappa) \cdot \mathrm{poly}\log(\kappa\mu/\epsilon).$$

3. *For* $\epsilon > 0$ *and* δ *as in parts 1 and 2 and* $\mathcal{A} \in \{A, A^{-1}\}$*, an estimate* Λ *such that* $\Lambda \in (1 \pm \epsilon)\|\mathcal{A}b\|$ *with probability* $(1-\delta)$ *can be generated in time*
$$(T_U + T_b)(\kappa\mu/\epsilon) \cdot \mathrm{poly}\log(\kappa\mu/\epsilon).$$

10.6 Revisit of Robust Newton Step

In this chapter, one of our contributions is that we provide a more robust version of the potential function that considers the perturbation of gradients.

Classical interior point literature [Ren01] controls the potential function when the exact Newton step is taken. [JKL+20] introduces small errors in the framework. Recently, [HJS+22] provides the most general framework, which is introduced in Chapter 9. We slightly modify it to adapt the error analysis in quantum computing.

Lemma 10.11 (Robust IPM iteration, a restatement of Lemma 9.40)**.** *Let* $c_0 = 10^{-4}$*. Given any parameters*

- $\alpha_S \in [1, 1 + c_0]$,
- $\alpha_H \in [1, 1 + c_0]$,
- $\epsilon_g \in [0, c_0]$,
- $\epsilon_\delta \in [0, c_0]$,
- $\epsilon_N \in [0, c_0]$,

- $\eta > 0$.

We assume

- **Condition 0.** *A dimension-m vector $y \in \mathbb{R}^m$ (feasible dual solution) satisfies*

$$\|g(y,\eta)\|_{H(y)^{-1}} := g(y,\eta)^\top H(y)^{-1} g(y,\eta) \leq \epsilon_N.$$

- **Condition 1.** *A $n \times n$ symmetric (positive definite) matrix $\widetilde{S}$ satisfies*

$$\alpha_S^{-1} \cdot S(y) \preceq \widetilde{S} \preceq \alpha_S \cdot S(y).$$

- **Condition 2.** *A $n \times n$ symmetric (positive definite) matrix $\widetilde{H}$ satisfies*

$$\alpha_H^{-1} \cdot H(\widetilde{S}) \preceq \widetilde{H} \preceq \alpha_H \cdot H(\widetilde{S}).$$

- **Condition 3.** *A dimension-m vector $\widetilde{g}$ satisfies*

$$\|\widetilde{g} - g(y, \eta^{\mathrm{new}})\|_{H(y)^{-1}} \leq \epsilon_g \cdot \|g(y, \eta^{\mathrm{new}})\|_{H(y)^{-1}}.$$

- **Condition 4.** *A dimension-m vector $\widetilde{\delta}_y$ satisfies*

$$\|\widetilde{\delta}_y - (-\widetilde{H}^{-1}\widetilde{g})\|_{H(y)} \leq \epsilon_\delta \cdot \|\widetilde{H}^{-1}\widetilde{g}\|_{H(y)}.$$

Then following the update rule of η and y: $\eta^{\mathrm{new}} = \eta \cdot (1 + \frac{\epsilon_N}{20\sqrt{n}})$ and $y^{\mathrm{new}} = y + \widetilde{\delta}_y$, we have y^{new} and η^{new} satisfy the following:

$$\|g(y^{\mathrm{new}}, \eta^{\mathrm{new}})\|_{H(y^{\mathrm{new}})^{-1}} \leq \epsilon_N.$$

Remark 10.1. **Condition 3 and 4** can be derived by the following ℓ_2 guarantees:

- $\|\widetilde{g} - g(y, \eta^{\mathrm{new}})\|_2 \leq \epsilon_g \cdot \|g(y, \eta^{\mathrm{new}})\|_2 / \kappa(H(y))$,
- $\|\widetilde{\delta}_y - (-\widetilde{H}^{-1}\widetilde{g})\|_2 \leq \epsilon_\delta \cdot \|\widetilde{H}^{-1}\widetilde{g}\|_2 / \kappa(H(y))$.

Based on the robust IPM framework, we can conclude the following convergence result for the barrier method, which is proved in Chapter 9. We restate the theorem below.

Theorem 10.12 (Robust barrier method, a restatement of Theorem 9.46)**.** *Consider a semidefinite program in Eq* (9.2)*. Suppose in each iteration, the* $\widetilde{S}, \widetilde{H}, \widetilde{g}, \widetilde{\delta}$ *computed in Line 6, Line 8, Line 7, Line 9 of Algorithm 48 satisfies Condition 1, 2, 3, 4 in Lemma 10.11.*

Let y *denote a dimension-*m *vector (can be viewed as a feasible initial solution). Suppose that* y *satisfies the invariant* $g(y,\eta)^\top H(y)^{-1} g(y,\eta) \leq \epsilon_N^2$*, for any error parameter* $0 < \epsilon \leq 0.01$ *and Newton step size* ϵ_N *satisfying* $\sqrt{\epsilon} < \epsilon_N \leq 0.1$*, Algorithm 48 outputs, in* $T = 40\epsilon_N^{-1}\sqrt{n}\log(n/\epsilon)$ *iterations, a vector* $y \in \mathbb{R}^m$ *that satisfies*

$$\langle b, y \rangle \leq \langle b, y^* \rangle + \epsilon^2. \tag{10.3}$$

where y^* *is an optimal solution to the dual formulation* (9.2)*.*

Further, consider the Algorithm 48, we have

$$\|S^{-1/2} S^{\mathrm{new}} S^{-1/2} - I\|_F \leq 1.1 \cdot \epsilon_N \tag{10.4}$$

holds for each iteration.

10.7 Quantum Second-Order SDP Solver

We propose a fast quantum second-order SDP solver in this section. The detailed algorithm is given in Algorithm 49. We follow the framework of the robust interior-point method (Algorihtm 48). In Section 10.7.1, we show the quantum implementation of the procedure ApproxSlack that directly computes the slack matrix inverse. In Section 10.7.2, we show how to implement the procedure ApproxGradient that computes the gradient in quantum. In Section 10.7.3, we show how to

Algorithm 48 A restatement of the framework (Algorithm 44).

```
1: procedure SDPFRAMEWORK($\{A_i\}_{i=1}^m, b, C, m, n$)    ▷ Let $C \in \mathbb{R}^{n \times n}$.
   Let $A_1, A_2, \cdots, A_m$ denote a list of $m$ matrices where each matrix has size $n \times n$.
   Let $b$ denote a length-$m$ vector. Let $\mathsf{A}$ denote a matrix that has size $m \times n^2$.
2:     Choose $\eta$ and $T$ according to Definition 10.2
3:     Find $y \in \mathbb{R}^m$ according to Lemma E.1    ▷ Condition 0 in Lemma 10.11
4:     for $t = 1 \to T$ do
5:         $\eta^{\text{new}} \leftarrow \eta \cdot (1 + \frac{\epsilon_N}{20\sqrt{n}})$
6:         $\widetilde{S} \leftarrow$ Approximate Slack    ▷ Condition 1 in Lemma 10.11
7:         $\widetilde{H} \leftarrow$ Approximate Hessian    ▷ Condition 2 in Lemma 10.11
8:         $\widetilde{g} \leftarrow$ Approximate Gradient    ▷ Condition 3 in Lemma 10.11
9:         $\delta_y \leftarrow$ Approximate Delta    ▷ Condition 4 in Lemma 10.11
10:        $y^{\text{new}} \leftarrow y + \delta_y$
11:        $y \leftarrow y^{\text{new}}$
12:    end for
13:    Return a solution via Lemma E.1
14: end procedure
```

implement the procedure APPROXDELTA that computes the Newton step in quantum. Then, in Section 10.7.4, we combine them together and show that the robustness conditions are satisfied, which gives the running time and correctness guarantees of Algorithm 49.

10.7.1 Slack matrix

The goal of this section is to prove the following lemma, which shows how to efficiently compute the slack matrix inverse S^{-1} using quantum singular value transformation (QSVT). We will show in Section 10.7.4 that the error guarantee in the Frobenius norm suffices for our robust IPM framework.

Lemma 10.13 (Quantum speedup for computing the slack matrix inverse S^{-1})**.** *Let $S \in \mathbb{R}^{n \times n}$ be defined as $S = \sum_{i=1}^m y_i A_i - C$. Let $\epsilon_S \in (0, 1/10)$ denote an accuracy parameter. For the* APPROXSLACK *procedure (Line 8), there is an algorithm that* *runs in time*

$$\widetilde{O}(n^2\mu(S)\kappa(S)\epsilon_S^{-1})$$

and outputs a classical symmetric matrix $\widetilde{S}^{-1} \in \mathbb{R}^{n\times n}$ *such that*

$$\|\widetilde{S}^{-1} - S^{-1}\|_F \leq \epsilon_S \cdot \|S^{-1}\|_F.$$

Proof. Using QRAM, we can construct P_L, P_R, a $(1+\|y\|_1, \log m, 0)$-state-preparation pair for $\begin{bmatrix} y \\ -1 \end{bmatrix} \in \mathbb{R}^{m+1}$. Since $A_1, \dots, A_m$ and C are stored in QRAM, we can implement U_S, which is a $(O(\mu), O(\log n), 0)$-block encoding for $S = \sum_{i=1}^m y_i A_i - C$, where $\mu := \|y\|_1 \cdot \max\{\max_{i\in[n]} \mu(A_i), \mu(C)\}$. Then, for each basis state $|e_i\rangle$, we compute $S^{-1}|e_i\rangle$ using Theorem 10.10. More specifically, we can prepare a state $|(\widetilde{S}^{-1})_i\rangle$ that is ϵ_1-close to $|(S^{-1})_i\rangle$ in time $\widetilde{O}(\mu\cdot\kappa(S))$. And we can estimate the norm $N_i \in (1\pm\epsilon_2)\|S^{-1}|e_i\rangle\|_2$ in time $\widetilde{O}(\mu\cdot\kappa(S)\cdot\epsilon_2^{-1})$. Finally, by Theorem 10.5, it takes $\widetilde{O}(n\cdot\mu\kappa(S)\cdot\epsilon_3^{-1})$-time to get a classical vector v_i such that $\|v_i - |(\widetilde{S}^{-1})_i\rangle\|_2 \leq \epsilon_3$. Define $(\widetilde{S}^{-1})_i := N_i \cdot v_i \in \mathbb{R}^n$.

We have

$$\begin{aligned}
\|(\widetilde{S}^{-1})_i - (S^{-1})_i\|_2 &= \|N_i \cdot v_i - (S^{-1})_i\|_2 \\
&\leq \|(S^{-1})_i\|_2 \cdot \|(1\pm\epsilon_2)v_i - |(S^{-1})_i\rangle\|_2 \\
&\leq \|(S^{-1})_i\|_2 \cdot (\|(1\pm\epsilon_2)v_i - |(\widetilde{S}^{-1})_i\rangle\|_2 + \||(\widetilde{S}^{-1})_i\rangle - |(S^{-1})_i\rangle\|_2) \\
&\leq \|(S^{-1})_i\|_2 \cdot (\|(1\pm\epsilon_2)v_i - |(\widetilde{S}^{-1})_i\rangle\|_2 + \epsilon_1) \\
&\leq \|(S^{-1})_i\|_2 \cdot (\|v_i - |(\widetilde{S}^{-1})_i\rangle\|_2 + \epsilon_1 + \epsilon_2\|v_i\|_2) \\
&\leq \|(S^{-1})_i\|_2 \cdot (\epsilon_1 + \epsilon_3 + \epsilon_2\|v_i\|_2) \\
&\leq \|(S^{-1})_i\|_2 \cdot (\epsilon_1 + \epsilon_3 + \epsilon_2(\||(\widetilde{S}^{-1})_i\rangle\|_2 + \||(\widetilde{S}^{-1})_i\rangle - v_i\|_2)) \\
&\leq \|(S^{-1})_i\|_2 \cdot (\epsilon_1 + \epsilon_3 + \epsilon_2(1+\epsilon_3)) \\
&\leq O(\epsilon_1 + \epsilon_2 + \epsilon_3) \cdot \|(S^{-1})_i\|_2.
\end{aligned}$$

By taking $\epsilon_1 = \epsilon_2 = \epsilon_3 = O(\epsilon_S)$, we get that $(\widetilde{S}^{-1})_i$ can be computed in $\widetilde{O}(n\mu(S)\kappa(S)\epsilon_S^{-1})$-time such that $\|(\widetilde{S}^{-1})_i - (S^{-1})_i\|_2 \leq \epsilon_S\|(S^{-1})_i\|_2$. We apply this procedure for all

$i \in [n]$. Then, we obtain $\widetilde{S}^{-1} \in \mathbb{R}^{n\times n}$ such that

$$\|\widetilde{S}^{-1} - S^{-1}\|_F \le \epsilon_S \|S^{-1}\|_F$$

in $\widetilde{O}(n^2\mu(S)\kappa(S)\epsilon_S^{-1})$-time. Since we know that S and S^{-1} are symmetric matrices, we can easily symmetrize $\widetilde{S}^{-1}$ by $(\widetilde{S}^{-1} + (\widetilde{S}^{-1})^\top)/2$ without increasing the approximation error. $\square$

10.7.2 Gradient

The goal of this section is to prove the following lemma, which computes a quantum state that encodes the gradient g_η. We will use the quantum state and the estimated norm to quantumly compute the Newton step later. Furthermore, we also bound the approximation error between the true gradient $g_{\eta,S}$ and the vector encoded in the quantum state, which will be useful in the latter error analysis.

Lemma 10.14 (Computing the gradient state $|g_\eta\rangle$ and norm $\|g_\eta\|$)**.** *Let $g_\eta \in \mathbb{R}^m$ be defined as $g_\eta = \eta \cdot b - \mathsf{A} \cdot \mathrm{vec}(\widetilde{S}^{-1}) := \mathsf{A}' \cdot s' \in \mathbb{R}^m$, where $\mathsf{A}' := \begin{bmatrix} \mathsf{A} & b \end{bmatrix}$ and $s' := \begin{bmatrix} -\mathrm{vec}(\widetilde{S}^{-1}) \\ \eta \end{bmatrix}$. Then, a quantum state $|\widetilde{g}_\eta\rangle$ can be prepared in time $T_g = \widetilde{O}(\mu(\mathsf{A})\kappa(\mathsf{A}))$ that is ϵ_g-close to the state $|g_\eta\rangle$. Moreover, $\|g_\eta\|_2$ can be estimated with ϵ_g'-multiplicative error in time $T_{g,\mathsf{norm}} = \widetilde{O}(\mu(\mathsf{A})\kappa(\mathsf{A})\epsilon_g'^{-1})$.*

Let $g_{\eta,S} = \eta \cdot b - \mathsf{A} \cdot \mathrm{vec}(S^{-1})$. Then, we have

$$\|\widetilde{g}_\eta - g_{\eta,S}\|_2 \le O(\epsilon_g + \epsilon_g' + \epsilon_S \cdot \kappa(\mathsf{A}))\|g_{\eta,S}\|_2.$$

Proof. Since A and b are stored in QRAM, we can form the block-encoding of A. Moreover, by Lemma 10.13, $\widetilde{S}^{-1}$ are obtained in the classical form such that $\|\widetilde{S}^{-1} - S^{-1}\|_F \le \epsilon_S\|S^{-1}\|_F$. We can prepare the state $|s'\rangle$. By Theorem 10.10 (part 2), $|\widetilde{g}_\eta\rangle$ can be prepared in time $\widetilde{O}(\mu(\mathsf{A}')\kappa(\mathsf{A}')) = \widetilde{O}(\mu(\mathsf{A})\kappa(\mathsf{A}))$ that is ϵ_g-close to $|g_\eta\rangle$. Furthermore, by Theorem 10.10 (part 3), the norm $\|g_\eta\|_2$ can be estimated within relative error ϵ_g' in time $\widetilde{O}(\mu(\mathsf{A})\kappa(\mathsf{A})\epsilon_g'^{-1})$. We analyze the approximation error below.

Let $\widetilde{g}_\eta := N_g \cdot |\widetilde{g}_\eta\rangle$, where $N_g \in (1 \pm \epsilon_g')\|g_\eta\|_2$ and $\||\widetilde{g}_\eta\rangle - |g_\eta\rangle\| \leq \epsilon_g$. Then, we have

$$\begin{aligned}
\|\widetilde{g}_\eta - g_\eta\|_2 &= \|N_g \cdot |\widetilde{g}_\eta\rangle - \|g_\eta\|_2 \cdot |g_\eta\rangle\|_2 \\
&\leq \|N_g \cdot |\widetilde{g}_\eta\rangle - N_g \cdot |g_\eta\rangle\|_2 + \|N_g \cdot |g_\eta\rangle - \|g_\eta\|_2 \cdot |g_\eta\rangle\|_2 \\
&\leq \epsilon_g N_g + \|N_g \cdot |g_\eta\rangle - \|g_\eta\|_2 \cdot |g_\eta\rangle\|_2 \\
&= \epsilon_g N_g + |N_g - \|g_\eta\|_2| \\
&\leq (\epsilon_g(1+\epsilon_g') + \epsilon_g')\|g_\eta\|_2 \\
&= O(\epsilon_g + \epsilon_g')\|g_\eta\|_2, \qquad (10.5)
\end{aligned}$$

where the second step follows from triangle inequality, the third step follows from the definition of ϵ_g, the fourth step follows from $\||g_\eta\rangle\|_2 = 1$, the fifth step follows from the definition of ϵ_g', and the last step is straightforward.

Define $g_{\eta,S} = \eta \cdot b - \mathsf{A} \cdot \mathrm{vec}(S^{-1})$. We have

$$\begin{aligned}
\|g_\eta - g_{\eta,S}\|_2 &= \|\mathsf{A} \cdot (\mathrm{vec}(\widetilde{S}^{-1}) - \mathrm{vec}(S^{-1}))\|_2 \\
&\leq \|\mathsf{A}\| \cdot \|\mathrm{vec}(\widetilde{S}^{-1}) - \mathrm{vec}(S^{-1})\|_2 \\
&= \|\mathsf{A}\| \cdot \|\widetilde{S}^{-1} - S^{-1}\|_F \\
&\leq \|\mathsf{A}\| \cdot \epsilon_S \|S^{-1}\|_F,
\end{aligned}$$

where the second step follows from the definition of the spectral norm, and the third step follows from Lemma 10.13.

Hence, we get that

$$\begin{aligned}
\frac{\|g_\eta - g_{\eta,S}\|_2}{\|g_{\eta,S}\|_2} &\leq \frac{\epsilon_S \|\mathsf{A}\| \cdot \|S^{-1}\|_F}{\|\eta \cdot b - \mathsf{A} \cdot \mathrm{vec}(S^{-1})\|_2} \\
&= O(\epsilon_S) \cdot \frac{\|\mathsf{A}\| \cdot \|\widetilde{S}^{-1}\|}{\|\mathsf{A} \cdot \mathrm{vec}(S^{-1})\|_2} \\
&\leq O(\epsilon_S) \cdot \kappa(\mathsf{A}) \cdot \frac{\|S^{-1}\|_F}{\|S^{-1}\|_F} \\
&= O(\epsilon_S \cdot \kappa(\mathsf{A})), \qquad (10.6)
\end{aligned}$$

where the second step follows from the fact that $\|b\|_2$ can be re-scaled to be $n^{-O(1)}$ so that $\|\eta b\|_2 \ll \|\mathsf{A} \cdot \mathrm{vec}(S^{-1})\|_2$, the third step follows from $\|\mathsf{A} \cdot \mathrm{vec}(S^{-1})\|_2 \geq \sigma_{\min}(\mathsf{A})\|\mathrm{vec}(S^{-1})\|_2 = \sigma_{\min}(\mathsf{A})\|S^{-1}\|_F$, and the third step is straightforward.

Therefore,

$$\begin{aligned}
\|\widetilde{g}_\eta - g_{\eta,S}\|_2 &\leq \|\widetilde{g}_\eta - g_\eta\|_2 + \|g_\eta - g_{\eta,S}\|_2 \\
&= O(\epsilon_g + \epsilon_g')\|g_\eta\|_2 + O(\epsilon_S \cdot \kappa(\mathsf{A}))\|g_{\eta,S}\|_2 \\
&\leq O((\epsilon_g + \epsilon_g')(1 + \epsilon_S \cdot \kappa(\mathsf{A})) + \epsilon_S \cdot \kappa(\mathsf{A}))\|g_{\eta,S}\|_2 \\
&= O(\epsilon_g + \epsilon_g' + \epsilon_S \cdot \kappa(\mathsf{A}))\|g_{\eta,S}\|_2.
\end{aligned}$$

where the first step follows from triangle inequality, the second step follows from Eqs. (10.5) and (10.6), the third step follows from Eq. (10.6) again, and the last step follows directly. $\square$

10.7.3 Update the changes of the dual

The goal of this section is to prove the following lemma that computes a classical vector that is close to the Newton step δ_y, based on QSVT and tomography. We also give an ℓ_2-relative error guarantee.

Lemma 10.15 (Quantum speedup for computing the update δ_y)**.** *For the* APPROX-DELTA *procedure (Line 23), there is an algorithm that outputs a classical vector $\widetilde{\delta}_y \in \mathbb{R}^m$ such that*

$$\|\widetilde{\delta}_y - (-\widetilde{H}^{-1}\widetilde{g}_{\eta^{\mathrm{new}}})\|_2 \leq O(\epsilon_\delta + \epsilon_n + \epsilon_\delta') \cdot \|\widetilde{H}^{-1}\widetilde{g}_{\eta^{\mathrm{new}}}\|_2$$

in time

$$\widetilde{O}\Big(\Big((m\epsilon_n^{-1} + \epsilon_\delta')\|\widetilde{H}\| + \epsilon_g'^{-1}\Big)\mu(\mathsf{A})\kappa(\mathsf{A})\kappa(\widetilde{H}) + (m\epsilon_n^{-1} + \epsilon_\delta'^{-1})\|\widetilde{H}\|\mu(\widetilde{S}^{-1})\kappa(\widetilde{S})\kappa(\widetilde{H})\Big),$$

where $\widetilde{H} = \mathsf{A}(\widetilde{S}^{-1} \otimes \widetilde{S}^{-1})\mathsf{A}^\top$.

Proof. We have access to $\widetilde{S}^{-1}$ in QRAM. Then, we can compute $(\widetilde{S}^{-1})^{1/2}$ classically and construct a unitary $U_{\widetilde{S}^{-1/2}}$, which is a $(\mu(\widetilde{S}^{-1/2}), O(\log n), 0)$-block encoding of $(\widetilde{S}^{-1})^{1/2}$.

By Theorem 10.9, U_1 implements a $(\mu(\widetilde{S}^{-1}), O(\log n), 0)$-block encoding of $\widetilde{S}^{-1/2} \otimes \widetilde{S}^{-1/2}$ in time $T_1 = \widetilde{O}(1)$.

Then, by Lemma 10.8, U_2 implements a $(O(\|\mathsf{A}\|\|\widetilde{S}^{-1}\|), O(\log n), \delta_2)$-block encoding of $\mathsf{A}(\widetilde{S}^{-1/2} \otimes \widetilde{S}^{-1/2})$ in time

$$\begin{aligned} T_2 &= \widetilde{O}_{\delta_2}\Big(\mu(\mathsf{A})/\|\mathsf{A}\| \cdot (T_A + \log n) + \mu(\widetilde{S}^{-1})/\|\widetilde{S}^{-1}\| \cdot (T_1 + \log n)\Big) \\ &= \widetilde{O}_{n,\delta_2}\Big(\mu(\mathsf{A})/\|\mathsf{A}\| + \mu(\widetilde{S}^{-1})/\|\widetilde{S}^{-1}\|\Big). \end{aligned}$$

Since

$$\widetilde{H} = (\mathsf{A}(\widetilde{S}^{-1/2} \otimes \widetilde{S}^{-1/2})) \cdot (\mathsf{A}(\widetilde{S}^{-1/2} \otimes \widetilde{S}^{-1/2}))^\top,$$

U_H implements a $(O(\|\widetilde{H}\|), O(\log n), O(\delta_2))$-block encoding of $\widetilde{H}$ in time

$$\begin{aligned} T_H &= \widetilde{O}_{n,\delta_2}\Big(\|\mathsf{A}\|\|\widetilde{S}^{-1}\|/\|\mathsf{A}(\widetilde{S}^{-1/2} \otimes \widetilde{S}^{-1/2})\| \cdot T_2\Big) \\ &= \widetilde{O}_{n,\delta_2}\Big(\Big(\mu(\mathsf{A})\|\widetilde{S}^{-1}\| + \mu(\widetilde{S}^{-1})\|\mathsf{A}\|\Big)\|\widetilde{H}\|^{-1/2}\Big). \end{aligned}$$

Then, we can prepare a quantum state $|\widetilde{\delta}_y\rangle$ that is ϵ_δ-close to $|\widetilde{H}^{-1}\widetilde{g}_\eta\rangle$ in time

$$\begin{aligned} T_\delta &= \widetilde{O}\Big(\kappa(\widetilde{H})(T_H\|\widetilde{H}\| + T_g)\Big) \\ &= \widetilde{O}\Big(\kappa(\widetilde{H})\Big((\mu(\mathsf{A})\|\widetilde{S}^{-1}\| + \mu(\widetilde{S}^{-1})\|\mathsf{A}\|)\|\widetilde{H}\|^{-1/2} \cdot \|\widetilde{H}\| + \mu(\mathsf{A})\kappa(\mathsf{A})\Big)\Big) \\ &= \widetilde{O}\Big(\kappa(\widetilde{H})\Big((\mu(\mathsf{A})\|\widetilde{S}^{-1}\| + \mu(\widetilde{S}^{-1})\|\mathsf{A}\|)\|\widetilde{H}\|^{1/2} + \mu(\mathsf{A})\kappa(\mathsf{A})\Big)\Big). \end{aligned}$$

And the norm $\|\widetilde{H}^{-1}|\widetilde{g}_\eta\rangle\|_2$ can be estimated with ϵ'_δ-multiplicative error in time $T_{\delta,\mathsf{norm}} = \widetilde{O}(T_\delta \cdot \epsilon_\delta'^{-1})$.

Finally, by the tomography procedure, we can obtain a classical vector $\widetilde{\delta}_{y,1} \in \mathbb{R}^m$ such that $\||\widetilde{\delta}_y\rangle - \widetilde{\delta}_{y,1}\|_2 \le \epsilon_n$ in time $T_{\delta,\mathsf{tomo}} = \widetilde{O}(T_\delta \cdot m\epsilon_n^{-1})$. And we let $\widetilde{\delta}_y := N_\delta \cdot N_g \cdot \widetilde{\delta}_{y,1}$, where $N_\delta \in (1 \pm \epsilon'_\delta)\|\widetilde{H}^{-1}|\widetilde{g}_\eta\rangle\|_2$ and $N_g \in (1 + \epsilon'_g)\|g_\eta\|_2$.

We have

$$\begin{aligned}
\|\widetilde{H}^{-1}\widetilde{g}_\eta - \widetilde{\delta}_y\|_2 &= \|\widetilde{H}^{-1}\widetilde{g}_\eta - N_\delta \cdot N_g \cdot \widetilde{\delta}_{y,1}\|_2 \\
&= \|\widetilde{g}_\eta\|_2 \cdot \|\widetilde{H}^{-1}|\widetilde{g}_\eta\rangle - N_\delta \cdot \widetilde{\delta}_{y,1}\|_2 \\
&= \|\widetilde{g}_\eta\|_2 \cdot \|\widetilde{H}^{-1}|\widetilde{g}_\eta\rangle\|_2 \cdot \||\widetilde{H}^{-1}\widetilde{g}_\eta\rangle - (1 \pm \epsilon_\delta')\widetilde{\delta}_{y,1}\|_2 \\
&= \|\widetilde{H}^{-1}g_\eta\|_2 \cdot (\||\widetilde{H}^{-1}g_\eta\rangle - \widetilde{\delta}_{y,1}\|_2 + \epsilon_\delta'\|\widetilde{\delta}_{y,1}\|_2) \\
&= \|\widetilde{H}^{-1}\widetilde{g}_\eta\|_2 \cdot (\||\widetilde{H}^{-1}\widetilde{g}_\eta\rangle - \widetilde{\delta}_{y,1}\|_2 + O(\epsilon_\delta')) \\
&= \|\widetilde{H}^{-1}\widetilde{g}_\eta\|_2 \cdot (\||\widetilde{H}^{-1}\widetilde{g}_\eta\rangle - |\widetilde{\delta}_y\rangle\|_2 + \||\widetilde{\delta}_y\rangle - \widetilde{\delta}_{y,1}\|_2 + O(\epsilon_\delta')) \\
&= \|\widetilde{H}^{-1}\widetilde{g}_\eta\|_2 \cdot O(\epsilon_\delta + \epsilon_n + \epsilon_\delta'),
\end{aligned}$$

where the first step follows from the definition of $\widetilde{\delta}_y$, the second step follows from $\|\widetilde{g}_\eta\|_2 = N_g$, the third step follows from $N_\delta \in (1 \pm \epsilon_\delta')\|\widetilde{H}^{-1}|\widetilde{g}_\eta\rangle\|_2$, the forth step follows from triangle inequality, the fifth step follows from $\|\widetilde{\delta}_{y,1}\|_2 = O(1)$, the sixth step follows from triangle inequality, and the last step follows from the approximation guarantees of $\||\widetilde{H}^{-1}\widetilde{g}_\eta\rangle - |\widetilde{\delta}_g\rangle\|_2 \leq \epsilon_\delta$ and $\||\widetilde{\delta}_y\rangle - \widetilde{\delta}_{y,1}\|_2 \leq \epsilon_n$.

The total time complexity of getting $\widetilde{\delta}_y$ is:

$$\begin{aligned}
T &= T_{\delta,\mathsf{tomo}} + T_{\delta,\mathsf{norm}} + T_{g,\mathsf{norm}} \\
&= \widetilde{O}((m\epsilon_n^{-1} + \epsilon_\delta'^{-1})T_\delta + \epsilon_g'^{-1}\mu(\mathsf{A})\kappa(\mathsf{A})) \\
&= \widetilde{O}\Big((m\epsilon_n^{-1} + \epsilon_\delta'^{-1})\kappa(\widetilde{H})(\mu(\mathsf{A})\|\widetilde{S}^{-1}\| + \mu(\widetilde{S}^{-1})\|\mathsf{A}\|)\|\widetilde{H}\|^{1/2} + \epsilon_g'^{-1}\kappa(\widetilde{H})\mu(\mathsf{A})\kappa(\mathsf{A})\Big) \\
&= \widetilde{O}\Big(\Big((m\epsilon_n^{-1} + \epsilon_\delta')\|\widetilde{H}\| + \epsilon_g'^{-1}\Big)\mu(\mathsf{A})\kappa(\mathsf{A})\kappa(\widetilde{H}) + (m\epsilon_n^{-1} + \epsilon_\delta'^{-1})\|\widetilde{H}\|\mu(\widetilde{S}^{-1})\kappa(\widetilde{S})\kappa(\widetilde{H})\Big).
\end{aligned}$$

□

10.7.4 Combine

In this section, we wrap up the three components and show the cost-per-iteration of our quantum SDP solver in the following lemma. By our error analysis of each component, we can prove that the robustness conditions are satisfied by proper choices of parameters. Then, the main theorem (Theorem 10.17) follows immediately.

Lemma 10.16 (Quantum SDP solver cost per iteration)**.** *Let $t \in [T]$. The t-th iteration of Algorithm 49 takes*

$$\widetilde{O}((m\mu(\mathsf{A}) + n^2\mu(S)) \cdot \kappa(A)\kappa(S)\kappa(H)^3)$$

time such that ***Cond. 0.*** *to* ***Cond. 4.*** *in Lemma 10.11 are satisfied.*

Proof. For **Cond. 0.**, if $t = 1$, then a feasible solution y can be found in the initialization step (Line 5). For $t > 1$, we will prove by induction. Suppose it is satisfied by the y computed in the $(t-1)$-th iteration.

For **Cond. 1.**, let $\widetilde{S}^{-1}$ be the matrix computed by ApproxSlack, and let $S := \mathrm{mat}(\mathsf{A}^\top y) - C$. By Lemma 10.13, the algorithm takes

$$T_S = \widetilde{O}(n^2\mu(S)\kappa(S)\epsilon_S^{-1}) \tag{10.7}$$

time such that

$$\|\widetilde{S}^{-1} - S^{-1}\|_F \leq \epsilon_S \cdot \|S^{-1}\|_F.$$

By Fact 9.5, we get that $(1-\epsilon_S)S^{-1} \preceq \widetilde{S}^{-1} \preceq (1+\epsilon_S)S^{-1}$. Hence, $\alpha_S^{-1}S^{-1} \preceq \widetilde{S}^{-1} \preceq \alpha_S S^{-1}$ holds for some $\alpha_S \in (1, 1+10^{-4})$, as long as ϵ_S is a small enough constant.

For **Cond. 2.**, since we compute $\widetilde{S}^{-1}$ and store it in QRAM, we may assume that there is no error in this step, i.e., $\widetilde{H} = H(\widetilde{S})$. Hence, this condition is satisfied by taking $\alpha_H = 1$.

For **Cond. 3.**, let $\widetilde{g}_{\eta^{\mathrm{new}}} := N_g \cdot |\widetilde{g}_{\eta^{\mathrm{new}}}\rangle$ produced by Lemma 10.14. It guarantees that

$$\|\widetilde{g}_{\eta^{\mathrm{new}}} - g_{\eta^{\mathrm{new}}}\|_2 \leq O(\epsilon_g + \epsilon_g' + \epsilon_S \cdot \kappa(\mathsf{A}))\|g_{\eta^{\mathrm{new}}}\|_2.$$

By the first part of Remark 10.1, if we have

$$\|\widetilde{g}_{\eta^{\mathrm{new}}} - g_{\eta^{\mathrm{new}}}\|_2 \leq \tau_g\|g_{\eta^{\mathrm{new}}}\|_2/\kappa(H).$$

for some $\tau_g \in (0, 10^{-4})$, then **Cond. 3.** will be satisfied, i.e.,

$$\|\widetilde{g}_{\eta^{\text{new}}} - g_{\eta^{\text{new}}}\|_{H^{-1}} \leq \tau_g \|g_{\eta^{\text{new}}}\|_{H^{-1}}.$$

Hence, we can take

$$\epsilon_g = \epsilon'_g = O(\kappa(H)^{-1}) \quad \text{and} \quad \epsilon_S = O(\kappa(\mathsf{A})^{-1}\kappa(H)^{-1}). \tag{10.8}$$

For **Cond. 4.**, let $\widetilde{\delta}_y$ be the vector computed by ApproxDelta. By Lemma 10.15, it takes

$$T_\delta = \widetilde{O}\Big(\Big((m\epsilon_n^{-1} + \epsilon'_\delta)\|\widetilde{H}\| + \epsilon_g'^{-1}\Big)\, \mu(\mathsf{A})\kappa(\mathsf{A})\kappa(\widetilde{H}) + (m\epsilon_n^{-1} + \epsilon_\delta'^{-1})\|\widetilde{H}\|\mu(\widetilde{S}^{-1})\kappa(\widetilde{S})\kappa(\widetilde{H})\Big) \tag{10.9}$$

time such that

$$\|\widetilde{\delta}_y - (-\widetilde{H}^{-1}\widetilde{g}_{\eta^{\text{new}}})\|_2 \leq O(\epsilon_\delta + \epsilon_n + \epsilon'_\delta) \cdot \|\widetilde{H}^{-1}\widetilde{g}_{\eta^{\text{new}}}\|_2.$$

By the second part of Remark 10.1, if we have

$$\|\widetilde{\delta}_y - (-\widetilde{H}^{-1}\widetilde{g}_{\eta^{\text{new}}})\|_2 \leq \tau_\delta \|\widetilde{H}^{-1}\widetilde{g}_{\eta^{\text{new}}}\|_2 / \kappa(H)$$

for some $\tau_\delta \in (0, 10^{-4})$, then **Cond. 4.** will be satisfied, i.e.,

$$\|\widetilde{\delta}_y - (-\widetilde{H}^{-1}\widetilde{g}_{\eta^{\text{new}}})\|_H \leq \tau_\delta \cdot \|\widetilde{H}^{-1}\widetilde{g}_{\eta^{\text{new}}}\|_H.$$

Hence, we can take

$$\epsilon_\delta = \epsilon_n = \epsilon'_\delta = O(\kappa(H)^{-1}). \tag{10.10}$$

Then, Eq. (10.9) can be simplified as follows:

$$\begin{aligned} T_\delta &= \widetilde{O}(m(\mu(\mathsf{A})\kappa(\mathsf{A}) + \mu(\widetilde{S}^{-1})\kappa(\widetilde{S})) \cdot \|H\|\kappa(H)^2) \\ &= \widetilde{O}(m(\mu(\mathsf{A})\kappa(\mathsf{A}) + \mu(S)\kappa(S)) \cdot \|H\|\kappa(H)^2). \end{aligned} \tag{10.11}$$

Then, by Lemma 10.11, for y^{new} computed in Line 32, we have

$$\|g(y^{\text{new}}, \eta^{\text{new}})\|_{H(y^{\text{new}})^{-1}} \leq \epsilon_N,$$

which means y^{new} satisfies the **Cond. 0.** in the $(t+1)$-th iteration.

Therefore, for all $t \in [T]$, **Cond. 0.** to **Cond. 4.** in Lemma 10.11 are satisfied.

For the running time per iteration, by Eqs. (10.7) and (10.11):

$$\begin{aligned}\mathcal{T}_{\text{iter}} &:= T_S + T_\delta \\ &= \widetilde{O}(n^2\mu(S)\kappa(S)\kappa(\mathsf{A})\kappa(H)) + \widetilde{O}(m(\mu(\mathsf{A})\kappa(\mathsf{A}) + \mu(S^{-1})\kappa(S)) \cdot \|H\|\kappa(H)^2) \\ &\le \widetilde{O}((m\mu(\mathsf{A}) + n^2\mu(S)) \cdot \kappa(A)\kappa(S)\kappa(H)^3).\end{aligned}$$

□

Theorem 10.17 (Quantum second-order SDP solver)**.** *Suppose the input matrices $\{A_i\}_{i=1}^m, C$ and vector b are stored in QRAM. Let $T := O(\sqrt{n}\log(1/\epsilon))$. Then, the quantum second-order SDP solver (Algorithm 49) runs T iterations with*

$$\widetilde{O}((m\mu(\mathsf{A}) + n^2\mu(S)) \cdot \kappa(A)\kappa(S)\kappa(H)^3)$$

time per iteration and outputs a classical vector $y \in \mathbb{R}^m$ such that with probability at least $1 - 1/\mathrm{poly}(n)$,

$$\begin{aligned}&\langle b, y\rangle \le \langle b, y^*\rangle + \epsilon, \quad \textit{and} \\ &\sum_{i=1}^n y_i A_i - C \succeq -\epsilon I,\end{aligned}$$

In the above formula, we use y^* to represent an optimal dual solution.

Furthermore, the algorithm can also output a PSD matrix $X \in \mathbb{R}^{n\times n}_{\ge 0}$ in the same running time that satisfies

$$\langle C, X\rangle \ge \langle C, X^*\rangle - \epsilon \cdot \|C\|_2 \cdot R \quad \textit{and} \quad \sum_{i=1}^m \left|\langle A_i, \widehat{X}\rangle - b_i\right| \le 4n\epsilon \cdot \Big(R\sum_{i=1}^m \|A_i\|_1 + \|b\|_1\Big). \tag{10.12}$$

In the above formula, we use X^* to represent an optimal solution to the semi-definite program in Definition 10.1.

Proof. Note that the running time per iteration follows from Lemma 10.16. And the number of iterations to achieve the stated error guarantee is by Theorem 10.12.

For the primal solution, by Lemma E.2, we can find X using y classically in time $O(n^{\omega})$, which is less than the SDP solver's running time. $\square$

Remark 10.2. Plugging-in the general upper bounds for $\mu(\mathsf{A}) \leq n$ and $\mu(S) \leq \sqrt{n}$ (see Theorem 10.6), we get that the total time complexity of Algorithm 49 is

$$\widetilde{O}\left(\sqrt{n}(mn + n^{2.5})\mathrm{poly}(\kappa, \log(1/\epsilon))\right).$$

10.8 Well-Conditioned SDP Instances

Here we list some SDP cases in high dimensions where the condition number of Hessian does not depend on $1/\epsilon$.

Case 1: A simple case is where the central path is *nearly isotropic* and thus the condition number of Hessian is close to 1, e.g.

$$\max_{y \in \mathbb{R}^m} \; y_1 + \cdots + y_m \; \text{ subject to } \; y_i \leq 0 \;\; \forall i \in [m].$$

Case 2: Another SDP example is as follows:

$$\max_{X \in \mathbb{R}^{n \times n}} \; \langle C, X \rangle \text{ subject to } \; \langle A_i, X \rangle = b_i, \;\; \forall i \in [m], \; X \succeq 0,$$

where $n = m = 3$ and

$$C = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}, A_1 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, A_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, A_3 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, b = \begin{bmatrix} -1 \\ -1 \\ 0 \end{bmatrix}.$$

The dual problem is given by

$$\min_{y \in \mathbb{R}^m} \; b^{\top} y \quad \text{subject to } S = \sum_{i=1}^{m} y_i A_i - C, \quad S \succeq 0.$$

The central path can be found by solving the following penalized objective function

$$\min_{y\in\mathbb{R}^m} \ \eta \cdot b^\top y - \log\det(S)$$

which yields

$$y = \begin{bmatrix} -\eta^{-1} \\ -\eta^{-1} \\ 0 \end{bmatrix}, S = \begin{bmatrix} \eta^{-1} & 0 & 0 \\ 0 & \eta^{-1} & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Therefore the Hessian is given by

$$H = \begin{bmatrix} \eta^2 & 0 & 0 \\ 0 & \eta^2 & 0 \\ 0 & 0 & 2\eta^2 \end{bmatrix}$$

and the condition number is $\kappa(H) = 2$. Notice that the primal and dual solutions are

$$X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, y = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, S = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

thus the strict complementary holds. In addition, another intermediate-matrix related term in the running time $\|\mathsf{A}^\top y\|_2^2\|S^{-1}\|^2 = 2$ in this example. Notice also the strict complementarity holds. Therefore, our algorithm will have $\log(1/\epsilon)$ dependence in the total running time in this example. In brief, the "good cases" (where our quantum algorithm's running time depends on $\log(1/\epsilon)$) for SDP are much more non-trivial than those for LP.

We also note that in all previous quantum interior-point method papers ([KP20b, CMD20, KPS21, ANTZ21]), their running times also depend on the condition number of the linear system that appears in each iteration. In addition, their running times explicitly depend on $\mathrm{poly}(1/\epsilon)$ as well. Thus, even in the well-conditioned cases, their running times still depend on $\mathrm{poly}(1/\epsilon)$. And within the quantum linear algebra framework, almost all matrix-related algorithms depend on the condition number of the matrix. Hence, it seems that the dependence on the condition number is unavoidable for the current techniques in quantum computing.

Algorithm 49 Quantum SDP Solver. Let $A_1, \cdots, A_m$ denote a list of $n \times n$ matrices. The input A has size $m \times n^2$. The input vector b has length m. And the input matrix C has size n by n.

```
1: procedure SolveSDPinQuantum($\{A_i\}_{i=1}^m$, $b, C, m, n$)
2:                                                    ▷ The input $\mathsf{A}, b, C$ are stored in QRAM.
3:                                                    ▷ Initialization
4:     Choose $\eta$ and $T$ as Definition 10.2.
5:     Find initial feasible $y$ and store in QRAM    ▷ Lemma E.1, $O(m+n)$-time
6:     Update QRAM for the modified SDP instance      ▷ $\widetilde{O}(m+n)$-time
7:     for $t = 1 \to T$ do
8:                                                    ▷ ApproxSlack, Lemma 10.13
9:         $U_S \leftarrow$ the block-encoding for $S = \sum_{i=1}^m y_i A_i - C$
10:        for $i = 1 \to n$ do                        ▷ $\widetilde{O}(n^{2.5})$-time in total
11:            $|(\widetilde{S}^{-1})_i\rangle \leftarrow$ apply quantum linear system solver with $U_S$ and $|e_i\rangle$
12:            $v_i \leftarrow$ apply $\ell_2$-tomography on the state $|(\widetilde{S^{-1}})_i\rangle$    ▷ $\widetilde{O}(n^{1.5})$-time
13:            $N_i \leftarrow$ estimate the norm $\|(S^{-1})_i\|_2$    ▷ $\widetilde{O}(n^{0.5})$-time
14:            $(\widetilde{S}^{-1})_i \leftarrow N_i \cdot v_i$    ▷ $O(n)$-time
15:        end for
16:        $\widetilde{S}^{-1} \leftarrow \frac{1}{2}(\widetilde{S}^{-1} + (\widetilde{S}^{-1})^\top)$ and store it in QRAM    ▷ $O(n^2)$-time
17:        $\eta^{\mathrm{new}} \leftarrow \eta \cdot (1 + \frac{\epsilon_N}{20\sqrt{n}})$
18:                                                    ▷ ApproxGradient, Lemma 10.14
19:        $U_{\mathsf{A}'} \leftarrow$ the block-encoding of $\begin{bmatrix} \mathsf{A} & b \end{bmatrix}$
20:        $|s'\rangle_\leftarrow$ the vector state for $\begin{bmatrix} -\mathrm{vec}(\widetilde{S}^{-1}) & \eta \end{bmatrix}^\top$
21:        $|\widetilde{g}_{\eta^{\mathrm{new}}}\rangle \leftarrow$ apply $U_{\mathsf{A}'}$ to $|s'\rangle$
22:        $N_g \leftarrow$ estimate the norm $\|\mathsf{A}'s'\|_2$    ▷ $\widetilde{O}(n^2)$-time
23:                                                    ▷ ApproxDelta, Lemma 10.15
24:        $\widetilde{S}^{-1/2} \leftarrow$ classically compute $\sqrt{\widetilde{S}^{-1}}$ and store in QRAM    ▷ $O(n^\omega)$-time
25:        $U_1 \leftarrow$ the block-encoding of $\widetilde{S}^{-1/2} \otimes \widetilde{S}^{-1/2}$
26:        $U_2 \leftarrow$ the block-encoding of $W = \mathsf{A}(\widetilde{S}^{-1/2} \otimes \widetilde{S}^{-1/2})$ using $U_{\mathsf{A}}$ and $U_1$
27:        $U_H \leftarrow$ the block-encoding of $\widetilde{H} = WW^\top$ using $U_2$
28:        $|\widetilde{H}^{-1}|\widetilde{g}_{\eta^{\mathrm{new}}}\rangle\rangle \leftarrow$ apply quantum linear system solver with $U_H$ and $|\widetilde{g}_{\eta^{\mathrm{new}}}\rangle$
29:        $v_\delta \leftarrow$ apply $\ell_2$-tomography on the state $|\widetilde{H}^{-1}|\widetilde{g}_{\eta^{\mathrm{new}}}\rangle\rangle$    ▷ $\widetilde{O}(mn)$-time
30:        $N_\delta \leftarrow$ estimate the norm $\|\widetilde{H}^{-1}|\widetilde{g}_{\eta^{\mathrm{new}}}\rangle\|_2$    ▷ $\widetilde{O}(n)$-time
31:        $\widetilde{\delta}_y \leftarrow -N_\delta \cdot N_g \cdot v_\delta$    ▷ $O(m)$-time
32:        $y^{\mathrm{new}} \leftarrow y + \widetilde{\delta}_y$ and store in QRAM    ▷ $O(m)$-time
33:    end for
34: end procedure
```

Chapter 11: A Unified Approach to Fourier Set-Query

11.1 Introduction

The fast Fourier transform (FFT) [CT65] is a fundamental tool in engineering, signal processing, mathematics, and theoretical computer science, with profound applications in theory and practice. Over the years, many variations of FFT have been studied and developed, depending on the underlying domain and time-invariance properties of the signal [OWN+97, Osg02, Opp11]. In this chapter, we study the *sparse Fourier transform* (SFT), where the signal is either discrete or continuous in time domain, but *k-sparse* in the frequency domain, i.e., $\widehat{x}$ is a discrete set of size k:

$$x(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} \langle f_j, t \rangle}.$$

Band-limited (i.e., Fourier-sparse) signals arise in many real-world datasets and applications, from image compression and analysis [Wat94], to compressed sensing [Don06] and (deep) learning with frequency-invariant kernels [MMM21]; for a broader exposition of SFT and its applications, we refer the reader to the survey [GIIS14].

A prototypical problem in this setting is *band-limited signal interpolation* [CKPS16] and the related[1] *Fourier Set-Query* problem [Pri11], which ask how to reconstruct (a subset of) the signal from few (ideally $\sim k$) *noisy* samples of $x(t)$ in a time domain $[0, T]^d$. In this model, the algorithm has access to samples $x(t) + g(t)$, where the signal-to-noise ratio is guaranteed to be above a certain constant threshold (e.g., $\|x\|_T \gtrsim \|g\|_T$, where $\|x\|_T^2 := T^{-d} \int_{[0,T]^d} |x(t)|^2 \mathrm{d}t$ is the *energy* of the signal).

[1]The Fourier interpolation literature typically focuses on frequency estimation followed by magnitude estimation. The magnitude estimation can be formulated as a Set-Query problem, where we are given a set of locations and we only try to recover the Fourier coefficients $\widehat{x}$ at the given locations. In our setup, frequencies are assumed to lie on a lattice, hence the problems are essentially equivalent, see Section 11.3.

In general, sparse-recovery problems have two computational aspects: the sample complexity, i.e., the number of (noisy) samples required by the algorithm, and the reconstruction time (decoding the signal from the measurements). This problem has a long history in signal-processing and TCS [CT65, Rey89, ASSN08, Voe11, HIKP12a, IK14, IKP14, Bub15, Kap16, Kap17, NSW19, JLS23]. A fundamental fact, pointed out in [Moi15], is that when the frequency gap is small ($\eta := \min_{i \neq j \in [k]} |f_i - f_j| < 1/T$), exact recovery of the signal is informational-theoretically impossible. Complementing this negative result, [PS15] gave a $k \cdot \mathrm{polylog}(k, FT/\delta)$-time δ-error reconstruction algorithm for one-dimensional signals where F is the band-limit, assuming the time domain satisfies $T > \Omega(\log^2(k/\delta)/\eta)$, and that the frequency gap η is known. Later, [JLS23] generalizes the [PS15] from one-dimensional space to high dimensional space. [CKPS16] strengthened this result [PS15] by showing that even if the frequency gap is unknown, *approximate* reconstruction of one-dimensional signals in $\mathrm{poly}(k, \log(FT))$-samples and time is possible, in the sense that the output signal is close to the original signal in the time domain albeit with worse sparsity in the frequency domain[2]. Subsequent works [CP19b, CP19a, LLM21] have improved this result, both in sample-complexity and decoding time. Recently, [LLM21] improved the sparsity of the output signal from $\mathrm{poly}(k)$ to $k\mathrm{poly}\log(k)$, settling for a somewhat weaker notion of approximation[3] than that of [CKPS16].

In the discrete setting, [Pri11] defined and studied the Set Query problem in the standard compressed-sensing model [Don06], where the design of the sensing matrix is unrestricted. [Kap17] defined and studied the Set Query problem in Fourier domain, where the sensing matrix is applied to the *Fourier transform* of x (i.e., measurements are $S\widehat{x} = S \cdot \mathsf{FFT} \cdot x$). As such, Fourier Set Query is a more challenging problem than the former one. The current best discrete Fourier set query algorithm in 1D is due to Kapralov [Kap17], who gave an algorithm with near-linear sample complexity

[2]More precisely, the error guarantee is $\|y(t) - x^*(t)\|_T \leq O(\|g(t)\|_T + \delta\|x^*(t)\|_T)$, where $x^*(t)$ is the original signal, $y(t)$ is the reconstructed signal, and $g(t)$ is the noise distribution.

[3]$\|y(t) - x^*(t)\|_{(1-c)T} \leq \mathrm{poly}(\log(k/c\delta)) \cdot \|g(t)\|_T + \delta\|x^*\|_T$.

$(O(k/\varepsilon))$ and decoding time $\widetilde{O}(\varepsilon^{-1}k\log^{2.001} n \log R^*)$, where n is the length of signal and R^* is (roughly) the ℓ_∞ norm of the signal in the time domain.

Unfortunately, much less was known in higher dimensions – The "curse of dimensionality" of the Filter function [Kap17] drastically deteriorates the sample complexity, which grows *exponentially* with the dimension, hence filter-based algorithms (a-la [Kap17]) are near-optimal only for small d. Indeed, this drawback was one of the principal motivations of this chapter.

Given the abundance of largely incomparable results and the diversity of techniques mentioned above, one might wonder whether there is a systematic, unified approach for analyzing band-limited signal reconstruction. We propose such an approach, which *decouples* the band-limited signal interpolation problem into two sub-problems:

1. **Frequency Estimation:** Find L net-frequencies $\{f_i'\}_{i\in[L]}$ such that each f_i is close to some f_j' for $j \in [L]$.
2. **Signal Estimation:** Based on the net-frequencies, approximate the original signal.

We note that almost all the previous works [PS15, CKPS16, JLS23] fall into this way, yet differ in the techniques used to solve these two sub-problems. Our starting point is the observation that the assumption of an η-frequency gap of the signal is roughly equivalent to assuming that the frequencies lie on the grid $\eta \cdot \mathbb{Z}^d$. It is natural to generalize this assumption by considering the case where signal frequencies lie on a *d-dimensional lattice* $\mathcal{L} = \Lambda(\mathcal{B})$, where $\mathcal{B}$ is a basis of $\mathcal{L}$ and is given. We interchangeably call this problem *lattice Fourier interpolation*, or *semi-continuous signal reconstruction*, as lattice frequencies can be viewed as interpolating between discrete and continuous domains. Solving this problem is our key tool en route to faster sparse recovery, but it is also interesting in its own right.

In this chapter, we focus on the second sub-problem, that is, given the result of frequency estimation for a semi-continuous signal, how to efficiently reconstruct the coefficients. It can be formulated as a semi-continuous Fourier set-query problem, where the queried frequency set L (a set of lattice points) is obtained from Frequency Estimation, and our goal is to recover the coefficients of these frequencies. The generality of our framework allows us to apply it to the *discrete* Fourier Set-Query problem as well, where the goal is to (approximately) compute DFT only on a subset $S \subset [n]$ of k coordinates. (See Remark 11.2 for more detailed discussions of the relationship between semi-continuous signal estimation and discrete Fourier set-query.)

11.1.1 Our results

Semi-continuous signal estimation Our first main result shows that for semi-continuous signals (whose frequencies lie on a lattice), Signal Estimation can be efficiently reduced to Frequency Estimation:

Theorem 11.1 (1D Semi-continuous Signal Estimation, informal version of Theorems 11.57 and 11.60)**.** *Let $\Lambda(\mathcal{B}) \subset \mathbb{R}$ denote the lattice $\Lambda(\mathcal{B}) = \{z \in \mathbb{R} | z = c\eta, c \in \mathbb{Z}\}$, and suppose $x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$ with $f_j \in \Lambda(\mathcal{B})$. Given observations of the form $x(t) = x^*(t) + g(t)$ for $t \in [0, T]$ for arbitrary noise distribution $g(t)$, let L be the result of Frequency Estimation for $x(t)$ such that for each f_i, there is an $f_i' \in L$ with $|f_i - f_i'| \leq D/T$. Then given the set L as input, we can obtain:*

- **A Sample-optimal algorithm:** *There is an algorithm that takes $O(\widetilde{k})$ samples and outputs a $\widetilde{k}$-sparse signal $y(t)$ in $O(\widetilde{k}^{\omega+1})$-time*[4]*, such that $\|y - x^*\|_T^2 \leq O(\|g\|_T^2)$ holds with high probability, where $\widetilde{k} := O(|L|(1 + D/(T\eta)))$ is the output sparsity.*

- **A High-accuracy algorithm:** *For any $\varepsilon \in (0, 1)$, there is an algorithm that takes $\widetilde{O}(\varepsilon^{-1}\widetilde{k})$ samples and outputs a $\widetilde{k}$-sparse signal $y(t)$ in $O(\varepsilon^{-1}\widetilde{k}^{\omega})$-time such*

[4]$\omega \approx 2.373$ is the fast matrix-multiplication exponent [AW21].

that $\|y - x^*\|_T^2 \leq (1+\varepsilon)\|g\|_T^2$ holds with high probability.

Remark 11.1. We note that the semi-continuous signal estimation is similar to a sub-problem studied in [PS15, JLS23]. However, they cannot achieve a $(1+\varepsilon)$-approximation using the HashtoBin techniques [HIKP12a]. We also note that our algorithms can be easily adapted to high-dimensional signal estimation (see Section 11.10).

Discrete Fourier set query Our next result provides an efficient, high-accuracy algorithm for the discrete Fourier Set Query problem in any dimension.

Theorem 11.2 (Discrete Fourier Set Query, informal version of Theorems 11.73)**.** *For any $d \geq 1$, let $n = p^d$ for some $p \in \mathbb{N}_+$. Given a vector $x \in (\mathbb{C}^p)^{\otimes d}$, for $k \geq 1$, any $S \subseteq [n]$, $|S| = k$, there exists an algorithm that takes $O(\varepsilon^{-1}k)$ samples from x, runs in $O(\varepsilon^{-1}k^{\omega+1} + \varepsilon^{-1}k^{\omega-1}d)$ time, and outputs a vector $x' \in \mathbb{C}^n$ such that*

$$\|(x' - \widehat{x})_S\|_2^2 \leq \varepsilon\|\widehat{x}_{[n]\setminus S}\|_2^2$$

holds with probability at least 0.9. Note that $\widehat{x}$ is the d-dimensional discrete Fourier transform of x, $\widehat{x}_f = \sum_{t\in[p]^d} x_t e^{-2\pi \mathbf{i}\langle f,t\rangle/p}$, $f \in [p]^d$.

Remark 11.2. Discrete Fourier set query can be viewed as a special case of semi-continuous signal estimation, where the queried frequencies (supported on S) lie on the integer lattice $\mathbb{Z}^d$. And the remaining part of the signal with frequencies outside of S (i.e., $\widehat{x}_{[n]\setminus S}$) corresponds to the noise g in the semi-continuous signal estimation problem. One difference between these two problems is that for semi-continuous signal estimation, we assume the signal has a continuous time domain $[0, T]$, while for discrete Fourier set query, the signal has a discrete time domain $[p]^d$.

Remark 11.3. In one-dimension ($d = 1$), the runtime of our set query algorithm can be simplified to $O(\varepsilon^{-1}k^{\omega+1})$. And prior to this chapter, the only known result for ℓ_2/ℓ_2 Fourier set query (due to [Kap17]) achieves the same sample complexity but runs in time $O(\varepsilon^{-1} k \log^{2.001} n \log R^*)$. Here, R^* is an upper bound on the $\|\cdot\|_\infty$ norm of the vector, typically assumed to be $\mathrm{poly}(n)$. We emphasize that our approach immediately yields a $\mathrm{poly}(k)$-time algorithm, independent of $\log(n)$ and R^*, and thus generalizes to arbitrary dimension (see next result). This, however, comes at a price of a slower dependence on k, as opposed to the linear dependence of [Kap17].

In high dimensions ($d > 1$), the decoding time of [Kap17] scales as $\log^d(n)$ due to the curse of dimensionality of the Filter function used[5], while our algorithm scales linearly with d.

11.2 Technical Overview

The following section contains a streamlined technical overview of the main ideas and techniques required to prove our results in Section 11.1.1. Section 11.2.1 develops a unified framework for a wide range of Fourier set query-type problems. Section 11.2.2 focuses on implementing the framework for semi-continuous signals estimation and proving Theorem 11.1. Section 11.2.3 focuses on implementing the framework for discrete Fourier set query and proving Theorem 11.2.

11.2.1 A general framework for Fourier set query-type problems

Many Fourier-related problems can be expressed as set query problems in different domains. For example, the sparse Fourier transformation only cares about the Fourier coefficients of a small set of frequencies, which is essentially equivalent to a set query problem in the frequency domain. Another example is recovering the actual Fourier coefficients given its *support* (set of non-zero frequencies). We propose a general framework for efficiently solving such Fourier set-query problems.

To this end, consider the following general form of the problem: Given a signal $x(t)$ (either continuous or discrete), we wish to recover the Fourier coefficients $\widehat{x}$ only

[5]See [NSW19] for more detailed discussions about the time complexity in high dimensions.

at the coordinates of a predetermined set S. The observed signal $x(t)$ can be accessed only through samples, either noiseless or noisy; In the latter case, one is only allowed to access $x(t) + g(t)$ for an arbitrary function $g(t)$.

A natural approach to such problem is to use *linear regression* – Notice that the observed signal $x(t)$ can be decomposed into $x_S(t) + (x_{\overline{S}}(t) + g(t))$, where $x_S(t)$ is a portion of the noiseless signal $x(t)$ with frequencies in S, and $x_{\overline{S}}(t)$ is the remaining part such that $(x_{\overline{S}} + g)(t)$ together can be treated as noise. In this terminology, recovering $\widehat{x}_S$ is equivalent to solving a linear regression problem in the subspace spanned by the *Fourier basis* functions with frequencies in S. Implementing this approach, however, has two substantial challenges:

1. **Sample complexity:** How should one select the sample points $\{t_1, \ldots, t_s\}$ in the time domain to solve the linear regression? Recall we wish to use as few samples as possible.

2. **Estimation accuracy:** How can one guarantee that the solution of the linear regression will not be corrupted by the noise? In other words, how can one ensure the recovered signal is close to the relevant projection $x_S(t)$ of the true signal?

To resolve these two issues, our framework consists of the following four steps, and we note that this framework can be applied to both continuous and discrete signals.

Step 1: Prove an energy bound for the signal The energy bound is an important property of Fourier signals, as it characterizes how far the maximum magnitude of a signal in a family $\mathcal{F}$ can deviate from its energy $\|x\|_T^2 := \frac{1}{|T|}\int_T |x(t)|^2 \mathrm{d}t$, in a given time duration, i.e.,

$$R := \sup_{x \in \mathcal{F}} \frac{\sup_t |x(t)|^2}{\|x\|_T^2},$$

This quantity is of particular interest to us because it captures what is the smallest size of a subset $W \subset T$ such that the signal's energy on W, $\|x\|_W^2 = \frac{1}{|W|}\sum_{t\in W}|x(t)|^2$, is close to its energy $\|x\|_T$ on T, which is a necessary condition for estimation accuracy if we take W as the sample set to solve the regression. As such, the energy bound is closely related to the *sample complexity* of the problem. In [BE06, Kós08, CKPS16, CP19b], energy bounds for one-dimensional continuous Fourier-sparse signals are proved. An appealing property of these energy bounds is that they only depend on the signal's Fourier-sparsity, but not on the time-interval length nor on the frequency band-limit. The first step of our framework therefore entails proving energy bounds for high-dimensional discrete and continuous band-limited signals.

Step 2: Oblivious sketching A straightforward approach for choosing the sample points for the linear regression is to uniformly[6] sample points in the time domain. Let S_0 denote the set of i.i.d uniform samples. An obvious question is how many samples are sufficient to guarantee that S_0 is a good sketch for the signal, i.e., $\|x_S\|_{S_0} \approx \|x_S\|_T$ will hold with high probability? We show that, while this upper bound may be quite large (poly(k)), it can be determined in an *oblivious* fashion: we can upper-bound the number of random samples as a function of only the Fourier sparsity, independent of the actual signal $x(t)$ (!) In this sense, the oblivious sketching step acts as a "preconditioner" for our regression problem – It shrinks the continuous universe to a discrete set of size independent of n and T, which in turn allows to apply (more expensive) data-dependent sketching (Step 3) in poly(k)-time. This is where we use the energy bounds proved in Step 1, along with an ε-net constructed for various kinds of Fourier-sparse signals.

[6]In the one-dimensional case, it is possible to design a more clever *nonuniform* oblivious distribution that achieves near-linear sample complexity [CP19a]. However, in $d \geq 2$ dimensions, the best known construction is a uniform oblivious sketch with polynomially many samples, more on this in Section 11.2.2.

Step 3: Sketch distillation An obvious shortcoming of Step 2 is that S_0 is too large. For example, if we uniformly sample a continuous k-Fourier-sparse signal, we need $\mathrm{poly}(k)$ many samples. This is an artifact of the worst-case nature of the energy bound, which holds for *any* signal in the family $\mathcal{F}$. We resolve this issue by a method called "sketch distillation" to sub-sample a *linear-sized* subset S_1 from S_0, in a data-dependent fashion, such that S_1 is still a good sketch of x_S. More specifically, suppose the frequencies of x_S are given. We can use a *well-balanced sampling procedure* defined by [CP19a] to sample a set of $s = O(k)$ points such that the i-th point t_i is sampled from a distribution D_i supported by S_0, and a coefficient vector $\alpha \in \mathbb{R}^s$. Then, this sub-sampler satisfies the following properties:

- For the weights $w_i := \alpha_i \frac{1/|T|}{D_i(t_i)}$ for $i \in [s]$, we have $\|x\|_{S_1,w} \approx \|x\|_T$ for all signals $x \in \mathcal{F}$, where $\|x\|_{S_1,w} := \sum_{t\in S_1} w_t \cdot |x(t)|^2$ and $\mathcal{F}$ is the signal family spanned by the frequencies in S.
- The sum of coefficients α_i is small and each distribution D_i is not "ill-conditioned"[7].

On the one hand, the first property guarantees that the sketch distillation outputs a sample set S_1 of linear size and $\|x_S\|_{S_1,w} \approx \|x_S\|_T$. On the other hand, the sketch distillation process is *robust to noise*. An easy information-theoretic argument shows that the estimation error must be proportional to the energy of the noise, i.e., $\|x_{\overline{S}} + g\|_T$. However, after the sketch distillation the weighted energy of the noise $\|x_{\overline{S}} + g\|_{S_1,w}$ could be amplified (i.e., $\|x_{\overline{S}} + g\|_{S_1,w} \gg \|x_{\overline{S}} + g\|_T$), resulting in a large estimation error. Fortunately, the second property ensures that it will not happen and in expectation $\|x_{\overline{S}} + g\|_{S_1,w} \lesssim \|x_{\overline{S}} + g\|_T$.

Step 4: Weighted linear regression The last step is to solve a (weighted) linear regression with the noisy samples $\{\widetilde{x}(t_i)\}_{i\in[s]}$ and weight $w \in \mathbb{R}^s_{>0}$. For simplicity,

[7]Formal definition is in Definition 11.10.

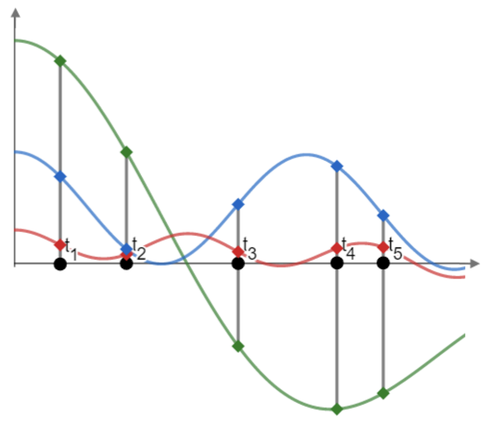


Figure 11.1: An illustration of sketching the set query signals. The green curve is the queried signal x_S, the green curve is the signal with remaining frequencies $x_{\overline{S}}$, and the blue curve is the noise g. Suppose $S_1 = \{t_1, \dots, t_5\}$ are the sampling points. Then, we need to guarantee that $\|x_S\|_{S_1,w} \approx \|x_S\|_T$ as well as $\|x_{\overline{S}} + g\|_{S_1,w} \lesssim \|x_{\overline{S}} + g\|_T$.

take the one-dimensional continuous k-Fourier-sparse signal as an example. Suppose the set-query frequencies of x_S are $S = \{f_1, \dots, f_k\}$. Then, we consider the following weighted linear regression:

$$\min_{v' \in \mathbb{C}^k} \left\| \sqrt{w} \circ (Av' - b) \right\|_2, \tag{11.1}$$

where $\sqrt{w} := (\sqrt{w_1}, \dots, \sqrt{w_s})$, the coefficients matrix $A \in \mathbb{C}^{s \times k}$ and the target vector $b \in \mathbb{C}^s$ are defined as follows:

$$\begin{aligned} A_{i,j} := &\ \exp(2\pi \mathbf{i} f_j t_i) \quad \forall (i,j) \in [s] \times [k], \quad \text{and} \\ b_i := &\ \widetilde{x}(t_i) \quad \forall i \in [s]. \end{aligned}$$

Let v' denote an optimal solution of Eq. (11.1). Then, we output a signal $y(t) := \sum_{i=1}^k v_i' \exp(2\pi \mathbf{i} f_i t)$.

Using the error analysis in sketch distillation (Step 3), we can upper-bound the estimation error by $\|y - x_S\|_T \lesssim \|x_{\overline{S}} + g\|_T$.

11.2.2 Our techniques for signal estimation algorithms

In this section, we show how to instantiate the framework and get Signal Estimation algorithms in the semi-continuous setting (Theorem 11.1).

We first note that given the output of Frequency Estimation algorithm, a set L such that all the true frequencies are close to L, the Signal Estimation problem can be formulated as a Fourier set-query problem. The idea is that by the guarantee of L and the semi-continuous assumption, there will not be too many lattice points that are close to L, and we can efficiently find all of them. Let's denote this set of candidate frequencies by $\widetilde{S}$ of size $\widetilde{k}$. Then, Signal Estimation problem is reduced to a set-query for $\widehat{x^*}_{\widetilde{S}}$. In Section 11.2.2.1, we discuss how to apply our set-query framework to obtain sample-optimal and high-accuracy algorithms for 1-D signals and how to generalize to higher dimensions. In Section 11.2.2.2, we show how to implement Sketch Distillation very efficiently by speeding up the randomized spectral sparsification, which may be of independent interest.

11.2.2.1 Sample-optimal and high-accuracy algorithms

Based on the aforementioned reduction, we briefly show the instantiation of our framework for the Signal Estimation problem. For convenience, suppose $\widetilde{k} = O(k)$ throughout this part.

Sample-optimal signal estimation For one-dimensional k-Fourier-sparse signals, [Kós08] proved that the energy bound is $O(k^2)$, which implies that the uniform sketching for the signal needs at least $\Omega(k^3 \log k)$ samples. Although the sketch size can be reduced to $O(k)$ via Sketch Distillation, the algorithm has to first sample $\widetilde{O}(k^3)$ points in $[0, T]$, which already takes $O(k^3)$-time. It is possible to improve this straightforward approach using *weighted oblivious sketching* [CP19a], namely, we sample a set S_0 of points in $[0, T]$ from a carefully-chosen non-uniform distribution D, and assign each point a weight (this works only in 1D, see last section). Using the following distribution constructed by [CP19a]:

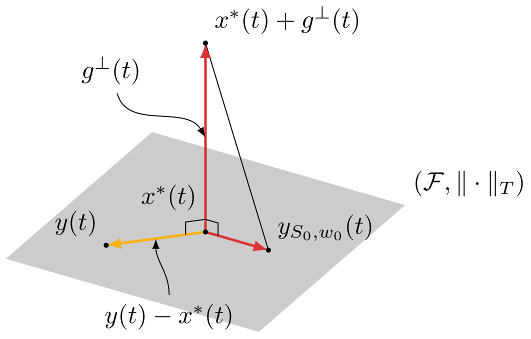
$$D(t) := \begin{cases} c/(1-|t/T|), & \text{for } |t| \le T(1-1/k) \\ c \cdot k, & \text{for } |t| \in [T(1-1/k), T] \end{cases}$$

we only need to take $|S_0| = O(k \log k)$ samples to guarantee that $\|x^*\|_{S_0,w_0} \approx \|x^*\|_T$ holds with high probability. In this way, oblivious sketching in Step 2 will not be a time-consuming step. In Step 3, the sketch S_0 will be distilled to a subset S_1 of size $O(k)$. Finally, in Step 4, we sample the signal at the points in S_1 and solve the weighted linear regression to recover a k-sparse signal $y(t)$. This gives us a *linear-sample* reduction from Frequency Estimation to Signal Estimation with $O(1)$-estimation error with high probability.

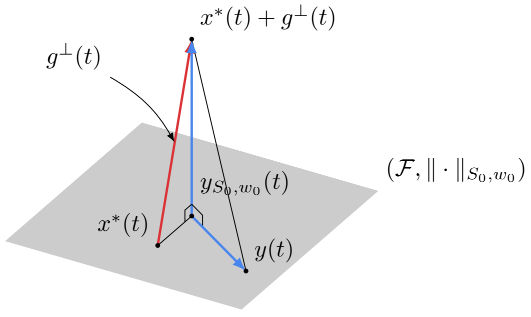
High-accuracy signal estimation For one-dimensional semi-continuous signals, we also discover a "sample-accuracy trade-off". If we can use nearly-linear (i.e., $\widetilde{O}(k)$) samples, then we can skip Step 3 and directly use (S_0, w_0) to solve the linear regression in Step 4.

The advantage of this approach is, the sampling procedure for (S_0, w_0) is well-balanced, and a sharper error analysis shows that we can achieve much smaller errors. The main observation is that the noise can be decomposed into $g^{\parallel} \in \mathcal{F}$ and $g^{\perp}$ orthogonal to $\mathcal{F}$. Since we will solve a linear regression (in Step 4) in the space $\mathcal{F}$, the contribution of $g^{\parallel}$ to the final estimation error will not blow-up. For the orthogonal part, we find an orthonormal basis $\{u_1, \dots, u_k\}$ for $\mathcal{F}$ and look at the weighted projection $\langle g^{\perp}, u_i \rangle_{S_1,w} := \sum_{t \in S_1} w_t \overline{u_i(t)} g^{\perp}(t)$, whose magnitude indicates how much the noise is amplified due to sketching in each direction. The well-balanced sampling procedure gives that the total magnitudes $\sum_{i=1}^{k} |\langle g^{\perp}, u_i \rangle_{S_1,w}|^2 \le \varepsilon \cdot \|g^{\perp}\|_T^2$ holds with high probability, which will imply an $\varepsilon \cdot \|x_{\overline{S}} + g\|_T^2$-estimation error for any small ε.

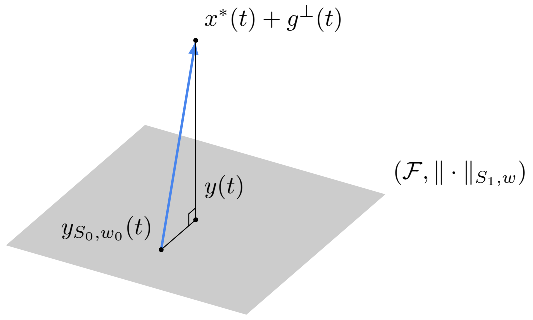
Then, in step 4, we can also decompose the final estimation error $\|y - x^*\|_T^2$ into the two parts: one contributed by $g^{\parallel}$ with energy $\|g^{\parallel}\|_T^2$, and another contributed by



(a) The function space with norm $\|\cdot\|_T$



(b) The function space with norm $\|\cdot\|_{S_0,w_0}$



(c) The function space with norm $\|\cdot\|_{S_1,w}$

Figure 11.2: The composition of two WBSPs may not be a WBSP. In (a), the central point is the ground-truth signal $x^*(t)$, and the yellow vector $y(t)$ is the final output of the composited WBSPs. Our goal is to bound the estimation error $\|y(t) - x^*(t)\|_T$. Let $y_{S_0,w_0}(t) \in \mathcal{F}$ be the optimal approximation of the observing signal in the basis (S_0, w_0), which is obtained by Step 2 of the framework. In (b), we show that by the property of the first WBSP, the estimation error $\|y_{S_0,w_0}(t) - x^*(t)\|_T$ can be bounded by the projection of $g^\perp$ in the space $(\mathcal{F}, \|\cdot\|_{S_0,w_0})$, which is about $\varepsilon\|g^\perp(t)\|_T$. In (c), the space is reduced to $(\mathcal{F}, \|\cdot\|_{S_1,w_1})$ by Step 3 of the framework, and the estimation error $\|y_{S_0,w_0} - y\|_{S_1,w_1}$ is bounded by the projection of $(x^*(t) + g^\perp - y_{S_0,w_0})$, which is about $\varepsilon\|x^*(t) + g^\perp - y_{S_0,w_0}(t)\|_{S_0,w_0}$. However, these two error bounds $\varepsilon\|g^\perp(t)\|_T$ and $\varepsilon\|x^*(t) + g^\perp - y_{S_0,w_0}(t)\|_{S_0,w_0}$ could not imply any bound on $\|y(t) - x^*(t)\|_T$.

$g^\perp$ with energy at most $(1+\varepsilon)\|g^\perp\|_T^2$. Combining the two parts together, we achieve a a high-accuracy guarantee:

$$\|y - x^*\|_T^2 \le (1+\varepsilon)\|g\|_T^2.$$

We remark that this error analysis does not work for the sample-optimal algorithm. Roughly speaking, the weighted sketch (S_1, w) in the sample-optimal algorithm is generated by a two-step sampling procedure: one in the oblivious sketching (Step 2) and another in the sketch distillation (Step 3). Even if both of them are well-balanced, when they are composited together and considered as a single sampling procedure, they may not be well-balanced.

High-dimensional signal estimation For high dimensional signals, we prove a $k^{O(d)}$-energy bound for k-Fourier-sparse signals in d-dimensions in Section 11.5.2, where we also provide a nearly-matching lower bound. We view it as a new generic tool in Fourier analysis and are quite a tour-de-force. Thus, we basically follow the 4-step framework. One tricky thing in high-dimension is, how to bound the output signal's sparsity, which is equivalent to the number of d-dimensional lattice points close to a fixed set L. We further reduce it to a clean math problem about lattices: let $\Lambda(\mathcal{B})$ be a lattice in $\mathbb{R}^d$. For $r > 0$, how many lattice points can be within an r-radius ball, i.e., $\sup_{x \in \mathbb{R}^d} |\Lambda(\mathcal{B}) \cap B_d(x, r)|$. We upper-bound this quantity via different approaches, which might be of independent interest. More details are deferred to the appendix.

11.2.2.2 Speed up randomized spectral sparsification

In Section 11.2.2.1, we do not discuss how to implement a well-balanced sampling procedure. [CP19a] proved that *Randomized BSS* algorithm in [BSS12, LS15] yields a well-balanced sampling procedure. However, this algorithm is slow when the sampling domain S_0 is very large. One contribution of this work is to improve the time and space costs of the randomized BSS algorithm.

We observe that the bottleneck of each iteration in the original Randomized BSS algorithm is to sample a point $t \in S_0$ from the distribution D_j defined by $D_j(t) = v(t)^\top E_j v(t)$, where $v(t)$ is a k-dimensional vector and E_j is a k-by-k positive semi-definite matrix determined by the potential function value at the j-th iteration. Suppose E_j and $\{v(t)\}_{t \in S_0}$ have already been computed. A naive approach to sampling from D_j is to compute the probability $D_j(t)$ for each $t \in S_0$, which takes $O(nk^2)$-time per iteration, where $n = |S_0|$ is the size of the sampling domain. To improve the algorithm, we consider a more general data structure problem—*Online Quadratic-Form Sampling.* In this problem, we are given n vectors $v_1, \dots, v_n \in \mathbb{R}^k$. In each query, the input is a positive semi-definite matrix $A \in \mathbb{R}^{k \times k}$ and we need to sample an $i \in [n]$ with probability proportional to $v_i^\top A v_i$. (See Problem 11.36 for formal definition). The naive space and query time for generating a sample is $O(nk^2)$. We design two data structures with substantially faster time-space tradeoff:

Theorem 11.3 (Online Quadratic-Form Sampling, informal version of Theorems 11.37 and 11.40)**.** *The Online Quadratic-Form Sampling problem admits the following two data structures:*

- **Data Structure 1:** *$O(nk^2)$ preprocessing time of the vectors $\{v_i\}_{i\in[n]}$, $O(k^2 \log n)$ query-time for generating a sample, and $O(nk^2)$-space.*
- **Data Structure 2:** *$O(nk^{\omega-1})$ preprocessing time, $O(k^2 \log(n/k) + k^\omega)$ query-time for generating a sample, and $O(nk)$-space.*

The main idea is to construct a range search tree for $\{1, 2, \dots, n\}$, and for a node corresponding to range $[l, r]$, stores the matrix $\sum_{i=l}^r v_i v_i^\top$. Subsequently, for each query matrix A, we traverse the root-to-leaf path in the tree (which corresponds to an element in $[n]$), and output this element as a sample. The rule for descendin the tree resembles a form of rejection sampling: At a node with range $[l, r]$, we know that its left child has range $[l, m]$ and right child has right $[m + 1, r]$, where $m = \lfloor (l + r)/2 \rfloor$. Then, we decide whether move to the left or the right subtree by tossing a coin with probability:

$$p_{\mathsf{left}} := \frac{\langle \sum_{i=l}^m v_i v_i^\top, A \rangle}{\langle \sum_{i=l}^r v_i v_i^\top, A \rangle}, \quad \text{and} \quad p_{\mathsf{right}} := 1 - p_{\mathsf{left}},$$

where $\langle \sum_{i=l}^r v_i v_i^\top, A \rangle := \sum_{i=l}^r v_i^\top A v_i$ is the trace-product of matrices. Therefore, the probability p_{left} equals to the *conditional* probability $\Pr_{\mathbf{i} \sim \mathcal{D}_A}[\mathbf{i} \in [l, m] \mid \mathbf{i} \in [l, r]]$.

By the chain rule of conditional probability, we get that the output distribution of this procedure is exactly equal to $\mathcal{D}_A$. This data structure can be built in $O(nk^2)$ and each query only takes $O(k^2 \log n)$-time. To store the matrices in each node, this data structure uses $O(nk^2)$-space. We can further improve the space complexity to $O(nk)$ by trading-off the preprocessing time and the query time. By plugging-in this data structure to the Randomized BSS algorithm, we can improve the time and space complexity by a factor of k when n is large.

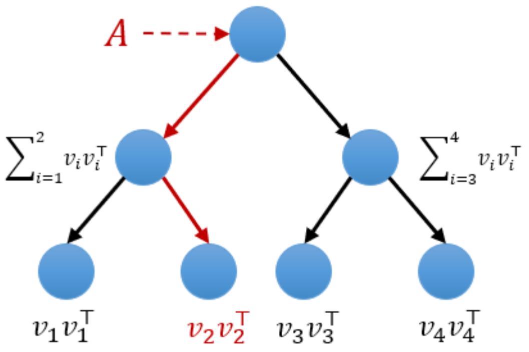


Figure 11.3: An example of the outer-product range tree with $n = 4$. For a query matrix A, the sampling probability of the red path is $\frac{\langle \sum_{i=1}^2 v_i v_i^\top, A\rangle}{\langle \sum_{i=1}^4 v_i v_i^\top \rangle} \cdot \frac{\langle v_2 v_2^\top, A\rangle}{\langle \sum_{i=1}^2 v_i v_i^\top, A\rangle} = \frac{v_2^\top A v_2}{\sum_{i=1}^4 v_i^\top A v_i}$.

11.2.3 Our techniques for discrete Fourier set query

For simplicity, we consider one-dimensional discrete Fourier signal $x(t)$, which is a length-n vector such that $x(t) = \sum_{j=1}^n \widehat{x}_j e^{2\pi \mathbf{i} jt/n}$ for any $t \in [n]$. And the set query problem asks to recover $\widehat{x}_S$, for a given k-subset $S \subset [n]$. Let x_S be the part of signal with frequencies in S, i.e., $x_S(t) = \sum_{f\in S} \widehat{x}_f e^{2\pi \mathbf{i} ft/n}$, which is a k-Fourier-sparse signal.

The high-level idea of obtaining Theorem 11.2 is as follows. We prove that the energy bound for discrete k-sparse signal is k, which implies that uniformly sample a set $S_0 \subset [n]$ of $O(k \log k)$ points can form a good sketch of the signal x_S. Then, by Sketch Distillation, we can find a subset $S_1 \subset S_0$ of linear size together with a weight vector w such that $\|x_S\|_{S_1,w} \approx \|x_S\|_2/n$. Finally, we can recover $\widehat{x}_S$ by solving a weighted linear regression on the samples $\{x(t)\}_{t\in S_1}$. By a direct error analysis, it is easy to see that this algorithm can achieve $O(1)$-estimation error. In the followings, we will show that it can perform much better.

Composition of well-balanced samplers The key step to proving a $(1+\varepsilon)$-error guarantee is to show that the final weighted sketch (S_1, w) can be generated by a well-balanced sampling procedure. In general, compositing two well-balanced sampling procedures may not be well-balanced. More specifically, we can show that the composition sampler satisfies the first property of a well-balance, that is, the output set and weight can well-approximate the energy of every function in the family $\mathcal{F}$. However, the second property about the sum of composition sampler's coefficients and the condition number of the composition sampling distribution may not hold.

For the above discrete set query algorithm, it is a very special case of composition in the sense that the first sampler sample each element from the same, simplest distribution—uniform distribution over $[n]$. Then, we can prove that in the composition sampler, each sample is equivalent to directly sampled from uniform distribution. Furthermore, for discrete Fourier-sparse signal, we also have a tight energy bound of $R = k$. Using these results, we can show that the composition sampler in our algorithm is well-balanced! Then, we are able to apply our sharper error analysis and get that

$$\|\widehat{y}_S - \widehat{x}_S\|_2 \leq \varepsilon \cdot \|\widehat{x}_{\overline{S}}\|_2^2$$

holds with high probability. Therefore, we obtain a linear-sample and high-accuracy algorithm for discrete Fourier set query.

Remark 11.4. This result is a fundamental departure from [CP19a] since they assumed that the noise has a zero-mean (i.e., $\mathbb{E}[g(t)] = 0$), which makes the WBSPs composition much easier. However, we do not have such an assumption. A natural fix is to ensure the orthogonality of the noise $g(t)$ to our basis functions $\{\exp(-2\pi\mathbf{i}f_i t/n)\}$. Unfortunately, this only holds for the first WBSP, but not for the second one (since sampling from the weighted output of the first WBSP can break the orthogonality).

Our main novelty is to directly find an equivalent sampling procedure to the combination of two WBSPs (just for the analysis). Then, using the special properties of DFT, we prove that the composition also works in our setting (see Section 11.11.2 Lemma 11.76 for more details). We believe this technique will be useful for other set-query or active learning problems.

Algorithm 50 Discrete One-Dimensional Signal Set-Query Algorithm (Informal)

```
1: procedure SetQuery($x$, $n$, $k$, $S$, $\varepsilon$)    ▷ Theorem 11.2
2:     $\{f_1, f_2, \cdots, f_k\} \leftarrow S$
   /*Step 2: Oblivious Sketching*/
3:     $S_0 \leftarrow O(\varepsilon^{-2}k\log(k))$ i.i.d. samples from Uniform($[n]$)
   /*Step 3: Sketch Distillation*/
4:     Let $\mathcal{F} = \{\sum_{j\in[k]} v_j \exp(2\pi\mathbf{i}f_j t/n) \mid v_j \in \mathbb{C}\}$.
5:     $\{t_1, t_2, \cdots, t_s\}, w \leftarrow$ RandBSS+$(k, \mathcal{F}, \mathrm{Uniform}(S_0), (\varepsilon/4)^2)$    ▷ Algorithm 52
   /*Step 4: Weighted linear regression*/
6:     $A_{i,j} \leftarrow \exp(2\pi\mathbf{i}f_j t_i/n)$ for each $(i,j) \in [s] \times [k]$
7:     $b_i \leftarrow x(t_i)$ for each $i \in [s]$    ▷ Observe the signal at time $t_i$
8:     $v' \leftarrow \arg\min_{v'\in\mathbb{C}^k} \|\sqrt{w} \circ (Av' - b)\|_2$
9:     return $v'$
10: end procedure
```

Organization

The remainder of the chapter is organized as follows. In Sections 11.3 we formally define and study the "semi-continuous" Fourier interpolation and set query problems over lattices. And in Section 11.4, we provide some preliminaries on Fourier transformation, lattices, etc.

Then, we focus on developing Fourier set query algorithms based on our 4-step framework (Section 11.2.1). We first build some technical components in Sections 11.5

- 11.8. Section 11.5 is for Step 1, where we prove energy bounds and concentration properties for high-dimensional Fourier-sparse signals and discrete Fourier-sparse signals. Section 11.6 is for Step 2, where we describe fast *oblivious* sketching methods and we use them as "preconditioners" for continuous and discrete Fourier-sparse signals in one and higher dimensions. Sections 11.7 and 11.8 are for Step 3. More specifically, in Sections 11.7, we design a data structure for improving the time/space complexity of the Randomized BSS algorithm (Theorem 11.3). Using this data structure, in Section 11.8 we describe the *sketch distillation* technique for different kinds of Fourier-sparse signals and analyze its robustness to noise. Finally, these components are wrapped up in Sections 11.9 - 11.11. Section 11.9 shows sample-optimal and high-accuracy signal estimation algorithms for one-dimensional semi-continuous signals (Theorem 11.1). Section 11.10 generalizes to high-dimensional signals using lattice theory. Section 11.11 gives a discrete Fourier set query algorithm (Theorem 11.2).

Section 11.12 discusses a special case of the Signal Estimation problem where the observation has no noise. We present a straightforward algorithm for one-dimensional Fourier sparse signals. Section 11.13 proves that any signal can be approximated by a semi-continuous signal with the same sparsity and polynomially-small frequency gap[8], which implies a Fourier interpolation algorithm with optimal output-sparsity with a different error guarantee.

[8]We also show an approximation with $O(1)$-frequency gap but slightly worse sparsity.

Notations. For any positive integer n, we use $[n]$ to denote $\{1, 2, \cdots, n\}$. We use $\mathbf{i}$ to denote $\sqrt{-1}$. For a complex number $z \in \mathbb{C}$ where $z = a + \mathbf{i}b$ and $a, b \in \mathbb{R}$. We use $\overline{z}$ to denote the complex conjugate of z, i.e., $\overline{z} = a - \mathbf{i}b$. Then it is obvious that $|z|^2 = z \cdot \overline{z} = a^2 + b^2$. We use $f \lesssim g$ to denote that there exists a constant C such that $f \leq Cg$, and $f \eqsim g$ to denote $f \lesssim g \lesssim f$. We use $\widetilde{O}(f)$ to denote $f \log^{O(1)}(f)$. We say $x(t)$ is a k-Fourier-sparse when $x(t) = \sum_{j=1}^{k} v_j \exp(2\pi \mathbf{i} f_j t)$. We use $\widehat{x}(f)$ to denote the Fourier transform of $x(t)$. More specifically, $\widehat{x}(f) = \int_{-\infty}^{\infty} x(t) \exp(-2\pi \mathbf{i} f t) \mathrm{d}t$. We define our discrete norm as $\|g(t)\|_W^2 = \frac{1}{|W|} \sum_{t \in W} |g(t)|^2$ for function g. We define our weighted discrete norm as $\|g(t)\|_{S,w}^2 = \sum_{t \in S} w_t |g(t)|^2$ for function g. We define the continuous T-norm as $\|g(t)\|_T^2 = \frac{1}{T} \int_0^T |g(t)|^2 \mathrm{d}t$ for function g.

In general, we assume $x^*(t)$ is our ground truth and is a k-Fourier-sparse signal. We can observe function $x(t) = x^*(t) + g(t)$ for $g(t)$ being a noise function. We can observe $x(t)$ in duration $[0, T]$. The ground truth $x^*(t)$ has frequencies in $[-F, F]$.

11.3 Definitions of Semi-Continuous Fourier Set Query and Interpolation

In this section, we give the formal definitions of the problems studied in this chapter. In Section 11.3.1, we define the Fourier set query for discrete and continuous signals. In Section 11.3.2, we define the Fourier interpolation problem and its two sub-problems: frequency estimation and signal estimation.

11.3.1 Formal definitions of Fourier set query

The discrete Fourier set query problem is defined as follows:

Definition 11.1 (Discrete Fourier set query problem)**.** Let $x \in \mathbb{C}^n$ and $\widehat{x}$ be its discrete Fourier transformation. Let $\varepsilon > 0$. Given a set $S \subseteq [n]$ and query access to x, the goal is to use a few queries to compute a vector x' with support $\mathrm{supp}(x') \subseteq S$

such that

$$\|(x' - \widehat{x})_S\|_2^2 \leq \varepsilon \cdot \|\widehat{x}_{[n] \setminus S}\|_2^2.$$

We also define the continuous Fourier set query problem as follows:

Definition 11.2 (Continuous Fourier set query problem)**.** For $d \geq 1$, let $x^*(t)$ be a signal in time duration $[0, T]^d$. Let $\widehat{x^*}(f)$ denote the continuous Fourier transformation of $x^*(t)$. Let $\varepsilon > 0$. Given a set $S \subseteq \mathbb{R}^d$ of frequencies such that $\mathrm{supp}(\widehat{x^*}) \subseteq S$, and observations of the form $x(t) = x^*(t) + g(t)$, where $g(t)$ denotes the noise. The goal is to output a Fourier-sparse signal $x'(t)$ with support $\mathrm{supp}(\widehat{x'}) \subseteq S$ such that

$$\|x' - x^*\|_T^2 \leq (1 + \varepsilon) \cdot \|g\|_T^2.$$

11.3.2 Formal definitions of semi-continuous Fourier interpolation

In this section, we provide the following formal definition of the semi-continuous Fourier interpolation problem, where we assume that the frequencies of the signal are contained in a lattice.

Problem 11.4 (Semi-continuous Fourier interpolation problem)**.** Given a basis $\mathcal{B}$ of m known vectors $b_1, b_2, \cdots b_m \in \mathbb{R}^d$, let $\Lambda(\mathcal{B}) \subset \mathbb{R}^d$ denote the lattice

$$\Lambda(\mathcal{B}) = \Big\{ z \in \mathbb{R}^d : z = \sum_{i=1}^m c_i b_i, c_i \in \mathbb{Z}, \forall i \in [m] \Big\}$$

Suppose that $f_1, f_2, \cdots, f_k \in \Lambda(\mathcal{B})$, $\forall i \in [k], |f_i| \leq F$. Let $x^*(t) = \sum_{j=1}^k v_j e^{2\pi \mathbf{i} \langle f_j, t \rangle}$, and let $g(t)$ denote the noise. Given observations of the form $x(t) = x^*(t) + g(t)$, $t \in [0, T]^d$. Let $\eta = \min_{i \neq j} \|f_j - f_i\|_\infty$. There are three goals:

1. The first goal is to design an algorithm that output $f_1, f_2, \cdots, f_k$ *exactly* given query access to the signal $x(t)$ for $t \in [0, T]^d$.

2. The second goal is to design an algorithm that output a set L of frequencies such that, for each f_i, there is $f_i' \in L$, $\|f_i - f_i'\|_2 \leq D/T$.

3. The third goal is to design an algorithm that output $y(t) = \sum_{j=1}^{\widetilde{k}} v'_j \cdot e^{2\pi\mathbf{i} f'_j t}$ such that $\int_{[0,T]^d} |y(t) - x(t)|^2 \mathrm{d}t \lesssim \int_{[0,T]^d} |g(t)|^2 \mathrm{d}t$.

Then, we extract two sub-problems from Problem 11.4: Frequency Estimation and Signal Estimation. We give their definitions below.

We first define the d-dimensional frequency estimation under the semi-continuous as follows. In this problem, we want to recover each frequencies in a small range.

Problem 11.5 (Frequency estimation)**.** Given a basis $\mathcal{B}$ of m known vectors $b_1, b_2, \cdots b_m \in \mathbb{R}^d$, let $\Lambda(\mathcal{B}) \subset \mathbb{R}^d$ denote the lattice

$$\Lambda(\mathcal{B}) = \Big\{ z \in \mathbb{R}^d : z = \sum_{i=1}^{m} c_i b_i, c_i \in \mathbb{Z}, \forall i \in [m] \Big\}$$

Suppose that $f_1, f_2, \cdots, f_k \in \Lambda(\mathcal{B})$. Let $x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi\mathbf{i}\langle f_j, t\rangle}$, and let $g(t)$ denote the noise. Given observations of the form $x(t) = x^*(t) + g(t)$, $t \in [0,T]^d$. Let $\eta = \min_{i \neq j} \|f_j - f_i\|_\infty$.

The goal is to design an algorithm that output a set L of frequencies such that, for each f_i, there is $f'_i \in L$, $\|f_i - f'_i\|_2 \leq D/T$.

We remark that the recovered frequencies in L are not necessary to be in $\Lambda(\mathcal{B})$, and D is a parameter that can depend on k.

Next, we define the d-dimensional Signal Estimation under the semi-continuous setting as follows. In this problem, we want to recover a signal that can approximate the ground-truth signal in the time domain.

Problem 11.6 (Signal Estimation problem)**.** Given a basis $\mathcal{B}$ of m known vectors $b_1, b_2, \cdots b_m \in \mathbb{R}^d$, let $\Lambda(\mathcal{B}) \subset \mathbb{R}^d$ denote the lattice

$$\Lambda(\mathcal{B}) = \Big\{ z \in \mathbb{R}^d : z = \sum_{i=1}^{m} c_i b_i, c_i \in \mathbb{Z}, \forall i \in [m] \Big\}$$

Suppose that $f_1, f_2, \cdots, f_k \in \Lambda(\mathcal{B})$. Let $x^*(t) := \sum_{j=1}^{k} v_j e^{2\pi\mathbf{i}\langle f_j, t\rangle}$, and let $g(t)$ denote the noise. Given observations of the form $x(t) = x^*(t) + g(t)$, $t \in [0,T]^d$. Let $\eta = \min_{i\neq j} \|f_j - f_i\|_\infty$.

The goal is to design an algorithm that outputs $y(t) = \sum_{j=1}^{\widetilde{k}} v'_j \cdot e^{2\pi\mathbf{i}f'_j t}$ such that

$$\int_{[0,T]^d} |y(t) - x(t)|^2 \mathrm{d}t \lesssim \int_{[0,T]^d} |g(t)|^2 \mathrm{d}t.$$

Note that outputting $y(t) = \sum_{j=1}^{\widetilde{k}} v'_j \cdot e^{2\pi\mathbf{i}f'_j t}$ means outputting $\{v'_j, f'_j\}_{j\in[\widetilde{k}]}$.

Remark 11.5. We note that given the solution of Frequency Estimation (Problem 11.5), Signal Estimation (Problem 11.6) can be formulated as a Fourier set query problem (Problem 11.2). More specifically, by Frequency Estimation, we will find a set that contains all frequencies of the ground truth signal $x^*(t)$. Then, we only need to recover the coefficients with frequencies in this set, which is equivalent to a set query problem.

11.4 Preliminaries

This section is organized as follows. In Section 11.4.1, we provide some technical tools in probability theory and linear algebra. In Section 11.4.2, we review the Fourier transformation for different types of signals. In Section 11.4.3, we show some facts about Lattices. And in Section 11.4.4, we discuss the importance sampling method.

11.4.1 Tools and inequalities

Definition 11.3 (ε-net)**.** Let T be a metric space with distance measure d. Consider a subset $K \subset T$ and let $\varepsilon > 0$. A subset $\mathcal{N} \subseteq K$ is called an ε-net of K if every point in K is within distance ε of some point of $\mathcal{N}$, i.e.

$$\forall x \in K, \exists y \in \mathcal{N} \text{ s.t. } d(x,y) \leq \varepsilon.$$

Fact 11.7 (Fast matrix multiplication)**.** *We use $\mathcal{T}_{\mathrm{mat}}(a,b,c)$ to denote the time of multiplying an $a \times b$ matrix with another $b \times c$ matrix.*

We use ω to denote the exponent of matrix multiplication, i.e., $\mathcal{T}_{\mathrm{mat}}(n,n,n) = n^{\omega}$. Currently $\omega \approx 2.373$ [Wil12, LG14, AW21].

Fact 11.8 (Weighted linear regression)**.** *Given a matrix $A \in \mathbb{C}^{n\times d}$, a vector $b \in \mathbb{C}^n$ and a weight vector $w \in \mathbb{R}^n_{>0}$, it takes $O(nd^{\omega-1})$ time to output an x' such that*

$$x' = \arg\min_x \|\sqrt{W}(Ax-b)\|_2 = (A^*WA)^{-1}A^*Wb.$$

where $\sqrt{W} := \mathrm{diag}(\sqrt{w_1}, \ldots, \sqrt{w_n}) \in \mathbb{R}^{n\times n}$, and $\omega \approx 2.373$ is the exponent of matrix multiplication [Wil12, LG14, AW21].

Fact 11.9. *For any $x \in (0,1)$, we have $\cos(x) \le \exp(-x^2/2)$.*

11.4.2 Basics of Fourier transformation

The definition of high dimensional Fourier transform is as follows:

$$\widehat{x}(f) = \int_{(-\infty,\infty)^d} x(t)\exp(-2\pi\mathbf{i}\langle f,t\rangle)\mathrm{d}t, \text{ where } f \in \mathbb{R}^d,$$

and the definition of high dimensional inverse Fourier transform is as follows:

$$x(t) = \int_{(-\infty,\infty)^d} \widehat{x}(f)\exp(2\pi\mathbf{i}\langle f,t\rangle)\mathrm{d}f, \text{ where } t \in \mathbb{R}^d.$$

Note that when we replace $d = 1$ in the definition of high dimensional Fourier transform and inverse Fourier transform above, we get the definition of one-dimensional Fourier transform and inverse Fourier transform.

The definition of discrete Fourier transform is as follows:

$$\widehat{x}_f = \sum_{t=1}^{n} x_t \exp(-2\pi\mathbf{i}ft/n), \text{ where } f \in [n],$$

and the definition of discrete inverse Fourier transform is as follows:

$$x_t = \frac{1}{n}\sum_{f=1}^{n} \widehat{x}_f \exp(2\pi\mathbf{i}ft/n), \text{ where } t \in [n].$$

A continuous k-Fourier sparse signal $x(t) : \mathbb{R}^d \to \mathbb{C}$ can be represented as follows:

$$x(t) = \sum_{j=1}^{k} v_j \exp(2\pi \mathbf{i} \langle f_j, t\rangle),\ v_j \in \mathbb{C}, f_j \in \mathbb{R}^d,\ \forall j \in [k].$$

Thus, $\widehat{x}(f)$ is:

$$\widehat{x}(f) = \sum_{j=1}^{k} v_j \delta(t - f_j).$$

A discrete k-Fourier sparse signal $x \in \mathbb{C}^n$ can be represented as follows:

$$x_t = \sum_{j \in S} v_j \exp(2\pi \mathbf{i} jt/n),\ S \subseteq [n], |S| = k,\ v_j \in \mathbb{C}, \forall j \in S.$$

So, $\widehat{x}_f$ is:

$$\widehat{x}_f = \begin{cases} v_j & , j \in S \\ 0 & , \text{o.w.} \end{cases}$$

11.4.3 Facts about lattices

Definition 11.4 (Lattice)**.** A lattice $\mathcal{L}$ in $\mathbb{R}^d$ is defined as follows:

$$\mathcal{L} := \Big\{ \sum_{i=1}^{k} \lambda_i b_i : \lambda_1, \dots, \lambda_k \in \mathbb{Z} \Big\},$$

where $b_1, \dots, b_k \in \mathbb{R}^d$ are linearly independent vectors. And we denote the matrix $B := \begin{bmatrix} b_1 & \cdots & b_k \end{bmatrix} \in \mathbb{R}^{n \times k}$ as the basis of the lattice $\mathcal{L}$.

Definition 11.5 (Fundamental parallelepiped)**.** For a lattice $\mathcal{L}$ with basis B, its fundamental parallelepiped is defined to be:

$$\mathcal{P}(B) := \{ Bx \mid x \in [0, 1)^d \}.$$

Fact 11.10. *For any lattice with basis B, we have*

$$\mathrm{vol}(\mathcal{P}(B)) = \sqrt{\det(B^\top B)}.$$

In particular, if B is full-rank, $\mathrm{vol}(\mathcal{P}(B)) = |\det(B)|$.

Lemma 11.11 (The number of lattice points within a ball[9]). *Let $\mathcal{L}$ be any lattice with basis B such that the spectral norm $\|B\| \leq \ell$. Then, the number of lattice points inside a ball centered at 0 with radius R is upper bounded by:*

$$|\mathcal{L} \cap B_d(0,R)| \leq (1 + \frac{\sqrt{k}\ell}{R})^k \cdot \frac{\mathrm{vol}(B_k(0,R))}{\mathrm{vol}(\mathcal{P}(B))}.$$

Proof. We first show that for two different lattice points $x, y \in \mathcal{L} \cap B_d(0,R)$, the translations of $\mathcal{P}$ at x and at y are disjoint, i.e., $(x + \mathcal{P}) \cap (y + \mathcal{P}) = \emptyset$.

Suppose $(x + \mathcal{P}) \cap (y + \mathcal{P}) \neq \emptyset$ for some $x, y \in \mathcal{L} \cap B_d(0,R)$. Then, we have

$$x + \sum_{i=1}^{k} \lambda_i b_i = y + \sum_{i=1}^{k} \tau_i b_i,$$

where $\lambda_i, \tau_i \in [0,1)$. It gives

$$x - y = \sum_{i=1}^{k} (\tau_i - \lambda_i) b_i.$$

Note that $x - y \in \mathcal{L}$, which means

$$\sum_{i=1}^{k} (\tau_i - \lambda_i) b_i = \sum_{i=1}^{k} c_i b_i,$$

where $c_1, \ldots, c_k \in \mathbb{Z}$. Since $\tau_i - \lambda_i \in (-1,1)$, we get that $c_i = 0$ for all $i \in [k]$. Thus, $x = y$.

Then, for any point $y \in x + \mathcal{P}$, where $x \in \mathcal{L} \cap B_d(0,R)$, we have

$$\|y\|_2 = \|x + z\|_2 \leq \|x\|_2 + \|z\|_2 \leq R + \sqrt{k}\ell, \tag{11.2}$$

where $z \in \mathcal{P}$ and the last step follows from $x \in B_d(0,R)$ and $\|z\|_2 = \|B\lambda\|_2 \leq \|B\|\|\lambda\|_2 \leq \sqrt{k}\ell$, for some $\lambda \in [0,1)^k$.

[9]We thank Thomas Rothvoss for providing the proof of this bound.

Then, we have

$$\begin{aligned}|\mathcal{L} \cap B_d(0,R)| &\leq \sum_{x \in \mathcal{L} \cap B(0,R)} \frac{\mathrm{vol}(x+\mathcal{P})}{\mathrm{vol}(\mathcal{P}(B))} \\ &\leq \frac{\mathrm{vol}(B_k(0, R+\sqrt{k}\ell))}{\mathrm{vol}(\mathcal{P}(B))} \\ &\leq (1+\frac{\sqrt{k}\ell}{R})^k \cdot \frac{\mathrm{vol}(B_k(0,R))}{\mathrm{vol}(\mathcal{P}(B))},\end{aligned}$$

where the second step follows from the disjointness of translations and the bound on the total width (Eq. (11.2)).

The lemma is then proved. □

We define the shortest vector problem (SVP) as follows:

Definition 11.6 (SVP)**.** Let $\mathcal{L}$ denote a Lattice. We define $\mathrm{SVP}(\mathcal{L})$,

$$\mathrm{SVP}(\mathcal{L}) := \min\{\|x\|_2 \mid x \in \mathcal{L} \backslash \{\mathbf{0}\}\}.$$

Given a basis of $\mathcal{L}$, the goal is to compute $\mathrm{SVP}(\mathcal{L})$.

In fact, compute SVP (or even approximations of SVP) is an NP-hard problem. The following theorem shows a well-known lower bound for the shortest vector length.

Theorem 11.12 (Theorem 1.10 in [Rot16], a lower bound on shortest vector)**.** *Let $\mathcal{L}$ denote a lattice with basis $\mathcal{B}$. Let $(b_1^*, \cdots, b_n^*)$ be its Gram-Schmit orthogonalization. Then*

$$\mathrm{SVP}(\mathcal{L}) \geq \min_{i \in [n]} \|b_i^*\|_2$$

Fact 11.13. *The Gram–Schmidt process takes a finite, linearly independent set of vectors $S = \{v_1, \cdots, v_k\}$ for $k \leq n$, runs $O(nk^2)$ time, and generates an orthogonal set $S' = \{u_1, \cdots, u_k\}$ that spans the same k-dimensional subspace of $\mathbb{R}^n$ as S.*

11.4.4 Facts about importance sampling

Important sampling try to estimate a statistic value in one distribution by taking samples in another distribution. In particular, [CP19a] considered the importance sampling for estimating the norm of functions in a linear family $\mathcal{F}$.

In this followings, we first provide some basic definitions about linear function family.

Definition 11.7 (Condition number of sampling distribution)**.** Let G be any domain and $\mathcal{F}$ is a linear function family from G to $\mathbb{C}$. Let D be an arbitrary distribution over G. Then the condition number of D with respect to $\mathcal{F}$ is defined as follows:

$$K_D := \sup_{t \in G} \sup_{f \in \mathcal{F}} \frac{|f(t)|^2}{\|f\|_D^2},$$

where

$$\|f\|_D^2 := \int_G D(t) \cdot |f(t)|^2 \mathrm{d}t.$$

Definition 11.8 (Orthonormal basis for linear function family)**.** Let G be any domain. Given a linear function family $\mathcal{F}$ from G to C, and a probability distribution D over G. We say $\{v_1, \dots, v_d\}$ form an orthonormal basis of $\mathcal{F}$ with respect to D, if they satisfy the following properties:

- for any $i, j \in [d]$, $\int_G D(t) v_i(t) \overline{v_j(t)} \mathrm{d}t = \mathbf{1}_{i=j}$, and
- for any $f \in \mathcal{F}$, $f \in \mathrm{span}\{v_1, \dots, v_d\}$.

Fact 11.14. *Let $\{v_1, \dots, v_k\}$ be an orthonormal basis of $\mathcal{F}$ with respect to D. For any function $f \in \mathcal{F}$, let $\alpha(f)$ denote the coefficients under the basis $\{v_1, \dots, v_d\}$, i.e., $h = \sum_{i=1}^d \alpha(h)_i \cdot v_i$. Then,*

$$\|\alpha(h)\|_2 = \|h\|_D.$$

For an unknown function $f \in \mathcal{F}$, the goal of importance sampling is to estimate $\|f\|_D$, given samples from another distribution D'. The following definition introduces the importance sampling procedure and condition number of the importance sampling distribution.

Definition 11.9 (Definition 3.1 of [CP19a])**.** For any unknown distribution D' over the domain G and any function $f \in \mathcal{F}$, let $f^{(D')}(t) := \sqrt{\frac{D(t)}{D'(t)}} \cdot f(t)$ be the importance sampling function for some known distribution D such that

$$\mathop{\mathbb{E}}_{t \sim D'}\left[|f^{(D')}(t)|^2\right] = \mathop{\mathbb{E}}_{t \sim D'}\left[\frac{D(t)}{D'(t)}|f(t)|^2\right] = \mathop{\mathbb{E}}_{t \sim D}\left[|f(t)|^2\right].$$

Then, we can use samples from D' to estimate $\|f^{(D')}\|_{D'}$, which gives an estimate of $\|f\|_D$.

When the family $\mathcal{F}$ and D is clear, we use $K_{\mathsf{IS},D'}$ to denote the condition number of importance sampling from D':

$$K_{\mathsf{IS},D'} = \sup_t \left\{ \sup_{f \in \mathcal{F}} \left\{ \frac{|f^{(D')}(t)|^2}{\|f^{D'}\|_{D'}^2} \right\} \right\} = \sup_t \left\{ \frac{D(t)}{D'(t)} \cdot \sup_{f \in \mathcal{F}} \left\{ \frac{|f(t)|^2}{\|f\|_D^2} \right\} \right\}. \tag{11.3}$$

From Definition 11.9, we know that the efficiency of importance sampling depends on how many samples we need to estimate $\|f^{D'}\|_{D'}$. The following lemma provide a criteria for judging whether a set of samples gives a good estimation for the norm of function.

Lemma 11.15 (Lemma 4.2 in [CP19a])**.** *For any $\varepsilon \in (0,1)$, let $S = \{t_1, \ldots, t_s\}$ and the weight vector $w \in \mathbb{R}^s_{>0}$. Define a matrix $A \in \mathbb{R}^{s \times d}$ be the $s \times d$ matrix defined as $A_{i,j} = \sqrt{w_i} \cdot v_j(t_i)$, where $\{v_1, \ldots, v_d\}$ is an orthonormal basis for $\mathcal{F}$. Then*

$$\|h\|_{S,w}^2 := \sum_{j=1}^{s} w_j \cdot |h(x_j)|^2 \in [1 \pm \varepsilon] \cdot \|h\|_D^2 \quad \textit{for every } h \in \mathcal{F}$$

if and only if the eigenvalues of A^*A are in $[1-\varepsilon, 1+\varepsilon]$.

The following lemma shows that the sample complexity depends on the condition number $K_{\mathsf{IS},D'}$:

Lemma 11.16 (Lemma 6.6 in [CP19a]). *Let D' be an arbitrary distribution over G and let $K_{\mathsf{IS},D'}$ be the condition number of importance sampling from D' (defined by Eq. (11.3)). There exists an absolute constant C such that for any $\varepsilon \in (0,1)$ and $\delta \in (0,1)$, let $S = \{t_1, \dots, t_s\}$ be a set of i.i.d. samples from the distribution D' and let w be the weight vector defined by $w_j = \frac{D(t_j)}{s \cdot D'(t_j)}$ for each $j \in [s]$. Then, as long as*

$$s \geq \frac{C}{\varepsilon^2} \cdot K_{\mathsf{IS},D'} \log \frac{d}{\delta},$$

the $s \times d$ matrix $A_{i,j} = \sqrt{w_i} \cdot v_j(t_i)$ satisfies

$$\|A^* A - I\|_2 \leq \varepsilon \text{ with probability at least } 1 - \delta.$$

11.5 Energy Bounds for Fourier Signals

The energy bound shows that the maximum value of a Fourier sparse signal in a certain interval can be bounded by its energy on the interval. One interesting fact is that the approximation ratio in the energy bound is only relate to the sparsity k, and have no relationship with time duration T and band-limit F. An application of energy bound is preserving the norm, that is what is the least size of set S, such that $\|f\|_S = \|f\|_T$, for any function f in a certain function family. The relationship between energy bound and norm preserving can be build by Chernoff bound.

[BE06, Kós08, CKPS16, CP19b] proved energy bounds for sparse Fourier signal under one-dimensional continuous Fourier transform. We further generalize these results to discrete Fourier sparse signal under discrete Fourier transform and high-dimensional Fourier sparse signal under continuous Fourier transform.

This section is organized as follows:

- Section 11.5.1 reviews previous results for one-dimensional continuous Fourier-sparse signals.
- Section 11.5.2 proves a new energy bound for high dimensional Fourier-sparse signals, and also gives a nearly matching lower bound.

- Section 11.5.3 proves energy bound for discrete Fourier-sparse signals.
- Section 11.5.4 builds the connection between energy bound and the concentration property.

11.5.1 Energy bound for one-dimensional signals

In this section, we review the energy bound proved in prior work [BE06, Kós08, CKPS16, CP19b].

[Kós08] proved the following energy bound:

Theorem 11.17 ([Kós08, CKPS16])**.** *Define a family of F-band-limit, k-sparse Fourier signals:*

$$\mathcal{F} := \left\{ x(t) = \sum_{j=1}^{k} v_j \cdot e^{2\pi \mathbf{i} f_j t} \;\middle|\; f_j \in \mathbb{R} \cap [-F, F] \right\}$$

Then, for any $t \in (-1, 1)$,

$$\sup_{x \in \mathcal{F}} \frac{|x(t)|^2}{\|x\|_D^2} \lesssim k^2.$$

[BE06] also proved a time-dependent energy bound for one-dimensional signal:

Theorem 11.18 ([BE06, CP19a])**.** *Define a family of F-band-limit, k-sparse Fourier signals:*

$$\mathcal{F} := \left\{ x(t) = \sum_{j=1}^{k} v_j \cdot e^{2\pi \mathbf{i} f_j t} \;\middle|\; f_j \in \mathbb{R} \cap [-F, F] \right\}$$

Then, for any $t \in (-1, 1)$,

$$\sup_{x \in \mathcal{F}} \frac{|x(t)|^2}{\|x\|_D^2} \lesssim \frac{k}{1 - |t|}.$$

11.5.2 Energy bound for high-dimensional signals

The goal of this section is to prove Theorem 11.19, which gives an energy bound for d-dimensional Fourier signal. It can be viewed as a d-dimensional version of [CKPS16, Lemma 5.1]. We also prove a lower bound in Lemma 11.21.

Theorem 11.19 (Energy bound in d-dimensional)**.** *For any d-dimensional k-Fourier-sparse signal $x(t) : \mathbb{R}^d \to \mathbb{C}$ and any duration T, we have*

$$\max_{t \in [0,T]^d} |x(t)|^2 \le k^{O(d)} \|x\|_T^2,$$

where $\|x\|_T^2 = \frac{1}{T^d} \int_{[0,T]^d} |x(t)|^2 \mathrm{d}t$.

Proof. Without loss of generality, we fix $T = 1$. Then $\|x\|_T^2 = \int_{[0,1]^d} |x(t)| \mathrm{d}t$. Because $\|x\|_T^2$ is the average over the interval $[0,T]^d$, if the maximizer $t^* = \arg\max_{t \in [0,T]^d} |x(t)|^2$ is not 0^d or $T = 1$, we can scale the two intervals $[0^d, t^*]$ and $[t^*, T^d]$ to $[0,1]$ and prove the desired property separately. Hence we assume that $|x(0)|^2 = \max_{t \in [0,T]} |x(t)|^2$ in the proof.

In the next a few paragraphs, we show how to use Lemma 11.20 to prove Theorem 11.19.

We use 0^d to denote a length-d vector with 0 everywhere. Due to Lemma 11.20, we can choose $t_0 = 0^d$ such that $\forall \tau \in \mathbb{R}^d_{>0}$ there exist $C_1, \cdots, C_m \in \mathbb{C}$, and

$$x(0^d) = \sum_{j \in [m]} C_j \cdot x(j \cdot \tau).$$

By the Cauchy-Schwarz inequality, it implies that for any τ,

$$|x(0^d)|^2 \le m \sum_{j \in [m]} |C_j|^2 |x(j \cdot \tau)|^2$$

Then, we obtain

$$\begin{aligned}
|x(0^d)|^2 &= m^d \int_{[0,1/m]^d} |x(0^d)|^2 \mathrm{d}\tau \\
&\lesssim m^d \cdot \int_{[0,1/m]^d} \left(m \sum_{j=1}^{m} |x(j \cdot \tau)|^2 \right) \mathrm{d}\tau \\
&= m^{d+1} \cdot \sum_{j=1}^{m} \int_{[0,1/m]^d} |x(j \cdot \tau)|^2 \mathrm{d}\tau \\
&= m^{d+1} \cdot \sum_{j=1}^{m} \frac{1}{j^d} \int_{[0,j/m]^d} |x(\tau)|^2 \mathrm{d}\tau \\
&\leq m^{d+1} \cdot \sum_{j=1}^{m} \frac{1}{j^d} \int_{[0,1]^d} |x(\tau)|^2 \mathrm{d}\tau \\
&\lesssim m^{d+1} \log m \cdot \|x\|_T^2 \\
&\leq k^{O(d)} \|x\|_T^2, \qquad (11.4)
\end{aligned}$$

where the third step follows by moving m outside of the integral and swapping the integration and the summation, the fourth step follows by replacing $j\tau$ by τ, the fifth step follows by $j/m \leq 1$, the sixth step follows by $\sum_{j=1}^{m} 1/j^d \leq \sum_{j=1}^{m} 1/j = O(\log m)$ and the definition of $\|x\|_T^2$, and the last step follows from Lemma 11.20 that $m = \mathrm{poly}(k)$.

Thus, we have the desired bound. $\square$

The following lemma shows that each point of the signal can be expressed as a linear combination of about k^2 equally spaced signal points.

Lemma 11.20 (d-dimensional signal interpolation)**.** *For any k and d, there exists $m = O(k^2 \log k)$ such that for any d-dimensional k-Fourier-sparse signal $x(t)$, any $t_0 \in \mathbb{R}^d_{\geq 0}$ and $\tau \in \mathbb{R}^d_{>0}$, there always exist $C_1, C_2, \cdots, C_m \in \mathbb{C}$ such that the following properties hold,*

$$\begin{aligned}
&\text{Property I} \quad |C_j| \leq 11 \ \ \textit{for all } j \in [m], \\
&\text{Property II} \quad x(t_0) = \sum_{j \in [m]} C_j \cdot x(t_0 + j \cdot \tau).
\end{aligned}$$

Proof. Consider a specific signal $x(t) := \sum_{i=1}^k v_i e^{2\pi \mathbf{i} f_i^\top t}$ for $t \in \mathbb{R}^d$, where $f_i \in \mathbb{R}^d$ are given. We fix $t_0 \in \mathbb{R}^d$ and $\tau \in \mathbb{R}^d$, and then rewrite $x(t_0 + j \cdot \tau)$ as a polynomial of $b_i := v_i \cdot e^{2\pi \mathbf{i} f_i^\top t_0}$ and $z_i := e^{2\pi \mathbf{i} f_i^\top \tau}$ for each $i \in [k]$.

$$\begin{aligned} x(t_0 + j \cdot \tau) = & \sum_{i=1}^k v_i e^{2\pi \mathbf{i} f_i^\top (t_0 + j\tau)} \\ = & \sum_{i=1}^k v_i e^{2\pi \mathbf{i} f_i^\top t_0} e^{2\pi \mathbf{i} f_i^\top j\tau} \\ = & \sum_{i=1}^k b_i \cdot z_i^j. \end{aligned}$$

where the last step follows from the definition of b_i and z_i.

Given k and $z_1, \cdots, z_k$, let $P(z) = \sum_{j=0}^m c_j z^j$ be the degree m-polynomial in [CKPS16, Lemma 5.4].

$$\begin{aligned} \sum_{j=0}^m c_j x(t_0 + j\tau) = & \sum_{j=0}^m c_j \sum_{i=1}^k b_i \cdot z_i^j \\ = & \sum_{i=1}^k b_i \sum_{j=0}^m c_j \cdot z_i^j \\ = & \sum_{i=1}^k b_i P(z_i) \\ = & \, 0, \end{aligned}$$

where the last step follows by Property I of $P(z)$ in [CKPS16, Lemma 5.4].

By Property II and III in [CKPS16, Lemma 5.4], we have $x(t_0) = -\sum_{j=1}^m c_j x(t_0 + j\tau)$. □

The energy bound in Theorem 11.19 for d-dimensional signals is nearly optimal due to a $k^{\Omega(d)}$ lower bound as follows.[10]

[10]The proof is due to Liu Yang.

Lemma 11.21. *Given $d \geq 1, \delta \in (0, 0.1), k \in \mathbb{Z}$ such that $k \geq O(d^{1+\delta})$. Then, there is a d-dimensional k-Fourier-sparse signal $x(t) : \mathbb{R}^d \to \mathbb{C}$ and a duration T such that,*

$$\max_{t \in [0,T]^d} |x(t)|^2 \geq k^{\Omega(\delta d)} \|x(t)\|_T^2.$$

Proof. We consider the following construction of $x(t)$:

$$x(t) := 2^{-k}(1 + e^{2\pi \mathbf{i} \langle f_0, t\rangle})^k,$$

where $f_0 = \mathbf{1}/(100dT) \in \mathbb{R}^d$.

It is easy to see that x is a k-Fourier sparse signal, and

$$\begin{aligned} |x(t)|^2 &= 2^{-2k} |1 + e^{2\pi \mathbf{i}\langle f_0, t\rangle}|^{2k} \\ &= 2^{-2k} \big((1 + e^{2\pi \mathbf{i}\langle f_0, t\rangle})(1 + e^{-2\pi \mathbf{i}\langle f_0, t\rangle}) \big)^k \\ &= 2^{-2k} \big(2 + e^{2\pi \mathbf{i}\langle f_0, t\rangle} + e^{-2\pi \mathbf{i}\langle f_0, t\rangle} \big)^k \\ &= 2^{-2k} \cdot 2^k (1 + \cos(2\pi \langle f_0, t\rangle))^k \\ &= \Big(\frac{1 + \cos(2\pi\langle f_0, t\rangle)}{2} \Big)^k \\ &= \cos(\pi \langle f_0, t\rangle)^{2k}, \end{aligned}$$

where the first step follows from the definition, the second step follows from $|z|^2 = z\overline{z}$, the third step is straightforward, the fourth step follows from $e^{ia} + e^{-ia} = 2\cos(a)$, the fifth step is straightforward, the last step follows from $(\cos(a) + 1)/2 = \cos(a/2)^2$.

Then, we know that

$$\max_{t \in [0,T]^d} |x(t)|^2 = |x(\mathbf{0})|^2 = 1. \tag{11.5}$$

It remains to upper bound

$$\|x(t)\|_T^2 = T^{-d} \int_{[0,T]^d} |x(t)|^2 \mathrm{d}t = T^{-d} \int_{[0,T]^d} \cos(\pi \langle f_0, t\rangle)^{2k} \mathrm{d}t.$$

Let r be a parameter. We have

$$\begin{aligned}\int_{[0,T]^d} \cos(\pi\langle f_0, t\rangle)^{2k} \mathrm{d}t &= \int_{W_d(r)} \cos(\pi\langle f_0, t\rangle)^{2k} \mathrm{d}t + \int_{[0,T]\setminus W_d(r)} \cos(\pi\langle f_0, t\rangle)^{2k} \mathrm{d}t \\ &\leq \int_{W_d(r)} 1 \cdot \mathrm{d}t + T^d \cdot \max_{t\in[0,T]\setminus W_d(r)} \cos(\pi\langle f_0, t\rangle)^{2k},\end{aligned}$$

where $W_d(r) := \{t \in [0,T]^d \mid t_1 + \cdots + t_d \leq r\}$.

We first bound the second term:

$$\cos(\pi\langle f_0, t\rangle)^{2k} \leq \exp\big(-\pi^2 k\langle f_0, t\rangle^2\big) \leq \exp(-\Omega(kr^2/(dT)^2)),$$

where the first step follows from Fact 11.9, and the second step follows from the definition of f_0 and $t \notin W_d(r)$. Hence,

$$T^d \cdot \max_{t\in[0,T]\setminus W_d(r)} \cos(\pi\langle f_0, t\rangle)^{2k} \leq T^d \cdot \exp(-\Omega(kr^2/(dT)^2)). \tag{11.6}$$

Next, we bound the first term, which is equal to the volume of $W_d(r)$. Note that $W_d(r)$ is contained in the following simplex:

$$P_d(r) := \{t \in \mathbb{R}_+^d \mid t_1 + \cdots + t_d \leq r\}.$$

Thus, we have

$$\begin{aligned}\mathrm{Vol}(W_d(r)) &\leq \mathrm{Vol}(P_d(r)) \leq r^d \cdot \mathrm{Vol}(P_d(1)) \\ &\leq r^d/d! \\ &= O(r/d)^d, \end{aligned} \tag{11.7}$$

where the first step is straightforward, the second step follows from the scaling of the volume, the third step follows from a well-known fact on the volume of a d-dimensional simplex $P_d(1) = 1/d!$ (see e.g. [Ste66]), and the last step follows from Stirling's approximation.

Combining Eqs. (11.6) and (11.7) together, we get that

$$T^{-d}\|x(t)\|_T^2 \leq O(\frac{r}{dT})^d + \exp(-\Omega(kr^2/(dT)^2)).$$

By taking $r = Tdk^{-\delta/6}$, we have

$$O(\frac{r}{dT})^d = O(k^{-\delta/6})^d = k^{-\Omega(\delta d)},$$

and

$$\begin{aligned}\exp(-\Omega(kr^2/(dT)^2)) =& \exp(-\Omega(k^{1-\delta/3})) \\ =& \exp(-\Omega(d^{(1+\delta)(1-\delta/3)})) \\ \leq& \exp(-\Omega(\delta d \log k)) \\ =& k^{-\Omega(d)},\end{aligned}$$

where the first step is straightforward, the second step follows from $k = O(d^{1+\delta})$, and the last step follows from $\delta \in (0,1)$.

Therefore, we have

$$T^{-d}\|x(t)\|_T^2 \leq k^{-\Omega(d)} = k^{-\Omega(d)} \cdot \max_{t\in[0,T]^d} |x(t)|^2,$$

where the last step follows from Eq. (11.5).

The lemma is then proved.

$\square$

11.5.3 Energy bound for discrete Fourier signals

Recall the definition of one-dimensional discrete sparse Fourier signal: for $t \in \{0, 1, \ldots, n-1\}$,

$$x_t = \frac{1}{n}\sum_{i=1}^{k} \widehat{x}_{f_i} \exp\left(\frac{2\pi\mathbf{i}}{n} f_i t\right), \ f_i \in [n] \tag{11.8}$$

More generally, for $d \geq 1$, the d-dimensional discrete sparse Fourier signal can be defined as follows. Let $n = p^d$ where both p and d are positive integers. Recall the definition of high-dimensional discrete sparse Fourier signal (see e.g. [NSW19]):

$$x_t = \frac{1}{n}\sum_{i=1}^{k} \widehat{x}_{f_i} \exp\left(\frac{2\pi\mathbf{i}}{p}\langle f_i, t\rangle\right) \quad \forall t \in [p]^d, \tag{11.9}$$

where each $f_i \in [p]^d$.

In this section, we prove the following discrete Fourier signals energy bound that works for any dimension:

Theorem 11.22 (Discrete d-dimensional Fourier energy bound)**.** *For $d \geq 1$ and any discrete d-dimensional k-sparse Fourier signal $\{x_i\}_{i\in[n]}$, we have*

$$\|x\|_\infty^2 \leq k \cdot \frac{\|x\|_2^2}{n}$$

Proof.

$$\|x\|_\infty^2 \leq \frac{1}{n^2}\|\widehat{x}\|_1^2 \leq \frac{k}{n^2}\|\widehat{x}\|_2^2 = \frac{k}{n}\|x\|_2^2,$$

where the first step follows from Claim 11.23, the second step follows from Cauchy-Schwarz inequality, and the last step follows from Theorem 11.24. □

Claim 11.23. *For any $d \geq 1$ and any discrete d-dimensional Fourier signal x,*

$$\|x\|_\infty \leq \frac{1}{n}\|\widehat{x}\|_1.$$

Proof. By triangle inequality,

$$|x(t)| \leq \frac{1}{n}\left|\sum_{i=1}^k \widehat{x}_i \exp\left(\frac{2\pi\mathbf{i}}{p}\langle f_i, t\rangle\right)\right| \leq \frac{1}{n}\sum_{i=1}^k |\widehat{x}_i| = \frac{1}{n}\|\widehat{x}\|_1.$$

□

Theorem 11.24 (Parseval's theorem)**.** *For any $d \geq 1$ and any discrete d-dimensional Fourier signal (Eq. (11.9)),*

$$\|x\|_2^2 = \frac{1}{n}\|\widehat{x}\|_2^2.$$

11.5.4 Energy bounds imply concentrations

By using Chernoff bound, we prove the following lemma to show the performance of uniformly sampling.

11.5.4.1 Continuous case

Lemma 11.25. *Let $d \in \mathbb{Z}_+$. Let R be a parameter. Given any function $x(t) : \mathbb{R}^d \to \mathbb{C}$ with $\max\limits_{t \in [0,T]^d} |x(t)|^2 \leq R\|x(t)\|_T^2$. Let S denote a set of points chosen uniformly at random from $[0,T]^d$. We have that*

$$\mathsf{Pr}\left[\left|\frac{1}{|S|}\sum_{i \in S}|x(t_i)|^2 - \|x(t)\|_T^2]\right| \geq \varepsilon\|x(t)\|_T^2\right] \quad \leq \quad \exp(-\Omega(\varepsilon^2|S|/R)),$$

where $\|x(t)\|_T^2 = \frac{1}{T^d}\int_{[0,T]^d}|x(t)|^2 \mathrm{d}t$.

Proof. Let M denote $\max\limits_{t \in [0,T]^d}|x(t)|^2$. Replacing X_i by $\frac{|x(t_i)|^2}{M}$ and n by $|S|$ in Lemma A.2, we obtain that

$$\mathsf{Pr}[|X - \mu| > \varepsilon\mu] \leq 2\exp(-\frac{\varepsilon^2}{3}\mu)$$

The above equation implies

$$\mathsf{Pr}\left[\left|\sum_{i \in S}\frac{|x(t_i)|^2}{M} - |S|\frac{\|x(t)\|_T^2}{M}\right| > \varepsilon|S|\frac{\|x(t)\|_T^2}{M}\right] \leq 2\exp(-\frac{\varepsilon^2}{3}\mu)$$

Multiplying M on the both sides

$$\mathsf{Pr}\left[\left|\frac{1}{|S|}\sum_{i \in S}|x(t_i)|^2 - \|x(t)\|_T^2\right| \geq \varepsilon\|x(t)\|_T^2\right] \leq 2\exp(-\frac{\varepsilon^2}{3}\mu)$$

Applying bound on μ

$$\mathsf{Pr}\left[\left|\frac{1}{|S|}\sum_{i \in S}|x(t_i)|^2 - \|x(t)\|_T^2\right| \geq \varepsilon\|x(t)\|_T^2\right] \leq 2\exp(-\frac{\varepsilon^2}{3}|S|\frac{\|x(t)\|_T^2}{M})$$

which is less than $2\exp(-\frac{\varepsilon^2}{3}|S|/R)$, thus completes the proof. □

11.5.4.2 Discrete case

Lemma 11.26. *Let R be a parameter. Given any function $x \in \mathbb{C}^n$ with $\|x\|_\infty^2 \leq R\|x\|_2^2/n$. Let S denote a set of points chosen uniformly at random from $[n]$. We have that*

$$\mathsf{Pr}\left[\left|\frac{1}{|S|}\sum_{t \in S}|x_t|^2 - n^{-1}\|x\|_2^2]\right| \geq \varepsilon n^{-1}\|x\|_2^2\right] \quad \leq \quad \exp(-\Omega(\varepsilon^2|S|/R)).$$

Proof. Let M denote $\max_{t\in[n]}|x_t|^2$. Replacing X_i by $\frac{|x_t|^2}{M}$ and n by $|S|$ in Lemma A.2, we obtain that

$$\Pr[|X-\mu|>\varepsilon\mu]\le 2\exp(-\frac{\varepsilon^2}{3}\mu)$$

The above equation implies that

$$\Pr\left[\left|\sum_{t\in S}\frac{|x_t|^2}{M}-|S|\frac{\|x\|_2^2}{nM}\right|>\varepsilon|S|\frac{\|x\|_2^2}{nM}\right]\le 2\exp(-\frac{\varepsilon^2}{3}\mu)$$

Multiplying the normalization factor on both sides,

$$\Pr\left[\left|\frac{1}{|S|}\sum_{t\in S}|x_t|^2-n^{-1}\|x\|_2^2\right|\ge\varepsilon n^{-1}\|x\|_2^2\right]\le 2\exp(-\frac{\varepsilon^2}{3}\mu)$$

Applying bound on μ

$$\Pr\left[\left|\frac{1}{|S|}\sum_{t\in S}|x_t|^2-n^{-1}\|x\|_2^2\right|\ge\varepsilon n^{-1}\|x\|_2^2\right]\le 2\exp(-\frac{\varepsilon^2}{3}|S|\frac{\|x\|_2^2}{nM})$$

which is less than $2\exp(-\frac{\varepsilon^2}{3}|S|/R)$, thus completes the proof. $\square$

11.6 Oblivious Sketching Fourier Sparse Signals

In this section, we show an intermediate step in the reduction from Frequency estimation to Signal estimation: constructing a small sketching subset S of the time domain *obliviously* (without making any query to the signal), so that the signal discretized by S has norm close to the original continuous signal. More formally, we define the *oblivious sketching Fourier signal problem* as follows:

Problem 11.27 (Oblivious sketching Fourier sparse signal problem)**.** Suppose $f_1,f_2,\cdots,f_k\in\mathbb{R}^d$, and $v_1,\dots,v_k\in\mathbb{C}$. Define the continuous signal $x(t)=\sum_{j=1}^k v_j e^{2\pi\mathbf{i}\langle f_j,t\rangle}$. Let $\eta=\min_{i\neq j}\|f_j-f_i\|_\infty$.

Let $\varepsilon\in(0,0.1)$ denote the accuracy parameter. Find a set $S=\{t_1,\dots,t_s\}\subseteq[0,T]^d$ of size s such that

$$(1-\varepsilon)\|x\|_T\le\|x\|_S\le(1+\varepsilon)\|x\|_T,$$

where

$$\|x\|_T^2 := \frac{1}{T^d}\int_{[0,T]^d} |x(t)|^2 \mathrm{d}t, \quad \text{and} \quad \|x\|_S^2 := \frac{1}{|S|}\sum_{i\in[s]} |x(t_i)|^2.$$

In Section 11.6.1, we show how to sketch one-dimensional signals with nearly-optimal weighted sketching. Then, we show how to sketch high dimensional signals in Section 11.6.2, and discrete signals in Section 11.6.3.

11.6.1 Weighted oblivious sketching one-dimensional signals

For one-dimensional signals, the most natural approach to oblivious sketching is to uniformly sample some points in the time domain. However, by a standard concentration argument, we know that the sample complexity is $\mathrm{poly}(k)$, which is not time-efficient for our task. In this section, we show a more efficient sketching method for one-dimensional Fourier sparse signals by assigning different weights to each sample point. More precisely, let $S = \{t_1, \dots, t_s\} \subseteq [0, T]$ be a discrete sketching set and let $w \in \mathbb{R}_{\geq 0}^s$ be the weight vector. We define the weighted sketching norm of the signal as follows:

$$\|x\|_{S,w} := \sum_{i\in[s]} w_i \cdot |x(t_i)|^2.$$

And the goal of weighted oblivious sketching is to find a small set S and a weight vector w such that $\|x\|_{S,w} \approx \|x\|_T$.

In the following lemma, we give a sketch for any one-dimensional Fourier sparse signal with nearly-optimal size:

Lemma 11.28 (Nearly-optimal weighted sketch for one-dimensional signals)**.** *For $k \in \mathbb{N}_+$, define a probability distribution $D(t)$ as follows:*

$$D(t) := \begin{cases} c/(1 - |t/T|), & \text{for } |t| \leq T(1 - 1/k) \\ c \cdot k, & \text{for } |t| \in [T(1 - 1/k), T] \end{cases} \tag{11.10}$$

where $c = \Theta(T^{-1}\log^{-1}(k))$ is a normalization factor such that $\int_{-T}^{T} D(t)\mathrm{d}t = 1$.

For any $f_1, \cdots, f_k \in [-F, F]$ and $v_1, \cdots, v_k \in \mathbb{C}$, let the continuous signal $x(t) = \sum_{j=1}^k v_j \exp(2\pi \mathbf{i} f_j t)$. For any $\varepsilon, \rho \in (0,1)$, let $S_D = \{t_1, \cdots, t_s\}$ be a set of i.i.d. samples from $D(t)$ of size $s \geq O(\varepsilon^{-2} k \log(k) \log(1/\rho))$. Let the weight vector $w \in \mathbb{R}^s$ be defined by $w_i := 2/(TsD(t_i))$ for $i \in [s]$. Then with probability at least $1-\rho$, we have

$$(1-\varepsilon)\|x\|_T \leq \|x\|_{S_D,w} \leq (1+\varepsilon)\|x\|_T,$$

where $\|x\|_T^2 := \frac{1}{2T}\int_{-T}^T |x(t)|^2 \mathrm{d}t$.

Proof. For the convenient, in the proof, we use time duration $[-T, T]$. Let $\mathcal{F}$ be defined as:

$$\mathcal{F} := \left\{ x(t) = \sum_{j=1}^k v_j \cdot e^{2\pi \mathbf{i} f_j t} | f_j \in \mathbb{R} \cap [-F, F], v_j \in \mathbb{C} \right\}$$

Let $\{v_1(t), v_2(t), \cdots, v_k(t)\}$ be an orthonormal basis for $\mathcal{F}$ with respect to the distribution D, i.e.,

$$\int_0^T D(t) \cdot v_i(t)\overline{v_j(t)} \mathrm{d}t = \mathbf{1}_{i=j}, \quad \forall i, j \in [k].$$

We first prove that the distribution D is well-defined. By the condition that $\int_{-T}^T D(t)\mathrm{d}t = 1$, we have

$$2\int_0^{T(1-1/(k))} \frac{c}{(1-|t/T|)}\mathrm{d}t + 2\int_{T(1-1/(k))}^T c \cdot k^2 k \mathrm{d}t = 1,$$

which implies that

$$\begin{aligned} c^{-1} &= 2\int_0^{T(1-1/k)} \frac{1}{(1-|t/T|)}\mathrm{d}t + 2\int_{T(1-1/k)}^T k^2 \mathrm{d}t \\ &= 2T \log k + 2T \\ &= \Theta(T \log(k)). \end{aligned}$$

Thus, we get that $c = \Theta(T^{-1} \log^{-1}(k))$.

To show that sampling from distribution D give a good weighted sketch, we will use some technical tools in Section 11.4.4. Applying Lemma 11.16 with $D' = D$, $D = \mathrm{Uniform}([-T, T])$, $d = k$, $\delta = \rho$, we have that, with probability at least $1 - \rho$, the matrix $A \in \mathbb{C}^{s \times k}$ defined by $A_{i,j} := \sqrt{w_i} \cdot v_j(x_i)$ satisfying

$$\|A^* A - I\|_2 \leq \varepsilon,$$

as long as $s \geq \frac{C}{\varepsilon^2} \cdot K_{\mathsf{IS},D'} \log \frac{k}{\rho}$. Then, by Lemma 11.15, it implies that for every $x \in \mathcal{F}$,

$$(1 - \varepsilon)\|x\|_T^2 \leq \|x\|_{S_D,w}^2 \leq (1 + \varepsilon)\|x\|_T^2.$$

It remains to bound the size of S_D; or equivalently, we need to upper-bound the condition number of the importance sampling of D' (see Definition 11.9):

$$\begin{aligned}
K_{\mathsf{IS},D'} &:= \sup_t \{\frac{D(t)}{D'(t)} \cdot \sup_{f \in \mathcal{F}} \{\frac{|f(t)|^2}{\|f\|_D^2}\}\} \\
&= \sup_t \{\frac{1}{2TD(t)} \cdot \sup_{f \in \mathcal{F}} \{\frac{|f(x)|^2}{\|f\|_D^2}\}\} \\
&\leq \sup_t \{\frac{1}{2TD(t)} \cdot \min\{\frac{k}{1 - |t/T|}, k^2\}\} \\
&\leq \max\{\frac{(1 - |t/T|)}{2cT} \frac{k}{1 - |t/T|}, \frac{1}{2cTk} k^2\} \\
&= \frac{k}{2cT} \\
&= O(k \log k),
\end{aligned}$$

where the first step follows from the definition, the second step follows from $D(t) = \mathrm{Uniform}([-T, T])(t) = \frac{1}{2T}$, the third step follows from Theorem 11.17 and Theorem 11.18, and the remaining steps follow from direct calculations. Thus, we get that

$$|S_D| \geq \Omega\left(\varepsilon^{-2} k \log(k) \log(1/\rho)\right).$$

The lemma is then proved. $\square$

11.6.2 Oblivious sketching high-dimensional signals

For high-dimensional noiseless signals, we can still use uniformly random samples in $[0,T]^d$ to sketch the continuous signal.

Lemma 11.29 (Oblivious sketching high-dimensional signal)**.** *Let $d > 0$ be the dimension of the signal. For any $\varepsilon \in (0,1)$, let S_d be a set of i.i.d. samples chosen uniformly at random over $[0,T]^d$ of size $|S_d| \geq \varepsilon^{-2} k^{O(d)} \log(1/(\rho\varepsilon))$. Let $V := \{\exp(2\pi\mathbf{i}\langle f_i, t\rangle) \mid i \in [k]\}$. Then with probability at least $1-\rho$, for all $x \in \mathrm{span}\{V\}$, we have*

$$(1-\varepsilon)\|x\|_T \leq \|x\|_{S_d} \leq (1+\varepsilon)\|x\|_T.$$

Proof. Define the set of normalized k-sparse signals $\mathcal{Q} := \{u \in \mathrm{span}\{V\} \mid \|u\|_T = 1\}$. Without loss of generality, consider a signal $u \in \mathcal{Q}$.

Let $\mathcal{P}_1$ be an ε_0-net for $\mathcal{Q}$ constructed by Lemma 11.32.

Let $m = O(d \log_{1/\varepsilon_0}(k))$. We recursively define $\widetilde{u}_i, w_i, u_i, \alpha_i$ for $\{0,1,2,\cdots,m\}$ as follows:

- Initially, let $\widetilde{u}_0 := u$.
- For $i = 0,1,2,\cdots,m$, define

$$\begin{aligned} w_i &:= \arg\min_{w \in \mathcal{P}_1} \|w - \widetilde{u}_i\|_T, \\ u_{i+1} &:= \widetilde{u}_i - w_i, \ \ \alpha_{i+1} := \|u_{i+1}\|_T, \\ \widetilde{u}_{i+1} &:= u_{i+1}/\alpha_{i+1} \text{ if } \alpha_{i+1} \neq 0. \end{aligned}$$

Eventually, we have

$$u = w_0 + \alpha_1 \cdot w_1 + \alpha_1\alpha_2 \cdot w_2 + \cdots + (\prod_{j=1}^m \alpha_j) \cdot (w_m + u_{m+1}),$$

where each $|\alpha_i| \leq \varepsilon_0$ and each $w_i \in \mathcal{P}_1$ for $i \in [m]$.

Because $\widetilde{u}_i \in \mathcal{Q}$, $w_i \in \mathcal{P}_1$, $\mathcal{P}_1$ is a ε_0-net, $\min_{w\in\mathcal{P}_1}\|w - \widetilde{u}_i\|_T \leq \varepsilon_0$.

$$\|u_{m+1}\|_T = \|\widetilde{u}_m - w_m\| = \min\|w - \widetilde{u}_m\|_T \leq \varepsilon_0 \leq 1$$

where the first step follows from the definition of u_{m+1}, the second step follows from $w_i = \arg\min_{w\in\mathcal{P}_1}\|w - \widetilde{u}_i\|_T$.

Then we show by induction that $u_i \in \mathrm{span}\{V\}$ for all $i \in [m+1]$. First, we have $\widetilde{u}_0 = u \in \mathcal{Q}$, which implies $\widetilde{u}_0 \in \mathrm{span}\{V\}$. Suppose that $\widetilde{u}_i \in \mathrm{span}\{V\}$, and consider $\widetilde{u}_{i+1}$. By definition, $w_i \in \mathcal{P}_1 \subset \mathrm{span}\{V\}$. Since $u_{i+1} = \widetilde{u}_i - w_i \in \mathrm{span}\{V\}$ and $\widetilde{u}_{i+1} = u_{i+1}/\alpha_i$, we have $\widetilde{u}_{i+1} \in \mathrm{span}\{V\}$. Hence, by induction, we get that $\widetilde{u}_i \in \mathrm{span}\{V\}$ for all $i \in [m+1]$.

Note that for any $u \in \mathrm{span}\{V\}$, u is a k-Fourier-sparse signal with frequencies in the set $\{f_i \mid i \in [k]\}$. Hence, by Theorem 11.19, we have $\|u_{m+1}\|_{S_d} \leq k^{O(d)} \cdot \|u_{m+1}\|_T$.

Therefore, we can lower bound $\|u\|_{S_d}$ as follows:

$$\begin{aligned}
\|u\|_{S_d} &= \|w_0 + \alpha_1 w_1 + \alpha_1\alpha_2 w_2 + \cdots + (\prod_{j=1}^m \alpha_j)(w_m + u_{m+1})\|_{S_d} \\
&\geq \|w_0\|_{S_d} - \|\alpha_1 w_1\|_{S_d} - \|\alpha_1\alpha_2 w_2\|_{S_d} - \cdots - \|\prod_{j=1}^m \alpha_j w_m\|_{S_d} - \|\prod_{j=1}^m \alpha_j u_{m+1}\|_{S_d} \\
&\geq (1-\varepsilon_0) - \varepsilon_0(1+\varepsilon_0) - \varepsilon_0^2(1+\varepsilon_0) - \cdots - \varepsilon_0^m(1+\varepsilon_0) - \varepsilon_0^m\|u_{m+1}\|_{S_d} \\
&\geq 1 - \varepsilon_0 - \frac{(1+\varepsilon_0)\varepsilon_0}{1-\varepsilon_0} - \varepsilon_0^m \cdot k^{O(d)} \\
&\geq 1 - 3\varepsilon_0.
\end{aligned}$$

where the second step follows from triangle inequality, the third step follows from Claim 11.30, and $w_i \in \mathcal{P}_1 \subset \mathcal{Q}$, so $\|w_i\|_T = 1$, and the last step follows from direct computations.

Similarly, we have $\|u\|_{S_d} \leq 1 + 3\varepsilon_0$. Let $\varepsilon_0 = \varepsilon/10$, we have that, $1 - \varepsilon \leq \|u\|_{S_d} \leq 1 + \varepsilon$. The lemma is then proved.

$\square$

Claim 11.30 (Handling high-dimensional net points)**.** *For any $\varepsilon \in (0,1)$, let $\mathcal{P}_d$ be an ε-net for $\mathcal{Q}$ constructed by Lemma 11.32. For any failure probability $\rho \in (0, 0.1)$, and a set S_d of i.i.d. samples chosen uniformly at random over $[0,T]^d$ of size $|S_d| \geq \varepsilon^{-2} k^{O(d)} \log(1/(\rho\varepsilon))$, then with probability at least $1-\rho$, for all $x \in \mathcal{P}_d$, we have*

$$(1-\varepsilon)\|x\|_T \leq \|x\|_{S_d} \leq (1+\varepsilon)\|x\|_T.$$

Proof. By the energy bound for high-dimensional signal (Theorem 11.19), we have

$$R := \sup_{x\in\mathcal{Q}} \sup_{t\in[0,T]^d} |x(t)|^2 = k^{O(d)}.$$

Then, From Lemma 11.25 and Theorem 11.19, for each $x \in \mathcal{P}_d$,

$$\begin{aligned}\Pr[\|x\|_{S_d} \notin [(1-\varepsilon)\|x\|_T, (1+\varepsilon)\|x\|_T]] \leq &~ \exp(-\Omega(\frac{|S_d|\varepsilon^2}{k^{O(d)}})) \\ \leq &~ \exp(-2k\log(k/(\rho\varepsilon))) \\ \leq &~ \rho(0.01\varepsilon^2/k^2)^k.\end{aligned}$$

where the first step follows form Lemma 11.25, the second step follows from the lower bound on W.

From the union bound, $\|x\|_{S_d} \in [(1-\varepsilon)\|x\|_T, (1+\varepsilon)\|x\|_T]$ for any $x \in \mathcal{P}_d$ with probability at least $1-\rho(0.01\varepsilon^2/k^2)^k \cdot |\mathcal{P}_d| \geq 1-\rho$. □

11.6.3 Oblivious sketching discrete signals

In this section, we show that discrete Fourier sparse signals can also be sketched in the offline. The following lemma works for discrete signals in any dimension.

Lemma 11.31 (Oblivious sketching discrete signal)**.** *For $d \geq 1$, let $n = p^d$ for some positive integer p. Let $k \in \mathbb{N}_+$ and $f_1, \dots, f_k \in [p]^d$. Define*

$$V := \left\{ (e^{2\pi\mathbf{i}\langle f_i, t\rangle/p})_{t\in[p]^d} \mid \forall i \in [k] \right\} \subseteq \mathbb{C}^{[p]^d}$$

be the basis of discrete Fourier sparse signals. For any $\varepsilon, \rho \in (0,1)$*, let* S *be a set of i.i.d. samples chosen uniformly at random over* $[n]$ *of size* $|S| \geq O(\varepsilon^{-2} k \log(k/\rho))$*. Then, with probability at least* $1-\rho$*, it holds that for all* $u \in \mathrm{span}\{V\}$*,*

$$(1-\varepsilon)\|u\|_2^2 \leq n\|u\|_S^2 \leq (1+\varepsilon)\|u\|_2^2,$$

where $\|u\|_S^2 = \sum_{i \in S} |u_i|^2/|S|$*.*

Proof. To show that sampling from distribution $D = \mathrm{Uniform}([n])$ give a good sketch, we will use some technical tools in Section 11.4.4.[11] Applying Lemma 11.16 with $D' = D$, $d = k$, $\delta = \rho$, $w_i = 1/s$ we have that, with probability at least $1-\rho$, the matrix $A \in \mathbb{C}^{s \times k}$ defined by $A_{i,j} := \sqrt{w_i} \cdot v_j(x_i)$ satisfying

$$\|A^* A - I\|_2 \leq \varepsilon,$$

as long as $s \geq \frac{C}{\varepsilon^2} \cdot K_{\mathsf{IS},D'} \log \frac{k}{\rho}$. Then, by Lemma 11.15, it implies that for every $x \in \mathcal{F}$,

$$(1-\varepsilon)\|x\|_2^2 \leq n\|x\|_S^2 \leq (1+\varepsilon)\|x\|_2^2.$$

It remains to bound the size of S. By Theorem 11.22,

$$K_{\mathsf{IS},D} := \sup_{t \in D}\{\sup_{f \in \mathcal{F}}\{\frac{|f(t)|^2}{\|f\|_D^2}\}\} = k.$$

Thus, we get that

$$|S| \geq \Omega\left(\varepsilon^{-2} k \log(k/\rho)\right).$$

The lemma is then proved. □

[11] A more straightforward approach is by ε-net (Lemma 11.33), as we did in previous sections. However, it requires $O(k^2)$ samples.

11.6.4 ε-net for sparse Fourier signals

In this section, we construct ε-nets for high-dimensional sparse Fourier continuous and discrete signals.

Lemma 11.32 (ε-net construction for continuous signals)**.** *Given $k \in \mathbb{Z}_+$ unknown frequencies $f_1, f_2, \ldots, f_k \in [-F, F]^d$. Let $V := \left\{e^{2\pi \mathbf{i}\langle f_i, t\rangle} \mid i \in [k]\right\}$ be a family of Fourier basis. Let $\mathcal{Q} := \{u \in \operatorname{span}\{V\} \mid \|u\|_T^2 = 1\}$ be the set of all signals in $[0,T]^d$ with frequency $f_1, \ldots, f_k$, where $\|x\|_T^2 = \frac{1}{T^d}\int_{[0,T]^d} |x(t)|^2 \mathrm{d}t$.*

Then, there exists an ε-net $\mathcal{P}_d \subset \mathcal{Q}$ such that

1. *$\forall u \in \mathcal{Q}, \exists w \in \mathcal{P}_d, \|u - w\|_T \leq \varepsilon$.*
2. *$|\mathcal{P}_d| \leq \left(5\frac{k}{\varepsilon}\right)^{2k}$.*

Proof. We first construct an $\frac{\varepsilon}{k}$-net for the unit disk in $\mathbb{C}$, i.e., $\{z \in \mathbb{C} \mid |z| \leq 1\}$. Let $\mathcal{P}'$ denote

$$\mathcal{P}' := \left\{\frac{\varepsilon}{2k}j_1 + \mathbf{i}\frac{\varepsilon}{2k}j_2 \mid j_1, j_2 \in \mathbb{Z}, |j_1| \leq \frac{2k}{\varepsilon}, |j_2| \leq \frac{2k}{\varepsilon}\right\}.$$

Notice that $|\varepsilon/(2k)j_1| \leq \varepsilon/(2k) \cdot 2k/\varepsilon = 1$; and similarly, $|\varepsilon/(2k)j_2| \leq 1$. Thus, for any $a \in \mathbb{C}$, $|a| \leq 1$, there is a $b \in \mathcal{P}'$ such that

$$|a - b| \leq \varepsilon/(2k) + \varepsilon/(2k) \leq \varepsilon/k.$$

Moreover,

$$|\mathcal{P}'| \leq (2 \cdot 2k/\varepsilon + 1) \cdot (2 \cdot 2k/\varepsilon + 1) = \left(4\frac{k}{\varepsilon} + 1\right)^2.$$

Hence, we conclude that,

- $\mathcal{P}'$ is an $\frac{\varepsilon}{k}$-net in the unit circle of $\mathbb{C}$.
- $\mathcal{P}'$ has size at most $(4\frac{k}{\varepsilon} + 1)^2$.

Then, we use $\mathcal{P}'$ to construct an ε-net for $\mathcal{Q}$. Since the dimension of $\mathcal{Q}$ is at most k, we take an orthonormal basis $w_1, \cdots, w_k \in \mathcal{Q}$ such that,

$$\int_{[0,T]^d} w_i(t)\overline{w_j(t)}\mathrm{d}t = \mathbf{1}_{i=j}.$$

And we define

$$\mathcal{P}'' := \{\sum_{i=1}^k \alpha_i w_i \mid \forall i \in [k], \alpha_i \in \mathcal{P}'\}.$$

First, for any $u \in \mathcal{Q}$, we have

$$u = \sum_{i=1}^k v_i \exp(2\pi\mathbf{i}\langle f_i, t\rangle) = \sum_{i=1}^k \alpha_i' w_i,$$

which implies that $|\alpha_i'| \leq 1$ for all $i \in [k]$. So, for any $a \in \mathcal{Q}$, there is a $b \in \mathcal{P}''$ such that $\|a-b\|_T \leq k \cdot \varepsilon/k = \varepsilon$. Moreover, $|\mathcal{P}''| \leq ((4\frac{k}{\varepsilon}+1)^2)^k \leq (5\frac{k}{\varepsilon})^{2k}$. Therefore, we conclude that $\mathcal{P}''$ is an ε-net for $\mathcal{Q}$ and $|\mathcal{P}''| \leq \left(5\frac{k}{\varepsilon}\right)^{2k}$.

Then we define

$$\mathcal{P}_d := \{v \in \mathcal{Q} \mid \forall u \in \mathcal{P}'', v = \mathrm{argmin}_{v\in\mathcal{Q}}\{\|v-u\|_T\}\}.$$

therefore we have that, for any $a \in \mathcal{Q}$, there is a $b \in \mathcal{P}''$ such that $\|a-b\|_T \leq \varepsilon$, because there is a $c \in \mathcal{P}_d$, such that $\|c-b\|_T = \min_{d\in\mathcal{Q}}\|d-b\|_T \leq \|a-b\|_T \leq \varepsilon$. Then, $\|c-a\|_T \leq \|c-b\|_T + \|b-a\|_T \leq 2\varepsilon$. $\square$

Lemma 11.33 (ε-net construction for discrete signals)**.** *Let $n, k \in \mathbb{Z}_+, \varepsilon \in (0,1), n \geq k$. Given unknown frequencies $f_1, f_2, \cdots, f_k \in [n]$. Let $V := \left\{(e^{2\pi\mathbf{i}f_i 1/n}, \cdots, e^{2\pi\mathbf{i}f_i n/n}) \mid i \in [k]\right\} \subseteq \mathbb{C}^n$. Let $\mathcal{Q}_{\mathsf{dis}} := \{u \in \mathrm{span}\{V\} \mid \|u\|_2^2 = 1\}$. There exists an ε-net $\mathcal{P}_{\mathsf{dis}} \subset \mathrm{span}\{V\}$ such that*

1. *$\forall u \in \mathcal{Q}, \exists w \in \mathcal{P}_{\mathsf{dis}}, \|u-w\|_2^2 \leq \varepsilon^2$.*

2. *$|\mathcal{P}_{\mathsf{dis}}| \leq \left(5\frac{k}{\varepsilon}\right)^{2k}$.*

Proof. Let $\mathcal{P}'$ be the $\frac{\varepsilon}{k}$-net for the unit disk in $\mathbb{C}$, i.e., $\{z \in \mathbb{C} \mid |z| \leq 1\}$:

$$\mathcal{P}' = \left\{ \frac{\varepsilon}{2k} j_1 + \mathbf{i}\frac{\varepsilon}{2k} j_2 \mid j_1, j_2 \in \mathbb{Z}, |j_1| \leq \frac{2k}{\varepsilon}, |j_2| \leq \frac{2k}{\varepsilon} \right\}.$$

Then, we use $\mathcal{P}'$ to construct an ε-net for $\mathcal{Q}_{\mathsf{dis}}$. Since the dimension of $\mathcal{Q}_{\mathsf{dis}}$ is at most k, we take an orthonormal basis $w_1, \cdots, w_k \in \mathcal{Q}_{\mathsf{dis}}$ such that,

$$\sum_{t=1}^{n} w_{i,t} \overline{w_{j,t}} = \mathbf{1}_{i=j}.$$

And we take

$$\mathcal{P}_{\mathsf{dis}} := \{\sum_{i=1}^{k} \alpha_i w_i \mid \forall i \in [k], \alpha_i \in \mathcal{P}'\}.$$

Then, for any $u(t) = \sum_{i=1}^{k} v_i \exp(2\pi \mathbf{i} f_i, t/n) = \sum_{i=1}^{k} \alpha'_i w_i \in \mathcal{Q}_{\mathsf{dis}}$, $|\alpha'_i| \leq 1$. So, for any $a \in \mathcal{Q}_{\mathsf{dis}}$, there is a $b \in \mathcal{P}_{\mathsf{dis}}$ such that $\|a - b\|_2^2 \leq k \cdot \varepsilon^2/k^2 \leq \varepsilon$. Moreover, $|\mathcal{P}_{\mathsf{dis}}| \leq ((4\frac{k}{\varepsilon} + 1)^2)^k \leq (5\frac{k}{\varepsilon})^{2k}$. Therefore, we conclude that $\mathcal{P}_{\mathsf{dis}}$ is an ε-net for $\mathcal{Q}$ and $|\mathcal{P}_{\mathsf{dis}}| \leq \left(5\frac{k}{\varepsilon}\right)^{2k}$. $\square$

11.7 Fast Implementation of Well-Balanced Sampling Procedure

Well-balanced sampling procedure was first defined in [CP19a] to study the active linear regression problem. Our signal estimation algorithm will call it as a sub-procedure. In this section, we give a fast implementation of well-balanced sampling procedure based on the Randomized BSS algorithm [BSS12, LS15].

First, we restate the definition of well-balanced sampling procedure in [CP19a].

Definition 11.10 (Well-balanced sampling procedure (WBSP), [CP19a])**.** Given a linear family $\mathcal{F}$ and underlying distribution D, let P be a random sampling procedure that terminates in m iterations (m is not necessarily fixed) and provides a coefficient α_i and a distribution D_i to sample $x_i \sim D_i$ in every iteration $i \in [m]$.

We say P is an ε-WBSP if it satisfies the following two properties:

1. With probability 0.9, for weight $w_i = \alpha_i \cdot \frac{D(x_i)}{D_i(x_i)}$ of each $i \in [m]$,

$$\sum_{i=1}^{m} w_i \cdot |h(x_i)|^2 \in \left[1 - 10\sqrt{\varepsilon}, 1 + 10\sqrt{\varepsilon}\right] \cdot \|h\|_D^2 \quad \forall h \in \mathcal{F}.$$

2. The coefficients always have $\sum_{i=1}^{m} \alpha_i \le \frac{5}{4}$ and $\alpha_i \cdot K_{\mathsf{IS},D_i} \le \frac{\varepsilon}{2}$ for all $i \in [m]$.

This definition describes a general sampling procedure that uses a few samples to represent the whole continuous signal, and the sampling procedure should satisfy two properties: one guarantees that the norm of any function in a function family is preserved, and another guarantees that the norm of noise is also preserved.

In Section 11.7.1, we review some results in [CP19a] and show that WBSP can be implemented via randomized spectral sparsification. In Section 11.7.2, we design a data structure and improve the time efficiency of the WBSP. In Section 11.7.3, we discover a tradeoff between the preprocessing cost and the query cost, which can improve the space complexity.

11.7.1 Randomized BSS implies a WBSP

In this section, we review the result of [CP19a], which shows that the Randomized BSS algorithm [BSS12, LS15] implies a well-balanced sampling procedure.

Lemma 11.34 (Lemma 5.1 in [CP19a])**.** *Let G be any domain. Given any dimension d linear function family $\mathcal{F}$ of function $f : G \to \mathbb{C}$,*

$$\mathcal{F} = \{f(t) = \sum_{j=1}^{d} v_j u_j(t) | v_j \in \mathbb{C}\},$$

where $u_j : G \to \mathbb{C}$. Given any distribution D over G, and any $\varepsilon > 0$, there exists an efficient procedure (Algorithm 51) that runs in $O(\varepsilon^{-1} d^3 |G| + \varepsilon^{-1} d^{\omega+1})$ time and outputs a set $S \subseteq G$ and weight w such that

- $|S| = O(d/\varepsilon),\ w \in \mathbb{R}^{|S|}$,

- *the procedure is an ε-WBSP,*

holds with probability $1 - \frac{1}{200}$.

Algorithm 51 A well-balanced sampling procedure based on Randomized BSS (see [CP19a])

```
1: procedure RandBSS($d, \mathcal{F}, D, \varepsilon$)
2:     Find an orthonormal basis $v_1, \dots, v_d$ of $\mathcal{F}$ under $D$
3:     Set $\gamma \leftarrow \sqrt{\varepsilon}/3$ and $\mathsf{mid} \leftarrow \frac{4d/\gamma}{1/(1-\gamma)-1/(1+\gamma)}$
4:     $j \leftarrow 0, B_0 \leftarrow 0$
5:     $l_0 \leftarrow -2d/\gamma, u_0 \leftarrow 2d/\gamma$
6:     while $u_{j+1} - l_{j+1} < 8d/\gamma$ do
7:         $\Phi_j \leftarrow \mathrm{tr}[(u_j I - B_j)^{-1}] + \mathrm{tr}[(B_j - l_j I)^{-1}]$    ▷ The potential function at iteration $j$.
8:         Set the coefficient $\alpha_j \leftarrow \frac{\gamma}{\Phi_j} \cdot \frac{1}{\mathsf{mid}}$
9:         Set $v(x) \leftarrow \big(v_1(x), \dots, v_d(x)\big)$
10:        for $x \in \mathrm{supp}(D)$ do
11:            Set the distribution
                 $$D_j(x) \leftarrow D(x) \cdot \left(v(x)^\top (u_j I - B_j)^{-1} v(x) + v(x)^\top (B_j - l_j I)^{-1} v(x)\right) / \Phi_j$$
12:        end for
13:        Sample $x_j \sim D_j$ and set a scale $s_j \leftarrow \frac{\gamma}{\Phi_j} \cdot \frac{D(x_j)}{D_j(x_j)}$
14:        $B_{j+1} \leftarrow B_j + s_j \cdot v(x_j) v(x_j)^\top$
15:        $u_{j+1} \leftarrow u_j + \frac{\gamma}{\Phi_j(1-\gamma)}$,    $l_{j+1} \leftarrow l_j + \frac{\gamma}{\Phi_j(1+\gamma)}$
16:        $j \leftarrow j + 1$
17:    end while
18:    $m \leftarrow j$
19:    Assign the weight $w_j \leftarrow s_j/\mathsf{mid}$ for each $x_j$
20:    return $\{x_1, x_2, \cdots, x_m\}, w$
21: end procedure
```

11.7.2 Fast implementation of WBSP

In this section, we give a fast implementation of Algorithm 51:

Theorem 11.35 (Fast implementation of WBSP)**.** *Let G be any domain. Given any* *dimension d linear function family $\mathcal{F}$ of function $f : G \to \mathbb{C}$,*

$$\mathcal{F} = \{f(t) = \sum_{j=1}^{d} v_j u_j(t) | v_j \in \mathbb{C}\},$$

where $u_j : G \to \mathbb{C}$. Given any distribution D over G, and any $\varepsilon > 0$, there exists an efficient procedure (Algorithm 52) that runs in $O(d^2|G| + \varepsilon^{-1} d^3 \log |G| + \varepsilon^{-1} d^{\omega+1})$ time and outputs a set $S \subseteq G$ and weight $w \in \mathbb{R}^{|S|}$ such that the following properties hold with probability at least 0.995:

- $|S| = O(d/\varepsilon)$,
- *the procedure is an ε-WBSP.*

Our algorithm is based on a data structure for solving the *online quadratic-form sampling problem* defined as follows:

Problem 11.36 (Online Quadratic-Form Sampling Problem)**.** Given n vectors $v_1, \dots, v_n \in \mathbb{R}^d$ and n coefficients $\alpha_1, \dots, \alpha_n$, for any PSD matrix $A \in \mathbb{R}^{d\times d}$, output a sample $i \in [n]$ from the following distribution $\mathcal{D}_A$:

$$\Pr_{\mathcal{D}_A}[i] := \frac{\alpha_i \cdot v_i^\top A v_i}{\sum_{j=1}^{n} \alpha_j \cdot v_j^\top A v_j} \quad \forall i \in [n]. \tag{11.11}$$

Theorem 11.37. *There is a data structure (Algorithm 53) that uses $O(nd^2)$ spaces for the Online Quadratic-Form Sampling Problem with the following procedures:*

- $\textsc{Init}(n, d, \{v_1, \dots, v_n\} \subset \mathbb{R}^d, \{\alpha_1, \dots, \alpha_n\} \subset \mathbb{R})$*: the data structure preprocesses in time $O(nd^2)$.*
- $\textsc{Query}(A \in \mathbb{R}^{d\times d})$*: Given a PSD matrix A, the* $\textsc{Query}$ *operation samples $i \in [n]$ exactly from the probability distribution $\mathcal{D}_A$ defined in Problem 11.36 in $O(d^2 \log n)$-time.*

Algorithm 52 Our fast implementation of well-balanced sampling procedure

```
1:  procedure RANDBSS+($d, \mathcal{F}, D, \varepsilon$)                    ▷ Theorem 11.35
2:      /*Preprocessing*/
3:      Find an orthonormal basis $v_1, \dots, v_d$ of $\mathcal{F}$ under $D$
4:      $\gamma \leftarrow \sqrt{\varepsilon}/3$ and $\mathsf{mid} \leftarrow \frac{4d/\gamma}{1/(1-\gamma)-1/(1+\gamma)}$
5:      $j \leftarrow 0, B_0 \leftarrow 0$
6:      $l_0 \leftarrow -2d/\gamma, u_0 \leftarrow 2d/\gamma$
7:      $\delta \leftarrow 1/\mathrm{poly}(d)$
8:                                  ▷ Let $v(x) = \big(v_1(x), \dots, v_d(x)\big) \in \mathbb{R}^d$
9:      DS.INIT($|D|, d, \{v(x_1), \cdots, v(x_{|D|})\} \subset \mathbb{R}^d, \{D(x_1), \dots, D(x_{|D|})\} \subset \mathbb{R}$)    ▷ Algorithm 53
10:     /*Iterative step*/
11:     while $u_{j+1} - l_{j+1} < 8d/\gamma$ do
12:         $\Phi_j \leftarrow \mathrm{tr}[(u_j I - B_j)^{-1}] + \mathrm{tr}[(B_j - l_j I)^{-1}]$    ▷ The potential function at iteration $j$.
13:         $\alpha_j \leftarrow \frac{\gamma}{\Phi_j} \cdot \frac{1}{\mathsf{mid}}$
14:         $E_j \leftarrow (u_j I - B_j)^{-1} + (B_j - l_j I)^{-1}$
15:         $q \leftarrow$ DS.QUERY($E_j/\Phi_j$)                    ▷ $q \in [|D|]$, Algorithm 53
16:         $\mathsf{x}_j \leftarrow x_q$ and set a scale $s_j \leftarrow \frac{\gamma}{v(\mathsf{x}_j)^\top E_j v(\mathsf{x}_j)}$
17:         $B_{j+1} \leftarrow B_j + s_j \cdot v(\mathsf{x}_j) v(\mathsf{x}_j)^\top$
18:         $u_{j+1} \leftarrow u_j + \frac{\gamma}{\Phi_j(1-\gamma)}$,    $l_{j+1} \leftarrow l_j + \frac{\gamma}{\Phi_j(1+\gamma)}$
19:         $j \leftarrow j + 1$
20:     end while
21:     $m \leftarrow j$
22:     Assign the weight $w_j \leftarrow s_j/\mathsf{mid}$ for each $x_j$
23:     return $\{\mathsf{x}_1, \mathsf{x}_2, \cdots, \mathsf{x}_m\}, w$
24: end procedure
```

Proof. The pseudo-code of the algorithm is given as Algorithm 53. The idea is to build a binary tree such that each node has an interval in $[l, \dots, r] \subset [1, \dots, n]$ and stores a matrix $\sum_{i=l}^r \alpha_i \cdot v_i v_i^\top$. For each internal node with interval $[l, \dots, r]$, its left child node has interval $[l, \dots, \lfloor (l+r)/2 \rfloor]$, and its right child node has interval $[\lfloor (l+r)/2 \rfloor + 1, \dots, r]$.

We first prove the correctness. Suppose the output of QUERY is $i \in [n]$. We compute its probability. Let $u_0 = \mathsf{root}, u_1, \dots, u_t$ be the path from the root of the tree to the leaf with id $= i$. Then, we have

$$\Pr[u_t] = \prod_{j=1}^{t} \Pr[u_j | u_{j-1}] = \prod_{j=1}^{t} \frac{\sum_{k=l_j}^{r_j} \alpha_k \cdot v_k^\top A v_k}{\sum_{k=l_{j-1}}^{r_{j-1}} \alpha_k \cdot v_k^\top A v_k} = \frac{\alpha_i \cdot v_i^\top A v_i}{\sum_{k=1}^{n} \alpha_k \cdot v_k^\top A v_k},$$

where $[l_j, \ldots, r_j]$ is the range of the node u_j, the first step follows from the conditional probability, the second step follows from Line 7 in Algorithm 53, and the last step follows from the telescoping products. Hence, we get that

$$\Pr[\textsc{Query}(A) = i] = \Pr_{\mathcal{D}_A}[i] \quad \forall i \in [n].$$

Hence, the sampling distribution is the same as the Online Quadratic-Form Sampling Problem's distribution.

For the running time, in the preprocessing stage, we build the binary tree recursively. It is easy to see that the number of nodes in the tree is $O(n)$ and the depth is $O(\log n)$. For a leaf node, we take $O(d^2)$-time to compute the matrix $\alpha_i \cdot v_i v_i^\top \in \mathbb{R}^{d\times d}$. For an internal node, we take $O(d^2)$-time to add up the matrices of its left and right children. Thus, the total preprocessing time is $O(nd^2)$.

In the query stage, we walk along a path from the root to a leaf, which has $O(\log n)$ steps. In each step, we compute the inner product between A and the current node's matrix, which takes $O(d^2)$-time. And we compute the inner product between A and its left child node's matrix, which also takes $O(d^2)$-time. Then, we toss a coin and decide which subtree to move. Hence, each query takes $O(d^2 \log n)$-time.

The theorem is then proved. $\square$

Lemma 11.38 (Running time of Procedure RANDBSS+ in Algorithm 52)**.** *Algorithm 52 runs in*

- $O(|D|d^2)$*-time for preprocessing,*
- $O(d^2 \log(|D|) + d^\omega)$*-time per iteration, and*

Algorithm 53 Quadratic-form sampling data structure

```
1:  structure Node
2:      $V \in \mathbb{R}^{d\times d}$
3:      left, right                    ▷ Point to the left/right child in the tree
4:  end structure
5:  data structure DS
6:  members
7:      $n \in \mathbb{N}$                          ▷ The number of vectors
8:      $v_1, \dots, v_n \in \mathbb{R}^d$                 ▷ $d$-dimensional vectors
9:      $\alpha_1, \dots, \alpha_n \in \mathbb{R}$                       ▷ Coefficients
10:     root: Node                             ▷ The root of the tree
11: end members
12: procedure BuildTree($l, r$)        ▷ $[l, \dots, r]$ is the range of the current node
13:     p ← new Node
14:     if $l = r$ then                                     ▷ Leaf node
15:         p.$V \leftarrow \alpha_l \cdot v_l v_l^\top$                    ▷ It takes $O(d^2)$-time
16:     else                                            ▷ Internal node
17:         $mid \leftarrow \lfloor (l+r)/2 \rfloor$
18:         p.left ← BuildTree($l, mid$)
19:         p.right ← BuildTree($mid + 1, r$)
20:         p.$V \leftarrow$ (p.left).$V$ + (p.right).$V$          ▷ It takes $O(d^2)$-time
21:     end if
22:     return p
23: end procedure
24: procedure Init($n, d, \{v_i\}_{i\in[n]} \subseteq \mathbb{R}^d, \{\alpha_i\}_{i\in[n]} \subseteq \mathbb{R}$)
25:     $v_i \leftarrow v_i$, $\alpha_i \leftarrow \alpha_i$ for $i \in [n]$
26:     root ← BuildTree($1, n$)
27: end procedure
```

- *$O(\varepsilon^{-1}d)$ iterations.*

Thus, the total running time is,

$$O(|D|d^2 + \varepsilon^{-1}d \cdot (d^2 \log |D| + d^\omega)).$$

Proof. In each call of the Procedure RandBSS+ in Algorithm 52,

- Finding orthonormal basis takes $O(|D|d^2)$.

Algorithm 54 Quadratic-form sampling data structure, Continue

```
1: procedure QUERY($A \in \mathbb{R}^{d \times d}$)
2:     $\mathsf{p} \leftarrow \mathsf{root}$, $l \leftarrow 1$, $r \leftarrow n$
3:     $s \leftarrow 0$
4:     while $l \neq r$ do                                  ▷ There are $O(\log n)$ iterations
5:         $w \leftarrow \langle \mathsf{p}.V, A \rangle$                          ▷ It takes $O(d^2)$-time
6:         $w_\ell \leftarrow \langle (\mathsf{p}.\mathrm{left}).V, A \rangle$
7:         Sample $c$ from Bernoulli$(w_\ell / w)$
8:         if $c = 0$ then
9:             $\mathsf{p} \leftarrow \mathsf{p}.\mathrm{left}$, $r \leftarrow \lfloor (l + r)/2 \rfloor$
10:        else
11:            $\mathsf{p} \leftarrow \mathsf{p}.\mathrm{right}$, $l \leftarrow \lfloor (l + r)/2 \rfloor + 1$
12:        end if
13:    end while
14:    return $l$
15: end procedure
16: end data structure
```

- In the line 9, it runs $O(|D|d^2)$ times.
- The while loop repeat $O(\varepsilon^{-1}d)$ times.
  - Line 14 is computing $(u_j I - B_j) \in \mathbb{C}^{d \times d}$, $(u_j I - B_j)^{-1}$. This part takes $O(d^\omega)$ time[12].
  - Note that line 15 of Procedure RANDBSS+ in Algorithm 52 runs $O(d^2 \log |D|)$ times.

So, the time complexity of Procedure RANDBSS+ in Algorithm 52 is

$$O(|D|d^2 + \varepsilon^{-1}d \cdot (d^2 \log |D| + d^\omega)).$$

□

[12]Note that this step seems to be very difficult to speed up via the Sherman-Morrison formula since u_j changes in each iteration and the update is of high rank.

Lemma 11.39 (Correctness of Procedure RANDBSS+ in Algorithm 52)**.** *Given any dimension d linear space $\mathcal{F}$, any distribution D over the domain of $\mathcal{F}$, and any $\varepsilon > 0$,* RANDBSS+$(d, \mathcal{F}, D, \varepsilon)$ *is an ε-WBSP that terminates in $O(d/\varepsilon)$ rounds with probability $1 - 1/200$.*

Proof. We first claim that, for each $j \in [m]$, x_j has the same distribution as D_j, where

$$D_j(x) = D(x) \cdot (v(x)^\top E_j v(x))/\Phi_j \quad \forall x \in D$$

Notice that sampling from distribution D_j can be reformulated as an Online Quadratic-Form Sampling Problem: the vectors are $\{v(x)\}_{x\in D}$, the coefficients are $\{D(x)\}_{x\in D}$, and the query matrix is $E_j' := E_j/\Phi_j$. Then, we have $D_j = \mathcal{D}_{E_j'}$ defined in Problem 11.36. Hence, by Theorem 11.37, we can use the data structure (Algorithm 53) to efficiently sample from D_j.

Therefore, the sample x_j in each iteration is generated from the same distribution as the original randomized BSS algorithm (Algorithm 51). Then, the WBSP guarantee and the number of iterations immediately follow from the proof of [CP19a, Lemma 5.1].

The proof of the lemma is then completed. □

Proof of Theorem 11.35. The running time of the algorithm follows from Lemma 11.38, and the correctness follows from Lemma 11.39. □

11.7.3 Trade-off between preprocessing and query

In this section, we consider the preprocessing and query trade-off in the data structure for quadratic form sampling problem. In the following theorem, we give a new data structure that takes less time in preprocessing and more time for each query than Theorem 11.37, and the space complexity is also reduced from $O(nd^2)$ to $O(nd)$.

Theorem 11.40. *There is a data structure (Algorithms 55 and 56) that uses $O(nd)$ spaces for the Online Quadratic-Form Sampling Problem with the following procedures:*

- $\textsc{Init}(n, d, \{v_1, \dots, v_n\} \subset \mathbb{R}^d, \{\alpha_1, \dots, \alpha_n\} \subset \mathbb{R})$*: the data structure preprocesses in time $O(nd^{\omega-1})$.*
- $\textsc{Query}(A \in \mathbb{R}^{d\times d})$*: Given a PSD matrix A, the* $\textsc{Query}$ *operation samples $i \in [n]$ exactly from the probability distribution $\mathcal{D}_A$ defined in Problem 11.36 in $O(d^2 \log(n/d) + d^\omega)$-time.*

Proof. The time and space complexities follow from Lemma 11.41. And the correctness follows from Lemma 11.42. □

Lemma 11.41 (Time and space complexities of Algorithms 55 and 56)**.** *The* $\textsc{Init}$ *procedure takes $O(nd^{\omega-1})$-time. The* $\textsc{Query}$ *procedure takes $O(d^2 \log(n/d)+d^\omega)$-time. The data structure uses $O(nd)$-space.*

Proof. We prove the space and time complexities of the data structure as follows:
Space complexity: Let $m = n/d$. It is easy to see that there are $O(m)$ nodes in the data structure. And each node has two d-by-d matrices. Hence, the total space used by the data structure is $O(n/d) \cdot O(d^2) = O(nd)$.

Time complexity: In the preprocessing stage, the time-consuming step is the call of $\textsc{BuildTree}$. There are $O(m)$ internal nodes and $O(m)$ leaf nodes. Each internal node takes $O(d^2)$-time to construct the matrix V_1 (Line 22). For each leaf node, it takes $O(d^2)$-time to form the matrix V_2 (Line 15). And it takes $O(d^\omega)$-time to compute the matrix V_1 (Line 16). Hence, the total running time of $\textsc{BuildTree}$ is $O(md^\omega) = O(nd^{\omega-1})$.

In the query stage, the While loop in the QUERY procedure (Line 17) is the same as in Algorithm 53. Since there are $O(m)$ nodes in the tree, it takes $O(d^2 \log m)$-time. Then, in the BLOCKSAMPLING procedure, it takes $O(d^\omega)$-time to compute the matrix U (Line 9), and it takes $O(d)$-time to sample an index from the distribution $\mathcal{D}_l$ (Line 11). Hence, the total running time for each query is $O(d^2 \log m + d^\omega) = O(d^2 \log(n/d) + d^\omega)$.

The proof of the lemma is then completed. $\square$

Lemma 11.42 (Correctness of Algorithm 56)**.** *The distribution of the output of the* QUERY*(A) is $\mathcal{D}_A$ defined by Eq.* (11.11)*.*

Proof. For simplicity, we assume that all the coefficients $\alpha_i = 1$.

Let $u_0 = \mathsf{root}, u_1, \ldots, u_t$ be the path in the While loop (Line 17) from the root of the tree to the leaf with index $l \in [m]$. By the construction of leaf node, we have

$$V_1 = V_2 V_2^\top = \begin{bmatrix} v_{(l-1)d+1} & \cdots & v_{ld} \end{bmatrix} \begin{bmatrix} v_{(l-1)d+1}^\top \\ \vdots \\ v_{ld}^\top \end{bmatrix} = \sum_{i=(l-1)d+1}^{ld} v_i v_i^\top,$$

which is the same as the V-matrix in Algorithm 53. Hence, similar to the proof of Theorem 11.37, we have

$$\Pr[u_t] = \prod_{j=1}^{t} \Pr[u_j | u_{j-1}] = \frac{\sum_{i=(l-1)d+1}^{ld} v_i^\top A v_i}{\sum_{i=1}^{n} v_i^\top A v_i}.$$

where $\{(l-1)d+1, \ldots, ld\}$ is the range of the node u_t and $\{1, \ldots, n\}$ is the range of u_0.

Then, consider the BLOCKSAMPLING procedure. Let $\{v_1, \ldots, v_d\}$ be the vectors in the input block. At Line 9, we have

$$U = V_2^\top A V_2 = \begin{bmatrix} v_1^\top \\ \vdots \\ v_d^\top \end{bmatrix} A \begin{bmatrix} v_1 & \cdots & v_d \end{bmatrix}.$$

For $i \in [d]$, the i-th element in the diagonal of U is

$$U_{i,i} = v_i^\top A v_i.$$

Hence,

$$\Pr[\textsc{BlockSampling} = i] = \frac{v_i^\top A v_i}{\sum_{j=1}^d v_j^\top A v_j}.$$

Therefore, for any $k \in [n]$, if $k = (l-1)d + r$ for some $l, r \in \mathbb{N}$, then the sample probability is

$$\begin{aligned}\Pr[\textsc{Query}(A) = k] = & \Pr[\textsc{BlockSampling} = k \mid u_t = \text{Block } l] \cdot \Pr[u_t = \text{Block } l] \\ = & \frac{v_k^\top A v_k}{\sum_{i=(l-1)d+1}^{ld} v_i^\top A v_i} \cdot \frac{\sum_{i=(l-1)d+1}^{ld} v_i^\top A v_i}{\sum_{i=1}^n v_i^\top A v_i} \\ = & \frac{v_k^\top A v_k}{\sum_{i=1}^n v_i^\top A v_i} \\ = & \mathcal{D}_A(k).\end{aligned}$$

The lemma is then proved. □

As a corollary, we get a WBSP using less space:

Corollary 11.43 (Space efficient implementation of WBSP)**.** *By plugging-in the new data structure (Algorithms 55 and 56) to* FasterRandSamplingBSS *(Algorithm 52), we get an algorithm taking $O(|D|d^2 + \gamma^{-2}d \cdot (d^2 \log |D| + d^\omega))$-time and using $O(|D|d)$-space.*

Proof. In the preprocessing stage of FasterRandSamplingBSS, we take $O(|D|d^2)$-time for Gram-Schmidt process and $O(|D|d^{\omega-1})$-time for initializing the data structure (Algorithm 55).

The number of iterations is $\gamma^{-2}d$. In each iteration, the matrix E_j can be computed in $O(d^\omega)$-time. And querying the data structure takes $O(d^2 \log(|D|/d) + d^\omega)$-time.

Hence, the total running time is

$$O\left(|D|d^2 + |D|d^{\omega-1} + \gamma^{-2}d(d^2\log(|D|/d) + d^\omega)\right) = O\left(|D|d^2 + \gamma^{-2}d^{\omega+1} + \gamma^{-2}d^2\log|D|\right).$$

For the space complexity, the data structure uses $O(|D|d)$-space. The algorithm uses $O(d^2)$ extra space in preprocessing and each iteration. Hence, the total space complexity is $O(|D|d)$. $\square$

11.8 Sketch Distillation for Fourier Sparse Signals

In Section 11.6, we show an oblivious approach for sketching Fourier sparse signals. However, there are two issues of using this sketching method in Signal estimation: 1. The sketch size too large. 2. The noise in the observed signal could have much larger energy on the sketching set than its average energy. To resolve these two issues, in this section, we propose a method called *sketch distillation* to post-process the sketch obtained in Section 11.6 that can reduce the sketch size to $O(k)$ and prevent the energy of noise being amplified too much. However, we need some extra information about the signal $x^*(t)$: we assume that the frequencies of the noiseless signal $x(t)$ are known. But the sketch distillation process can still be done *partially oblivious*, i.e., we do not need to access/sample the signal.

In Section 11.8.1, we show our distillation algorithms for one-dimensional signals. Then, we generalize the sketch distillation for high-dimensional signals in Section 11.8.2 and for discrete signals in Section 11.8.3.

11.8.1 Sketch distillation for one-dimensional signals

In this section, we show how to distill the sketch produced by Lemma 11.28 from $O(k\log k)$-size to $O(k)$-size, using an ε-well-balanced sampling procedure developed in Section 11.7.

Lemma 11.44 (Fast distillation for one-dimensional signal)**.** *Given $f_1, f_2, \cdots, f_k \in \mathbb{R}$. Let $x^*(t) = \sum_{j=1}^k v_j \exp(2\pi \mathbf{i} f_j t)$. Let $\eta = \min_{i\neq j}|f_j - f_i|$. For any accuracy* *parameter $\varepsilon \in (0, 0.1)$, there is an algorithm* FastDistill1D *(Algorithm 57) that runs in $O(\varepsilon^{-2}k^{\omega+1})$-time and outputs a set $S \subset [-T, T]$ of size $s = O(k/\varepsilon^2)$ and a weight vector $w \in \mathbb{R}^s_{\geq 0}$ such that,*

$$(1-\varepsilon)\|x^*(t)\|_T \leq \|x^*(t)\|_{S,w} \leq (1+\varepsilon)\|x^*(t)\|_T$$

holds with probability 0.99.

Furthermore, for any noise signal $g(t)$, the following holds with high probability:

$$\|g\|^2_{S,w} \lesssim \|g\|^2_T,$$

where $\|x\|^2_T := \frac{1}{2T}\int_{-T}^{T} |x(t)|^2 \mathrm{d}t$.

Proof. For the convenient, in the proof, we use time duration $[-T, T]$. Let $D(t)$ be defined as follows:

$$D(t) = \begin{cases} c/(1-|t/T|), & \text{for } |t| \leq T(1-1/k) \\ c \cdot k, & \text{for } |t| \in [T(1-1/k), T] \end{cases}$$

where $c = O(T^{-1}\log^{-1}(k))$ a fixed value such that $\int_{-T}^{T} D(t)\mathrm{d}t = 1$.

First, we randomly pick up a set $S_0 = \{t_1, \cdots, t_{s_0}\}$ of $s_0 = O(\varepsilon_0^{-2}k\log(k)\log(1/\rho_0))$ i.i.d. samples from $D(t)$, and let $w'_i := 2/(Ts_0D(t_i))$ for $i \in [s_0]$ be the weight vector, where ε_0, ρ_0 are parameters to be chosen later.

By Lemma 11.28, we know that (S_0, w') gives a good weighted sketch of the signal that can preserve the norm with high probability. More specifically, with probability $1 - \rho_0$,

$$(1-\varepsilon_0)\|x^*(t)\|^2_T \leq \|x^*(t)\|^2_{S_0,w'} \leq (1+\varepsilon_0)\|x^*(t)\|^2_T. \tag{11.12}$$

Then, we will select $s = O(k/\varepsilon_1^2)$ elements from S_0 and output the corresponding weights $w_1, w_2, \cdots, w_s$ by applying RandBSS+ with the following parameter: replacing d by k, ε by ε_1^2, and D by $D(t_i) = w'_i / \sum_{j\in[s_0]} w'_j$ for $i \in [s_0]$.

By Theorem 11.35 and the property of WBSP (Definition 11.10), we obtain that with probability 0.995,

$$(1-\varepsilon_1)\|x^*(t)\|_{S_0,w'}^2 \leq \|x^*(t)\|_{S,w}^2 \leq (1+\varepsilon_1)\|x^*(t)\|_{S_0,w'}^2.$$

Combining with Eq. (11.12), we conclude that

$$\begin{aligned}\|x^*\|_{S,w}^2 &\in [1-\varepsilon_1, 1+\varepsilon_1]\cdot\|x^*\|_{S_0,w'}^2\\ &\in [(1-\varepsilon_0)(1-\varepsilon_1),(1+\varepsilon_0)(1+\varepsilon_1)]\cdot\|x^*\|_T^2\\ &\in [1-\varepsilon,1+\varepsilon]\cdot\|x^*\|_T^2,\end{aligned}$$

where the second step follows from Eq. (11.12) and the last stpe follows by taking $\varepsilon_0=\varepsilon_1=\varepsilon/4$.

The overall success probability follows by taking union bound over the two steps and taking $\rho_0=0.001$. The running time of Algorithm 57 follows from Claim 11.45. And the furthermore part follows from Claim 11.46.

The proof of the lemma is then completed. □

Claim 11.45 (Running time of Procedure FastDistill1D in Algorithm 57)**.** *Procedure* FastDistill1D *in Algorithm 57 runs in*

$$O(\varepsilon^{-2}k^{\omega+1})$$

time.

Proof. First, it is easy to see that Procedure WeightedSketch takes $O(\varepsilon^{-2}k\log(k))$-time.

By Theorem 11.35 with $|D|=O(\varepsilon^{-2}k\log(k))$, $d=k$, we have that the running time of Procedure RandBSS+ is

$$\begin{aligned}&O\left(k^2\cdot\varepsilon^{-2}k\log(k)+\varepsilon^{-2}k^3\log\left(\varepsilon^{-2}k\log(k)\right)+\varepsilon^{-2}k^{\omega+1}\right)\\ =&\ O\left(\varepsilon^{-2}k^{\omega+1}\right).\end{aligned}$$

Hence, the total running time of Algorithm 57 is $O\left(\varepsilon^{-2}k^{\omega+1}\right)$.

$\square$

Claim 11.46 (Preserve the energy of noise)**.** *Let (S, w) be the outputs of Algorithm 57. Then, we have that*

$$\|g(t)\|_{S,w}^2 \lesssim \|g(t)\|_T^2,$$

holds with probability 0.99*.*

Proof. For the convenient, in the proof, we use time duration $[-T, T]$. Algorithm 57 has two stages of sampling.

In the first stage, Procedure WEIGHTEDSKETCH samples a set $S_0 = \{t_1', \ldots, t_{s_0}'\}$ of i.i.d. samples from the distribution D, and a weight vector w'. Then, we have

$$\begin{aligned}
\mathbb{E}\left[\|g(t)\|_{S_0,w'}^2\right] &= \mathbb{E}\left[\sum_{i=1}^{s_0} w_i' |g(t_i')|^2\right] \\
&= \sum_{i=1}^{s_0} \mathbb{E}_{t_i' \sim D}[w_i' |g(t_i')|^2] \\
&= \sum_{i=1}^{s_0} \mathbb{E}_{t_i' \sim D}\left[\frac{2}{T s_0 D(t_i')} |g(t_i')|^2\right] \\
&= \sum_{i=1}^{s_0} \mathbb{E}_{t_i' \sim \mathrm{Uniform([-T,T])}}[s_0^{-1} |g(t_i')|^2] \\
&= \mathbb{E}_{t \sim \mathrm{Uniform([-T,T])}}[|g(t)|^2] \\
&= \|g(t)\|_T^2
\end{aligned}$$

where the first step follows from the definition of the norm, the third step follows from the definition of w_i, the forth step follows from $\mathbb{E}_{t \sim D_0(t)}[\frac{D_1(t)}{D_0(t)} f(t)] = \mathbb{E}_{t \sim D_1(t)} f(t)$.

In the second stage, let P denote the Procedure RANDBSS+. With high probability, P is a ε-WBSP (Definition 11.10). By the Definition 11.10, each sample $t_i \sim D_i(t)$ and $w_i = \alpha_i \cdot \frac{D'(t_i)}{D_i(t_i)}$ in every iteration $i \in [s]$, where $\sum_{i=1}^{s} \alpha_i \leq 5/4$ and

$D'(t) = \frac{w'_t}{\sum_{t' \in S_0} w'_{t'}}$. As a result,

$$\begin{aligned}
\mathbb{E}_P[\|g(t)\|^2_{S,w}] =& \ \mathbb{E}_P\left[\sum_{i=1}^{s} w_i |g(t_i)|^2\right] \\
=& \sum_{i=1}^{s} \mathbb{E}_{t_i \sim D_i(t_i)}[w_i |g(t_i)|^2] \\
=& \sum_{i=1}^{s} \mathbb{E}_{t_i \sim D_i(t_i)}\left[\alpha_i \cdot \frac{D'(t_i)}{D_i(t_i)} |g(t_i)|^2\right] \\
=& \sum_{i=1}^{s} \mathbb{E}_{t_i \sim D'(t_i)}[\alpha_i |g(t_i)|^2] \\
\leq& \sup_P \{\sum_{i=1}^{s} \alpha_i\} \, \mathbb{E}_{t \sim D'(t)}[|g(t)|^2] \\
=& \sup_P \{\sum_{i=1}^{s} \alpha_i\} \|g(t)\|^2_{S_0,w'} \cdot (\sum_{t' \in S_0} w'_{t'})^{-1} \\
\lesssim& \ \rho^{-1} \cdot \|g(t)\|^2_{S_0,w'}.
\end{aligned}$$

where the first step follows from the definition of the norm, the third step follows from $w_i = \alpha_i \cdot \frac{D'(t_i)}{D_i(t_i)}$, the forth step follows from $\mathbb{E}_{t \sim D_0(t)} \frac{D_1(t)}{D_0(t)} f(t) = \mathbb{E}_{t \sim D_1(t)} f(t)$, the sixth step follows from $D'(t) = \frac{w'_t}{\sum_{t' \in S_0} w'_{t'}}$ and the definition of the norm, the last step follows from $\sum_{i=1}^{s} \alpha_i \leq 5/4$ and $(\sum_{t' \in S_0} w'_{t'})^{-1} = O(\rho^{-1})$ with probability at least $1 - \rho/2$.

Hence, combining the two stages together, we have

$$\mathbb{E}\left[\mathbb{E}_P[\|g(t)\|^2_{S,w}]\right] \lesssim \rho^{-1} \cdot \mathbb{E}\left[\|g(t)\|^2_{S_0,w'}\right] = \rho^{-1} \cdot \|g\|^2_T.$$

And by Markov inequality and union bound, we have

$$\Pr\left[\|g(t)\|^2_{S,w} \lesssim \rho^{-2} \|g(t)\|^2_T\right] \leq 1 - \rho.$$

$\square$

11.8.1.1 Sharper bound for the energy of orthogonal part of noise

In this section, we give a sharper analysis for the energy of $g^{\perp}$ on the sketch, which is the orthogonal projection of g to the space $\mathcal{F}$. More specifically, we can decompose an arbitrary function g into $g^{\parallel}+g^{\perp}$, where $g^{\parallel} \in \mathcal{F}$ and $\int_{[0,T]} \overline{h(t)} g^{\perp}(t) \mathrm{d}t = 0$ for all $h \in \mathcal{F}$. The motivation of considering $g^{\perp}$ is that $g^{\parallel}$ is also a Fourier sparse signal and its energy will not be amplified in the Signal Estimation problem. And the nontrivial part is to avoid the blowup of the energy of $g^{\perp}$, which is shown in the following lemma:

Lemma 11.47 (Preserving the orthogonal energy)**.** *Let $\mathcal{F}$ be an m-dimensional linear function family with an orthonormal basis $\{v_1, \ldots, v_m\}$ with respect to a distribution D. Let P be the ε-WBSP that generate a sample set $S = \{t_1, \ldots, t_s\}$ and coefficients $\alpha \in \mathbb{R}^s_{>0}$, where each t_i is sampled from distribution D_i for $i \in [s]$. Define the weight vector $w \in \mathbb{R}^s$ be such that $w_i := \alpha_i \frac{D(t_i)}{D_i(t_i)}$ for $i \in [s]$.*

For any noise function $g(t)$ that is orthogonal to $\mathcal{F}$ with respect to D, the following property holds with probability 0.99:

$$\sum_{i=1}^{m} |\langle g, v_i \rangle_{S,w}|^2 \lesssim \varepsilon \|g\|_D^2,$$

where $\langle g, v \rangle_{S,w} := \sum_{j=1}^{s} w_j \overline{v(t_j)} g(t_j)$.

Remark 11.6. We note that this lemma works for both continuous and discrete signals.

Remark 11.7. $|\langle g, v_i \rangle_{S,w}|^2$ corresponds to the energy of g on the sketch points in S. On the other hand, if we consider the energy on the whole time domain, we have $\langle g, v_i \rangle = 0$ for all $i \in [m]$. The above lemma indicates that this part of energy could be amplified by at most $O(\varepsilon)$, as long as the sketch comes from a WBSP.

Proof. We can upper-bound the expectation of $\sum_{i=1}^m |\langle g, v_i\rangle_{S,w}|^2$ as follows:

$$\begin{aligned}
\mathbb{E}\Big[\sum_{i=1}^m |\langle g, v_i\rangle_{S,w}|^2\Big] =&\ \mathbb{E}_{D_1,\ldots,D_s}\Big[\|w\|_1^2 \sum_{i=1}^m \big|\mathbb{E}_{t\sim D'}[\overline{v_j(t)} g(t)]\big|^2\Big] \\
=&\ \mathbb{E}_{D_1,\ldots,D_s}\Big[\sum_{i=1}^m \big|\sum_{j=1}^s w_j \overline{v_i(t_j)} g(t_j)]\big|^2\Big] \\
=&\ \sum_{i=1}^m \mathbb{E}_{D_1,\ldots,D_s}\Big[\big|\sum_{j=1}^s w_j \overline{v_i(t_j)} g(t_j)\big|^2\Big] \\
=&\ \sum_{i=1}^m \mathbb{E}_{D_1,\ldots,D_s}\Big[\sum_{j=1}^s w_j^2 |v_i(t_j)|^2 |g(t_j)|^2\Big] \\
=&\ \sum_{j=1}^s \mathbb{E}_{D_j}\Big[\sum_{i=1}^m w_j |v_i(t_j)|^2 \cdot w_j |g(t_j)|^2\Big] \\
\leq&\ \sum_{j=1}^s \sup_{t\in D_j}\Big\{w_j \sum_{i=1}^m |v_i(t)|^2\Big\} \cdot \mathbb{E}_{D_j}[w_j |g(t_j)|^2],
\end{aligned}$$

where the first step follows from Fact 11.48, the second step follows from the definition of D', the third follows from the linearity of expectation, the forth step follows from Fact 11.49, the last step follows by pulling out the maximum value of $w_j \sum_{i=1}^k |v_i(t)|^2$ from the expectation.

Next, we consider the first term:

$$\begin{aligned}
\sup_{t\in D_j}\Big\{w_j \sum_{i=1}^m |v_i(t)|^2\Big\} =&\ \sup_{t\in D_j}\Big\{\alpha_j \frac{D(t)}{D_j(t)} \sum_{i=1}^m |v_i(t)|^2\Big\} \\
=&\ \alpha_j \sup_{t\in D_j}\Big\{\frac{D(t)}{D_j(t)} \sup_{h\in\mathcal{F}}\big\{\frac{|h(t)|^2}{\|h\|_D^2}\big\}\Big\} \\
=&\ \alpha_j K_{\mathsf{IS},D_j}.
\end{aligned}$$

where the first step follows from the definition of w_j, the second step follows from Fact 11.50 that $\sup\limits_{h\in\mathcal{F}}\{\frac{|h(t_j)|^2}{\|h\|_D^2}\} = \sum_{i=1}^k |v_i(t_j)|^2$, the last step follows from the definition of K_{IS,D_j} (Eq. (11.3)).

Then, we bound the last term:

$$\mathbb{E}_{D_j}[w_j |g(t_j)|^2] = \mathop{\mathbb{E}}_{t_j\sim D_j}\Big[\alpha_j \frac{D(t_j)}{D_j(t_j)} |g(t_j)|^2\Big] = \alpha_j \mathop{\mathbb{E}}_{t_j\sim D}[|g(t_j)|^2] = \alpha_j \|g\|_D^2.$$

Combining the two terms together, we have

$$\begin{aligned}\mathbb{E}\Big[\sum_{i=1}^m |\langle g, v_i\rangle_{S,w}|^2\Big] \le&~ \sum_{j=1}^s (\alpha_j K_{\mathsf{IS},D_j}\cdot \alpha_j \|g\|_D^2)\\ \le&~ \Big(\sum_{j=1}^s \alpha_j\Big)\cdot \max_{j\in[s]}\{\alpha_j K_{\mathsf{IS},D_j}\}\cdot \|g\|_D^2\\ \le&~ \varepsilon\|g\|_D^2.\end{aligned}$$

where the last step follows from P being a ε-WBSP (Definition 11.10), which implies that $\sum_{j=1}^s \alpha_j = \frac{5}{4}$ and $\alpha_j K_{\mathsf{IS},D_j} \le \varepsilon/2$ for all $j\in[s]$.

Finally, by Markov's inequality, we have that

$$\sum_{i=1}^m |\langle g, v_i\rangle_{S,w}|^2 \lesssim \varepsilon\|g\|_D^2$$

holds with probability 0.99. □

Fact 11.48.

$$\sum_{i=1}^m |\langle g, v_i\rangle_{S,w}|^2 = \|w\|_1^2\cdot \sum_{i=1}^m \Big|\mathbb{E}_{t\sim D'}[\overline{v_i(t)}g(t)]\Big|^2,$$

where D' is a distribution defined by $D'(t_i) := \frac{w_i}{\|w\|_1}$ for $i\in[s]$.

Proof. We have:

$$\begin{aligned}\sum_{i=1}^m |\langle g, v_i\rangle_{S,w}|^2 =&~ \sum_{i=1}^m \Big|\sum_{j=1}^s w_j\overline{v_i(t_j)}g(t_j)\Big|^2\\ =&~ \sum_{i=1}^m \Big|\sum_{j=1}^s \frac{w_j\overline{v_i(t_j)}g(t_j)}{\sum_{j'=1}^s w_{j'}}\Big|^2\cdot\Big(\sum_{j'=1}^s w_{j'}\Big)^2\\ =&~ \Big(\sum_{j'=1}^s w_{j'}\Big)^2\cdot \sum_{i=1}^m \Big|\mathbb{E}_{t\sim D'}[\overline{v_i(t)}g(t)]\Big|^2.\end{aligned}$$

□

Fact 11.49. *For any $i\in[m]$, we have*

$$\mathbb{E}_{D_1,\dots,D_s}\Big[\big|\sum_{j=1}^s w_j\overline{v_i(t_j)}g(t_j)\big|^2\Big] = \mathbb{E}_{D_1,\dots,D_s}\Big[\sum_{j=1}^s w_j^2|v_i(t_j)|^2|g(t_j)|^2\Big].$$

Proof. We first show that for any $i \in [m]$ and $j \in [s]$,

$$\begin{aligned}
\mathop{\mathbb{E}}_{t_j \sim D_j}[w_j \overline{v_i(t_j)} g(t_j)] &= \mathop{\mathbb{E}}_{t_j \sim D_j}[\alpha_j \frac{D(t_j)}{D_j(t_j)} \overline{v_i(t_j)} g(t_j)] \\
&= \alpha_j \mathop{\mathbb{E}}_{t_j \sim D}[\overline{v_i(t_j)} g(t_j)] \\
&= 0. \qquad (11.13)
\end{aligned}$$

where the first step follows from the definition of w_i, the third step follows from $g(t)$ is orthonormal with $v_i(t)$ for any $i \in [k]$.

Then, we can expand LHS as follows:

$$\begin{aligned}
&\mathbb{E}_{D_1,\ldots,D_s}\Big[\big|\sum_{j=1}^{s} w_j \overline{v_i(t_j)} g(t_j)\big|^2\Big] \\
= \; &\mathbb{E}_{D_1,\ldots,D_s}\Big[\big(\sum_{j=1}^{s} w_j \overline{v_i(t_j)} g(t_j)\big)^* \big(\sum_{j=1}^{s} w_j \overline{v_i(t_j)} g(t_j)\big)\Big] \\
= \; &\mathbb{E}_{D_1,\ldots,D_s}\Big[\sum_{j,j'=1}^{s} w_j w_{j'} v_i(t_j) \overline{g(t_j) v_i(t_{j'})} g(t_{j'})\Big] \\
= \; &\sum_{j,j'=1}^{s} \mathbb{E}_{D_1,\ldots,D_s}[w_j w_{j'} v_i(t_j) \overline{g(t_j) v_i(t_{j'})} g(t_{j'})] \\
= \; &\sum_{j=1}^{s} \mathbb{E}[w_j^2 |v_i(t_j)|^2 |g(t_j)|^2] + \sum_{1 \le j < j' \le s} 2\Re\, \mathbb{E}_{D_1,\ldots,D_j}[w_j w_{j'} v_i(t_j) \overline{g(t_j) v_i(t_{j'})} g(t_{j'})] \\
= \; &\mathrm{RHS} + \sum_{1 \le j < j' \le s} 2\Re\, \mathbb{E}_{D_1,\ldots,D_j}\Big[w_j v_i(t_j) \overline{g(t_j)}\, \mathbb{E}_{D_{j+1},\ldots,D_{j'}}[w_{j'} \overline{v_i(t_{j'})} g(t_{j'})]\Big] \\
= \; &\mathrm{RHS} + \sum_{1 \le j < j' \le s} 2\Re\, \mathbb{E}_{D_1,\ldots,D_j}[w_j v_i(t_j) \overline{g(t_j)} \cdot 0] \\
= \; &\mathrm{RHS},
\end{aligned}$$

where the third step follows from the linearity of expectation, the fifth step follows from t_j only depends on $t_1, \ldots, t_{j-1}$, and the sixth step follows from Eq. (11.13). $\square$

Fact 11.50. *Let $\{v_1, \ldots, v_k\}$ be an orthonormal basis of $\mathcal{F}$ with respect to the distribution D. Then, we have*

$$\sup_{h \in \mathcal{F}} \Big\{\frac{|h(t)|^2}{\|h\|_D^2}\Big\} = \sum_{i=1}^{k} |v_i(t)|^2$$

Proof. Then,

$$\begin{aligned}
\sup_{h\in\mathcal{F}}\Big\{\frac{|h(t)|^2}{\|h\|_D^2}\Big\} &= \sup_{a\in\mathbb{C}^k}\Big\{\frac{|\sum_{i=1}^k a_i v_i(t)|^2}{\|a\|_2^2}\Big\} \\
&= \sup_{a\in\mathbb{C}^k:\|a\|_2=1}\Big|\sum_{i=1}^k a_i v_i(t)\Big|^2 \\
&= \sum_{i=1}^k |v_i(t)|^2,
\end{aligned}$$

where the first step follows from each $h\in\mathcal{F}$ can be expanded as $h=\sum_{i=1}^k a_i v_i$ and $\|h(t)\|_D^2 = \|a\|_2^2$ (Fact 11.14), the second step follows from the Cauchy-Schwartz inequality and taking $a=\frac{v(t)}{\|v(t)\|_2}$. □

11.8.2 Sketch distillation for high-dimensional signals

The goal of this section is to prove Lemma 11.51, which can reduce the sketch size of Lemma 11.29 for high-dimensional signals.

Lemma 11.51 (Distillation for high-dimensional signal)**.** *Given $f_1,f_2,\cdots,f_k\in\mathbb{R}^d$. Let $x^*(t)=\sum_{j=1}^k v_j e^{2\pi\mathbf{i}\langle f_j,t\rangle}$ for $t\in[0,T]^d$. Let $\eta=\min_{i\neq j}\|f_j-f_i\|_\infty$. For any accuracy parameter $\varepsilon\in(0,1)$, there is an algorithm* DistillHD *(Algorithm 58) that runs in $\widetilde{O}(\varepsilon^{-2}k^{O(d)})$-time and outputs a set $S\subset[0,T]^d$ of size $s=O(k/\varepsilon^2)$ and a weight vector $w\in\mathbb{R}^s_{\geq 0}$ such that*

$$(1-\varepsilon)\|x^*\|_T \leq \|x^*\|_{S,w} \leq (1+\varepsilon)\|x^*\|_T$$

holds with probability 0.99*.*

Furthermore, for any noise function $g(t)$, with high probability, it holds that

$$\|g\|_{S,w} \lesssim \|g\|_T.$$

Proof. First, we randomly and uniformly sample a set S_0 of $s_0=O(\varepsilon_0^{-2}k^{O(d)}\log(1/(\rho_0\varepsilon_0)))$ real number in $[0,T]^d$, where ε_0,ρ_0 are parameters to be chosen later.

By Lemma 11.29, we know that those points are good sketch of the high-dimensional signal and can preserve the norm with a large probability. More precisely, with probability $1-\rho_0$,

$$(1-\varepsilon_0)\|x^*\|_T^2 \leq \|x^*\|_{S_0}^2 \leq (1+\varepsilon_0)\|x^*\|_T^2. \tag{11.14}$$

Then, we will select $s = O(k)$ real number from S_0 and output s corresponding weight $w_1, w_2, \cdots, w_s$ by applying the Procedure RandBSS+ with setting the following parameter: replacing d by k, ε by ε_1^2, D by Uniform(S_0), and $\mathcal{F}$ by

$$\mathcal{F} = \Big\{ f(t) = \sum_{j=1}^{k} v_j \exp(2\pi \mathbf{i} \langle f_j, t\rangle) \;\Big|\; v_j \in \mathbb{C} \Big\}.$$

Then, by Theorem 11.35 and the property of WBSP (Definition 11.10), we obtain that with probability 0.995,

$$(1-\varepsilon_1)\|x^*\|_{S_0}^2 \leq \|x^*\|_{S,w}^2 \leq (1+\varepsilon_1)\|x^*\|_{S_0}^2.$$

Combining with Eq. (11.14), we conclude that

$$\begin{aligned}
\|x^*\|_{S,w}^2 &\in [1-\varepsilon_1, 1+\varepsilon_1] \cdot \|x^*\|_{S_0}^2 \\
&\in [(1-\varepsilon_0)(1-\varepsilon_1), (1+\varepsilon_0)(1+\varepsilon_1)] \cdot \|x^*\|_T^2 \\
&\in [1-\varepsilon, 1+\varepsilon] \cdot \|x^*\|_T^2,
\end{aligned}$$

where the second step follows from Eq. (11.14), and the last step follows from $\varepsilon_0 = \varepsilon_1 = \varepsilon/4$.

The running time of Algorithm 58 follows from Claim 11.52 and the success probability follows from setting $\rho_0 = 0.001$. The furthermore part follows from Claim 11.53.

The lemma is then proved. □

Claim 11.52 (Running time of Procedure DistillHD in Algorithm 58)**.** *Procedure* DistillHD *in Algorithm 58 runs in time*

$$O\left(\varepsilon^{-2}k^{O(d)}\log(1/\varepsilon)\right).$$

Proof. The first step of sampling S_0 takes $O(\varepsilon^{-2}k^{O(d)}\log(1/\varepsilon))$-time.

Then, by Theorem 11.35 with $|D| = O(\varepsilon^{-2}k^{O(d)}\log(1/\varepsilon))$, $d = k$, we have that the running time of RandBSS+ is

$$\begin{aligned} & O\left(k^2 \cdot \varepsilon^{-2}k^{O(d)}\log(1/\varepsilon) + \varepsilon^{-2}k^3\log\left(\varepsilon^{-2}k^{O(d)}\log(1/\varepsilon)\right) + \varepsilon^{-2}k^{\omega+1}\right) \\ = & O\left(\varepsilon^{-2}k^{O(d)}\log^3(k)\log(1/\varepsilon)\right). \end{aligned}$$

Hence, the total running time of Algorithm 58 is $O\left(\varepsilon^{-2}k^{O(d)}\log^3(k)\log(1/\varepsilon)\right)$.

□

Claim 11.53 (Preserve the energy of noise (high Dimension))**.** *Let* (S, w) *be the outputs of Algorithm 58. Then, for any function* $g(t)$*,*

$$\|g(t)\|_{S,w}^2 \lesssim \rho^{-2}\|g(t)\|_T^2,$$

holds with probability $1-\rho$*.*

Proof. Let P denote the Procedure ImportantSampling$(k, \varepsilon, \rho, F, T, \mathcal{B})$. Because P is a ε-well-balanced sampling procedure (Definition 11.10). By the Definition 11.10, we have that $t_i \sim D_i(t)$ and $w_i = \alpha_i \cdot \frac{D(t_i)}{D_i(t_i)}$ in every iteration $i \in [s]$, where $\sum_{i=1}^s \alpha_i \leq 5/4$, $D(t) = \mathrm{Uniform}(S_0)$.

As a result,

$$\begin{aligned}
\mathbb{E}_P[\|g(t)\|_{S,w}^2] =& \ \mathbb{E}_P[\sum_{i=1}^s w_i |g(t_i)|^2] \\
=& \ \sum_{i=1}^s \mathbb{E}_{t_i \sim D_i(t_i)}[w_i |g(t_i)|^2] \\
=& \ \sum_{i=1}^s \mathbb{E}_{t_i \sim D_i(t_i)}[\alpha_i \cdot \frac{D(t_i)}{D_i(t_i)} |g(t_i)|^2] \\
=& \ \sum_{i=1}^s \mathbb{E}_{t_i \sim D(t_i)}[\alpha_i |g(t_i)|^2] \\
\leq& \ \sup_P \{\sum_{i=1}^s \alpha_i\} \, \mathbb{E}_{t \sim D(t)}[|g(t)|^2] \\
=& \ \sup_P \{\sum_{i=1}^s \alpha_i\} \|g(t)\|_{S_0}^2 \\
\leq& \ 2\|g(t)\|_{S_0}^2.
\end{aligned}$$

where the first step follows from the definition of the norm, the third step follows from $w_i = \alpha_i \cdot \frac{D(t_i)}{D_i(t_i)}$, the forth step follows from $\mathbb{E}_{t \sim D_0(t)} \frac{D_1(t)}{D_0(t)} f(t) = \mathbb{E}_{t \sim D_1(t)} f(t)$, the sixth step follows from $D(t) = \text{Uniform}(S_0)$ and the definition of the norm, the last step follows from $\sum_{i=1}^s \alpha_i \leq 5/4$.

Moreover,

$$\begin{aligned}
\mathbb{E}_{S_0}[\|g(t)\|_{S_0}^2] =& \ \mathbb{E}_{t \sim \text{Uniform}([0,T])} |g(t)|^2 \\
=& \ \|g(t)\|_T^2
\end{aligned}$$

So, by Markov's inequality,

$$\Pr[\|g(t)\|_{S,w}^2 \leq \|g(t)\|_{S_0}^2/\varepsilon_0] \geq 1 - \varepsilon_0/2,$$

and

$$\Pr[\|g(t)\|_{S_0}^2 \leq \|g(t)\|_T^2/\varepsilon_1] \geq 1 - \varepsilon_1.$$

Then, with probability at least $(1-\varepsilon_0/2)(1-\varepsilon_1)$ holds,

$$\|g(t)\|_{S,w}^2 \leq \|g(t)\|_{S_0}^2/\varepsilon_0 \leq \|g(t)\|_T^2/(\varepsilon_0\varepsilon_1).$$

Set $\varepsilon_0 = \rho/10, \varepsilon_1 = \rho/10$, we have that,

$$\|g(t)\|_{S,w}^2 \lesssim \|g(t)\|_T^2/\rho^2,$$

holds with probability $1-\rho$.

□

11.8.3 Sketch distillation for discrete signals

The goal of this section is to prove Lemma 11.54, which can reduce the sketch size of Lemma 11.31 for discrete signals in any dimension.

Lemma 11.54 (Distillation for discrete signal)**.** *For any $d \geq 1$, let $n = p^d$ for some positive integer p. Let $x^* \in \mathbb{C}^{[p]^d}$, such that $\mathrm{supp}(\widehat{x^*}) \subset [p]^d$ and $|\mathrm{supp}(\widehat{x^*})| = k$. For any accuracy parameter $\varepsilon \in (0, 0.1)$, there is an algorithm (Algorithm 59) that runs in $O(\varepsilon^{-2}k^{\omega+1})$-time and outputs a set $S \subset [n]$ of size $s = O(k/\varepsilon^2)$ and a weight vector $w \in \mathbb{R}_{\geq 0}^s$ such that,*

$$(1-\varepsilon)\|x^*\|_2 \leq n\|x^*\|_{S,w} \leq (1+\varepsilon)\|x^*\|_2$$

holds with probability 0.99*.*

Proof. For the convenient, in the proof, we use x to denote the x^*.

First, we randomly pick up a set $S_0 = \{t_1, \cdots, t_{s_0}\}$ of $s_0 = O(\varepsilon^{-2}k\log(k/\rho))$ i.i.d. samples from Uniform$([n])$, where ε_0, ρ_0 are parameters to be chosen later.

By Lemma 11.31, with probability $1-\rho_0$,

$$(1-\varepsilon_0)\|x\|_2^2 \leq n\|x\|_{S_0}^2 \leq (1+\varepsilon_0)\|x\|_2^2. \tag{11.15}$$

Then, we will select $s = O(k/\varepsilon_1^2)$ elements from S_0 and output the corresponding weights $w_1, w_2, \cdots, w_s$ by applying Procedure RandBSS+ with the following parameter: replacing d by k, ε by ε_1^2, and D by Uniform(S_0).

By Theorem 11.35 and Definition 11.10, we obtain that with probability 0.995,

$$(1-\varepsilon_1)\|x\|_{S,w}^2 \leq \|x\|_{S_0}^2 \leq (1+\varepsilon_1)\|x\|_{S,w}^2.$$

Combining with Eq. (11.15), we conclude that

$$\begin{aligned}\|x\|_{S,w}^2 &\in [1-\varepsilon_1, 1+\varepsilon_1] \cdot \|x\|_{S_0}^2 \\ &\in [(1-\varepsilon_0)(1-\varepsilon_1), (1+\varepsilon_0)(1+\varepsilon_1)] \cdot \|x\|_2^2/n \\ &\in [1-\varepsilon, 1+\varepsilon] \cdot \|x\|_2^2/n,\end{aligned}$$

where the second step follows from Eq. (11.15), and the last step follows by taking $\varepsilon_0 = \varepsilon_1 = \varepsilon/4$.

By taking $\rho = 0.001$, we get that the overall success probability is at least 0.99.

Regarding the running time, if $d = 1$, we run Procedure DistillDisc in Algorithm 59, whose runtime follows from Claim 11.55. And if $d > 1$, we run Procedure DistillDiscHD in Algorithm 59, whose runtime follows from Claim 11.56.

The lemma is then proved. □

Claim 11.55 (Running time of Procedure DistillDisc in Algorithm 59)**.** *Procedure* DistillDisc *in Algorithm 59 runs in*

$$O\left(\varepsilon^{-2} k^{\omega+1}\right)$$

time.

Proof. The first step of sampling S_0 takes $O(\varepsilon^{-2} k \log(k))$-time.

Then, by Theorem 11.35 with $|D| = O(\varepsilon^{-2}k\log(k))$, $d = k$, we have that the running time of RandBSS+ is

$$\begin{aligned} & O\left(k^2 \cdot \varepsilon^{-2}k\log(k) + \varepsilon^{-2}k^3\log\left(\varepsilon^{-2}k\log(k)\right) + \varepsilon^{-2}k^{\omega+1}\right) \\ = & O\left(\varepsilon^{-2}k^{\omega+1}\right). \end{aligned}$$

Hence, the total running time is $O\left(\varepsilon^{-2}k^{\omega+1}\right)$. $\square$

Claim 11.56 (Running time of Procedure DistillDiscHD in Algorithm 59)**.** *Procedure* DistillDiscHD *in Algorithm 59 runs in*

$$O\left(\varepsilon^{-2}k^{\omega+1} + \varepsilon^{-2}dk^{\omega-1}\log k\right)$$

time.

Proof. The first step of sampling S_0 takes $O(\varepsilon^{-2}k\log(k)d)$-time.

Then, we need to implement the function family

$$\mathcal{F} = \{f(t) = \sum_{j=1}^{k} v_j \exp(2\pi\mathbf{i}\langle f_j, t\rangle/p) | v_j \in \mathbb{C}\}.$$

Naively, for each $f \in \mathcal{F}$, it takes $O(d)$-time per evaluation. We observe that in the distribution sent to RandBSS+ is Uniform(S_0), which is discrete with support size $s_0 = |S_0|$. And in Procedure RandBSS+, we only need to find an orthonormal basis for $\mathcal{F}$ with respect to this distribution, which is equivalent to orthogonalize the columns of the matrix defined at Line 11. To compute the matrix $\mathcal{F}$, we need to multiply an s_0-by-d matrix with a d-by-k matrix. By fast matrix multiplication, by Fact 11.7, it takes

$$\mathcal{T}_{\mathrm{mat}}(k, d, s_0) = \begin{cases} O(\varepsilon^{-2}k^{\omega}\log k) & \text{if } d \le k, \\ O(\varepsilon^{-2}dk^{\omega-1}\log k) & \text{if } d > k. \end{cases}$$

For Procedure RandBSS+, by Theorem 11.35 with $|D| = O(\varepsilon^{-2}k\log(k))$, $d = k$, we have that the running time of RandBSS+ is

$$\begin{aligned} & O\left(k^2 \cdot \varepsilon^{-2}k\log(k) + \varepsilon^{-2}k^3\log\left(\varepsilon^{-2}k\log(k)\right) + \varepsilon^{-2}k^{\omega+1}\right) \\ = & O\left(\varepsilon^{-2}k^{\omega+1}\right). \end{aligned}$$

Hence, the total running time of the procedure is

$$O\left(\varepsilon^{-2}dk\log(k) + \varepsilon^{-2}k^{\omega+1} + \mathcal{T}_{\mathrm{mat}}(k,d,s_0)\right) = O\left(\varepsilon^{-2}k^{\omega+1} + \varepsilon^{-2}dk^{\omega-1}\log k\right).$$

□

11.9 One-Dimensional Signal Estimation

In this section, we apply the tools developed in previous sections to show two efficient reductions from Frequency Estimation to Signal Estimation for one-dimensional semi-continuous Fourier signals. The first reduction in Section 11.9.1 is optimal in sample complexity, which takes linear number of samples from the signal but only achieves constant accuracy. The section reduction in Section 11.9.2 takes nearly-linear number of samples but can achieve very high-accuracy (i.e., $(1+\varepsilon)$-estimation error).

11.9.1 Sample-optimal reduction

The main theorem of this section is Theorem 11.57. The optimal sample complexity is achieved via the sketch distillation in Lemma 11.44.

Theorem 11.57 (Sample-optimal algorithm for one-dimensional Signal Estimation)**.** *For $\eta \in \mathbb{R}$, let $\Lambda(\mathcal{B}) \subset \mathbb{R}$ denote the lattice $\Lambda(\mathcal{B}) = \{c\eta \mid c \in \mathbb{Z}\}$. Suppose that $f_1, f_2, \cdots, f_k \in \Lambda(\mathcal{B})$. Let $x^*(t) = \sum_{j=1}^k v_j \exp(2\pi \mathbf{i} f_j t)$, and let $g(t)$ denote the noise. Given observations of the form $x(t) = x^*(t) + g(t)$, $t \in [0, T]$. Let $\eta = \min_{i\neq j} |f_j - f_i|$.*

Given $D, \eta \in \mathbb{R}_+$. Suppose that there is an algorithm FreqEst *that*

- *takes $\mathcal{S}_{\mathsf{freq}}$ samples,*
- *runs in $\mathcal{T}_{\mathsf{freq}}$-time, and*
- *outputs a set $\mathcal{L}$ of frequencies such that with probability 0.99, the following condition holds:*

$$\forall i \in [k],\ \exists f_i' \in \mathcal{L} \text{ s.t. } |f_i - f_i'| \le \frac{D}{T}.$$

Then, there is an algorithm (Algorithm 60) such that

- *takes $O(\widetilde{k} + \mathcal{S}_{\mathsf{freq}})$ samples*
- *runs $O(\widetilde{k}^{\omega+1} + \mathcal{T}_{\mathsf{freq}})$ time,*
- *outputs $y(t) = \sum_{j=1}^{\widetilde{k}} v_j' \cdot \exp(2\pi \mathbf{i} f_j' t)$ with $\widetilde{k} = O(|\mathcal{L}|(1 + D/(T\eta)))$ such that with probability at least 0.9, we have*

$$\|y(t) - x(t)\|_T^2 \lesssim \|g(t)\|_T^2.$$

Proof. First, we recover the frequencies by utilizing the algorithm FreqEst. Let L be the set of frequencies output by the algorithm $\textsc{FreqEst}(x, k, D, T, F, \mathcal{B})$.

We define $\widetilde{L}$ as follows:

$$\widetilde{L} := \left\{ \widetilde{f} \in \Lambda(\mathcal{B}) \mid \exists f' \in L,\ |f' - \widetilde{f}| < D/T \right\}.$$

We use $\widetilde{k}$ to denote the size of set $\widetilde{L}$. And we use $\widetilde{f}_1, \widetilde{f}_2, \cdots, \widetilde{f}_{\widetilde{k}}$ to denote the frequencies in the set $\widetilde{L}$. It is easy to see that

$$\widetilde{k} \le |\mathcal{L}|(1 + D/(T\eta)).$$

Next, we focus on recovering magnitude $v' \in \mathbb{C}^{\widetilde{k}}$. First we run Procedure FastDistill1D in Algorithm 57 and obtain a set $S = \{t_1, t_2, \cdots, t_s\} \subset [0, T]$ of size $s = O(\widetilde{k})$ and a weight vector $w \in \mathbb{R}^s_{>0}$. Then, we sample the signal at $t_1, \ldots, t_s$ and let $x(t_1), \ldots, x(t_s)$ be the samples. Consider the following weighted linear regression problem:

$$\min_{v' \in \mathbb{C}^{\widetilde{k}}} \left\| \sqrt{w} \circ (Av' - b) \right\|_2, \tag{11.16}$$

where $\sqrt{w} := (\sqrt{w_1}, \ldots, \sqrt{w_s})$, and the coefficients matrix $A \in \mathbb{C}^{s \times \widetilde{k}}$ and the target vector $b \in \mathbb{C}^s$ are defined as follows:

$$A := \begin{bmatrix} \exp(2\pi\mathbf{i}\widetilde{f}_1 t_1) & \exp(2\pi\mathbf{i}\widetilde{f}_2 t_1) & \cdots & \exp(2\pi\mathbf{i}\widetilde{f}_{\widetilde{k}} t_1) \\ \exp(2\pi\mathbf{i}\widetilde{f}_1 t_2) & \exp(2\pi\mathbf{i}\widetilde{f}_2 t_2) & \cdots & \exp(2\pi\mathbf{i}\widetilde{f}_{\widetilde{k}} t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \exp(2\pi\mathbf{i}\widetilde{f}_1 t_s) & \exp(2\pi\mathbf{i}\widetilde{f}_2 t_s) & \cdots & \exp(2\pi\mathbf{i}\widetilde{f}_{\widetilde{k}} t_s) \end{bmatrix} \text{ and } b := \begin{bmatrix} x(t_1) \\ x(t_2) \\ \vdots \\ x(t_s) \end{bmatrix}$$

Then, we output a signal

$$y(t) = \sum_{j=1}^{\widetilde{k}} v'_j \cdot \exp(2\pi\mathbf{i}\widetilde{f}_j t),$$

where v' is an optimal solution of Eq. (11.16).

The running time follows from Lemma 11.58. And the estimation error guarantee $\|y(t) - x(t)\|_T \lesssim \|g(t)\|_T$ follows from Lemma 11.59.

The theorem is then proved. □

Lemma 11.58 (Running time of Algorithm 60)**.** *Algorithm 60 takes $O(\widetilde{k}^{\omega+1})$-time, giving the output of Procedure* FreqEst.

Proof. At Line 5, we run Procedure FastDistill1D, which takes $O(\widetilde{k}^{\omega+1})$-time by Lemma 11.44.

At Line 8, we solve the weighted linear regression, which takes

$$O(s\widetilde{k}^{\omega-1}) = O(\widetilde{k}^{\omega})$$

time by Fact 11.8.

Thus, the total running time is $O(\widetilde{k}^{\omega+1})$. □

Lemma 11.59 (Estimation error of Algorithm 60)**.** *Let $y(t)$ be the output signal of Algorithm 60. With high probability, we have*

$$\|y(t) - x(t)\|_T \lesssim \|g(t)\|_T.$$

Proof. We have

$$\begin{aligned}
\|y(t) - x(t)\|_T &\leq \|y(t) - x^*(t)\|_T + \|g(t)\|_T \\
&\leq (1+\varepsilon)\|y(t) - x^*(t)\|_{S,w} + \|g(t)\|_T \\
&\leq (1+\varepsilon)\|y(t) - x(t)\|_{S,w} + (1+\varepsilon)\|g(t)\|_{S,w} + \|g(t)\|_T \\
&\leq (1+\varepsilon)\|x^*(t) - x(t)\|_{S,w} + (1+\varepsilon)\|g(t)\|_{S,w} + \|g(t)\|_T \\
&\lesssim \|x^*(t) - x(t)\|_{S,w} + \|g(t)\|_T \\
&\lesssim \|x^*(t) - x(t)\|_T + \|g(t)\|_T \\
&\lesssim \|g(t)\|_T, \qquad (11.17)
\end{aligned}$$

where the first step follows from triangle inequality, the second step follows from Lemma 11.44 with 0.99 probability, the third step follows from triangle inequality, the forth step follows from $y(t)$ is the optimal solution of the linear system, the fifth step follows from Claim 11.46, the sixth step follows from Lemma 11.44, and the last step follows from the definition of $g(t)$. □

11.9.2 High-accuracy reduction

In this section, we prove Theorem 11.60, which achieves $(1+\varepsilon)$-estimation error by a sharper bound on the energy of noise in Lemma 11.47.

Theorem 11.60 (High-accuracy algorithm for one-dimensional Signal Estimation)**.** *For $\eta \in \mathbb{R}$, let $\Lambda(\mathcal{B}) \subset \mathbb{R}$ denote the lattice $\Lambda(\mathcal{B}) = \{c\eta \mid c \in \mathbb{Z}\}$. Suppose that $f_1, f_2, \cdots, f_k \in \Lambda(\mathcal{B})$. Let $x^*(t) = \sum_{j=1}^k v_j \exp(2\pi\mathbf{i} f_j t)$, and let $g(t)$ denote the noise. Given observations of the form $x(t) = x^*(t) + g(t)$, $t \in [0, T]$. Let $\eta = \min_{i \neq j} |f_j - f_i|$.*

Given $D, \eta \in \mathbb{R}_+$. Suppose that there is an algorithm FreqEst *that*

- *takes* $\mathcal{S}_{\mathsf{freq}}$ *samples,*
- *runs in* $\mathcal{T}_{\mathsf{freq}}$*-time, and*
- *outputs a set* $\mathcal{L}$ *of frequencies such that, for each* f_i*, there exists an* $f'_i \in \mathcal{L}$ *with* $|f_i - f'_i| \leq D/T$*, holds with probability* 0.99*.*

Then, there is an algorithm (Algorithm 61) such that

- *takes* $O(\varepsilon^{-1}\widetilde{k}\log(\widetilde{k}) + \mathcal{S})$ *samples,*
- *runs* $O(\varepsilon^{-1}\widetilde{k}^{\omega}\log(\widetilde{k}) + \mathcal{T})$ *time,*
- *outputs* $y(t) = \sum_{j=1}^{\widetilde{k}} v'_j \cdot \exp(2\pi \mathbf{i} f'_j t)$ *with* $\widetilde{k} = O(|\mathcal{L}|(1 + D/(T\eta)))$ *such that with probability at least* 0.9*, we have*

$$\|y(t) - x^*(t)\|_T^2 \leq (1+\varepsilon)\|g(t)\|_T^2.$$

Remark 11.8. For simplicity, we state the constant failure probability. It is straightforward to get failure probability ρ by blowing up a $\log(1/\rho)$ factor in both samples and running time.

Proof. Let L be the set of frequencies output by the Frequency Estimation algorithm FreqEst. We have the guarantee that with probability 0.99, for each true frequency f_i, there exists an $f'_i \in \mathcal{L}$ with $|f_i - f'_i| \leq D/T$. Conditioning on this event, we define a set $\widetilde{L}$ as follows:

$$\widetilde{L} := \{f \in \Lambda(\mathcal{B}) \mid \exists f' \in L,\ |f' - f| < D/T\}.$$

Since we assume that $\{f_1, \ldots, f_k\} \subset \Lambda(\mathcal{B})$, we have $\{f_1, \ldots, f_k\} \subset \widetilde{L}$. We use $\widetilde{k}$ to denote the size of set $\widetilde{L}$, and we denote the frequencies in $\widetilde{L}$ by $\widetilde{f}_1, \widetilde{f}_2, \cdots, \widetilde{f}_{\widetilde{k}}$.

Next, we need to recover magnitude $v' \in \mathbb{C}^{\widetilde{k}}$.

We first run Procedure WeightedSketch in Algorithm 57 and obtain a set $S = \{t_1, t_2, \cdots, t_s\} \subset [0, T]$ of size $s = O(\varepsilon^{-2}\widetilde{k}\log(\widetilde{k}))$ and a weight vector $w \in$

$\mathbb{R}^s_{>0}$. Then, we sample the signal at $t_1, \dots, t_s$ and let $x(t_1), \dots, x(t_s)$ be the samples. Consider the following weighted linear regression problem:

$$\min_{v' \in \mathbb{C}^{\widetilde{k}}} \left\| \sqrt{w} \circ (Av' - b) \right\|_2, \tag{11.18}$$

where $\sqrt{w} := (\sqrt{w_1}, \dots, \sqrt{w_s})$, and the coefficients matrix $A \in \mathbb{C}^{s \times \widetilde{k}}$ and the target vector $b \in \mathbb{C}^s$ are defined as follows:

$$A := \begin{bmatrix} \exp(2\pi \mathbf{i} \widetilde{f}_1 t_1) & \exp(2\pi \mathbf{i} \widetilde{f}_2 t_1) & \cdots & \exp(2\pi \mathbf{i} \widetilde{f}_{\widetilde{k}} t_1) \\ \exp(2\pi \mathbf{i} \widetilde{f}_1 t_2) & \exp(2\pi \mathbf{i} \widetilde{f}_2 t_2) & \cdots & \exp(2\pi \mathbf{i} \widetilde{f}_{\widetilde{k}} t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \exp(2\pi \mathbf{i} \widetilde{f}_1 t_s) & \exp(2\pi \mathbf{i} \widetilde{f}_2 t_s) & \cdots & \exp(2\pi \mathbf{i} \widetilde{f}_{\widetilde{k}} t_s) \end{bmatrix} \text{ and } b := \begin{bmatrix} x(t_1) \\ x(t_2) \\ \vdots \\ x(t_s) \end{bmatrix}$$

Note that if v' corresponds to the true coefficients v, then we have $\|\sqrt{w} \circ (Av' - b)\|_2 = \|\sqrt{w} \circ g(S)\|_2 = \|g\|_{S,w}$. Let v' be the exact solution of the weighted linear regression in Eq. (11.18), i.e.,

$$v' := \arg \min_{v' \in \mathbb{C}^{\widetilde{k}}} \left\| \sqrt{w} \circ (Av' - b) \right\|.$$

And we define the output signal to be:

$$y(t) := \sum_{j=1}^{\widetilde{k}} v'_j \cdot \exp(2\pi \mathbf{i} f'_j t).$$

The estimation error guarantee $\|y(t) - x^*(t)\|_T \leq (1 + \varepsilon)\|g(t)\|_T$ follows from Lemma 11.62. The running time follows from Lemma 11.61.

The theorem is then proved. □

Lemma 11.61 (Running time of Algorithm 61)**.** *Algorithm 61 takes $O(\varepsilon^{-1}\widetilde{k}^{\omega} \log(\widetilde{k}))$-time, giving the output of Procedure* FreqEst.

Proof. At Line 7, the regression solver takes

$$O(s\widetilde{k}^{\omega-1}) = O(\varepsilon^{-1}\widetilde{k} \log(\widetilde{k}) \cdot \widetilde{k}^{\omega-1}) = O(\varepsilon^{-1}\widetilde{k}^{\omega} \log(\widetilde{k}))$$

time. The remaining part of Algorithm 61 takes at most $O(s)$-time. □

Lemma 11.62 (Estimation error of Algorithm 61)**.** *Let $y(t)$ be the output signal of Algorithm 61. With high probability, we have*

$$\|y(t) - x^*(t)\|_T \le (1+\varepsilon)\|g(t)\|_T.$$

Proof. Let $\mathcal{F}$ be the family of signals with frequencies in $\widetilde{L}$:

$$\mathcal{F} = \Big\{ h(t) = \sum_{j=1}^{\widetilde{k}} v_j \cdot e^{2\pi \mathbf{i} \widetilde{f}_j t} \mid \forall v_j \in \mathbb{C}, j \in [\widetilde{k}] \Big\}.$$

Suppose the dimension of $\mathcal{F}$ is $m \le k$. Let $\{u_1, u_2, \cdots, u_m\}$ be an orthonormal basis of $\mathcal{F}$, i.e.,

$$\frac{1}{T}\int_{[0,T]} u_i(t)\overline{u_j(t)}\mathrm{d}t = \mathbf{1}_{i=j}, \quad \forall i,j \in [m],$$

On the other hand, since $u_i \in \mathcal{F}$, we can also expand these basis vectors in the Fourier basis. Let $V \in \mathbb{C}^{m\times \widetilde{k}}$ be an linear transformation[13] such that

$$u_i = \sum_{j=1}^{\widetilde{k}} V_{i,j} \cdot \exp(2\pi \mathbf{i} \widetilde{f}_j t) \quad \forall i \in [m].$$

Then, we have

$$\begin{bmatrix} \exp(2\pi\mathbf{i}\widetilde{f}_1 t) \\ \vdots \\ \exp(2\pi\mathbf{i}\widetilde{f}_{\widetilde{k}} t)\end{bmatrix} = V^+ \cdot \begin{bmatrix} u_1 \\ \vdots \\ u_m \end{bmatrix},$$

where $V^+ \in \mathbb{C}^{\widetilde{k}\times m}$ is the pseudoinverse of V; or equivalently, the i-th row of V^+ contains the coefficients of expanding $\exp(2\pi\mathbf{i}\widetilde{f}_i t)$ under $\{u_1, \ldots, u_m\}$. Define a linear operator $\alpha : \mathcal{F} \to \mathbb{C}^m$ such that for any $h(t) = \sum_{j=1}^{\widetilde{k}} v_j \exp(2\pi\mathbf{i} f_j t)$,

$$\alpha(h) := V^+ \cdot v,$$

which gives the coefficients of h under the basis $\{u_1, \cdots, u_{\widetilde{k}}\}$.

[13]When $m < \widetilde{k}$, V is not unique, and we take any one of such linear transformation.

Define an s-by-m matrix B as follows:

$$B := A \cdot V^\top = \begin{bmatrix} u_1(t_1) & u_2(t_1) & \cdots & u_m(t_1) \\ u_1(t_2) & u_2(t_2) & \cdots & u_m(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ u_1(t_s) & u_2(t_s) & \cdots & u_m(t_s) \end{bmatrix}.$$

$B = AV$. It is easy to see that $\mathrm{Im}(B) = \mathrm{Im}(A)$. Thus, solving Eq. (11.18) is equivalent to solving:

$$\min_{z \in \mathbb{C}^m} \|\sqrt{w} \circ (Bz - b)\|_2. \tag{11.19}$$

Since $y(t)$ is an solution of Eq. (11.18), we also know that $\alpha(y)$ is an solution of Eq. (11.19).

For convenience, we define some notations. Let $\sqrt{W} := \mathrm{diag}(\sqrt{w})$ and define

$$\begin{aligned} B_w &:= \sqrt{W} \cdot B, \\ X_w &:= \sqrt{W} \cdot \begin{bmatrix} x(t_1) & x(t_2) & \cdots & x(t_s) \end{bmatrix}^\top \\ X_w^* &:= \sqrt{W} \cdot \begin{bmatrix} x^*(t_1) & x^*(t_2) & \cdots & x^*(t_s) \end{bmatrix}^\top \end{aligned}$$

By Fact 11.8, we know that the solution of the weighted linear regression Eq. (11.19) has the following closed form:

$$\alpha(y) = (B^* W B)^{-1} B^* W b = (B_w^* B_w)^{-1} B_w^* X_w. \tag{11.20}$$

Then, consider the noise in the signal. Since g is an arbitrary noise, let $g^\parallel$ be the projection of $g(x)$ to $\mathcal{F}$ and $g^\perp = g - g^\parallel$ be the orthogonal part to $\mathcal{F}$ such that

$$g^\parallel(t) \in \mathcal{F}, \text{ and } \int_{[0,T]} g^\parallel(t) \overline{g^\perp(t)} \mathrm{d}t = 0.$$

Similarly, we also define

$$\begin{aligned} g_w &:= \sqrt{W} \cdot \begin{bmatrix} g(t_1) & g(t_2) & \cdots & g(t_s) \end{bmatrix}^\top \\ g_w^\parallel &:= \sqrt{W} \cdot \begin{bmatrix} g^\parallel(t_1) & g^\parallel(t_2) & \cdots, g^\parallel(t_s) \end{bmatrix}^\top, \\ g_w^\perp &:= \sqrt{W} \cdot \begin{bmatrix} g^\perp(t_1) & g^\perp(t_2) & \cdots, g^\perp(t_s) \end{bmatrix}^\top. \end{aligned}$$

By Claim 11.63, the error can be decomposed into two terms:

$$\|y(t) - x^*(t)\|_T \leq \left\|(B_w^* B_w)^{-1} B_w^* \cdot g_w^{\perp}\right\|_2 + \left\|(B_w^* B_w)^{-1} B_w^* \cdot g_w^{\|}\right\|_2 .$$

By Claim 11.65, we have

$$\left\|(B_w^* B_w)^{-1} B_w^* \cdot g_w^{\perp}\right\|_2^2 \lesssim \varepsilon \left\|g^{\perp}(t)\right\|_T^2 .$$

And by Claim 11.68, we have

$$\left\|(B_w^* B_w)^{-1} B_w^* \cdot g_w^{\|}\right\|_2^2 = \left\|g^{\|}\right\|_T^2 .$$

Combining them together (and re-scaling ε be an constant factor), we have that

$$\|y(t) - x^*(t)\|_T \leq \|g^{\|}\|_T + \sqrt{\varepsilon}\|g^{\perp}\|_T.$$

Since $\|g^{\|}\|_T^2 + \|g^{\perp}\|_T^2 = \|g\|_T^2$, by Cauchy–Schwarz inequality, we have that

$$(\|g^{\|}\|_T + \sqrt{\varepsilon}\|g^{\perp}\|_T)^2 \leq (\|g^{\|}\|_T^2 + \|g^{\perp}\|_T^2) \cdot (1+\varepsilon) = (1+\varepsilon) \cdot \|g\|_T^2.$$

That is,

$$\|y(t) - x^*(t)\|_T^2 \leq (1+\varepsilon)\|g(t)\|_T^2.$$

□

Claim 11.63 (Error decomposition)**.**

$$\|y(t) - x^*(t)\|_T \leq \left\|(B_w^* B_w)^{-1} B_w^* \cdot g_w^{\perp}\right\|_2 + \left\|(B_w^* B_w)^{-1} B_w^* \cdot g_w^{\|}\right\|_2 .$$

Proof. Since $y, x^* \in \mathcal{F}$ and $\{u_1, \ldots, u_{\widetilde{k}}\}$ is an orthonormal basis, we have $\|y - x^*\|_T = \|\alpha(y) - \alpha(x^*)\|_2$. Furthermore, by Eq. (11.20), we have $\alpha(y) = (B_w^* B_w)^{-1} B_w^* \cdot X_w$. And by Fact 11.64, since $x^* \in \mathcal{F}$, we have $\alpha(x^*) = (B_w^* B_w)^{-1} B_w^* \cdot X_w^*$.

Thus, we have

$$\begin{aligned}\|\alpha(y)-\alpha(x^*)\|_2 &= \|(B_w^*B_w)^{-1}B_w^*\cdot(X_w-X_w^*)\|_2\\ &= \|(B_w^*B_w)^{-1}B_w^*\cdot g_w\|_2\\ &= \|(B_w^*B_w)^{-1}B_w^*\cdot(g_w^\perp+g_w^\parallel)\|_2\\ &\le \|(B_w^*B_w)^{-1}B_w^*\cdot g_w^\perp\|_2+\|(B_w^*B_w)^{-1}B_w^*\cdot g_w^\parallel\|_2\end{aligned}$$

where the second step follows from the definition of g_w, the forth step follows from $g_w = g^\parallel + g^\perp$, and the last step follows from triangle inequality.

Hence, we get that $\|y(t)-x^*(t)\|_T \le \|(B_w^*B_w)^{-1}B_w^*\cdot g_w^\perp\|_2+\|(B_w^*B_w)^{-1}B_w^*\cdot g_w^\parallel\|_2$. □

Fact 11.64. *For any $h\in\mathcal{F}$,*

$$\alpha(h) = (B_w^*B_w)^{-1}B_w^*\cdot h_w,$$

where $h_w = \sqrt{W}\begin{bmatrix}h(t_1) & \cdots & h(t_s)\end{bmatrix}^\top$.

Proof. Suppose $h(t)=\sum_{j=1}^{\widetilde{k}} v_j\exp(2\pi\mathbf{i}\widetilde{f}_j t)$. We have

$$\begin{aligned}B_w\alpha(h) &= \sqrt{W}B\cdot\alpha(h)\\ &= \sqrt{W}B\cdot(V^+v)\\ &= h_w,\end{aligned}$$

where the second step follows from V^+ is a change of coordinates.

Hence, by the Moore-Penrose inverse, we have

$$\alpha(h) = B_w^\dagger h_w = (B_w^*B_w)^{-1}B_w^* h_w.$$

□

Claim 11.65 (Bound the first term)**.** *The following holds with high probability:*

$$\left\|(B_w^*B_w)^{-1}B_w^*\cdot g_w^\perp\right\|_2^2 \lesssim \varepsilon\left\|g^\perp(t)\right\|_T^2.$$

Proof. By Lemma 11.28, with high probability, we have

$$(1-\varepsilon)\|x\|_T \le \|x\|_{S,w} \le (1+\varepsilon)\|x\|_T,$$

where (S,w) is the output of Procedure WEIGHTEDSKETCH. Conditioned on this event, by Lemma 11.15,

$$\lambda(B_w^* B_w) \in [1-\varepsilon, 1+\varepsilon],$$

since B_w is the same as the matrix A in the lemma.

Hence,

$$\begin{aligned}\|(B_w^* B_w)^{-1} B_w^* \cdot g_w^{\perp}\|_2^2 &\le \lambda_{\max}((B_w^* B_w)^{-1})^2 \cdot \|B_w^* \cdot g_w^{\perp}\|_2^2 \\ &\le (1-\varepsilon)^{-2} \|B_w^* \cdot g_w^{\perp}\|_2^2 \\ &\lesssim \varepsilon \|g^{\perp}(t)\|_T^2\end{aligned}$$

where the second step follows from $\lambda_{\max}((B_w^* B_w)^{-1}) \le (1-\varepsilon)^{-1}$, and the third step follows from Lemma 11.47 and Corollary 11.67. $\square$

Lemma 11.66 (Lemma 6.2 of [CP19a])**.** *There exists a universal constant C_1 such that given any distribution D' with the same support of D and any $\varepsilon > 0$, the random sampling procedure with $m = C_1(K_{D'} \log d + \varepsilon^{-1} K_{D'})$ i.i.d. random samples from D' and coefficients $\alpha_1 = \cdots = \alpha_m = 1/m$ is an ε-*well-balanced sampling procedure*.*

Corollary 11.67. *Procedure* WEIGHTEDSKETCH *in Algorithm 57 is a ε-WBSP (Definition 11.10).*

Claim 11.68 (Bound the second term)**.**

$$\left\|(B_w^* B_w)^{-1} B_w^* \cdot g_w^{\|}\right\|_2^2 = \left\|g^{\|}\right\|_T^2.$$

Proof.

$$\|(B_w^* B_w)^{-1} B_w^* \cdot g_w^{\|}\|_2^2 = \|\alpha(g^{\|})\|_2^2 = \|g^{\|}\|_T^2,$$

where the first step follows from Fact 11.64 and $g^{\|} \in \mathcal{F}$, the second step follows from the definition of α. $\square$

11.10 High-dimensional Signal Estimation

In this section, we show a sample-optimal reduction from Frequency Estimation to Signal Estimation for high-dimensional signals in Section 11.10.1, which generalize Theorem 11.57. The key difference is that in high dimensions, we need to upper-bound the number of lattice points within a d-dimensional sphere, which turns out to be related to the output signal's Fourier sparsity, and the results are given in Section 11.10.2.

11.10.1 Sample-optimal reduction

Theorem 11.69 (Sample-optimal algorithm for high dimension Signal Estimation)**.** *Given a basis $\mathcal{B}$ of m known vectors $b_1, b_2, \cdots b_m \in \mathbb{R}^d$, let $\Lambda(\mathcal{B}) \subset \mathbb{R}^d$ denote the lattice*

$$\Lambda(\mathcal{B}) = \left\{ z \in \mathbb{R}^d : z = \sum_{i=1}^{m} c_i b_i, c_i \in \mathbb{Z}, \forall i \in [m] \right\}$$

Suppose that $f_1, f_2, \cdots, f_k \in \Lambda(\mathcal{B})$. Let $x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} \langle f_j, t \rangle}$ and let $g(t)$ denote the noise. Given observations of the form $x(t) = x^*(t) + g(t)$, $t \in [0, T]^d$. Let $\eta = \min_{i \neq j} \|f_j - f_i\|_\infty$.

Given $D, \eta \in \mathbb{R}_+$. Suppose that there is an algorithm FreqEst *that*

- *takes $\mathcal{S}_{\mathsf{freq}}$ samples,*
- *runs in $\mathcal{T}_{\mathsf{freq}}$-time,*
- *outputs a set $\mathcal{L}$ of frequencies such that with probability 0.99, the following condition holds:*

$$\forall i \in [k], \ \exists f_i' \in \mathcal{L} \ s.t. \ \|f_i - f_i'\|_2 \leq \frac{D}{T}.$$

Then, there is an algorithm that

- *takes* $O(\widetilde{k} + \mathcal{S}_{\mathsf{freq}})$ *samples*
- *runs in* $O(\widetilde{k}^{O(d)} + \mathcal{T}_{\mathsf{freq}})$ *time,*
- *output* $y(t) = \sum_{j=1}^{\widetilde{k}} v'_j \cdot \exp(2\pi \mathbf{i} \langle f'_j, t\rangle)$ *with* $\widetilde{k} \leq |L| \cdot (D/T + \sqrt{m}\|\mathcal{B}\|)^m \cdot \frac{\pi^{m/2}}{(m/2)!} \cdot \frac{1}{|\det(\mathcal{B})|}$ *such that with probability 0.9, we have*

$$\int_{[0,T]^d} |y(t) - x(t)|^2 \mathrm{d}t \lesssim \int_{[0,T]^d} |g(t)|^2 \mathrm{d}t.$$

Proof. The algorithm is almost the same as Algorithm 60. First, we recover the frequencies by calling Procedure FreqEst$(x, k, d, T, F, \mathcal{B})$. Let L be the set of frequencies output by the algorithm.

We define $\widetilde{L}$ as follows:

$$\widetilde{L} := \{f \in \Lambda(\mathcal{B}) \mid \exists f' \in L,\ \|f' - f\|_2 < D/T\}.$$

We use $\widetilde{k}$ to denote the size of set $\widetilde{L}$. We use $f'_1, f'_2, \cdots, f'_{\widetilde{k}}$ to denote the frequencies in the set $\widetilde{L}$. By applying Lemma 11.70, we have that

$$\widetilde{k} \leq |L| \cdot (D/T + \sqrt{m}\|\mathcal{B}\|)^m \cdot \frac{\pi^{m/2}}{(m/2)!} \cdot \frac{1}{|\det(\mathcal{B})|}.$$

Next, we focus on recovering magnitude $v' \in \mathbb{C}^{\widetilde{k}}$. We run Procedure DistillHD in Algorithm 58 and obtain a set $S = \{t_1, t_2, \cdots, t_s\}$ of $s = O(\widetilde{k})$ samples in the duration $[0, T]^d$, and a weight vector $w \in \mathbb{R}^s$.

Then, we consider the following weighted linear regression problem

$$\min_{v' \in \mathbb{C}^{\widetilde{k}}} \|\sqrt{w} \circ (Av' - b)\|_2,$$

where $A \in \mathbb{C}^{s \times \widetilde{k}}$ and $b \in \mathbb{C}^s$ are defined as follows:

$$A := \begin{bmatrix} \exp(2\pi\mathbf{i}\langle \widetilde{f}_1, t_1\rangle) & \cdots & \exp(2\pi\mathbf{i}\langle \widetilde{f}_{\widetilde{k}}, t_1\rangle) \\ \vdots & \ddots & \vdots \\ \exp(2\pi\mathbf{i}\langle \widetilde{f}_1, t_s\rangle) & \cdots & \exp(2\pi\mathbf{i}\langle \widetilde{f}_{\widetilde{k}}, t_s\rangle) \end{bmatrix} \text{ and } b := \begin{bmatrix} x(t_1) \\ \vdots \\ x(t_s) \end{bmatrix}$$

Let v' be an optimal solution of the regression and we output the signal

$$y(t) := \sum_{j=1}^{\widetilde{k}} v'_j \cdot \exp(2\pi\mathbf{i}\langle f'_j, t\rangle).$$

Finally, we prove that $\|y(t)-x(t)\|_T \lesssim \|g(t)\|_T$, holds with a large constant probability.

$$\begin{aligned}
\|y(t)-x(t)\|_T \leq{}& \|y(t)-x^*(t)\|_T + \|g(t)\|_T \\
\leq{}& (1+\varepsilon)\|y(t)-x^*(t)\|_{S,w} + \|g(t)\|_T \\
\leq{}& (1+\varepsilon)\|y(t)-x(t)\|_{S,w} + (1+\varepsilon)\|g(t)\|_{S,w} + \|g(t)\|_T \\
\leq{}& (1+\varepsilon)\|x^*(t)-x(t)\|_{S,w} + (1+\varepsilon)\|g(t)\|_{S,w} + \|g(t)\|_T \\
\lesssim{}& \|x^*(t)-x(t)\|_{S,w} + \|g(t)\|_T \\
\lesssim{}& \|x^*(t)-x(t)\|_T + \|g(t)\|_T \\
\lesssim{}& \|g(t)\|_T, \qquad (11.21)
\end{aligned}$$

where the first step follows from triangle inequality, the second step follows from Lemma 11.51 with 0.99 probability, the third step follows from triangle inequality, the forth step follows from $y(t)$ is the optimal solution of the linear system, the fifth step follows from Claim 11.53, the sixth step follows from Lemma 11.51, and the last step follows from the definition of $g(t)$.

The running time of the reduction follows from Lemma 11.51. □

11.10.2 Bounding the sparsity

In this section, we show that the Fourier sparsity of the output signal can be bounded by the number of lattice points within a sphere. The intuition is that for each frequency f' outputted by Procedure FreqEst, there could be $|B_d(f', D/T) \cap \Lambda(\mathcal{B})|$ many candidates of true frequencies, where $B_d(x, r)$ denotes the d-dimensional sphere centered at x with radius r. In Lemma 11.70, we upper-bound the sparsity for the case when D/T is larger than $\lambda_1(\Lambda(\mathcal{B}))$, the shortest vector length of the lattice.

When D/T is small, we show in Lemma 11.71 that Procedure FreqEst finds all true frequencies.

Lemma 11.70 (Bounding sparsity for large D/T)**.** *Given a basis $\mathcal{B}$ of m known vectors $b_1, b_2, \cdots b_m \in \mathbb{R}^d$, let $\Lambda(\mathcal{B}) \subset \mathbb{R}^d$ denote the lattice*

$$\Lambda(\mathcal{B}) = \Big\{ z \in \mathbb{R}^d : z = \sum_{i=1}^{m} c_i b_i, c_i \in \mathbb{Z}, \forall i \in [m] \Big\},$$

and let

$$\widetilde{k} := |\{ f \in \Lambda(\mathcal{B}) \mid \exists f' \in L, \ \|f' - f\|_2 < D/T \}|$$

be the output sparsity. Then, we have

- *(Spectral bound, which is better when $D/T < O(\|\mathcal{B}\|)$)*
$$\widetilde{k} \leq |L| \cdot (1 + 2D/(T\sigma_{\min}(\mathcal{B})))^m.$$
- *(Volume bound, which is better when $D/T > O(\|\mathcal{B}\|)$)*
$$\widetilde{k} \leq |L| \cdot (D/T + \sqrt{m}\|\mathcal{B}\|)^m \cdot \frac{\pi^{m/2}}{(m/2)!} \cdot \frac{1}{|\det(\mathcal{B})|}.$$

Proof. **Spectral bound:** Let $c = \begin{bmatrix} c_1 & c_2 & \cdots & c_m \end{bmatrix}^\top \in \mathbb{Z}^m$. Then $z = \mathcal{B}c \in \Lambda(\mathcal{B})$, and

$$\|z\|_2 = \|\mathcal{B}c\|_2 \geq \sigma_{\min}(\mathcal{B}) \cdot \|c\|_2$$

Then we have that,

$$\begin{aligned}
\widetilde{k} &= |\{ f \in \Lambda(\mathcal{B}) \mid \exists f' \in L, \ \|f' - f\|_2 < D/T \}| \\
&\leq |L| \cdot |\{ z \in \Lambda(\mathcal{B}) \mid \|z\|_2 < D/T \}| \\
&\leq |L| \cdot |\{ c \in \mathbb{Z}^m \mid \|c\|_2 \leq D/(T\sigma_{\min}(\mathcal{B})) \}| \\
&\leq |L| \cdot |\{ c \in \mathbb{Z}^m \mid \|c\|_\infty \leq D/(T\sigma_{\min}(\mathcal{B})) \}| \\
&\leq |L| \cdot (1 + 2D/(T\sigma_{\min}\mathcal{B}))^m.
\end{aligned}$$

where the first step follows from $f' - f \in \Lambda(\mathcal{B})$, the second step follows from if $\|c\|_2 \geq D/(T\sigma_{\min})$, then $\|z\|_2 \geq D/T$, the third step follows from $\|c\|_\infty \leq \|c\|_2$, and the last step follows from c is a bounded integer vector.

Volume bound: Using Lemma 11.11, we have

$$\widetilde{k} \leq |L| \cdot (1 + \frac{\sqrt{m}\|\mathcal{B}\|}{D/T})^m \cdot \frac{\mathrm{vol}(\mathcal{B}_m(0, D/T))}{\mathrm{vol}(\mathcal{P}(\mathcal{B}))}. \tag{11.22}$$

We can upper bound volume of a ball as follows:

$$\mathrm{vol}(\mathcal{B}_m(0, D/T)) \leq \frac{\pi^{m/2}}{(m/2)!} \cdot (D/T)^m. \tag{11.23}$$

Combining the above two equations, we have

$$\begin{aligned} \mathrm{LHS} &\leq |L| \cdot (1 + \frac{\sqrt{m}\|\mathcal{B}\|}{D/T})^m \cdot \frac{\pi^{m/2}}{(m/2)!} \cdot (D/T)^m \cdot \frac{1}{\mathrm{vol}(\mathcal{P}(\mathcal{B}))} \\ &\leq |L| \cdot (D/T + \sqrt{m}\|\mathcal{B}\|)^m \cdot \frac{\pi^{m/2}}{(m/2)!} \cdot \frac{1}{|\det(\mathcal{B})|}, \end{aligned}$$

where the first step follows from Eq. (11.22) and Eq. (11.23).

□

Lemma 11.71 (Bounding sparsity for tiny D/T)**.** *Given a basis $\mathcal{B}$ of m known vectors $b_1, b_2, \cdots b_m \in \mathbb{R}^d$, let $\Lambda(\mathcal{B}) \subset \mathbb{R}^d$ denote the lattice*

$$\Lambda(\mathcal{B}) = \Big\{ z \in \mathbb{R}^d : z = \sum_{i=1}^m c_i b_i, c_i \in \mathbb{Z}, \forall i \in [m] \Big\}$$

and let

$$\widetilde{k} := |\{ f \in \Lambda(\mathcal{B}) \mid \exists f' \in L, \ \|f' - f\|_2 < D/T \}|$$

be the output sparsity. If $D/T \leq \lambda_1(\Lambda(\mathcal{B}))$[14], then we have

$$\widetilde{k} \leq |L|.$$

Proof. Since the radius D/T is at most the shortest vector length of the lattice $\Lambda(\mathcal{B})$, for each $f' \in L$, the sphere $B_d(f', D/T)$ contains at most one lattice point. □

[14]When m is small, we can solve the shortest vector problem (SVP) exactly to decide the sparsity. Otherwise, we can check $D/T < \min_i \|b_i^*\|_2$ by Theorem 11.12.

11.10.3 High-accuracy reduction

Theorem 11.72 (High-dimensional Signal Estimation algorithm)**.** *Given a basis $\mathcal{B}$ of m known vectors $b_1, b_2, \cdots b_m \in \mathbb{R}^d$, let $\Lambda(\mathcal{B}) \subset \mathbb{R}^d$ denote the lattice*

$$\Lambda(\mathcal{B}) = \Big\{ z \in \mathbb{R}^d : z = \sum_{i=1}^m c_i b_i, c_i \in \mathbb{Z}, \forall i \in [m] \Big\}$$

Suppose that $f_1, f_2, \cdots, f_k \in \Lambda(\mathcal{B})$. Let $x^*(t) = \sum_{j=1}^k v_j e^{2\pi \mathbf{i} \langle f_j, t \rangle}$ and let $g(t)$ denote the noise. Given observations of the form $x(t) = x^*(t) + g(t)$, $t \in [0,T]^d$. Let $\eta = \min_{i \neq j} \|f_j - f_i\|_\infty$.

Given $D, \eta \in \mathbb{R}_+$. Suppose that there is an algorithm FreqEst *that*

- *takes $\mathcal{S}_{\mathsf{freq}}$ samples,*
- *runs in $\mathcal{T}_{\mathsf{freq}}$-time,*
- *outputs a set $\mathcal{L}$ of frequencies such that with probability 0.99, the following condition holds:*

$$\forall i \in [k],\ \exists f_i' \in \mathcal{L} \text{ s.t. } \|f_i - f_i'\|_2 \leq \frac{D}{T}.$$

Then, there is an algorithm that

- *takes $O(\varepsilon^{-1} \widetilde{k}^{O(d)} + \mathcal{S}_{\mathsf{freq}})$ samples*
- *runs in $O(\varepsilon^{-1} \widetilde{k}^{O(d)} + \mathcal{T}_{\mathsf{freq}})$ time,*
- *output $y(t) = \sum_{j=1}^{\widetilde{k}} v_j' \cdot \exp(2\pi \mathbf{i} \langle f_j', t \rangle)$ with $\widetilde{k} \leq |L| \cdot (D/T + \sqrt{m}\|\mathcal{B}\|)^m \cdot \frac{\pi^{m/2}}{(m/2)!} \cdot \frac{1}{|\det(\mathcal{B})|}$ such that with probability 0.9, we have*

$$\int_{[0,T]^d} |y(t) - x(t)|^2 \mathrm{d}t \leq (1 + O(\varepsilon)) \int_{[0,T]^d} |g(t)|^2 \mathrm{d}t.$$

Remark 11.9. The difference between Theorem 11.72 and Theorem 11.69 is one is achieving $(1+\varepsilon)$ error and the other is achieving $O(1)$ error.

Proof. We can prove this theorem by using Theorem 11.5.2. The proof is similar as Theorem 11.60. □

11.11 Discrete Fourier Set Query in One Dimension

In this section, we study the Fourier set-query problem, where we only care about the Fourier coefficients of a discrete signal in a given set of frequencies. We apply our framework and achieve optimal sample complexity and high-accuracy. In Section 11.11.1, we show our main result on discrete Fourier set query. A key step to prove this result is a WBSP Composition Lemma in Section 11.11.2, which might be of independent interest.

11.11.1 Sample-optimal set query algorithm

In this section, we show our discrete Fourier set query result in the following theorem, which works for discrete signals in any dimension.

Theorem 11.73 (Discrete Fourier Set Query). *For any $d \geq 1$, let $n = p^d$ where both p and d are positive integers. Given a vector $x \in \mathbb{C}^{[p]^d}$, for $1 \leq k \leq n$, any $S \subseteq [n]$, $|S| = k$, there exists an algorithm (Algorithm 62) that takes $O(\varepsilon^{-1}k)$ samples, runs in $O(\varepsilon^{-1}k^{\omega+1} + \varepsilon^{-1}dk^{\omega-1}\log k)$ time, and outputs a vector $x' \in \mathbb{C}^{[p]^d}$ such that*

$$\|(\widehat{x}' - \widehat{x})_S\|_2^2 \leq \varepsilon \|\widehat{x}_{\overline{S}}\|_2^2$$

holds with probability at least 0.9.

In particular, for $d = 1$, the runtime of Algorithm 62 is $O(\varepsilon^{-1}k^{\omega+1})$.

Proof. Let $\{f_1, f_2, \cdots, f_k\} \subseteq [p]^d$ denote S. If $d = 1$, we run Procedure DistillDisc in Algorithm 59, and if $d > 1$, we run Procedure DistillDiscHD in Algorithm 59. Then, we obtain a set $L = \{t_1, t_2, \cdots, t_s\} \subseteq [p]^d$ of $s = O(\varepsilon^{-1}k)$ samples together with a weight vector $w \in \mathbb{R}^s$.

Then, we consider the following weighted linear regression problem:

$$\min_{v' \in \mathbb{C}^k} \|\sqrt{w} \circ (Av' - b)\|_2. \tag{11.24}$$

where $A \in \mathbb{C}^{s \times k}$ and $b \in \mathbb{C}^s$ are defined as follows:

$$A := \begin{bmatrix} \exp(2\pi \mathbf{i} f_1 t_1/n) & \cdots & \exp(2\pi \mathbf{i} f_k t_1/n) \\ \vdots & \ddots & \vdots \\ \exp(2\pi \mathbf{i} f_1 t_s/n) & \cdots & \exp(2\pi \mathbf{i} f_k t_s/n) \end{bmatrix} \text{ and } b := \begin{bmatrix} x(t_1) \\ \vdots \\ x(t_s) \end{bmatrix}$$

Let v' be an optimal solution of Eq. (11.24). And we output a vector

$$\widehat{x}'_{f_i} = v'_i \quad \forall i \in [k].$$

The running time follows from Lemma 11.74, and the estimation error guarantee follows from Lemma 11.75.

The proof of the theorem is then completed. $\square$

Lemma 11.74 (Running time of Algorithm 62)**.** *The time complexity of Algorithm 62 is as follows:*

- *Procedure* SetQuery *runs in* $O(\varepsilon^{-1} k^{\omega+1})$*-time.*
- *Procedrue* SetQueryHD *runs in* $O(\varepsilon^{-1} k^{\omega+1} + \varepsilon^{-1} d k^{\omega-1} \log k)$*-time.*

Proof. We first show the time complexity of Procedure DistillDisc. At Line 3, Procedure DistillDisc takes $O(\varepsilon^{-1} k^{\omega+1})$-time by Lemma 11.54.

At Line 6, by Fact 11.8, it takes $O(\varepsilon^{-1} k \cdot k^{\omega-1}) = O(\varepsilon^{-1} k^{\omega})$-time.

Thus, the total running time is $O(\varepsilon^{-1} k^{\omega+1})$.

Then, we show the time complexity of Procedure DistillDiscHD. At Line 11, Procedure DistillDiscHD takes $O(\varepsilon^{-1} k^{\omega+1} + \varepsilon^{-1} d k^{\omega-1} \log k)$-time by Lemma 11.54.

At Line 14, by Fact 11.7, it takes the time $\mathcal{T}_{\mathrm{mat}}(k, d, s)$. We know that $s \geq k$. We can consider two cases.

- In case 1, $d \leq k$, we can just simply bound the time by $\mathcal{T}_{\mathrm{mat}}(k, k, s) = O(k^{\omega} \cdot (s/k)) = O(k^{\omega-1} s) = O(\varepsilon^{-1} k^{\omega})$. (In this regime, this part running time is dominated by Line 11)

- In case 2, $d \geq k$, we can just bound the time by $\mathcal{T}_{\mathrm{mat}}(k, d, s) = O(k^{\omega} \cdot (d/k) \cdot (s/k)) = dsk^{\omega-2} = O(\varepsilon^{-1} dk^{\omega-1})$

At Line 17, by Fact 11.8, it takes $O(\varepsilon^{-1} k \cdot k^{\omega-1}) = O(\varepsilon^{-1} k^{\omega})$-time.

Thus, the total running time is $O(\varepsilon^{-1} k^{\omega+1} + \varepsilon^{-1} dk^{\omega-1} \log k)$. $\square$

Lemma 11.75 (Estimation error of Algorithm 62)**.** *Let $\widehat{x}'$ be the output of Algorithm 62 (with $d = 1$ or $d > 1$). Then, with high probability,*

$$\|(\widehat{x}' - \widehat{x})_S\|_2^2 \lesssim \|\widehat{x}_{\overline{S}}\|_2^2.$$

Proof. Let $D := \mathrm{Uniform}([p]^d)$. Recall that $n = p^d$. Let $\mathcal{F}$ be the family of length-n discrete signals with frequencies in S:

$$\mathcal{F} = \Big\{ \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} \langle f_j, t\rangle / p} \mid v_j \in \mathbb{C} \Big\}$$

Then, it is well-known that $\{v_j(t) = \exp(2\pi \mathbf{i} \langle f_j, t\rangle/p)\}_{j \in [k]}$ forms an orthonormal basis for $\mathcal{F}$ with respect to the distribution D, i.e.,

$$\mathbb{E}_{t \sim D}[\overline{v_i(t)} v_j(t)] = \mathbf{1}_{i=j} \quad \forall i, j \in [k].$$

Now, we define some notations. Let $\alpha : \mathcal{F} \to \mathbb{C}^k$ be a linear operator such that for any $h(t) = \sum_{j=1}^{k} a_j \exp(2\pi \mathbf{i} \langle f_j, t\rangle/p)$,

$$\alpha(h) := \begin{bmatrix} a_1 & a_2 & \cdots & a_k \end{bmatrix}^{\top}.$$

Suppose the true discrete signal $x(t) = \sum_{j=1}^{n} v_j \exp(2\pi \mathbf{i} \langle j, t\rangle/p)$. Define

$$x_S(t) := \sum_{f \in S} v_f \exp(2\pi \mathbf{i} \langle f, t\rangle/p),$$
$$x_{\overline{S}}(t) := \sum_{f \in \overline{S}} v_f \exp(2\pi \mathbf{i} \langle f, t\rangle/p).$$

Let $\sqrt{W} \in \mathbb{R}^{s\times s}$ denote the diagonal matrix $\mathrm{diag}(\sqrt{w_1},\ldots,\sqrt{w_s})$. Define

$$\begin{aligned}
A_w &:= \sqrt{W}\cdot A,\\
X_w &:= \sqrt{W}\cdot \begin{bmatrix} x(t_1) & \cdots & x(t_s)\end{bmatrix}^\top,\\
X_w^S &:= \sqrt{W}\cdot \begin{bmatrix} x_S(t_1) & \cdots & x_S(t_s)\end{bmatrix}^\top,\\
X_w^{\overline{S}} &:= \sqrt{W}\cdot \begin{bmatrix} x_{\overline{S}}(t_1) & \cdots & x_{\overline{S}}(t_s)\end{bmatrix}^\top.
\end{aligned}$$

Notice that for any $h = \sum_{i=1}^k a_i \exp(2\pi\mathbf{i}\langle f_i, t\rangle/p) \in \mathcal{F}$,

$$A_w\alpha(h) = \sqrt{W}\cdot \begin{bmatrix} \exp(2\pi\mathbf{i}f_1t_1/n) & \cdots & \exp(2\pi\mathbf{i}f_kt_1/n)\\ \vdots & \ddots & \vdots\\ \exp(2\pi\mathbf{i}f_1t_s/n) & \cdots & \exp(2\pi\mathbf{i}f_kt_s/n)\end{bmatrix}\begin{bmatrix}a_1\\ \vdots\\ a_k\end{bmatrix} = \begin{bmatrix}\sqrt{w_1}h(t_1)\\ \vdots\\ \sqrt{w_s}h(t_s)\end{bmatrix}.$$

Thus, by Moore-Penrose inverse, we have

$$\alpha(h) = (A_w^*A_w)^{-1}A_w^*\cdot \begin{bmatrix}\sqrt{w_1}h(t_1)\\ \vdots\\ \sqrt{w_s}h(t_s)\end{bmatrix}. \tag{11.25}$$

Let $x'(t) := \sum_{j=1}^k \widehat{x}'_{f_j}\exp(2\pi\mathbf{i}\langle f_j, t\rangle/p)$ be the output signal in the time domain. Then we claim that

$$\begin{aligned}
\|x' - x_S\|_D^2 &= \|\alpha(x') - \alpha(x_S)\|_2^2\\
&= \|(A_w^*A_w)^{-1}A_w^*\cdot(X_w - X_w^S)\|_2^2\\
&= \|(A_w^*A_w)^{-1}A_w^*\cdot X_w^{\overline{S}}\|_2^2\\
&\le \lambda_{\max}((A_w^*A_w)^{-1})^2\cdot\|A_w^*\cdot X_w^{\overline{S}}\|_2^2\\
&\le \|A_w^*\cdot X_w^{\overline{S}}\|_2^2,
\end{aligned}$$

where the first step follows from the definition of α, the second step follows from $\alpha(x') = v'$ being the optimal solution of Eq. (11.24) and Eq. (11.25) for x_S, the third step follows from $x = x_S + x_{\overline{S}}$, the fifth step follows from Lemma 11.15 and Lemma 11.54 and holds with high probability.

Notice that $x_{\overline{S}}$ is orthogonal to $\mathcal{F}$. And by Lemma 11.76, we know that (L, w) is generated by an ε-WBSP. Hence, by Lemma 11.47, we have

$$\|x' - x_S\|_D^2 \leq \|A_w^* \cdot X_w^{\overline{S}}\|_2^2 \lesssim \varepsilon \|x_{\overline{S}}\|_D^2.$$

By Parseval's theorem (Theorem 11.24), we conclude that

$$\|\widehat{x}' - \widehat{x}_S\|_2^2 \leq \varepsilon \|\widehat{x}_{\overline{S}}\|_2^2$$

holds with high probability.

□

11.11.2 Composition of two WBSPs

In this section, we prove the following key lemma on the composition of two WBSPs for discrete signals.

Lemma 11.76 (WBSP Composition Lemma)**.** *Let $m_0, m_1, n \in \mathbb{Z}_+$, $m_1 \leq m_0 \leq n$. Let $\{f_1, \cdots, f_k\} \subseteq [n]$. Let $\mathcal{F}$ be the family of discrete k-sparse signals in $t \in [n]$:*

$$\mathcal{F} = \Big\{ v_0 + \sum_{j=1}^{k} v_j \cdot \exp(2\pi \mathbf{i} f_j t/n) \mid \forall v_j \in \mathbb{C}, j = \{0, \dots, k\} \Big\}$$

Define the following two WBSPs for $\mathcal{F}$:

- *Let P_1 be an ε-WBSP generating m_1 samples, with input distribution D_1, coefficients α_1, and output distributions $D_{1,i}$ for $i \in [m_1]$.*
- *Let P_2 be an ε-WBSP generating m_2 samples, with input distribution D_2, coefficients α_2, and output distributions $D_{2,i}$ for $i \in [m_2]$.*

We can composite P_1 and P_2 by taking $D_2(x_i) := \frac{w_{1,i}}{\sum_{j \in [m_1]} w_{1,j}}$ for $i \in [m_1]$. Let $P_1 \circ P_2$ denote the composition of P_1, P_2.

Then, if P_1 satisfies $D_{1,i} = D_1 = \mathrm{Uniform}([n])$, then $P_1 \circ P_2$ is an $O(\varepsilon)$-WBSP generating m_2 samples, with input distribution D_1, coefficients w_2, and output distributions D_1 for all $i \in [m_2]$.

Proof. Let $S_1 = \{x_1, \dots, x_{m_1}\}$ denote the set sampled by P_1 and $S_2 = \{x'_1, \dots, x'_{m_2}\}$ denote the set sampled by P_2. Then, we have $S_2 \subset S_1$. In the followings, we show that $P_1 \circ P_2$ satisfies all the stated properties.

Input distribution and the first WBSP property. We first show that $P_1 \circ P_2$ satisfies the first property of WBSP in Definition 11.10 with respect to distribution D_1, that is,

$$\|f\|_{S_2,w_2} \in [1 - O(\sqrt{\varepsilon}), 1 + O(\sqrt{\varepsilon})]\|f\|_{D_1}^2 \quad \forall f \in \mathcal{F}.$$

By definition of ε-WBSP (Definition 11.10), we have for any $f \in \mathcal{F}$,

$$\begin{aligned} &\|f\|_{S_1,w_1}^2 \in [1 - \sqrt{\varepsilon}, 1 + \sqrt{\varepsilon}] \cdot \|f\|_{D_1}^2, \text{ and} \\ &\|f\|_{S_2,w_2}^2 \in [1 - \sqrt{\varepsilon}, 1 + \sqrt{\varepsilon}] \cdot \|f\|_{D_2}^2. \end{aligned} \tag{11.26}$$

By the definition of D_2, we have $\|f\|_{D_2}^2 = \|f\|_{S_1,w_1}^2$ (assuming $\|w_1\|_1 = 1$ without loss of generality). Thus, we get that

$$\begin{aligned} \|f\|_{S_2,w_2}^2 &\in [1 - \sqrt{\varepsilon}, 1 + \sqrt{\varepsilon}]\|f\|_{S_1,w_1}^2 \\ &\in [1 - \sqrt{\varepsilon}, 1 + \sqrt{\varepsilon}] \cdot [1 - \sqrt{\varepsilon}, 1 + \sqrt{\varepsilon}]\|f\|_{D_1}^2 \\ &\in [1 - 3\sqrt{\varepsilon}, 1 + 3\sqrt{\varepsilon}]\|f\|_{D_1}^2. \end{aligned} \tag{11.27}$$

Coefficients. Then, consider the equivalent coefficients α_3 of $P_1 \circ P_2$. Let $D_{3,i}$ be the output distribution of the i-th sample x'_i produced by $P_1 \circ P_2$. By Fact 11.77,

$$D_{3,i}(x'_i) = \sum_{j=1}^{m_1} D_{2,i}(x_j) \cdot D_{1,j}(x'_i) = D_1(x'_i),$$

where the second step follows from the assumption that $D_{1,j} = D_1$ for all $j \in [m_1]$. Thus, we have $D_{3,i} = D_1$ for all $i \in [m_2]$. Since its weight vector is w_2 and input distribution is D_1, by definition, we have for $i \in [m_2]$,

$$\alpha_{3,i} = w_{2,i} \cdot \frac{D_{3,i}(x'_i)}{D_1(x'_i)} = w_{2,i}.$$

Thus, the coefficients of $P_1 \circ P_2$ is w_2.

The second WBSP property. We first bound $\sum_{i=1}^{m_2} \alpha_{3,i}$. Since $\alpha_3 = w_2$, we just need to bound $\sum_{i=1}^{m_2} w_{2,i}$. Let $f_1 := \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^\top \in \mathbb{C}^n$. Then, it is easy to see that $f_1 \in \mathcal{F}$ with $v_0 = 1$ and $v_i = 0$ for $i \in [k]$. By Eq. (11.27), we have

$$\begin{aligned}
\|f_1\|_{S_2,w_2}^2 &= \sum_{i=1}^{m_2} w_{2,i} \\
&\in [1-\sqrt{\varepsilon}, 1+\sqrt{\varepsilon}] \cdot \|f_1\|_{D_1}^2 \\
&= [1-\sqrt{\varepsilon}, 1+\sqrt{\varepsilon}],
\end{aligned}$$

where the last step follows from $\|f_1\|_{D_1}^2 = \sum_{i=1}^n D_1(i) = 1$. Hence,

$$\sum_{i=1}^{m_2} \alpha_{3,i} = \sum_{i=1}^{m_2} w_{2,i} \leq 1 + \sqrt{\varepsilon} \leq \frac{5}{4}.$$

We also need to show that $\alpha_{3,i} K_{\mathsf{IS},D_{3,i}} = O(\varepsilon)$ for all $i \in [m_2]$. By definition, we have

$$\begin{aligned}
K_{\mathsf{IS},D_{3,i}} &= \sup_t \left\{ \frac{D_1(t)}{D_{3,i}(t)} \cdot \sup_{f \in \mathcal{F}} \left\{ \frac{|f(t)|^2}{\|f\|_{D_1}^2} \right\} \right\} \\
&= \sup_t \sup_{f \in \mathcal{F}} \left\{ \frac{|f(t)|^2}{\|f\|_{D_1}^2} \right\} \\
&\leq k, \qquad (11.28)
\end{aligned}$$

where the second step follows from $D_{3,i} = D_1$ and the last step follows from the energy bound (Theorem 11.22) and the assumption that $D_1 = \mathrm{Uniform}([n])$.

Since P_2 is an ε-WBPS, we have

$$\begin{aligned}
K_{\mathsf{IS},D_{2,i}} &= \sup_t \left\{ \frac{D_2(t)}{D_{2,i}(t)} \cdot \sup_{f \in \mathcal{F}} \left\{ \frac{|f(t)|^2}{\|f\|_{D_2}^2} \right\} \right\} \\
&= \sup_t \left\{ \frac{D_2(t)}{D_{2,i}(t)} \cdot \sup_{f \in \mathcal{F}} \left\{ \frac{|f(t)|^2}{\|f\|_{S_1,w_1}^2} \right\} \right\} \\
&\geq (1+\sqrt{\varepsilon})^{-1} \cdot \sup_t \left\{ \frac{D_2(t)}{D_{2,i}(t)} \cdot \sup_{f \in \mathcal{F}} \left\{ \frac{|f(t)|^2}{\|f\|_{D_1}^2} \right\} \right\},
\end{aligned}$$

where the second step follows from $\|f\|_{D_2} = \|f\|_{S_1,w_1}$, the third step follows from Eq. (11.26). And for all $i \in [m_2]$,

$$\alpha_{2,i} K_{\mathsf{IS},D_{2,i}} = O(\varepsilon),$$

which implies that

$$\alpha_{2,i} \cdot \sup_t \left\{ \frac{D_2(t)}{D_{2,i}(t)} \cdot \sup_{f \in \mathcal{F}} \{ \frac{|f(t)|^2}{\|f\|_{D_1}^2} \} \right\} = O(\varepsilon).$$

Since D_1 is uniform, we know that $\{\exp(2\pi \mathbf{i} f_j t)\}_{j \in [k]}$ form an orthonormal basis with respect to D_1. Thus, by Fact 11.50, for any $t \in [n]$,

$$\sup_{f \in \mathcal{F}} \{ \frac{|f(t)|^2}{\|f\|_{D'}^2} \} = \sum_{j=1}^{k} |\exp(2\pi \mathbf{i} f_j t)|^2 = k.$$

Hence, we get that

$$\alpha_{2,i} \cdot \sup_t \left\{ \frac{D_2(t)}{D_{2,i}(t)} \right\} = O(\varepsilon / k)$$

Therefore,

$$\begin{aligned} \alpha_{3,i} K_{\mathsf{IS}, D_{3,i}} &\leq w_{2,i} \cdot k \\ &= \alpha_{2,i} \cdot \frac{D_2(x_i)}{D_{2,i}(x_i)} \cdot k \\ &\leq \alpha_{2,i} \cdot \sup_t \left\{ \frac{D_2(t)}{D_{2,i}(t)} \right\} \cdot k \\ &= O(\varepsilon / k) \cdot k \\ &= O(\varepsilon). \end{aligned}$$

where the first step follows from $\alpha_3 = w_2$ and Eq. (11.28), the second step follows from the definition of $w_{2,i}$.

Thus, we prove that $P_1 \circ P_2$ is an $O(\varepsilon)$-WBSP with input distribution D_1, output distributions D_1, coefficients w_2. □

Fact 11.77 (Double-sampling distribution)**.** *For $i \in [n]$, let D_i be a distribution over the domain G. Suppose we first sample x_i from D_i for each $i \in [n]$. Let $w_1, \cdots, w_n \in \mathbb{R}_+$ such that $\sum_{i=1}^n w_i = 1$. Conditioned on the samples $\{x_1, \ldots, x_n\}$, let D' be a distribution over these samples such that $D'(x_i) = w_i$. Then, we sample an x' from D'.*

Then, the distribution of x' is D'', where

$$D''(x) = \sum_{i=1}^{n} w_i D_i(x) \quad \forall x \in G.$$

Proof. Notice that the second sampling process is equivalent to sample an index $\mathsf{i} \in [n]$. Hence, for any $a \in G$,

$$\begin{aligned}
\Pr[x' = a] = & \sum_{j=1}^{n} \Pr[\mathsf{i} = j] \cdot \Pr_{D_j}[x_j = a \mid \mathsf{i} = j] \\
= & \sum_{j=1}^{n} w_j \cdot \Pr_{D_j}[x_j = a] \\
= & \sum_{j=1}^{n} w_j D_j(a) \\
= & \, D''(a).
\end{aligned}$$

where the first step follows from law of total probability, and the second step follows from sampling x_j from D_j is independent to sampling the index i from D'. □

11.12 High Dimensional Reduction Under Noiseless Assumption

This section is organized as follows:

- Section 11.12.1 shows that Fourier basis is linear independent.
- Section 11.12.2 shows that, there is straightforward algorithm for signal estimation under noiseless setting.

11.12.1 Fourier basis is linear independent on randomly sampled points

Lemma 11.78. *Given a basis $\mathcal{B}$ of m known vectors $b_1, b_2, \cdots b_m \in \mathbb{R}^d$, let $\Lambda(\mathcal{B}) \subset \mathbb{R}^d$ denote the lattice*

$$\Lambda(\mathcal{B}) = \left\{ z \in \mathbb{R}^d : z = \sum_{i=1}^{m} c_i b_i, c_i \in \mathbb{Z}, \forall i \in [m] \right\}$$

Suppose that $f_1, f_2, \cdots, f_k \in \Lambda(\mathcal{B})$. Randomly samples a vector $o \sim \mathcal{N}(0, I_d)$. Let $t = k^{-1} \cdot (T/2 + o \cdot \min(\max_{j\in[k]}|\langle f_j, o\rangle|^{-1}, k^{-1}T))$. Let $t_i := (i-1)\cdot t$ for $i \in [k]$.

Let $v_j = (\exp(2\pi\mathbf{i}\langle f_j, t_1\rangle), \exp(2\pi\mathbf{i}\langle f_j, t_2\rangle), \cdots, \exp(2\pi\mathbf{i}\langle f_j, t_k\rangle))$ for $j \in [k]$. We have that $v_1, v_2, \cdots, v_k$ are linear independent with probability 1.

Proof. We have that

$$u_j := \begin{bmatrix} e^{2\pi\mathbf{i}\langle f_1, 0\cdot t\rangle} \\ e^{2\pi\mathbf{i}\langle f_1, 1\cdot t\rangle} \\ \vdots \\ e^{2\pi\mathbf{i}\langle f_1, (k-1)\cdot t\rangle} \end{bmatrix} = \begin{bmatrix} 1 \\ e^{2\pi\mathbf{i}\langle f_1, t\rangle} \\ \vdots \\ (e^{2\pi\mathbf{i}\langle f_1, t\rangle})^{k-1} \end{bmatrix} = \begin{bmatrix} 1 \\ w_1 \\ \vdots \\ w_1^{k-1} \end{bmatrix},$$

where $w_1 := e^{2\pi\mathbf{i}\langle f_1, t\rangle}$. Similarly, we can define $w_j := e^{2\pi\mathbf{i}\langle f_j, t\rangle}$. And we have

$$\begin{bmatrix} | & | & & | \\ u_1 & u_2 & \cdots & u_k \\ | & | & & | \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ w_1 & w_2 & \cdots & w_k \\ w_1^2 & w_2^2 & \cdots & w_k^2 \\ \vdots & \vdots & \ddots & \vdots \\ w_1^{k-1} & w_2^{k-1} & \cdots & w_k^{k-1} \end{bmatrix},$$

which is a Vandermonde matrix. Hence, they are linearly independent as long as $w_1, \ldots, w_k$ are distinct. Or equivalently,

$$\langle f_1, t\rangle \mod 2\pi, \cdots, \langle f_k, t\rangle \mod 2\pi$$

are distinct.

Next, we will show that $w_1, \ldots, w_k$ are distinct with probability 1. We first show that:

$$\langle f_i, t\rangle \leq \langle f_i, o\rangle / \max_{j\in[k]}|\langle f_j, o\rangle| \in [-1, 1] \subset [-\pi, \pi].$$

Then, $\langle f_i, t\rangle = \langle f_j, t\rangle \mod 2\pi$ is equivalent to $\langle f_i, t\rangle = \langle f_j, t\rangle$, and is equivalent to $\langle f_i - f_j, t\rangle = 0$, and is equivalent to $\langle f_i - f_j, o\rangle = 0$. However, $\langle f_i - f_j, o\rangle$ follows from $\mathcal{N}(0, \|f_i - f_j\|_2^2)$. By our assumption, $\|f_i - f_j\|_2^2 \neq 0$. Thus,

$$\Pr_{t\sim\mathcal{N}(0, I_d)}[\langle f_i - f_j, t\rangle = 0] = 0.$$

Therefore, by union bound,

$$\Pr_{t\sim\mathcal{N}(0,I_d)}[\exists i \neq j \in [k] : \langle f_i - f_j, t\rangle = 0] = 0.$$

□

11.12.2 Reduction

Lemma 11.79 (High Dimension Noiseless)**.** *Given a basis $\mathcal{B}$ of m known vectors $b_1, b_2, \cdots b_m \in \mathbb{R}^d$, let $\Lambda(\mathcal{B}) \subset \mathbb{R}^d$ denote the lattice*

$$\Lambda(\mathcal{B}) = \Big\{ z \in \mathbb{R}^d : z = \sum_{i=1}^m c_i b_i, c_i \in \mathbb{Z}, \forall i \in [m] \Big\}$$

Suppose that $f_1, f_2, \cdots, f_k \in \Lambda(\mathcal{B})$. Let $x^*(t) = \sum_{j=1}^k v_j e^{2\pi\mathbf{i}\langle f_j, t\rangle}$. Given $x^*(t)$ is observable for $t \in [0,T]^d$. Let $\eta = \min_{i\neq j} \|f_j - f_i\|_\infty$.

Given $D, \eta \in \mathbb{R}_+$. Suppose that there is an algorithm FrequencyEstimation$(x^*, k, \rho, d, F, T, \mathcal{B}$ *that*

- *takes $\mathcal{S}(k, \rho, d, F, T, \eta)$ samples,*
- *runs $\mathcal{T}(k, \rho, d, F, T, \eta)$ time,*
- *output a set L of frequencies such that, for each f_i, there is $f_i' \in L$, $|f_i - f_i'| \leq D/T$, holds with probability $1 - \rho$.*

*Let $C = |L| \cdot (D/T + \sqrt{m}\|\mathcal{B}\|)^m \cdot \frac{\pi^{m/2}}{(m/2)!} \cdot \frac{1}{|\det(\mathcal{B})|}$. If $C \leq d$, then, there is an algorithm (*SignalEstimation$(x, k, d, F, T, \mathcal{B})$ *) that*

- *takes $O(\widetilde{k} + \mathcal{S}(k, d, F, T, \eta))$ samples,*
- *runs $O(\widetilde{k}^\omega + \widetilde{k}^2 d + \mathcal{T}(k, d, F, T, \eta))$ time,*
- *output $y(t) = \sum_{j=1}^{\widetilde{k}} v_j' \cdot \exp(2\pi\mathbf{i}\langle f_j', t\rangle)$ such that*
  - *$\widetilde{k} \leq C$,*

– $y(t) = x(t)$, *holds with probability* $(1-\rho)^2$.

Proof. First, we recover the frequencies by utilizing the algorithm FrequencyEstimation$(x, k, d, T, F$
Let L be the set of frequencies output by the algorithm FrequencyEstimation$(x, k, d, T, F, \mathcal{B})$.

We define $\widetilde{L}$ as follows:

$$\widetilde{L} := \{f \in \Lambda(\mathcal{B}) \mid \exists f' \in L,\ |f' - f| < D/T\}.$$

We use $\widetilde{k}$ to denote the size of set $\widetilde{L}$. We use $f'_1, f'_2, \cdots, f'_{\widetilde{k}}$ to denote the frequencies in the set $\widetilde{L}$.

By applying Lemma 11.70, we have that

$$\widetilde{k} = |\widetilde{L}| \leq |L| \cdot (D/T + \sqrt{m}\|\mathcal{B}\|)^m \cdot \frac{\pi^{m/2}}{(m/2)!} \cdot \frac{1}{|\det(\mathcal{B})|}.$$

Next, we focus on recovering magnitude $v' \in \mathbb{C}^{\widetilde{k}}$. Let $\rho_1 := \rho$. First, we randomly samples a vector $o \sim \mathcal{N}(0, I_d)$. Let $t = k^{-1} \cdot (T/2 + o \cdot \min(\max_{j \in [k]} |\langle f_j, o\rangle|^{-1}, \rho_1 T))$. Let $t_i := (i-1) \cdot t$ for $i \in [k]$. We have that $|o \cdot \min(\max_{j \in [k]} |\langle f_j, o\rangle|^{-1}, \rho_1 T)| \leq o\rho_1 T$. Then with probability $1 - \rho_1$, $|o| \leq 1/(2\rho_1)$, then $t_i \in [0, T]^d, \forall i \in [k]$.

Consider the matrix $A \in \mathbb{C}^{\widetilde{k} \times \widetilde{k}}$, where the (i,j)-th entry in A is $A_{i,j} = \exp(2\pi \mathbf{i} \langle f'_j, t_i\rangle)$ for each $i \in [\widetilde{k}]$ and $j \in [\widetilde{k}]$. Let $b = (x^*(t_1), x^*(t_2), \cdots, x^*(t_{\widetilde{k}}))^\top \in \mathbb{C}^{\widetilde{k}}$.

Then, if the following linear system solvable, then we solve the following linear system:

$$Av' = b. \tag{11.29}$$

We output $y(t) = \sum_{j=1}^{\widetilde{k}} v'_j \cdot \exp(2\pi \mathbf{i} \langle f'_j, t\rangle)$, and we have that $y(t) = x^*(t)$.

Finally, by Lemma 11.78, we have that Eq. (11.29) is solvable with a large probability.

□

11.13 Semi-continuous Approximation

In this section, we justify the usefulness of the semi-continuous setting by showing that for any k-Fourier-sparse, it can be approximated by a k-Fourier-sparse semi-continuous signal. This section is organized as follows:

- In Sections 11.13.1 and 11.13.2, we give some technical tools on the Gaussian multiplier.
- In Section 11.13.3, we prove the main result of this section (Theorem 11.86), which is incomparable to the existence result in [CKPS16].
- In Section 11.13.4, we give a fast optimal-sparsity Fourier interpolation algorithm with a different error guarantee.
- In Section 11.13.5, we also show that the frequency gap of the approximation can be increased to $\Theta(1/T)$ if we slightly blow up the sparsity of the approximation signal (Corollary 11.90).

11.13.1 Properties related to Gaussians

Definition 11.11 (Gaussian Multiplier)**.** For parameters μ, σ, we define

$$M_{\mu,\sigma^2}(x) = e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

i.e. it is a Gaussian scaled so that its maximum value is 1.

We stat a standard result for Gaussian multiplier (see [LLM21] for example).

Lemma 11.80. *Let $0 < \varepsilon < 0.1$ be a parameter. Let c be a real number such that $0 < c \leq (\log(1/\varepsilon))^{-1/2}$. Let M be defined as in Definition 11.11. Define*

$$f(x) := \sum_{j=-\infty}^{\infty} \frac{c}{\sqrt{2\pi}} M_{cj\sigma,\sigma^2}(x)\,.$$

Then

$$1 - \varepsilon^{10} \leq f(x) \leq 1 + \varepsilon^{10}$$

for all x*. Furthermore, let* $\alpha \leq 1$ *be a parameter and* $c = \alpha(\log(1/\varepsilon))^{-1/2}$*. Then*

$$1 - \varepsilon^{10/\alpha^2} \leq f(x) \leq 1 + \varepsilon^{10/\alpha^2}$$

for all x*.*

Proof. Without loss of generality $\sigma = 1$. Now the function f is c-periodic and even, so we may consider its Fourier expansion

$$\begin{aligned} f(x) &= a_0 + 2\sum_{j=1}^{\infty} a_j \cos(2j\pi x/c) \\ &= a_0 + 2a_1 \cos\left(\frac{2\pi x}{c}\right) + 2a_2 \cos\left(\frac{4\pi x}{c}\right) + \dots \end{aligned}$$

and we will now compute the Fourier coefficients. First note that

$$\begin{aligned} a_0 &= \frac{1}{c}\int_0^c f(x)\mathrm{d}x \\ &= \frac{1}{\sqrt{2\pi}} \sum_{j=-\infty}^{\infty} \int_{c(j+1)}^{cj} M_{0,1}(x)\mathrm{d}x \\ &= 1. \end{aligned}$$

where the second step follows from the definition of f.

Next, for any $j \geq 1$,

$$\begin{aligned}
a_j &= \frac{1}{c}\int_0^c f(x)\cos\left(\frac{2\pi j x}{c}\right)\mathrm{d}x \\
&= \frac{1}{c}\int_0^c \sum_{j=-\infty}^{\infty} \frac{c}{\sqrt{2\pi}} M_{cj,1}(x)\cos\left(\frac{2\pi j x}{c}\right)\mathrm{d}x \\
&= \frac{1}{\sqrt{2\pi}} \sum_{l=-\infty}^{\infty} \int_{c(l+1)}^{cl} M_{0,1}(x)\cos\left(\frac{2\pi j x}{c}\right)\mathrm{d}x \\
&= \frac{1}{\sqrt{2\pi}} \sum_{l=-\infty}^{\infty} \int_{c(l+1)}^{cl} \exp(-\frac{x^2}{2})\cos\left(\frac{2\pi j x}{c}\right)\mathrm{d}x \\
&= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \frac{1}{2}\left(\exp(\frac{-x^2}{2} + \frac{2\pi \mathbf{i} j x}{c}) + \exp(\frac{-x^2}{2} - \frac{2\pi \mathbf{i} j x}{c})\right)\mathrm{d}x \\
&= \exp(-\frac{2\pi^2 j^2}{c^2}).
\end{aligned}$$

where the second step follows from the definition of f. The fourth step follows from the definition of M.

Then we can claim that

$$\begin{aligned}
|f(x) - 1| &\leq 2\sum_{j=1}^{\infty} \exp(-\frac{2\pi^2 j^2}{c^2}) \\
&\leq 2\sum_{j=1}^{\infty} \exp(-15j/c^2) \\
&= \frac{2\exp(-15/c^2)}{1 - \exp(-15/c^2)} \\
&\leq \frac{2\varepsilon^{15/\alpha^2}}{1 - \varepsilon^{15}} \\
&\leq \varepsilon^{10/\alpha^2} \\
&\leq \varepsilon^{10}
\end{aligned}$$

where the first step follows from $a_0 = 1$, the forth step follows from $\alpha \leq 1$ and $c = \alpha(\log(1/\varepsilon))^{-1/2}$, the fifth step follows from $\varepsilon \in (0, 0.1)$, □

Claim 11.81. *Let $\varepsilon \in (0, 0.01)$ be a parameter. Let $\varepsilon_0 = 2\varepsilon$, $c = 0.01/\sqrt{\log(1/\varepsilon)}$,*

$K = \lceil \frac{1+0.5\varepsilon}{c\varepsilon^2} \rceil$. Let $M : \mathbb{R} \to \mathbb{R}$ be defined as in Definition 11.11. Define $f : \mathbb{R} \to \mathbb{R}$

$$f(x) = \sum_{j=-K}^{K} \frac{c}{\sqrt{2\pi}} M_{\mu_j, \sigma^2}(x), \quad \text{where} \quad \mu_j = cj\varepsilon^2 l, \quad \sigma^2 = \varepsilon^4 l^2 .$$

Then the following properties is satisfied

- *Part 1. $f(x) \in [0, 1+\varepsilon_0]$, for all x*
- *Part 2. $f(x) \in [1-\varepsilon_0, 1+\varepsilon_0]$, for all $x \in [-l, l]$*
- *Part 3. $f(x) \in [0, \varepsilon_0]$, for all $|x| \geq (1+\varepsilon)l$*

Proof. Using Lemma 11.80 with setting the following parameter choice $\sigma = \varepsilon^2 l$, $c = 0.01(\log(1/\varepsilon))^{-1/2}$, we define

$$f_0(x) = \sum_{j=-\infty}^{\infty} \frac{c}{\sqrt{2\pi}} M_{\mu_j, \sigma^2} .$$

we can get that $1 - \varepsilon \leq f_0(x) \leq 1 + \varepsilon$.

We will then upper bound the perturbation between f and f_0. Firstly, we will provide a upper bound for M when $x/(\varepsilon^2 l) - cj \geq 10$

$$\begin{aligned} \frac{c}{\sqrt{2\pi}} M_{\mu_j, \sigma^2}(x) &= \frac{c}{\sqrt{2\pi}} \exp(-(x - cj\varepsilon^2 l)^2 / (2\varepsilon^4 l^2)) \\ &= \frac{c}{\sqrt{2\pi}} \exp(-(x/(\varepsilon^2 l) - cj)^2/2) \\ &\leq \exp(-|x/(\varepsilon^2 l) - cj|) \end{aligned} \tag{11.30}$$

where the first step follows from the definition of M, the third step follows from $x/(\varepsilon^2 l) - cj \geq 10$.

Next we will claim that

$$\begin{aligned} \sum_{j=0}^{\infty} \exp(-cj) &= \frac{1}{1 - \exp(-c)} \\ &\leq \frac{2}{c} \\ &\leq \frac{200}{\varepsilon} \end{aligned} \tag{11.31}$$

where the first step follows from $c > 0$ and the sum of geometric sequence, the second step follows from $1 - \exp(-x) > x/2$ when $x < 0.1$, the third step follows from $c \geq 0.01\varepsilon$.

Because of the symmetry of M, We will also claim that

$$M_{\mu,\sigma^2}(x) = M_{-\mu,\sigma^2}(-x) \tag{11.32}$$

Part 1.

Because $M_{\mu,\sigma} = \exp(-(x-\mu)^2/\sigma^2) \geq 0$, we can obtain that $f(x) \geq 0$. Besides, we will claim that

$$\begin{aligned} f(x) \leq & ~ f_0(x) \\ \leq & ~ 1 + \varepsilon \\ \leq & ~ \varepsilon_0 \end{aligned}$$

where the first step follows from $M_{\mu,\sigma} \geq 0$.

Part 2.

Firstly, we upper bound f when $x \in [-l, l]$. To bound $\sum_{j=K+1}^{\infty} \frac{c}{\sqrt{2\pi}} M_{\mu_j,\sigma^2}(x)$, we obtain

$$\begin{aligned} cj - x/(\varepsilon^2 l) \geq & ~ cj - 1/(\varepsilon^2) \\ \geq & ~ c(K+1) - 1/\varepsilon^2 \\ \geq & ~ 0.5/\varepsilon \\ \geq & ~ 10 \end{aligned} \tag{11.33}$$

where the first step follows from $x \in [-l, l]$, the second step follows from $j \geq K + 1$, the third step follows from $K + 1 \geq (1 + 0.5\varepsilon)/(c\varepsilon^2)$.

Then we can upper bound

$$\begin{aligned}\sum_{j=K+1}^{\infty} \frac{c}{\sqrt{2\pi}} M_{\mu_j,\sigma^2}(x) \leq & \sum_{j=K+1}^{\infty} \exp(x/(\varepsilon^2 l) - cj) \\ \leq & \frac{200}{\varepsilon} \exp(x/(\varepsilon^2 l) - c(K+1)) \\ \leq & \frac{200}{\varepsilon} \exp(-0.5/\varepsilon) \\ \leq & 0.1\varepsilon \end{aligned} \tag{11.34}$$

where the first step follows from Eq. (11.30) and Eq. (11.33), the second step follows from Eq. (11.31), the third step follows from Eq. (11.33), the last step follows from $\varepsilon \in (0, 0.01)$.

We conclude

$$\begin{aligned}|f(x) - 1| = & \Big| f_0(x) - 1 - \sum_{j=K+1}^{\infty} \frac{c}{\sqrt{2\pi}} M_{\mu_j,\sigma^2}(x) - \sum_{j=-\infty}^{-K-1} \frac{c}{\sqrt{2\pi}} M_{\mu_j,\sigma^2}(x) \Big| \\ \leq & |f_0(x) - 1| + \Big| \sum_{j=K+1}^{\infty} \frac{c}{\sqrt{2\pi}} M_{\mu_j,\sigma^2}(x) \Big| + \Big| \sum_{j=-\infty}^{-K-1} \frac{c}{\sqrt{2\pi}} M_{\mu_j,\sigma^2}(x) \Big| \\ \leq & \varepsilon + 0.1\varepsilon + 0.1\varepsilon \\ \leq & \varepsilon_0 \end{aligned}$$

where the first step follows from the definition of f and f_0, the second step follows from the triangle inequality, the third step follows from $|f_0(x) - 1| \leq \varepsilon$, Eq.(11.34) and Eq. (11.32).

Part 3.

Secondly, we will provide the upper bound for f when $|x| > (1+\varepsilon)l$. Without loss of generality, we can only consider $x > (1+\varepsilon)l$ because of Eq. (11.32). To bound

$\sum_{j=-K}^{K} \frac{c}{\sqrt{2\pi}} M_{\mu_j,\sigma^2}(x)$. We obtain

$$\begin{aligned} x/(\varepsilon^2 l) - cj &\geq x/(\varepsilon^2 l) - cK \\ &\geq (1+\varepsilon)/\varepsilon^2 - cK \\ &\geq 0.25/\varepsilon \\ &\geq 10 \end{aligned} \tag{11.35}$$

where the first step follows from $j \leq K$, the second step follows from $x > (1+\varepsilon)l$, the third step follows from $K \leq (1+0.5\varepsilon)/(c\varepsilon^2) + 1$ and $c \leq 0.01 \leq 0.25/\varepsilon$.

Then we can upper bound

$$\begin{aligned} \sum_{j=-K}^{K} \frac{c}{\sqrt{2\pi}} M_{\mu_j,\sigma^2}(x) &\leq \sum_{j=-K}^{K} \exp(cj - x/(\varepsilon^2 l)) \\ &\leq \frac{200}{\varepsilon} \exp(cK - x/(\varepsilon^2 l)) \\ &\leq \frac{200}{\varepsilon} \exp(-0.25/\varepsilon) \\ &\leq \varepsilon \\ &\leq \varepsilon_0 \end{aligned} \tag{11.36}$$

where the first step follows from Eq. (11.30) and Eq. (11.35), the second step follows from Eq. (11.31), the third step follows from Eq. (11.35), the fourth step follows from $\varepsilon \in (0, 0.01)$.

$\square$

Lemma 11.82. *Given $c \in \mathbb{R}$, $\gamma, l, \sigma \in \mathbb{R}_+$, $\mu \in \mathbb{R}$. Let $M : \mathbb{R} \to \mathbb{R}$ be defined as in Definition 11.11. For $|a-b| \leq \gamma$, We have that*

$$|M_{\mu,\sigma^2}(a) \exp(c\mathbf{i}a) - M_{\mu,\sigma^2}(b) \exp(c\mathbf{i}b)| \leq (\sigma^{-1} + |c|)\gamma.$$

Proof. We have that

$$\begin{aligned}
& |M_{\mu,\sigma^2}(b) - M_{\mu,\sigma^2}(a)| \\
= & |\exp(-(b-\mu)^2/(2\sigma^2)) - \exp(-(a-\mu)^2/(2\sigma^2))| \\
= & |b-a||(\xi-\mu)\exp(-(\xi-\mu)^2/(2\sigma^2))/(\sigma^2)| \\
\leq & \gamma/\sigma
\end{aligned} \tag{11.37}$$

where the first step follows from the definition of M, the second step defines $\xi \in [a,b]$ and follows from Lagrange's Mean Value Theorem, the last step follows from $|x\exp(-x^2/2)| \leq 1$. We have that

$$\begin{aligned}
& |M_{\mu,\sigma^2}(a)\exp(c\mathbf{i}a) - M_{\mu,\sigma^2}(b)\exp(c\mathbf{i}b)| \\
\leq & |M_{\mu,\sigma^2}(a)\exp(c\mathbf{i}a) - M_{\mu,\sigma^2}(b)\exp(c\mathbf{i}a)| + |M_{\mu,\sigma^2}(b)\exp(c\mathbf{i}a) - M_{\mu,\sigma^2}(b)\exp(c\mathbf{i}b)| \\
\leq & |M_{\mu,\sigma^2}(a) - M_{\mu,\sigma^2}(b)| + M_{\mu,\sigma^2}(b)\cdot|\exp(c\mathbf{i}a) - \exp(c\mathbf{i}b)| \\
\leq & \frac{\gamma}{\sigma} + M_{\mu,\sigma^2}(b)\cdot|c(a-b)| \\
\leq & \frac{\gamma}{\sigma} + |c|\gamma \\
\leq & (\frac{1}{\sigma} + |c|)\gamma
\end{aligned}$$

where the first follows from triangle inequality, the second follows from $|\exp(\mathbf{i}\theta)| = 1$, the third step follows from Eq. (11.37) and that arc length is greater than chord length, the fourth step follows from $M \leq 1$ and the assume in the statement.

□

11.13.2 Continuous Fourier transform

11.13.2.1 Bounding the tails

The goal of this section is to prove Lemma 11.83,

Lemma 11.83. *Let $\mu \in \mathbb{R}, \alpha \in \mathbb{Z}_+, \sigma,\ F, F_0, \varepsilon_1 \in \mathbb{R}_+$. Let $\sigma \in (0,1)$, $\varepsilon_1 \in (0, 0.1)$, $F > 1$. Let $M : \mathbb{R} \to \mathbb{R}$ be defined as in Definition 11.11. Let $x : \mathbb{R} \to \mathbb{C}$ be a* *function. Let* $\mathrm{supp}(\widehat{x}) \subseteq [-F, F]$. *If*

$$F_0 > \sigma^{-1} \cdot \log^{1/2}(4F/\varepsilon_1) + F,$$

then we have that,

$$\Big| \int_{-\infty}^{\infty} (\widehat{M}_{\mu,\sigma^2}(f) * \widehat{x}(f))^\alpha \mathrm{d}f - \int_{-F_0}^{F_0} (\widehat{M}_{\mu,\sigma^2}(f) * \widehat{x}(f))^\alpha \mathrm{d}f \Big| \leq \|\widehat{x}\|_\infty^\alpha \varepsilon_1 / F^\alpha.$$

Proof. First, we will calculate $\widehat{M}$. Then, we provide a bound on the tail of $\widehat{M}$. Finally, we conclude our proof.

We can claim that

$$\widehat{M}_{\mu,\sigma^2}(f) = \sqrt{2\pi\sigma^2} \cdot \exp(-2\pi \mathbf{i} \mu f) \cdot M_{0,1/(4\pi^2\sigma^2)}.$$

Taking the $|\cdot|$ on both sides of the above equation, we get

$$|\widehat{M}_{\mu,\sigma^2}(f)| = \sqrt{2\pi\sigma^2} \cdot |M_{0,1/(4\pi^2\sigma^2)}|$$

We start with

$$\begin{aligned} \int_{F_0-F}^{\infty} |\widehat{M}_{\mu,\sigma^2}(f)|^\alpha \mathrm{d}f &= \sqrt{2\pi\sigma^2} \cdot \int_{F_0-F}^{\infty} |M_{0,1/(4\pi^2\sigma^2)}|^\alpha \mathrm{d}f \\ &= \sqrt{2\pi\sigma^2} \cdot \int_{F_0-F}^{\infty} \exp(-2\alpha\pi^2\sigma^2 f^2) \mathrm{d}f \\ &= \frac{1}{\sqrt{\alpha\pi}} \cdot \int_{\sqrt{2\alpha}\pi\sigma(F_0-F)}^{\infty} \exp(-\xi^2) \mathrm{d}\xi \\ &\leq \frac{1}{\sqrt{\alpha\pi}} \cdot \int_{\sqrt{2\alpha}\pi\sigma(F_0-F)}^{\infty} \exp(-\sqrt{2\alpha}\pi\sigma(F_0 - F)\xi) \mathrm{d}\xi \\ &= \frac{1}{\sqrt{\alpha\pi}} \cdot \frac{1}{\sqrt{2\alpha}\pi\sigma(F_0 - F)} \exp(-2\alpha\pi^2\sigma^2(F_0 - F)^2) \\ &\leq \varepsilon_1/(4F)^{2\alpha}. \end{aligned} \tag{11.38}$$

where the second step follows definition of M, the third step follows from $\xi = \sqrt{2\alpha}\pi\sigma f$, the forth step follows from $\xi \leq \sqrt{2\alpha}\pi\sigma(F_0 - F)$, the last step follows from lower bound on F_0 in the Lemma statement.

Finally, we can bound LHS in the statement as follows

$$\begin{aligned}
& \Big| \int_{-\infty}^{\infty} (\widehat{M}_{\mu,\sigma^2}(f) * \widehat{x}(f))^{\alpha} \mathrm{d}f - \int_{-F_0}^{F_0} (\widehat{M}_{\mu,\sigma^2}(f) * \widehat{x}(f))^{\alpha} \mathrm{d}f \Big| \\
= & \Big| \int_{-\infty}^{F_0} (\widehat{M}_{\mu,\sigma^2}(f) * \widehat{x}(f))^{\alpha} \mathrm{d}f + \int_{F_0}^{\infty} (\widehat{M}_{\mu,\sigma^2}(f) * \widehat{x}(f))^{\alpha} \mathrm{d}f \Big| \\
= & \Big| \int_{-\infty}^{F_0} (\int_{-F}^{F} \widehat{M}_{\mu,\sigma^2}(f-\xi) \cdot \widehat{x}(\xi) \mathrm{d}\xi)^{\alpha} \mathrm{d}f + \int_{F_0}^{\infty} (\int_{-F}^{F} \widehat{M}_{\mu,\sigma^2}(f-\xi) \cdot \widehat{x}(\xi) \mathrm{d}\xi)^{\alpha} \mathrm{d}f \Big| \\
\leq & \Big| \int_{F_0}^{\infty} (\int_{-F}^{F} \widehat{M}_{\mu,\sigma^2}(f-\xi) \cdot \widehat{x}(\xi) \mathrm{d}\xi)^{\alpha} \mathrm{d}f \Big| + \Big| \int_{-\infty}^{-F_0} (\int_{-F}^{F} \widehat{M}_{\mu,\sigma^2}(f-\xi) \cdot \widehat{x}(\xi) \mathrm{d}\xi)^{\alpha} \mathrm{d}f \Big| \\
\leq & \int_{F_0}^{\infty} (\int_{-F}^{F} |\widehat{M}_{\mu,\sigma^2}(f-\xi) \cdot \widehat{x}(\xi)| \mathrm{d}\xi)^{\alpha} \mathrm{d}f + \int_{-\infty}^{-F_0} (\int_{-F}^{F} |\widehat{M}_{\mu,\sigma^2}(f-\xi) \cdot \widehat{x}(\xi)| \mathrm{d}\xi)^{\alpha} \mathrm{d}f \\
= & \|\widehat{x}\|_{\infty}^{\alpha} \cdot (\int_{F_0}^{\infty} (\int_{-F}^{F} |\widehat{M}_{\mu,\sigma^2}(f-\xi)| \mathrm{d}\xi)^{\alpha} \mathrm{d}f + \int_{-\infty}^{-F_0} (\int_{-F}^{F} |\widehat{M}_{\mu,\sigma^2}(f-\xi)| \mathrm{d}\xi)^{\alpha} \mathrm{d}f)
\end{aligned} \tag{11.39}$$

where the third step follows from triangle inequality, the fourth step follows from triangle inequality.

For the second term in the above Eq. (11.39)

$$\begin{aligned}
\int_{F_0}^{\infty} (\int_{-F}^{F} |\widehat{M}_{\mu,\sigma^2}(f-\xi)| \mathrm{d}\xi)^{\alpha} \mathrm{d}f \leq & \int_{F_0}^{\infty} (\int_{-F}^{F} |\widehat{M}_{\mu,\sigma^2}(f-F)| \mathrm{d}\xi)^{\alpha} \mathrm{d}f \\
= & \int_{F_0}^{\infty} |2F \widehat{M}_{\mu,\sigma^2}(f-F)|^{\alpha} \mathrm{d}f \\
= & (2F)^{\alpha} \cdot \int_{F_0}^{\infty} |\widehat{M}_{\mu,\sigma^2}(f-F)|^{\alpha} \mathrm{d}f \\
= & (2F)^{\alpha} \cdot \int_{F_0-F}^{\infty} |\widehat{M}_{\mu,\sigma^2}(\xi)|^{\alpha} \mathrm{d}\xi \\
\leq & (2F)^{\alpha} \cdot \varepsilon_1/(4F)^{2\alpha} \\
\leq & \varepsilon_1/(2F^{\alpha})
\end{aligned} \tag{11.40}$$

where the first step follows from what $\widehat{M}_{\mu,\sigma^2}(f-\xi) \leq \widehat{M}_{\mu,\sigma^2}(f-F)$ because $f - \xi \geq f - F \geq F_0 - F > 0$, the forth step follows from $f - F = \xi$, second last step follows from Eq. (11.38).

Similarly, we have that

$$\int_{-\infty}^{-F_0} (\int_{-F}^{F} |\widehat{M}_{\mu,\sigma^2}(f-\xi)|\mathrm{d}\xi)^\alpha \mathrm{d}f \le \varepsilon_1/2. \tag{11.41}$$

Combining Eq. (11.39), (11.40) and (11.41) completes the proof. □

11.13.2.2 Bounding the convolution

The goal of this section is to prove Lemma 11.84.

Lemma 11.84. *Given $\mu', F, \sigma', \gamma \in \mathbb{R}_+$. Let $M : \mathbb{R} \to \mathbb{R}$ be defined as in Definition 11.11. Let $x : \mathbb{R} \to \mathbb{C}$ be a function. For simplicity, let $M = M_{\mu',\sigma'^2}$. We have that for $\forall f \in \mathbb{R}$,*

$$|\widehat{M} * \widehat{x}(f) - \sum_{\mu \in \mathbb{Z}\gamma} \int_{\mu-\gamma/2}^{\mu+\gamma/2} \widehat{x}(\xi)\mathrm{d}\xi \cdot \widehat{M}(f) * \delta(f-\mu)| \le (\sigma'^2 + \sigma'|\mu'|)\gamma^2 \cdot \|\widehat{x}\|_1.$$

Proof. We will separate $\widehat{M} * \widehat{x}(f)$ into $\widehat{M}(f) * (\widehat{x}(f) \cdot \mathrm{rect}_{\gamma/2}(f-\mu))$ where $\mu \in \gamma\mathbb{Z}$. This decomposes $\widehat{x}$ into different intervals. Then, we get a bound for each interval.

We can rewrite $\widehat{M}_{\mu',\sigma'^2}(f)$ in the following sense,

$$\begin{aligned}\widehat{M}_{\mu',\sigma'^2}(f) &= \sqrt{2\pi\sigma'^2} \cdot \exp(-2\pi^2\sigma'^2 f^2) \cdot \exp(-2\pi \mathbf{i} \mu' f) \\ &= \sqrt{2\pi\sigma'^2} \cdot M_{0,1/(4\pi^2\sigma'^2)}(f) \cdot \exp(-2\pi \mathbf{i} \mu' f)\end{aligned}$$

where the second step follows from the definition of M.

First, we consider the first term in the LHS of our statement

$$\begin{aligned}\widehat{M}(f) * (\widehat{x}(f) \cdot \mathrm{rect}_{\gamma/2}(f-\mu)) &= \widehat{M}(f) * \Big(\int_{\mu-\gamma/2}^{\mu+\gamma/2} \widehat{x}(\xi)\delta(f-\xi)\mathrm{d}\xi\Big) \\ &= \int_{\mu-\gamma/2}^{\mu+\gamma/2} \widehat{x}(\xi) \cdot \widehat{M}(f) * \delta(f-\xi)\mathrm{d}\xi \\ &= \int_{\mu-\gamma/2}^{\mu+\gamma/2} \widehat{x}(\xi) \cdot \widehat{M}(f-\xi)\mathrm{d}\xi \end{aligned} \tag{11.42}$$

where the first step follows from definition of rect function, the last step follows from $\widehat{M}(f) * \delta(f-\xi) = \widehat{M}(f-\xi)$.

Now, we consider the second term in the LHS of our statement

$$\begin{aligned} \int_{\mu-\gamma/2}^{\mu+\gamma/2} \widehat{x}(\xi)\mathrm{d}\xi \cdot \widehat{M}(f) * \delta(f-\mu) = & \int_{\mu-\gamma/2}^{\mu+\gamma/2} \widehat{x}(\xi) \cdot \widehat{M}(f) * \delta(f-\mu)\mathrm{d}\xi \\ = & \int_{\mu-\gamma/2}^{\mu+\gamma/2} \widehat{x}(\xi) \cdot \widehat{M}(f-\mu)\mathrm{d}\xi. \end{aligned} \tag{11.43}$$

where the last step follows from $\widehat{M}(f) * \delta(f-\mu) = \widehat{M}(f-\mu)$.

By Lemma 11.82 (with $\gamma = \gamma/2$, $\sigma = 1/(2\pi\sigma')$, $\mu = 0$, $c = -2\pi\mu'$), we have that

$$\begin{aligned} & |\widehat{M}(f-\xi) - \widehat{M}(f-\mu)| \\ = & \sqrt{2\pi\sigma'^2}|M_{0,1/(4\pi^2\sigma'^2)}(f-\xi)\exp(-2\pi\mathbf{i}\mu'(f-\xi)) - M_{0,1/(4\pi^2\sigma'^2)}(f-\mu)\exp(-2\pi\mathbf{i}\mu'(f-\mu))| \\ \leq & \sqrt{2\pi\sigma'^2}(2\pi\sigma' + 2\pi|\mu'|)\gamma \\ \lesssim & (\sigma'^2 + \sigma'|\mu'|)\gamma \end{aligned} \tag{11.44}$$

where the first step from the calculation of $\widehat{M}$, the second step follows from Lemma 11.82.

Thus, we can claim that $\forall f \in \mathbb{R}$,

$$\begin{aligned} & |\widehat{M}(f) * (\widehat{x}(f) \cdot \mathrm{rect}_{\gamma/2}(f-\mu)) - \int_{\mu-\gamma/2}^{\mu+\gamma/2} \widehat{x}(f)\mathrm{d}f \cdot \widehat{M}(f) * \delta(f-\mu)| \\ = & |\int_{\mu-\gamma/2}^{\mu+\gamma/2} \widehat{x}(\xi) \cdot \widehat{M}(f-\xi)\mathrm{d}\xi - \int_{\mu-\gamma/2}^{\mu+\gamma/2} \widehat{x}(\xi) \cdot \widehat{M}(f-\mu)\mathrm{d}\xi| \\ = & |\int_{\mu-\gamma/2}^{\mu+\gamma/2} \widehat{x}(\xi) \cdot (\widehat{M}(f-\xi) - \widehat{M}(f-\mu))\mathrm{d}\xi| \\ \leq & \int_{\mu-\gamma/2}^{\mu+\gamma/2} |\widehat{x}(\xi)| \cdot |\widehat{M}(f-\xi) - \widehat{M}(f-\mu)|\mathrm{d}\xi \\ \leq & \max_{\xi \in [\mu-\gamma/2, \mu+\gamma/2]} \{|\widehat{M}(f-\xi) - \widehat{M}(f-\mu)|\} \cdot \int_{\mu-\gamma/2}^{\mu+\gamma/2} |\widehat{x}(\xi)|\mathrm{d}\xi \\ \lesssim & (\sigma'^2 + \sigma'|\mu'|)\gamma \cdot \int_{\mu-\gamma/2}^{\mu+\gamma/2} |\widehat{x}(\xi)|\mathrm{d}\xi. \end{aligned} \tag{11.45}$$

where the first step follows from from Eq. (11.42), Eq. (11.43), the third step follows from triangle inequality, and the last step follows from Eq. (11.44).

As a result, we have that for f

$$\begin{aligned}
&|\widehat{M} * \widehat{x}(f) - \sum_{\mu \in \mathbb{Z}\gamma} \int_{\mu-\gamma/2}^{\mu+\gamma/2} \widehat{x}(\xi) \mathrm{d}\xi \cdot \widehat{M}(f) * \delta(f-\mu)| \\
\leq & | \sum_{\mu \in \mathbb{Z}\gamma} \widehat{M}(f) * (\widehat{x}(f) \cdot \mathrm{rect}_{\gamma/2}(f-\mu)) - \sum_{\mu \in \mathbb{Z}\gamma} \int_{\mu-\gamma/2}^{\mu+\gamma/2} \widehat{x}(\xi) \mathrm{d}\xi \cdot \widehat{M}(f) * \delta(f-\mu)| \\
\leq & \sum_{\mu \in \mathbb{Z}\gamma} |\widehat{M}(f) * (\widehat{x}(f) \cdot \mathrm{rect}_{\gamma/2}(f-\mu)) - \int_{\mu-\gamma/2}^{\mu+\gamma/2} \widehat{x}(f) \mathrm{d}f \cdot \widehat{M}(f) * \delta(f-\mu)| \\
\lesssim & (\sigma'^2 + \sigma'|\mu'|)\gamma \cdot \sum_{\mu \in \mathbb{Z}\gamma} \int_{\mu-\gamma/2}^{\mu+\gamma/2} |\widehat{x}(\xi)| \mathrm{d}\xi \\
\leq & (\sigma'^2 + \sigma'|\mu'|)\gamma \cdot \|\widehat{x}\|_1
\end{aligned}$$

where the first step follows from $\sum_{\mu \in \mathbb{Z}\gamma} \mathrm{rect}_{\gamma/2}(f-\mu) = 1$, the second step follows from triangle inequality, the third step follows from Eq. (11.45), the last step follows from the definition of ℓ_1 norm.

Thus we complete proof. $\square$

Choice of parameters The following lemma shows how to take the parameters in this section.

Lemma 11.85. *Let $\varepsilon_0, \varepsilon, F, c_0, \sigma', T, F_0, K, \varepsilon_1 \in \mathbb{R}_+$ such that*

- $\varepsilon_0 = 0.01$
- $\varepsilon_1 \leq \varepsilon^2/T$
- $\sigma' = \sqrt{2}\varepsilon_0^2 T$
- $c_0 = 0.01/\sqrt{\log(1/\varepsilon_0)}$

- $K = \lceil \frac{1+0.5\varepsilon_0}{c_0\varepsilon_0^2} \rceil \leq 2/(c_0\varepsilon_0^2)$
- $F_0 = \sigma'^{-1} \cdot \log^{1/2}(4F/\varepsilon_1) + F$

We have that

- **Part 1.** $c_0 K \lesssim 1$
- **Part 2.** $F_0 \lesssim (1/T) \cdot \log^{1/2}(FT/\varepsilon) + F$

Proof. We will prove them separately.

Part 1.

We can show

$$c_0 K = c_0 \lceil \frac{1+0.5\varepsilon_0}{c_0\varepsilon_0^2} \rceil \leq \frac{2c_0}{c_0\varepsilon_0^2} \lesssim 1.$$

where the first step follows from the definition of K, the third step follows from the definition of ε_0.

Part 2.

We have that

$$\begin{aligned} F_0 &= \sigma'^{-1} \cdot \log^{1/2}(4F/\varepsilon_1) + F \\ &\lesssim 1/(\varepsilon_0^2 T) \cdot \log^{1/2}(F/\varepsilon_1) + F \\ &\lesssim 1/T \cdot \log^{1/2}(F/\varepsilon_1) + F \\ &\lesssim 1/T \cdot \log^{1/2}(FT/\varepsilon^2) + F \\ &\lesssim 1/T \cdot \log^{1/2}(FT/\varepsilon) + F \end{aligned}$$

where the second step follows from the definition of σ', the third step follows from the definition of ε_0, the fourth step follows from the definition of $\varepsilon_1 \geq \varepsilon^2/T$.

$\square$

11.13.3 Semi-continuouse approximation of Fourier-sparse signals

The main theorem of this section is stated and proved below.

Theorem 11.86 (Sparse Signal is Semi-continuous)**.** *Given $\gamma, \varepsilon \in (0, 0.1), F, T \in \mathbb{R}_+$. Let $x : \mathbb{R} \to \mathbb{C}$ be a function that such that x is k-Fourier-sparse and* $\mathrm{supp}(\widehat{x}) \subseteq$ *$[-F, F]$. Let $\gamma > 0$ and $\varepsilon_1 > 0$. Then there is an algorithm output a k'-Fourier-sparse signal ($k' \leq k$),*

$$x'(t) = \sum_{i=1}^{k'} v_i \exp(2\pi \mathbf{i} f_i t)$$

such that

$$\|x' - x\|_T^2 \lesssim (F_0 T^3 \gamma^2 + \varepsilon_1/(F^2 T)) \cdot \|\widehat{x}\|_1^2$$

where $\gamma = \min_{i \neq j} |f_i - f_j|$, $f_i \in [-F, F]$, $f_i \in \gamma\mathbb{Z}$, $\forall i \in [k']$ and $F_0 = \Omega(T^{-1} \log^{1/2}(F/\varepsilon_1) + F)$.

Further, if $\varepsilon_1 \leq \varepsilon^2/T$ and $\gamma \leq \varepsilon/\sqrt{F_0 T^3}$, then we have

$$\|x' - x\|_T^2 \lesssim \varepsilon^2 \|\widehat{x}\|_1^2$$

Proof. First, we will introduce our choice for x'. Then, we bound $|\widehat{M_j} * \widehat{x} - \widehat{M_j} * \widehat{x}'|$. Finally, show the relationship between $|\widehat{M_j} * \widehat{x} - \widehat{M_j} * \widehat{x}'|$ and the LHS in our statement by utilizing M to bridge the integral calculated in the time domain $\|x' - x\|_T$ and the integral calculated in the frequency domain $\|\widehat{x}\|_1$.

Let x' be chosen as

$$x'(t) = \sum_{\mu \in \mathbb{Z}\gamma} \int_{\mu - \gamma/2}^{\mu + \gamma/2} \widehat{x}(f) \mathrm{d}f \cdot \exp(2\pi \mathbf{i} \mu t).$$

Then,

$$\|\widehat{x'}\|_1 = \sum_{\mu \in \mathbb{Z}\gamma} \int_{\mu - \gamma/2}^{\mu + \gamma/2} \widehat{x}(f) \mathrm{d}f = \int_{-\infty}^{\infty} \widehat{x}(f) \mathrm{d}f = \|\widehat{x}\|_1. \tag{11.46}$$

Let $c_0, \varepsilon_0 \in \mathbb{R}_+$, $K \in \mathbb{Z}_+$ such that $\varepsilon_0 = 0.01 \in (0, 0.1),\ c_0 = 0.01/\sqrt{\log(1/\varepsilon_0)},\ K = \lceil \frac{1+0.5\varepsilon_0}{c_0\varepsilon_0^2} \rceil \leq 2/(c_0\varepsilon_0^2)$.

For simplicity, let $M_j(t) = M_{\mu_j', \sigma'^2}(t) = M_{c_0 j \varepsilon_0^2 T, 2\varepsilon_0^4 T^2}(t)$, then

$$\widehat{M_j}(f) = \sqrt{2\pi\sigma'^2} \exp(-2\pi^2\sigma'^2 f^2) \exp(-2\pi \mathbf{i} \mu_j' f).$$

and

$$\begin{aligned}
\mu_j' &\leq c_0 K \varepsilon_0^2 T \\
&\leq c_0 (\frac{2}{c_0\varepsilon_0^2}) \varepsilon_0^2 T \\
&= 2T.
\end{aligned}$$

where the second step follows from upper bound on K.

We have that

$$\widehat{x}'(f) = \sum_{\mu \in \mathbb{Z}\gamma} \int_{\mu-\gamma/2}^{\mu+\gamma/2} \widehat{x}(f) \mathrm{d}f \cdot \delta(f - \mu).$$

So, we can convolute $\widehat{M}$ at the both sides and get

$$\widehat{M_j}(f) * \widehat{x}'(f) = \sum_{\mu \in \mathbb{Z}\gamma} \int_{\mu-\gamma/2}^{\mu+\gamma/2} \widehat{x}(f) \mathrm{d}f \cdot \widehat{M_j}(f) * \delta(f - \mu). \tag{11.47}$$

By Lemma 11.84, we have that for $\forall f \in \mathbb{R}$,

$$\begin{aligned}
&|\widehat{M_j} * \widehat{x} - \widehat{M_j} * \widehat{x}'| \\
= &\Big|\widehat{M_j} * \widehat{x} - \sum_{\mu \in \mathbb{Z}\gamma} \int_{\mu-\gamma/2}^{\mu+\gamma/2} \widehat{x}(f) \mathrm{d}f (\widehat{M_j}(f) * \delta(f - \mu))\Big| \\
\leq &(\sigma'^2 + \sigma'|\mu_j'|)\gamma \cdot \|\widehat{x}\|_1 \\
\lesssim &T^2 \gamma \cdot \|\widehat{x}\|_1.
\end{aligned} \tag{11.48}$$

where the first step follows from Eq. (11.47), the second step follows from Lemma 11.84, the third step follows the choice of μ_j' and σ and that ε_0 is a constant.

Let F_0 be defined as

$$F_0 > \sigma'^{-1} \cdot \log^{1/2}(4F/\varepsilon_1) + F.$$

Next, we can bound $|x'(t) - x(t)|^2$,

$$\begin{aligned}
\int_{-\infty}^{\infty} M_j^2(t) \cdot |x'(t) - x(t)|^2 \mathrm{d}t = & \int_{-\infty}^{\infty} |\widehat{M}_j * \widehat{x}'(f) - \widehat{M}_j * \widehat{x}(f)|^2 \mathrm{d}f \\
\leq & \int_{-F_0}^{F_0} |\widehat{M}_j * \widehat{x}'(f) - \widehat{M}_j * \widehat{x}(f)|^2 \mathrm{d}f + \varepsilon_1 \|\widehat{x}'(f) - \widehat{x}(f)\|_\infty^2 \\
\leq & \int_{-F_0}^{F_0} |\widehat{M}_j * \widehat{x}'(f) - \widehat{M}_j * \widehat{x}(f)|^2 \mathrm{d}f + 4\varepsilon_1 \|\widehat{x}\|_\infty^2 \\
\lesssim & \ 2F_0 T^4 \gamma^2 \cdot \|\widehat{x}\|_1^2 + 4\varepsilon_1 \|\widehat{x}\|_\infty^2 \\
\lesssim & \ \underbrace{(F_0 T^4 \gamma^2 + \varepsilon_1 / F^2) \cdot \|\widehat{x}\|_1^2}_{\text{err}} \qquad (11.49)
\end{aligned}$$

where the second step follows from Lemma 11.83, the third step follows from $\|\widehat{x}'(f) - \widehat{x}(f)\|_\infty^2 \leq (\|\widehat{x}'(f)\|_\infty + \|\widehat{x}(f)\|_\infty)^2 = (2\|\widehat{x}(f)\|_\infty)^2 = 4\|\widehat{x}\|_\infty^2$ due to Eq. (11.46), the forth step follows from Eq. (11.48), the fifth step follows from $\|\widehat{x}\|_\infty \leq \|\widehat{x}\|_1$.

Then we can upper bound the LHS $\int_0^T |x'(t) - x(t)|^2 \mathrm{d}t$ as follows:

$$\begin{aligned}
& \int_0^T |x'(t) - x(t)|^2 \mathrm{d}t \\
= & \int_0^T \mathrm{rect}_T(t) |x'(t) - x(t)|^2 \mathrm{d}t \\
\leq & \int_0^T \sum_{j=-K}^{K} (\frac{c_0}{(1 - 2\varepsilon_0)\sqrt{2\pi}}) M_j^2(t) \cdot |x'(t) - x(t)|^2 \mathrm{d}t \\
\leq & \int_0^T \sum_{j=-K}^{K} c_0 M_j^2(t) \cdot |x'(t) - x(t)|^2 \mathrm{d}t \\
\leq & \ c_0 \cdot \sum_{j=-K}^{K} \int_{-\infty}^{\infty} M_j^2(t) \cdot |x'(t) - x(t)|^2 \mathrm{d}t \\
\lesssim & \ c_0 K \cdot \text{err} \\
\lesssim & \ \text{err}
\end{aligned}$$

where the second step follows from Claim 11.81 and $M_j^2(t) = M_{c_0 j \varepsilon_0^2 T, \varepsilon_0^4 T^2}(t)$, the third step follows from $\varepsilon_0 = 0.01$, the forth step follows from relaxing integral range, the fifth step follows from Eq. (11.49), the last step follows from $c_0 K \lesssim 1$ due to Lemma 11.85.

□

11.13.4 Fast optimal-sparsity Fourier sparse recovery

Corollary 11.87 (Our result). *For any $F > 0, T > 0, \varepsilon > 0$. Let $x^*(t) = \sum_{j=1}^k v_j \exp(2\pi \mathbf{i} f_j t)$ with $|f_j| \leq F$ for $j \in [k]$. For observation $x(t) = x^*(t) + g(t)$, there exists an algorithm that takes*

$$m = O(\varepsilon^{-1} k^2 \log^3(k) \log(FT/(\delta\rho)))$$

random samples $t_1, \ldots, t_m \in [0, T]$, runs in $(\varepsilon^{-1} FT)^{O(k)}$ time, and outputs $y(t) = \sum_{j=1}^k \widetilde{v}_j \exp(2\pi \mathbf{i} \widetilde{f}_j t)$ such that

$$\|y(t) - x^*(t)\|_T \leq (1 + \varepsilon)\|g(t)\|_T + \delta \|\widehat{x}^*(f)\|_1,$$

holds with probability $1 - \rho$.

Proof. Let $N_f = O(\frac{\delta}{T\sqrt{FT \log(1/\delta)}}) \cdot \mathbb{Z} \cap [-F, F]$ denote a net of frequencies. Because

$$\begin{aligned}
\frac{\delta}{T\sqrt{F_0 T}} &\gtrsim \frac{\delta}{T\sqrt{\log^{1/2}(F/\varepsilon_1) + FT}} \\
&\geq \frac{\delta}{T\sqrt{\log^{1/2}(FT/\delta^2) + FT}} \\
&\geq \frac{\delta}{T\sqrt{\log(FT) + \log(1/\delta) + FT}} \\
&\geq \frac{\delta}{T\sqrt{FT \log(1/\delta)}}
\end{aligned}$$

where the first step follows from $F_0 = \Omega(T^{-1} \log^{1/2}(F/\varepsilon_1) + F)$, the second step follows from setting $\varepsilon_1 = \delta^2/T$.

By Theorem 11.86, for any signal $x^*(t) = \sum_{j=1}^k v_j \exp(2\pi \mathbf{i} f_j t)$, there exists a k-Fourier-sparse signal $\widetilde{x}(t) = \sum_{j=1}^k v_j' \exp(2\pi \mathbf{i} f_j' t)$ such that,

$$\|x^*(t) - \widetilde{x}(t)\|_T \leq \delta \|\widehat{x}^*(f)\|_1$$

and $f_1', \cdots, f_k' \subseteq N_f$.

Because we have that

$$\begin{aligned}
\|y(t) - x^*(t)\|_T &\leq \|y(t) - \widetilde{x}(t)\|_T + \|\widetilde{x}(t) - x^*(t)\|_T \\
&\leq \|y(t) - \widetilde{x}(t)\|_T + \delta \|\widehat{x}^*(f)\|_1.
\end{aligned}$$

Let S be the set of i.i.d samples from $D(t)$ of size $O(\varepsilon^{-1} k \log^3(k) \log(1/\rho_0))$, w be the corresponding weight in Algorithm 63 Procedure SparseFT line 3. By Lemma 11.28, we have that, for any $\mathcal{F}$, with probability at least $1 - \rho_0$,

$$(1 - \sqrt{\varepsilon}) \|x\|_T \leq \|x\|_{S,w} \leq (1 + \sqrt{\varepsilon}) \|x\|_T.$$

In total, we enumerate $(\delta^{-1} F T)^{O(k)}$ function family $\mathcal{F}$ in Algorithm 63 Procedure SparseFT line 6. By taking $\rho_0 = \rho(\delta (FT)^{-1})^{O(k)}$, we have that the total success probability is at least

$$(1 - \rho_0)^{(\delta^{-1} F T)^{O(k)}} \geq 1 - (\delta^{-1} F T)^{O(k)} \cdot \rho_0 \geq 1 - \rho$$

Thus, by Lemma 11.66, with probability at least $1 - \rho$, sampling S and w forms a ε-WBSP for every $\mathcal{F}$.

Finally, we bound $\|y(t) - \widetilde{x}(t)\|_T$ as follows,

$$\begin{aligned}
\|y(t) - \widetilde{x}(t)\|_T &\leq (1 + O(\varepsilon)) \|x(t) - \widetilde{x}(t)\|_T \\
&\leq (1 + O(\varepsilon)) (\|x(t) - x^*(t)\|_T + \|x^*(t) - \widetilde{x}(t)\|_T) \\
&\leq (1 + O(\varepsilon)) (\|g(t)\|_T + \delta \|\widehat{x}^*(f)\|_1)
\end{aligned}$$

where the first step follows from the proof of Theorem 11.60.

Combine the results above we have that

$$\begin{aligned} \|y(t) - x^*(t)\|_T &\leq \|y(t) - \widetilde{x}(t)\|_T + \delta \|\widehat{x}^*(f)\|_1 \\ &\leq (1 + O(\varepsilon))\|g(t)\|_T + O(\delta)\|\widehat{x}^*(f)\|_1. \end{aligned}$$

Note that our sample complexity is $|S| = O(\varepsilon^{-1} k \log^3(k) \log(1/\rho_0)) = O(\varepsilon^{-1} k^2 \log^3(k) \log(FT/(\delta$

□

Lemma 11.88 (Running time of Lemma 11.87)**.** *Procedure* SparseFT *in Algorithm 63 runs in* $O((\delta^{-1}FT)^{O(k)} \log(1/\rho))$ *times.*

Proof. In each call of the Procedure SparseFT in Algorithm 63,

- In the for loop, it repeats the line 8 for $(\delta^{-1}\log^{0.5}(1/\delta)(FT)^{1.5})^k$ times.
- Note that each 8 of Procedure SparseFT in Algorithm 63 is solving linear regression. This part takes $O(\varepsilon^{-1} k^{\omega+1} \log^3(k) \log(FT/(\delta\rho)))$ time.

So, the time complexity of Procedure SparseFT in Algorithm 63 is

$$O((\delta^{-1}\log^{0.5}(1/\delta)(FT)^{1.5})^k) \cdot O(\varepsilon^{-1} k^{\omega+1} \log^3(k) \log(FT/(\delta\rho))) = O((\varepsilon^{-1}FT)^{O(k)} \log(1/\rho)).$$

□

11.13.5 Semi-continuous approximation with a constant frequency gap

In this section, we show that the semi-continuous approximation result in previous section can be further improved in terms of the frequency gap.

We first consider the one-sparse case in the following lemma.

Lemma 11.89. *Let $0 < \delta < 0.1$ be a parameter. Let $x^* = v^* \exp(2\pi \mathbf{i} f^* t)$ be a function such that $f^* \in [-c/T, c/T]$. Then, there exists $k > \log(1/\delta)$, $F \lesssim k/T$, $f_1, \dots, f_k \in c/T\mathbb{Z} \cap [-F, F]$, and $v_1, \dots, v_k \in \mathbb{C}$ and for the function*

$$\widetilde{x}(t) = \sum_{j=1}^{k} v_j \exp(2\pi \mathbf{i} f_j t)$$

we have that

$$\|x^* - \widetilde{x}\|_T \lesssim \delta |v^*|$$

Proof. Note that $x^*(t)$ can be written as

$$x^*(t) = \int_{-F}^{F} \widehat{x}^*(f) \exp(2\pi \mathbf{i} f t) \mathrm{d} f.$$

Now consider the Taylor expansion of

$$\exp(2\pi \mathbf{i} f t) = \sum_{j=0}^{\infty} \frac{(2\pi \mathbf{i} f t)^j}{j!}$$

Note that since $f \in [-F, F]$, $t \in [0, T]$, $FT \lesssim k$, we have

$$\begin{aligned} \sum_{j=k+1}^{\infty} \left|\frac{(2\pi \mathbf{i} f t)^j}{j!}\right| \leq & \sum_{j=k+1}^{\infty} \left|\frac{(2e\pi \mathbf{i} f t)^j}{j^j}\right| \\ \leq & \sum_{j=k+1}^{\infty} \frac{1}{\exp(j)} \\ \leq & \frac{1}{\exp(k)} \cdot \frac{1}{1 - 1/e} \\ \lesssim & \delta_1 \end{aligned} \tag{11.50}$$

where the first step follows from Stirling's formula $n! \geq \sqrt{2\pi} n^{n+0.5} \exp(-n)$, the second step follows from $2e^2 \pi f \geq j$, the last step follows from $k \geq \log(1/\delta_1)$.

In particular, if we define the approximator of exponential function

$$g_f(x) = \sum_{j=0}^{k} \frac{(2\pi \mathbf{i} f t)^j}{j!}$$

then over the interval $t \in [0,T]$, $f \in [-F,F]$, $FT \lesssim k$,

$$\begin{aligned} |\exp(2\pi \mathbf{i} f t) - g_f(t)| &= |\sum_{j=k+1}^{\infty} \frac{(2\pi \mathbf{i} f t)^j}{j!}| \\ &\leq \sum_{j=k+1}^{\infty} |\frac{(2\pi \mathbf{i} f t)^j}{j!}| \\ &\leq \delta_1. \end{aligned} \tag{11.51}$$

where the first step follows from the definition of $g_f(x)$, the second step follows from the triangle inequality, the last step follows from Eq. (11.50).

Next, let $\mathcal{V}_k(f) = (1, f, \cdots, f^k)$. For $f^* \in [-c/T, c/T]$, we can write the vector

$$\mathcal{V}_k(f^*) = w_1 \mathcal{V}_k(f_1) + \cdots + w_k \mathcal{V}_k(f_k)$$

for some real numbers (depending on f^*) $w_1, \ldots, w_k$, because $\mathcal{V}_k(f_1), \cdots, \mathcal{V}_k(f_k)$ is linear independence. Thus,

$$g_{f^*}(t) = w_1 g_{f_1}(t) + \cdots + w_k g_{f_k}(t) \tag{11.52}$$

for the same weights. Now note that by (11.51), for all $t \in [0,T]$,

$$\begin{aligned} |x^*(t) - v^* g_{f^*}(t)| &= |v^*(g_{f^*}(t) - \exp(2\pi \mathbf{i} f^* t))| \\ &\leq |v^*| \delta_1. \end{aligned}$$

Then, because $g_f(t)$ can be expressed by $g_{f_j}(t), j \in [k]$, we have that

$$|x^*(t) - \sum_{j=0}^{k} v^* g_{f_j}(t) w_j| = |x^*(t) - v^* g_{f^*}(t)|$$

which is follows from Eq. (11.52).

Note that

$$\sum_{j=1}^{k} |w_j| \leq C_1$$

So by Eq. (11.51), we will transform the approximator of exponential function back to exponential function. For $\forall t \in [0,T]$,

$$|\sum_{j=1}^{k}(g_{f_j}(t) - \exp(2\pi \mathbf{i} f_j t))(v^* w_j)| \le C_1 |v^*| \delta_1$$

Therefore, we can conclude that for $\forall t \in [0,T]$

$$|x^*(t) - \sum_{j=1}^{k}(v^* w_j) \exp(2\pi \mathbf{i} f_j t)| \le C_1 |v^*| \delta_1$$

and setting

$$h(x) = \sum_{j=1}^{k}(v^* w_j) \exp(2\pi \mathbf{i} f_j t)$$

immediately leads to the desired conclusion. $\square$

Lemma 11.89 immediately gives the following corollary by taking linear summation over k frequencies.

Corollary 11.90. *Let $0 < \delta < 0.1$ be a parameter. Let x^* be any k-Fourier-sparse signal. Then, there exists $\widetilde{k} \lesssim k \log(k/\delta)$, universal constant $c \in (0,1)$, $f_1, \cdots, f_k \in c/T\mathbb{Z}$, and $v_1, \ldots, v_k \in \mathbb{C}$ and for the function*

$$\widetilde{x}(t) = \sum_{j=1}^{\widetilde{k}} v_j \exp(2\pi \mathbf{i} f_j t)$$

we have that

$$\|x^* - \widetilde{x}\|_T \lesssim \delta \|\widehat{x}^*(f)\|_1$$

Algorithm 55 Quadratic-form sampling with preprocessing-query trade-off: Preprocessing

```
1: structure Node
2:     $V_1, V_2 \in \mathbb{R}^{d\times d}$
3:     left, right                                  ▷ Point to the left/right child in the tree
4: end structure
5: data structure DS+                               ▷ Theorem 11.40
6: members
7:     $n \in \mathbb{N}$                           ▷ The number of vectors
8:     $m \in \mathbb{N}$                           ▷ The number of blocks
9:     $v_1, \dots, v_n \in \mathbb{R}^d$           ▷ $d$-dimensional vectors
10:    root: Node                                   ▷ The root of the tree
11: end members
12: procedure BuildTree($l, r$)                     ▷ $[l, \dots, r]$ is the range of the current node
13:     p ← new Node
14:     if $l = r$ then                              ▷ Leaf node
15:         p.$V_2 \leftarrow \begin{bmatrix} v_{(l-1)d+1} & \cdots & v_{ld} \end{bmatrix}$
16:         p.$V_1 \leftarrow (\mathsf{p}.V_2) \cdot (\mathsf{p}.V_2)^\top$     ▷ It takes $O(d^\omega)$-time
17:                                                  ▷ p.mat1 $= \sum_{i=(l-1)d+1}^{ld} v_i v_i^\top$
18:     else                                         ▷ Internal node
19:         $mid \leftarrow \lfloor (l+r)/2 \rfloor$
20:         p.left ← BuildTree($l, mid$)
21:         p.right ← BuildTree($mid+1, r$)
22:         p.$V_1 \leftarrow$ (p.left).$V_1$ + (p.right).$V_1$              ▷ It takes $O(d^2)$-time
23:     end if
24:     return p
25: end procedure
26: procedure Init($n, d, \{v_i\}_{i\in[n]} \subseteq \mathbb{R}^d, \{\alpha_i\}_{i\in[n]} \subseteq \mathbb{R}$)
27:     $v_i \leftarrow v_i \cdot \sqrt{\alpha_i}$ for $i \in [n]$
28:     $m \leftarrow n/d$                           ▷ We assume that $n$ is divisible by $d$
29:     Group $\{v_i\}_{i\in[n]}$ into $m$ blocks $B_1, \dots, B_m$      ▷ $B_i = \{v_{(i-1)d+1}, \dots, v_{id}\}$ for $i \in [m]$
30:     root ← BuildTree($1, m$)
31: end procedure
32: end data structure
```

Algorithm 56 Quadratic-form sampling with preprocessing-query trade-off: Query

```
1: data structure DS+                                              ▷ Theorem 11.40
2: members
3:     $n \in \mathbb{N}$                                          ▷ The number of vectors
4:     $m \in \mathbb{N}$                                          ▷ The number of blocks
5:     $v_1, \dots, v_n \in \mathbb{R}^d$                          ▷ $d$-dimensional vectors
6:     root: Node                                                  ▷ The root of the tree
7: end members
8: procedure BlockSampling($\mathsf{p}$, $l \in \mathbb{N}$, $A \in \mathbb{R}^{d \times d}$)  ▷ $\mathsf{p}$ is a leaf node with index $l$
9:     $U \leftarrow (\mathsf{p}.V_2)^\top \cdot A \cdot (\mathsf{p}.V_2)$     ▷ It takes $O(d^\omega)$-time
10:    Define a distribution $\mathcal{D}_l$ over $[d]$ such that $\mathcal{D}_l(i) \propto U_{i,i}$
11:    Sample $i \in [d]$ from $\mathcal{D}_l$                     ▷ It takes $O(d)$-time
12:    return $(l-1)d + i$
13: end procedure
14: procedure Query($A \in \mathbb{R}^{d \times d}$)
15:    $\mathsf{p} \leftarrow \mathsf{root}$, $l \leftarrow 1$, $r \leftarrow m$
16:    $s \leftarrow 0$
17:    while $l \neq r$ do                                         ▷ There are $O(\log m)$ iterations
18:        $w \leftarrow \langle \mathsf{p}.V_1, A \rangle$            ▷ It takes $O(d^2)$-time
19:        $w_\ell \leftarrow \langle (\mathsf{p}.\mathrm{left}).V_1, A \rangle$
20:        Sample $c$ from Bernoulli$(w_\ell / w)$
21:        if $c = 0$ then
22:            $\mathsf{p} \leftarrow \mathsf{p}.\mathrm{left}$, $r \leftarrow \lfloor (l+r)/2 \rfloor$
23:        else
24:            $\mathsf{p} \leftarrow \mathsf{p}.\mathrm{right}$, $l \leftarrow \lfloor (l+r)/2 \rfloor + 1$
25:        end if
26:    end while
27:    return BlockSampling($\mathsf{p}$, $l$, $A$)
28: end procedure
29: end data structure
```

Algorithm 57 Fast distillation for one-dimensional signal

```
1: procedure WeightedSketch($k, \varepsilon, T, \mathcal{B}$)                ▷ Lemma 11.28
2:     $c \leftarrow O(T^{-1}\log^{-1}(k))$
3:     $D(t)$ is defined as follows:
           $$D(t) \leftarrow \begin{cases} c/((1-|t/T|)\log k), & \text{if } |t| \le T(1-1/k), \\ c \cdot k, & \text{if } |t| \in [T(1-1/k), T]. \end{cases}$$
4:     $S_0 \leftarrow O(\varepsilon^{-2} k \log(k))$ i.i.d. samples from $D$
5:     for $t \in S_0$ do
6:         $w_t \leftarrow \frac{2}{T \cdot |S_0| \cdot D(t)}$
7:     end for
8:     Set a new distribution $D'(t) \leftarrow w_t / \sum_{t' \in S_0} w_{t'}$ for all $t \in S_0$
9:     return $D'$
10: end procedure
11: procedure FastDistill1D($k$, $\varepsilon$, $F = \{f_1, \ldots, f_k\}$, $T$)        ▷ Lemma 11.44
12:     Distribution $D' \leftarrow$ WeightedSketch($k, \varepsilon, T, \mathcal{B}$)
13:     Set the function family $\mathcal{F}$ as follows:
           $$\mathcal{F} := \Big\{ f(t) = \sum_{j=1}^{k} v_j \exp(2\pi \mathbf{i} f_j t) \ \Big| \ v_j \in \mathbb{C} \Big\}.$$
14:     $s, \{t_1, t_2, \cdots, t_s\}, w \leftarrow$ RandBSS+($k, \mathcal{F}, D', (\varepsilon/4)^2$)        ▷ $s = O(k/\varepsilon^2)$, Algorithm 52
15:     return $\{t_1, t_2, \cdots, t_s\}$ and $w$
16: end procedure
```

Algorithm 58 Distillation for high-dimensional signal.

```
1: procedure DistillHD($k, \varepsilon, d, F = \{f_1, \ldots, f_k\}, T$)        ▷ Lemma 11.51
2:     $S_0 \leftarrow O(\varepsilon^{-2} k^{O(d)} \log(1/\varepsilon))$ i.i.d. samples from Uniform($[0, T]^d$)
3:     Set the function family $\mathcal{F}$ as follows:
           $$\mathcal{F} = \Big\{ f(t) = \sum_{j=1}^{k} v_j \exp(2\pi \mathbf{i} \langle f_j, t \rangle) \ \Big| \ v_j \in \mathbb{C} \Big\}.$$
4:     $s, \{t_1, t_2, \cdots, t_s\}, w \leftarrow$ RandBSS+($k, \mathcal{F}$, Uniform($S_0$), $(\varepsilon/4)^2$)  ▷ $s = O(k/\varepsilon^2)$, Algorithm 52
5:     return $\{t_1, t_2, \cdots, t_s\}$ and $w$
6: end procedure
```

Algorithm 59 Distillation for discrete signal.

```
1: procedure DistillDisc($k, \varepsilon, F = \{f_1, \cdots, f_k\}, n$)    ▷ Lemma 11.54 (one-dimension)
2:     $S_0 \leftarrow O(\varepsilon^{-2} k \log(k))$ i.i.d. samples from Uniform($[n]$)
3:     Set the function family $\mathcal{F}$ as follows:
           $$\mathcal{F} = \{f(t) = \sum_{j=1}^{k} v_j \exp(2\pi \mathbf{i} f_j t/n) | v_j \in \mathbb{C}\}.$$
4:     $s, \{t_1, t_2, \cdots, t_s\}, w \leftarrow$ RandBSS+($k, \mathcal{F}$, Uniform($S_0$), $(\varepsilon/4)^2$)    ▷ $s = O(k/\varepsilon^2)$, Algorithm 52
5:     return $\{t_1, t_2, \cdots, t_s\}$ and $w$
6: end procedure
7: procedure DistillDiscHD($k, \varepsilon, F = \{f_1, \cdots, f_k\}, p, d$)    ▷ Lemma 11.54 (high-dimension)
8:     $S_0 \leftarrow O(\varepsilon^{-2} k \log(k))$ i.i.d. samples from Uniform($[p]^d$)
9:     $A \leftarrow \begin{bmatrix} {t'_1}^\top \\ \vdots \\ {t'_{s_0}}^\top \end{bmatrix} \in \mathbb{R}^{s_0 \times d}$, $B \leftarrow \begin{bmatrix} f_1 & \cdots & f_k \end{bmatrix} \in \mathbb{R}^{d \times k}$
10:    $C \leftarrow A \cdot B \in \mathbb{R}^{s_0 \times k}$
11:    $\mathcal{F}_{ij} \leftarrow \exp(2\pi \mathbf{i} C_{ij})$ for each $(i, j) \in [s_0] \times [k]$
12:    $s, \{t_1, t_2, \cdots, t_s\}, w \leftarrow$ RandBSS+($k, \mathcal{F}$, Uniform($S_0$), $(\varepsilon/4)^2$)    ▷ $s = O(k/\varepsilon^2)$, Algorithm 52
13:    return $\{t_1, t_2, \cdots, t_s\}$ and $w$
14: end procedure
```

Algorithm 60 Signal estimation algorithm for one-dimensional signals (sample optimal version)

```
1: procedure SignalEstimationFast($x, k, F, T, \mathcal{B}$)    ▷ Theorem 11.57
2:     $\varepsilon \leftarrow 0.01$
3:     $L \leftarrow$ FreqEst($x, k, D, F, T, \mathcal{B}$)
4:     $\{f_1', f_2', \cdots, f_{\widetilde{k}}'\} \leftarrow \{f \in \Lambda(\mathcal{B}) \mid \exists f' \in L,\ |f' - f| < D/T\}$
5:     $s, \{t_1, t_2, \cdots, t_s\}, w \leftarrow$ FastDistill1D($\widetilde{k}, \sqrt{\varepsilon}, \{f_i'\}_{i \in [\widetilde{k}]}, T, \mathcal{B}$)    ▷ $\widetilde{k}$, $w \in \mathbb{R}^{\widetilde{k}}$, Algorithm 57
6:     $A_{i,j} \leftarrow \exp(2\pi \mathbf{i} f_j' t_i)$, $A \in \mathbb{C}^{s \times \widetilde{k}}$
7:     $b \leftarrow (x(t_1), x(t_2), \cdots, x(t_s))^\top$
8:     Solving the following weighted linear regression    ▷ Fact 11.8
           $v' \leftarrow \arg\min_{v' \in \mathbb{C}^{\widetilde{k}}} \|\sqrt{w} \circ (Av' - b)\|_2.$
9:     return $y(t) = \sum_{j=1}^{\widetilde{k}} v_j' \cdot \exp(2\pi \mathbf{i} f_j' t)$.
10: end procedure
```

Algorithm 61 Signal estimation algorithm for one-dimensional signals (high-accuracy version)

```
1: procedure SignalEstimationAcc($x, \varepsilon, k, F, T, \mathcal{B}$)    ▷ Theorem 11.60
2:     $L \leftarrow$ FreqEst($x, k, D, F, T, \mathcal{B}$)
3:     $\{f_1', f_2', \cdots, f_{\widetilde{k}}'\} \leftarrow \{f \in \Lambda(\mathcal{B}) \mid \exists f' \in L,\ |f' - f| < D/T\}$
4:     $s, \{t_1, t_2, \cdots, t_s\}, w \leftarrow$ WeightedSketch($\widetilde{k}, \sqrt{\varepsilon}, T, \mathcal{B}$)    ▷ $\widetilde{k}$, $w \in \mathbb{R}^{\widetilde{k}}$, Algorithm 57
5:     $A_{i,j} \leftarrow \exp(2\pi \mathbf{i} f_j' t_i)$, $A \in \mathbb{C}^{s \times \widetilde{k}}$
6:     $b \leftarrow (x(t_1), x(t_2), \cdots, x(t_s))^\top$
7:     Solving the following weighted linear regression    ▷ Fact 11.8
           $v' \leftarrow \arg\min_{v' \in \mathbb{C}^{\widetilde{k}}} \|\sqrt{w} \circ (Av' - b)\|_2.$
8:     return $y(t) = \sum_{j=1}^{\widetilde{k}} v_j' \cdot \exp(2\pi \mathbf{i} f_j' t)$.
9: end procedure
```

Algorithm 62 Discrete signal set-query algorithm.

```
1:  procedure SetQuery($x$, $n$, $k$, $S$, $\varepsilon$)    ▷ Theorem 11.73 (one-dimension)
2:      $\{f_1, f_2, \cdots, f_k\} \leftarrow S$
3:      $s, \{t_1, t_2, \cdots, t_s\}, w \leftarrow$ DistillDisc($k, \sqrt{\varepsilon}, F, n$)    ▷ Algorithm 59
4:      $A_{i,j} \leftarrow \exp(2\pi \mathbf{i} f_j t_i / n)$, $A \in \mathbb{C}^{s \times k}$
5:      $b \leftarrow (x(t_1), x(t_2), \cdots, x(t_s))^\top$
6:      Solving the following weighted linear regression    ▷ Fact 11.8
            $$v' \leftarrow \arg\min_{v' \in \mathbb{C}^{\widetilde{k}}} \|\sqrt{w} \circ (Av' - b)\|_2.$$
7:      return $\widehat{x}'$ such that $\widehat{x}'_{f_j} = v'_j$ for $j \in [k]$
8:  end procedure
9:  procedure SetQueryHD($x$, $n$, $k$, $S$, $\varepsilon$)    ▷ Theorem 11.73 (high-dimension)
10:     $\{f_1, f_2, \cdots, f_k\} \leftarrow S$
11:     $s, \{t_1, t_2, \cdots, t_s\}, w \leftarrow$ DistillDiscHD($k, \sqrt{\varepsilon}, F, n$)    ▷ Algorithm 59
12:     $F_{\text{batch}} = [f_1, f_2, \cdots, f_k] \in [p]^{d \times k}$
13:     $T_{\text{batch}} = [t_1, t_2, \cdots, t_s] \in [p]^{d \times s}$
14:     $U = F_{\text{batch}}^\top T_{\text{batch}} \in \mathbb{Z}^{k \times s}$    ▷ Fact 11.7
15:     $A_{i,j} \leftarrow \exp(2\pi \mathbf{i} U_{j,i} / p)$, $A \in \mathbb{C}^{s \times k}$
16:     $b \leftarrow (x(t_1), x(t_2), \cdots, x(t_s))^\top$
17:     Solving the following weighted linear regression    ▷ Fact 11.8
            $$v' \leftarrow \arg\min_{v' \in \mathbb{C}^{\widetilde{k}}} \|\sqrt{w} \circ (Av' - b)\|_2.$$
18:     return $\widehat{x}'$ such that $\widehat{x}'_{f_j} = v'_j$ for $j \in [k]$
19: end procedure
```

Algorithm 63 Recover k-sparse FT

```
1:  procedure SparseFT($x, k, F, T, \varepsilon, \delta, \rho$)                ▷ Corollary 11.87
2:      $m \leftarrow O(\varepsilon^{-1} k^2 \log^3(k) \log(FT/(\delta\rho)))$
3:      $S, w \leftarrow$ WeightedSketch($m, k, T$)                       ▷ Algorithm 57
4:      We observe the signal $x(t)$ for each $t \in S$
5:      $N_f \leftarrow O(\frac{\varepsilon}{T\sqrt{FT \log(1/\varepsilon)}}) \cdot \mathbb{Z} \cap [-F, F]$
6:      for $\{f'_1, \dots, f'_k\} \in \binom{N_f}{[k]}$ do
7:          Let $\mathcal{F} = \mathrm{span}\{\exp(2\pi \mathbf{i} f'_1 t), \cdots, \exp(2\pi \mathbf{i} f'_k t)\}$
8:          $h(t) \leftarrow \mathrm{argmin}_{h \in \mathcal{F}} \|h(t) - x(t)\|_{S,w}$
9:          if $\|h(t) - x(t)\|_{S,w} \le \|\widetilde{f}(t) - x(t)\|_{S,w}$ then
10:             $\widetilde{f}(t) \leftarrow h(t)$
11:         end if
12:     end for
13:     return $\widetilde{f}(t)$
14: end procedure
```

Chapter 12: Quartic Samples Suffice for Fourier Interpolation

12.1 Introduction

Fourier transforms are the backbone of signal processing and engineering, with profound implications to nearly every field of scientific computing and technology. This is primarily due to the discovery of the well-known Fast Fourier Transform (FFT) algorithm [CT65], which is ubiquitous in engineering applications, from image and audio processing to fast integer multiplication and optimization. The classic FFT algorithm of [CT65] computes the *Discrete Fourier Transform* (DFT) of a length-n vector x, where both the time and frequency domains are assumed to be discrete. This algorithm takes $O(n)$ samples in the time domain, and constructs $\widehat{x} = \mathrm{DFT}(x)$ in $O(n \log(n))$ time. The discrete setting of DFT limits its applicability in two main aspects: The first one is that many real-world signals are continuous (analog) by nature; Secondly, many real-world applications (such as image processing) involve signals which are *sparse* in the frequency domain (i.e., $\|\widehat{x}\|_0 = k \ll n$) [ITU92, Wat94, Rab02]. This feature underlies the *compressed sensing* paradigm [CRT06], which leverages sparsity to obtain *sublinear* algorithms for signal reconstruction, with time and sample complexity depending only on the sparsity k. Unfortunately, the continuous case cannot simply be reduced to the discrete case via standard discretization (i.e., using a sliding-window function), as it “smears out" the frequencies and blows up the sparsity, which motivates a more direct approach for the continuous problem [PS15].

The study of Fourier-sparse signals dates back to the work of Prony in 1795 [dP95], who studied the problem of exact recovery of the “ground-truth" signal x in the vanilla *noiseless* setting. By contrast, the realistic setting of reconstruction from *noisy-samples* [PS15] is a different ballgame, and exact recovery is generally impossible [Moi15]. In the *Fourier Interpolation* problem, the ground-truth signal

$$x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}, \quad v_j \in \mathbb{C}, f_j \in [-F, F] \; \forall j \in [k],$$

is a k-Fourier-sparse signal with bandlimit F. Given noisy access to the ground truth $x(t) = x^*(t) + g(t)$ in limited time duration $t \in [0, T]$ (which means that we need to recover $x^*(t)$ by taking samples from $x(t)$), the goal is to reconstruct a $\widetilde{k}$-Fourier-sparse signal $y(t)$ (i.e., $y(t) = \sum_{j=1}^{\widetilde{k}} \widetilde{v}_j e^{2\pi \mathbf{i} \widetilde{f}_j t}$ for some $\widetilde{v}_j \in \mathbb{C}, \widetilde{f}_j \in [-F, F]$ for all $j \in [\widetilde{k}]$) such that

$$\|y(t) - x^*(t)\|_T^2 \le c(\|g\|_T^2 + \delta \|x^*(t)\|_T^2)$$

holds for some $c = O(1)$, where the T-norm of any function $f : \mathbb{R} \to \mathbb{C}$ is defined as

$$\|f(t)\|_T^2 := \frac{1}{T} \int_0^T |f(t)|^2 \mathrm{d}t.$$

We note that it is not necessary for $y(t)$'s frequencies and magnitudes $(\widetilde{f}_j, \widetilde{v}_j)$ being close to the ground-truth signal $x^*(t)$'s frequencies and magnitudes $(f_{j'}, v_{j'})$.

Prior to this work, the state-of-the-art algorithm for the Fourier interpolation problem was given by [CKPS16], which achieves $\widetilde{O}(k^{51})$ sample complexity, $\widetilde{O}(k^{10\omega+40})$ running time, $\widetilde{O}(k^{10})$ output sparsity, and $c \ge 2000$ approximation ratio. For calibration, we note that $o(k^4)$ sample complexity for Fourier interpolation is not known to be achievable even with *exponential* decoding time. In this chapter, we focus on improving the *efficiency* of [CKPS16]'s algorithm across all aspects: (i) runtime, (ii) sample complexity, and (iii) output-sparsity. Our main result is:

Theorem 12.1 (Main Theorem). *Let $x(t) = x^*(t) + g(t)$, where $x^*(t)$ is k-Fourier-sparse signal with frequencies in $[-F, F]$. Given samples of $x(t)$ over $[0, T]$, there is an algorithm that uses*

$$k^4 \log(FT) \cdot \operatorname{poly}\log(k, 1/\delta, 1/\rho)$$

References	**Samples**	**Time**	**Output Sparsity**
[CKPS16]	$\widetilde{O}(k^{51})$	$\widetilde{O}(k^{10\omega+40})$	$\widetilde{O}(k^{10})$
[CP19a]	$\widetilde{O}(k^4)$	$\exp(k^3)$	k
Ours (Theorem 12.1)	$\widetilde{O}(k^4)$	$\widetilde{O}(k^{4\omega})$	$\widetilde{O}(k^4)$

Table 12.1: Summary of the results. All the algorithms obtain $O(1)$ approximation ratio. We use ω to denote the exponent of matrix multiplication, currently $\omega \approx 2.373$ [Wil12, AW21].

samples, runs in

$$k^{4\omega}\log(FT)\cdot \operatorname{poly}\log(k,1/\delta,1/\rho)$$

time, and outputs a $k^4\cdot \operatorname{poly}\log(k/\delta)$*-Fourier-sparse signal* $y(t)$ *s.t with probability at least* $1-\rho$*,*

$$\|y(t)-x^*(t)\|_T \lesssim \|g(t)\|_T + \delta\|x^*(t)\|_T.$$

12.1.1 Related works

Sparse Fourier transform in the discrete setting The Fourier transform $\widehat{x}\in\mathbb{C}^N$ is a vector of length N. The goal of a sparse DFT algorithm is, given a bunch of samples x_i in the time domain and the sparsity parameter k, to output a k-Fourier-sparse signal x' with the ℓ_2/ℓ_2-guarantee

$$\|\widehat{x}'-\widehat{x}\|_2 \lesssim \min_{k\text{-sparse } z}\|z-\widehat{x}\|_2.$$

There are two different lines of work solving the above problem. One line [GMS05, HIKP12a, HIKP12b, IKP14, IK14, Kap16, Kap17] is carefully choosing samples (via hash function) and obtaining sublinear sample complexity and running time. The other line [CT06, RV08, Bou14, HR17, NSW19] is taking *random* samples (via RIP property [CT06] or others) and paying sublinear sample complexity but nearly linear running time.

Sparse Fourier transform in the continuous setting [PS15] defined the sparse Fourier transform in the continuous setting. It shows that as long as the sample duration T is large enough compared to the frequency gap η, then there is a sublinear time algorithm that recovers all the frequencies up to certain precision and further reconstructs the signal. [JLS23] improves and generalize several results in [PS15]. In particular, [PS15] only works for one-dimensional continuous Fourier transform, and [JLS23] generalizes it to d-dimensional Fourier transform. In order to convert the tone estimation guarantee to signal estimation guarantees, [PS15] provides a positive result which shows $T = O(\log^2(k)/\eta)$ is sufficient, and [Moi15] shows a lower bound result where $T = \Omega(1/\eta)$. [Son19] asked an open question about whether this gap can be closed. [JLS23] made positive progress on that problem by providing a new upper bound which is $T = O(\log(k)/\eta)$.

From the negative side, [Moi15] shows that in order to show tone estimation[1], we have to pay a lower bound in sample duration T. In [PS15], it shows that once we have tone estimation, we can obtain a signal estimation guarantee. Since [PS15] and [Moi15], there is an interesting question about whether we can reconstruct the signal without having a tone estimation guarantee, which is defined as the Fourier interpolation problem. [CKPS16] shows a positive answer to this problem. They provide a polynomial time algorithm to solve this problem. However, both sample complexity and running time in [CKPS16] have a huge polynomial factor in k. The major goal of our work is to significantly improve those polynomial factors.

12.2 Technical Overview

12.2.1 High-level approach

The high-level approach of Fourier Interpolation (also Fourier Signal reconstruction) has two steps: frequency estimation and signal estimation (also called sig-

[1]Tone refers to a (frequency, coefficient) pair in [PS15]. E.g., (f_i, v_i) is a tone of the signal $x(t) = \sum_{i=1}^{k} v_i e^{2\pi \mathbf{i} f_i t}$. And tone estimation means estimating each (f_i, v_i) precisely.

nal recovery or Fourier set query). This work mainly contributes to the first frequency estimation step.

Filters and HashToBins The core technique in Fourier sparse recovery and interpolation algorithms is filtering. There are two kinds of filters we are using. The first filter function applied to the signal is $H(t)$ (Figure 12.1a), which is the bounded band limit approximation of the rectangular window function $\mathrm{rect}_T(t)$. Intuitively, since the time duration is restricted to $[0,T]$, we should view the ground truth signal as $x^*(t)\cdot\mathrm{rect}_T(t)$. However, handling $\widehat{\mathrm{rect}_T}(f)$ is not easy due to its unbounded support in the frequency domain. Therefore, we use $H(t)$ instead, which truncates the frequency domain of $\mathrm{rect}_T(t)$ and makes the analysis much easier.

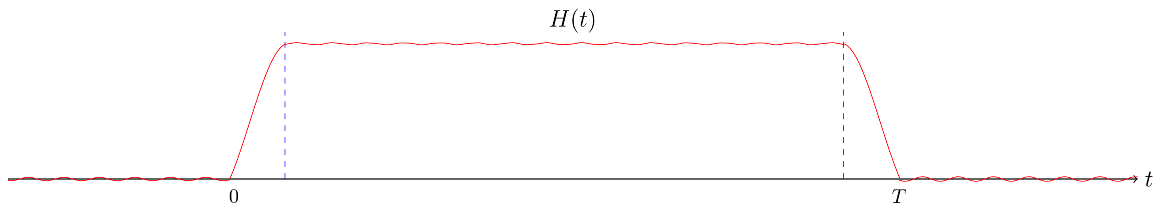
Another kind of filters we use is $G^{(j)}_{\sigma,b}(t)$ (Figure 12.1b), which "isolates" the signal through the procedure HashToBins and extracts the one-cluster signal in the j-th bin. More specifically, HashToBins divides the frequency domain into $B=O(k)$ bins. We can show that with high probability over the randomized hashing function, each bin contains a single cluster of frequencies. Hence, in the following frequency estimation step, we can just focus on recovering the frequency of a one-cluster *filtered signal* in each bin $j\in[B]$:

$$z_j(t)=(x\cdot H)(t)*G^{(j)}_{\sigma,b}(t).$$

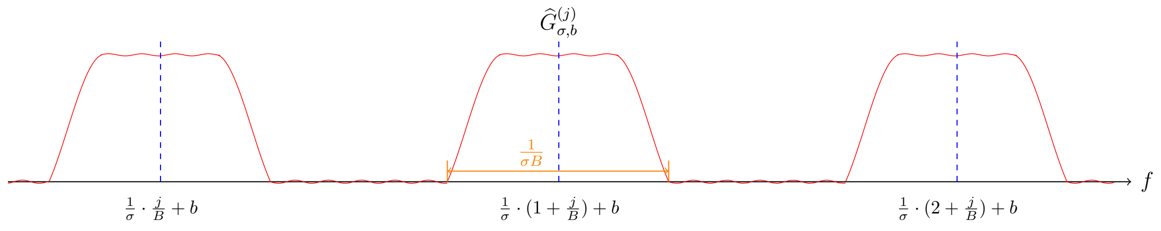
Frequency Estimation This step is the main focus on this chapter. To estimate the frequencies, our algorithm has two levels. The first level generates significant samples of the *local-test signal*:

$$d_z(t)=z(t)e^{2\pi\mathbf{i}f^*\beta}-z(t+\beta),$$

where $z(t)=z_j(t)$ is the filtered signal in the j-th bin and β is a perturbation parameter. A time point $\alpha\in[0,T]$ is defined to be significant with respect to the target frequency f^* if $|d_z(\alpha)|$ is small. In this case, $z(\alpha+\beta)/z(\alpha)$ is a good approximation of



(a) Time domain filter $H(t)$.



$\widehat{G}_{\sigma,b}^{(j)}$
$\frac{1}{\sigma B}$
f
$\frac{1}{\sigma} \cdot \frac{j}{B} + b$
$\frac{1}{\sigma} \cdot (1 + \frac{j}{B}) + b$
$\frac{1}{\sigma} \cdot (2 + \frac{j}{B}) + b$

(b) Frequency domain filter $\widehat{G}_{\sigma,b}^{(j)}(f)$ for the j-th bin.

Figure 12.1: Time and frequency domain filters.

$e^{2\pi \mathbf{i} f^* \beta}$,which further implies the target frequency f^*. The second level is a searching algorithm that iteratively estimates the target frequency f^*. In each iteration, it calls the significant sample generation algorithm and uses the significant sample to narrow the possible range of the target frequency until reaching the desired accuracy. Based on the two-level strategy, we design an efficient, high-accuracy frequency estimation algorithm, improving the time complexity, sample complexity, and the estimation error of the frequency estimation algorithms in previous works [CKPS16, CP19b]. The theorem is stated as follows.

Theorem 12.2 (Frequency estimation, Informal version of Theorem 12.61)**.** *There exists an algorithm takes $O(k^2 \log(1/\delta) \log(FT))$ samples, runs in $O(k^2 \log(1/\delta) \log^2(FT))$ time, returns a set L of $O(k)$ frequencies such that with probability $1 - \rho_0$, for any "important frequency" f, there exists an $\widetilde{f} \in L$ satisfying*

$$|f - \widetilde{f}| \lesssim \Delta,$$

where $\Delta = k \cdot |\mathrm{supp}(\widehat{H})|$, where $\widehat{H}$ is the Fourier transform of H.

Signal Estimation In signal estimation, a set of estimated frequencies of $y(t)$ has been found, and it remains to interpolate the signal under these frequencies. This is often done via *set-query* techniques [Pri11]. This step is not the focus of this chapter, and more discussions can be found in [CKPS16, SSWZ22]. [2]

12.2.2 Our techniques for frequency estimation

In the frequency estimation part, there are two central questions that need to be answered:

1. *Which frequencies or hashing bins are worth recovering?*
2. *How to recover a key frequency in a bin?*

Our answer to these questions substantially deviates from previous works, as we discuss below.

Answer to the first question: For the first question, [CKPS16]'s answer is the *heavy-cluster condition*, which is defined as follows:

$$[f^*-\Delta, f^*+\Delta] \text{ is heavy if } \int_{f^*-\Delta}^{f^*+\Delta} |\widehat{H\cdot x^*}(f)|^2 \mathrm{d}f \geq T\cdot \mathcal{N}^2/k, \tag{12.1}$$

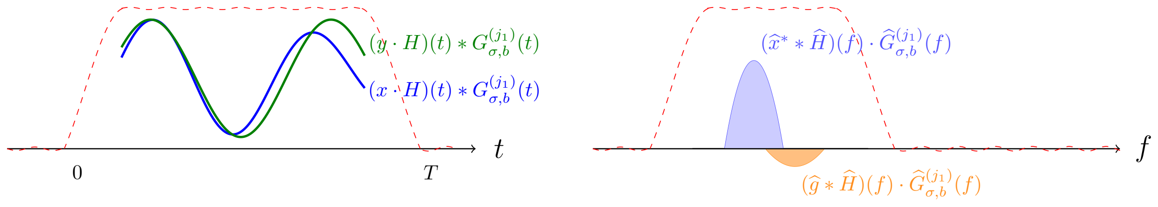
where $\mathcal{N}^2 := \|g\|_T^2 + \delta\|x^*\|_T^2$ represents the noisy-level of $x(t)$. However, only considering the energy of the ground-truth signal is not enough[3]. Indeed, their algorithm only works for "recoverable" clusters, which are defined as:

$$[f^*-\Delta, f^*+\Delta] \text{ is recoverable if } \int_{f^*-\Delta}^{f^*+\Delta} |\widehat{H\cdot x}(f)|^2 \mathrm{d}f \geq T\cdot \mathcal{N}^2/k.$$

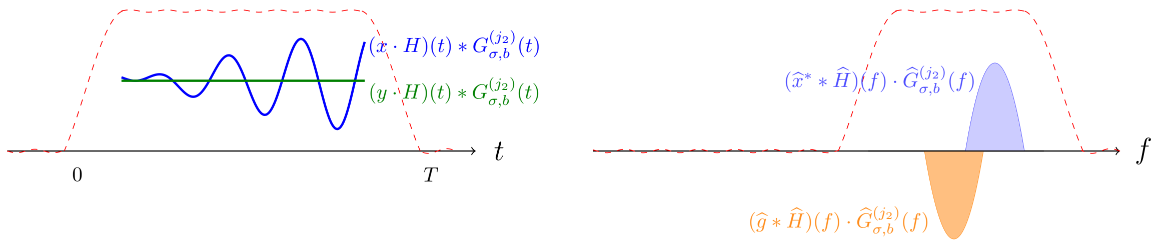
[2]We stress that this chapter is self-contained and we provide all the technical details of signal estimation in Section 12.16.

[3]For example, consider the ground-truth signal $x^*(t) = ve^{2\pi \mathbf{i} f^* t} + ve^{2\pi\mathbf{i}(f^*+10\Delta)t}$ and the noise $g(t) = -ve^{2\pi\mathbf{i}f^*t}$. Even if $f^*\pm\Delta$ is a heavy cluster, it is impossible to recover f^* from the observation $x(t) = x^*(t) + g(t)$, since $\widehat{x}(f)$ is zero around f^*.

The gap between heavy clusters and recoverable clusters is a bottleneck for improving the approximation ratio of the Fourier interpolation algorithms in [CKPS16] to an arbitrarily small constant. This gap also introduces many other technical difficulties in designing more efficient frequency estimation algorithms.



(a) Low-noise band recovery: high-accuracy frequency estimation is needed.



(b) High-noise band recovery: any frequency estimation output is acceptable.

Figure 12.2: The high SNR band condition. The red curves are the filters. On the left, the blue curves are the filtered noisy observation signal in the time domain, and the green curves are corresponding reconstructed signals. On the right, the light blue regions are the filtered frequencies of the ground-truth signal x^*, and the orange regions are the filtered frequencies of the noise g. Figure 12.2a shows a high-SNR case, where we can recover a good approximation of x^* in this band. Figure 12.2b shows an extremely low-SNR case, where g has almost the same energy as x^*, and a trivial signal ($y(t) =$ constant) suffices for the recovery of this band.

To overcome this gap, we introduce a new criterion for the frequency bands that need to be non-trivially reconstructed, which we call the *high signal-to-noise ratio (SNR) band condition*. Formally, we say a hashing bin $j \in [B]$ has a high SNR if the filtered signal $z_j^*(t) = (x^* \cdot H) * G^{(j)}_{\sigma,b}$ satisfies:

$$\|(g \cdot H) * G^{(j)}_{\sigma,b}(t)\|_T^2 \le c \cdot \|z_j^*(t)\|_T^2, \tag{12.2}$$

where c is a universal small constant. Our frequency estimation algorithm focuses solely on recovering *heavy frequencies in high-SNR bins*. The intuition behind this condition is as follows: if the noise in a band (i.e., $(g \cdot H) * G^{(j)}_{\sigma,b}(t)$) is too large, then we can simply use an all-zero signal as the reconstruction of the filtered signal. We show this new condition brings many advantages for designing more efficient frequency estimation algorithms. In particular, we show that the remaining frequencies in the low-SNR bins are inconsequential for the reconstruction error, and ignoring them in the signal estimation can still achieve the approximation guarantee of Fourier interpolation.[4]

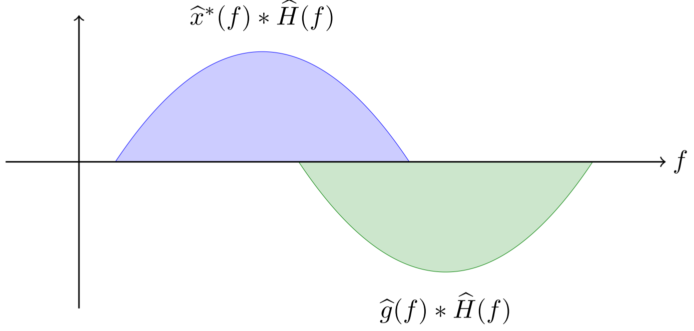


Figure 12.3: A case that violates our high SNR band assumption but [CKPS16] tries to recover. $\widehat{x}^*(f) * \widehat{H}(f)$ (in blue) is the filtered ground-truth signal, and $\widehat{g}(f) * \widehat{H}(f)$ (in green) is the filtered noise. This signal does not satisfy the high SNR band condition since the noise $\widehat{g}(f) * \widehat{H}(f)$ is too strong. However, the combined signal $(\widehat{x}^*(f) + \widehat{g}(f)) * \widehat{H}(f)$ still satisfies the recoverable-cluster condition since it has enough energy in the frequency domain.

Answer to the second question: As we discussed earlier, the key to answering this question is our novel "significant-samples" generation procedure (which produces samples α such that $|z(\alpha)e^{2\pi \mathbf{i} f^* \beta} - z(\alpha + \beta)|$ is small, where $z(t)$ is the filtered signal and β is a parameter). This is the content of the following lemma.

[4]We remark our algorithm never attempts to decide whether a bin satisfies the high-SNR condition or not, but rather assumes all bins are "good". The low-SNR bins may therefore produce totally wrong frequency estimates. However, for accurate signal estimation, we only need to guarantee that all the good frequencies are reconstructed by the frequency estimation algorithm, so even if the output set contains some wrong frequencies, they can be simply ignored.

Lemma 12.3 (Significant Sample Generation, Informal version of Lemma 12.55)**.** *There is a Procedure* GenerateSignificantSamples *in Algorithm 65 such that for $\beta \leq O(1/\Delta)$, it takes $\widetilde{O}(k^2)$ samples in $x(t)$ and runs in $\widetilde{O}(k^2)$ time. For each frequency f^* with $j := h_{\sigma,b}(f^*)$, if the j-th bin has "high SNR", and f^* is "heavy", then the output α_j satisfies:*

$$|z_j(\alpha_j + \beta) - z_j(\alpha_j)e^{2\pi \mathbf{i} f^* \beta}|^2 \leq 0.01|z_j(\alpha_j)|^2,$$

with a high constant probability, where $z_j(t) := (x \cdot H) * G_{\sigma,b}^{(j)}(t)$.

We first sketch the proof of Theorem 12.2 using Lemma 12.3. Intuitively, if $z(t)$ is exactly one-sparse, i.e., $z(t) = e^{2\pi \mathbf{i} f^* t}$, then we have $z(t)e^{2\pi \mathbf{i} f^* \beta} - z(t+\beta) = 0$, and $\frac{z(t+\beta)}{z(t)}$ gives the exact value of $e^{2\pi \mathbf{i} f^* \beta}$. More generally, by the guarantee of the significant sample, that ratio can well-approximate $e^{2\pi \mathbf{i} f^* \beta}$, which gives a good estimate of $f^*\beta \mod 1$ in a small constant range:

$$f^* \approx \frac{1}{2\pi\beta}\Big(\arg\Big(\frac{z(\alpha+\beta)}{z(\alpha)}\Big) + 2\pi s\Big)$$

for some unknown $s \in \mathbb{Z}$. To determine s, we use a search technique to narrow down the potential range of f^* from $[-F, F]$ to $[f^* - \Delta, f^* + \Delta]$. In each iteration, we divide the region of interest into $\mathsf{num} = O(1)$ regions, and repeatedly run the Procedure GenerateSignificantSamples with several different β and pick up the heavy-hitter among all possible regions, which can exponentially increase the success probability of finding the correct interval. Now, we consider the costs of this process. The initial frequency range is $[-F, F]$, and in the last iteration, the frequency range is $[f^* - \Theta(\Delta), f^* + \Theta(\Delta)]$. Thus, we can take the number of iterations to be $O(\log(F/\Delta)) \leq O(\log(FT))$. In each iteration, we call Procedure GenerateSignificantSamples for $O(\log\log(F/\Delta)) \leq O(\log\log(FT))$ times. Note that each run of Procedure GenerateSignificantSamples can generate significant samples for all B bins. Therefore, by Lemma 12.3, the total time and sample complexity for frequency estimation is $\widetilde{O}(k^2) \cdot O(\log(FT)) \cdot O(\log\log(FT)) = \widetilde{O}(k^2)$.

Then, we sketch the proof of Lemma 12.3, which contains three parts:

Algorithm 64 Frequency Estimation Algorithm, Informal version of Algorithm 66, 67, and 68

```
1: procedure FrequencyEstimationX($x, (\sigma, b)$)
2:     for $j \leftarrow [B]$ do
3:         $\widetilde{f}_j \leftarrow$ FrequencyEstimationZ($x, H, G_{\sigma,b}^{(j)}$)    ▷ recover the heavy frequency of $z^{(j)}$
4:         $L \leftarrow L \cup \{\widetilde{f}_j\}$
5:     end for
6:     return $L$
7: end procedure
8: procedure FrequencyEstimationZ($x, H, G_{\sigma,b}^{(j)}$)
9:     $\mathsf{num} \leftarrow O(1)$    ▷ num-ary search in each iteration
10:    $D \leftarrow O(\log(\frac{FT}{\Delta}))$    ▷ number of iterations
11:    $\mathsf{left}_1 \leftarrow -F$, $\mathsf{len}_1 \leftarrow 2F$    ▷ initial searching interval $[\mathsf{left}_1, \mathsf{left}_1 + \mathsf{len}_1]$
12:    for $d \in [D]$ do
13:        $\mathsf{left}_{d+1} \leftarrow$ ArySearch($x, H, G_{\sigma,b}^{(j)}, \mathsf{left}_d, \mathsf{len}_d, \mathsf{num}$)    ▷ new searching interval's left-end
14:        $\mathsf{len}_{d+1} \leftarrow 5\frac{\mathsf{len}_d}{\mathsf{num}}$    ▷ new searching interval's length
15:    end for
16:    return $\mathsf{left}_{D+1}$
17: end procedure
18: procedure ArySearch($x, H, G_{\sigma,b}^{(j)}, \mathsf{left}_i, \mathsf{len}_i, \mathsf{num}$)
19:    $I_q \leftarrow [\mathsf{left}_d + (q-1)\mathsf{len}_d/\mathsf{num}, \mathsf{left}_d + q\mathsf{len}_d/\mathsf{num}]$ for $q \in [\mathsf{num}]$    ▷ candidate regions
20:    $v_q \leftarrow 0$ for $q \in [\mathsf{num}]$    ▷ votes counter
21:    $R \leftarrow O(\log(\log(FT)))$
22:    for $r = 1 \rightarrow R$ do
23:        Sample $\beta \sim$ Uniform($[\frac{1}{2}\widehat{\beta}, \widehat{\beta}]$) for $\widehat{\beta} = O(\frac{\mathsf{num}}{\mathsf{len}_d})$    ▷ perturbation
24:        $z(\alpha+\beta), z(\alpha) \leftarrow$ GenerateSignificantSamples($x, H, G_{\sigma,b}^{(j)}$)    ▷ significant sample
25:        $\widetilde{S} \leftarrow \frac{1}{2\pi\beta}(\arg(\frac{z(\alpha+\beta)}{z(\alpha)}) + 2\pi\mathbb{Z})$    ▷ all possible frequencies
26:        $\widetilde{I} \leftarrow \{q \in [\mathsf{num}] \mid I_q \cap \widetilde{S} \neq \emptyset\}$    ▷ all possible regions
27:        $v_q \leftarrow v_q + 1$ for $q \in \widetilde{I}$    ▷ add votes to these regions
28:    end for
29:    return $\mathsf{left}_d + (q-1)\mathsf{len}_d/\mathsf{num}$ for any $q$ such that $v_q + v_{q+1} + v_{q+2} \geq R/2$
30: end procedure
```

I. A two-level sampling procedure (see Section 12.2.2.1).

II. Energy estimation and Signal Equivalent Method (see Section 12.2.2.2).

III. Time-domain concentration of filtered signals (see Section 12.2.2.3).

12.2.2.1 Two-level sampling for significant samples generation

We may assume that in frequency domain, the energy of $\widehat{z}(f)$ is concentrated around f^*:

$$\int_{f^*-\Delta}^{f^*+\Delta} |\widehat{z}(f)|^2 \mathrm{d}f \geq 0.7 \int_{-\infty}^{+\infty} |\widehat{z}(f)|^2 \mathrm{d}f.$$

This is a very natural and necessary assumption for the frequency estimation problem. [5] Then we can show that:

$$\|z(t)e^{2\pi \mathbf{i} f^* \beta} - z(t+\beta)\|_T^2 < \gamma \|z(t)\|_T^2 \tag{12.3}$$

where $\gamma \in (0, 0.001)$ is a small constant. We show how to find an α such that $|z(\alpha)e^{2\pi \mathbf{i} f^* \beta} - z(\alpha+\beta)|^2 < \gamma |z(\alpha)|^2$. For ease of discussion, we scale the time domain from $[0, T]$ to $[-T, T]$.

The main idea is to use a two-level sampling procedure, which is motivated by [CP19b]. In the first level, we take a set $S = \{t_1, \dots, t_s\}$ of $O(k \log(k))$ i.i.d. samples from the following distribution:

$$D_z(t) = \begin{cases} c \cdot (1 - |t/T|)^{-1} T^{-1} & \text{if } |t| \leq T(1 - 1/k) \\ c \cdot k T^{-1} & \text{if } |t| \in [T(1-1/k), T] \end{cases} \quad \forall t \in U, \tag{12.4}$$

where $U = \{t_0 \in \mathbb{R} \mid H(t) > 1 - \delta_1 \ \forall t \in [t_0, t_0 + \beta]\}$. Then, we assign weights $w_i := 1/(2T|S|D_z(t_i))$ for each sample $t_i \in S$.

[5] For the filtered signals that do not satisfy the frequency domain energy concentration assumption, it basically means that they do not contain enough information to recover f^*, and we can just ignore those "useless" clusters.

In the second level of the sampling procedure, we sub-sample a t_i from the set S as the output according to the following distribution:

$$D_S(t_i) = \frac{w_i \cdot |z(t_i)|^2}{\sum_{j\in[s]} w_j \cdot |z(t_j)|^2} \quad \forall i \in [s].$$

Now, we explain why the two-level sampling procedure works. By the energy estimation method discussed in Section 12.2.2.2, we know that:

$$\|z(t)\|_T^2 \approx \|z(t)\|_{S,w}^2 := \sum_{i=1}^{s} w_i \cdot |z(t_i)|^2, \quad \text{and}$$

$$\|z(t)e^{2\pi \mathbf{i} f^* t} - z(t+\beta)\|_T^2 \approx \|z(t)e^{2\pi \mathbf{i} f^* \beta} - z(t+\beta)\|_{S,w}^2 := \sum_{i=1}^{s} w_i \cdot |z(t_i)e^{2\pi \mathbf{i} f^* \beta} - z(t_i+\beta)|^2.$$

The second level of the sampling procedure ensures that

$$\begin{aligned}\mathbb{E}_{t\sim D_S}\left[\frac{|z(t)e^{2\pi \mathbf{i} f^* \beta} - z(t+\beta)|^2}{|z(t)|^2}\right] &= \frac{\sum_{i=1}^{s} w_i |z(t_i)e^{2\pi \mathbf{i} f^* \beta} - z(t_i+\beta)|^2}{\sum_{j=1}^{s} w_j |z(t_j)|^2} \\ &= \frac{\|z(t)e^{2\pi \mathbf{i} f^* \beta} - z(t+\beta)\|_{S,w}^2}{\|z(t)\|_{S,w}^2}.\end{aligned}$$

Hence, we get that

$$\mathbb{E}_{t\sim D_S}\left[\frac{|z(t)e^{2\pi \mathbf{i} f^* \beta} - z(t+\beta)|^2}{|z(t)|^2}\right] \approx \frac{\|z(t)e^{2\pi i f^* \beta} - z(t+\beta)\|_T^2}{\|z(t)\|_T^2} < \gamma,$$

where the last step follows from Eq. (12.3). Then by Markov's inequality, we get that the sample α generated by the two-level sampling procedure satisfies $|z(\alpha)e^{2\pi i f^* \beta} - z(\alpha+\beta)|^2 \lesssim \gamma |z(\alpha)|^2$ with high probability.

The costs of this two-level sampling procedure are calculated as follows. In the first level, we takes $|S| = \widetilde{O}(k)$ samples from $z(t)$, where each sample $z(t_i) = ((x \cdot H) * G_{\sigma,b}^{(j)})(t_i)$ can be computed by $|\mathrm{supp}(G_{\sigma,b}^{(j)}(t))| = \widetilde{O}(k)$ samples from $x(t)$ in $\widetilde{O}(k)$ time. Thus, the total time and sample complexity for the first level sampling procedure is $\widetilde{O}(k) \cdot \widetilde{O}(k) = \widetilde{O}(k^2)$. In the second level, we further select one sample from the output of the first level, which can be done in $\widetilde{O}(|S|) = \widetilde{O}(k)$ times and does not need any new sample.

We further discuss how large β we can choose in the sampling procedure since it controls the estimation accuracy of f^*.[6] We note that the range of β is determined by Eq. (12.3), which is an underlying assumption of our sampling procedure. To satisfy this inequality, we need to guarantee that $|e^{2\pi \mathbf{i} f^* \beta} - e^{2\pi \mathbf{i} f \beta}| \le \gamma$ for any $f \in f^* \pm \Delta$, which implies that $\beta \le O(\gamma/\Delta)$. For comparison, the upper bound of β in [CKPS16] is only $O(\gamma/(\Delta\sqrt{\Delta T}))$ due to a stronger accuracy requirement there.[7]

12.2.2.2 Energy estimation and Signal Equivalent Method

In this section, we show that the sampling and reweighing method we use in the significant sample generation procedure can accurately estimate the energy of $z(t)$ and $z(t)e^{2\pi \mathbf{i} f^* t} - z(t+\beta)$ with a sample complexity almost reaching the information-theoretic limit.

Lemma 12.4 (Informal version of Lemma 12.52 and Lemma 12.53)**.** *Suppose f^* is a heavy frequency hashed to the j-th bin which satisfies the high SNR condition. Let $z^*(t) = (x^* \cdot H) * G_{\sigma,b}^{(j)}$ and $z(t) = (x \cdot H) * G_{\sigma,b}^{(j)}$. Let $U \subseteq [0,T]$ be an interval. Let $S = \{t_1, \dots, t_s\}$ be a set of $O(k\log(k))$ i.i.d. samples from the distribution D defined by Eq. (12.4) with weights $w_i = 1/(TsD(t_i))$. Then, with probability at least 0.8,*

$$\|z(t)\|_{S,w}^2 \gtrsim \|z^*(t)\|_U^2 \quad \text{and} \quad \|z(t)e^{2\pi \mathbf{i} f^* t} - z(t+\beta)\|_{S,w}^2 \lesssim \|z^*(t)\|_U^2,$$

where $\|z(t)\|_U^2 = (1/|U|) \cdot \int_U |z(t)|^2 \mathrm{d}t$.

To prove Lemma 12.4, we develop a *Signal Equivalent Method.* Below, we sketch the proof of the first half of Lemma 12.4 on the energy estimation for $z(t)$. The second half follows similar ideas.

[6] By comparing $z(t+\beta)$ and $z(t)$, we get an estimate of $f^*\beta$ within some error $\pm b$, which implies an estimate of f^* within an error $\pm b/\beta$. Hence, larger β gives a higher accuracy of the frequency estimation.

[7] [CKPS16] give an ℓ_1-norm error guarantee in the frequency domain, i.e., $\int_{f^*-\Delta}^{f^*+\Delta} |e^{2\pi \mathbf{i} f^* \beta} - e^{2\pi \mathbf{i} f \beta}| \mathrm{d}f$ is small. To obtain an ℓ_2-norm guarantee (like Eq. (12.3)), they need to apply Cauchy-Schwarz inequality, which results in an extra $\sqrt{\Delta T}$ factor in their upper bound of β.

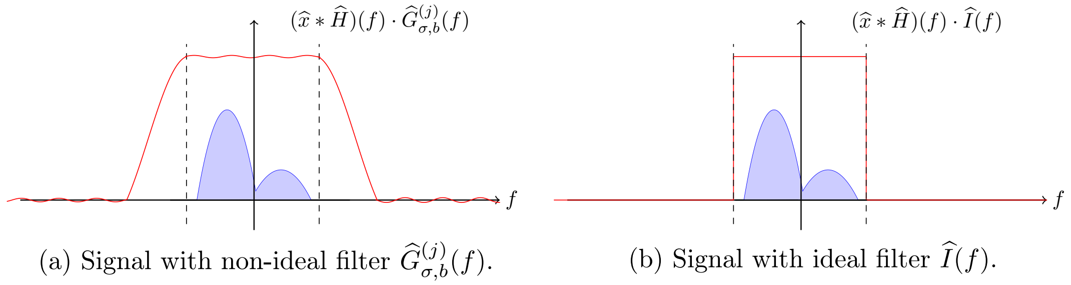


Figure 12.4: The Signal Equivalent Method. This figure demonstrates that $(\widehat{x} * \widehat{H})(f) \cdot \widehat{G}^{(j)}_{\sigma,b}(f)$ (left) can be approximated by $(\widehat{x} * \widehat{H})(f) \cdot \widehat{I}(f)$ (right). $\widehat{I}(f)$ (red curve on the right) is the ideal filter that approximate $\widehat{G}^{(j)}_{\sigma,b}(f)$ (red curve on the left).

Energy estimation is also used in prior works [CKPS16, CP19a, CP19b, SSWZ22], where a key component is the following energy bound for the interested function family $\mathcal{F}$:

$$\sup_{f \in \mathcal{F}} \sup_{t \in [0,T]} \frac{|f(t)|^2}{\|f(t)\|_T^2}.$$

However, this approach is unlikely to work directly for our filtered signal $z(t)$ since it depends on the randomized hashing function. And under some hashing parameter (σ, b), there always exists some signal $x(t)$ such that $z(t) = (x \cdot H)(t) * G^{(j)}_{\sigma,b}(t)$ is in ill-condition (e.g., the frequencies are not well-isolated, or large offset events happen). As a result, bounding $\frac{|z(t)|^2}{\|z(t)\|_T^2}$ for all $z(t)$ of the form $(x \cdot H) * G^{(j)}_{\sigma,b}(t)$ by a small number is not easy. We bypass the issue by proving an energy bound only for those $z(t)$ under some well-hashed conditions (e.g. frequency is isolated and do not have a large offset), and showing that such a "refined energy bound" is still sufficient to derive the sample complexity of our algorithm.

The motivation of the Signal Equivalent Method comes from the special structure of $z(t) = (x \cdot H)(t) * G^{(j)}_{\sigma,b}(t)$ in the frequency domain. Notices that the observed signal $x(t)$'s Fourier transform $\widehat{x}(f)$ only contains some spikes (assuming small noise). By convolution with $\widehat{H}(f)$ (which corresponds to multiplying by $H(t)$ in the time domain), $(\widehat{x} * \widehat{H})(f)$ fattens the spikes in the frequency domain (and by Parseval's theorem, the area of the signal in frequency domain equals to its energy). Then, convolution with $G_{\sigma,b}^{(j)}(t)$ "zooms-in" to a narrow band around a single frequency. This construction of $z(t)$ motivates us to build a new signal $\overline{z}(t) = (x \cdot H)(t) * I(t)$, where $I(t)$ is a filter function such that $\widehat{I}(f) = 1$ when $G_{\sigma,b}^{(j)}(t) > 1/2$, and $\widehat{I}(f) = 0$ otherwise. To analyze the equivalent signal $\overline{z}(t)$, we improve the analysis of the filter $H(t)$ in [CP19b] and give a tighter bound on its value in a sub-interval of $[0, T]$. Then, we show that the equivalent signal $\overline{z}(t)$ is *almost equivalent* to $z(t)$ under some "good conditions" (i.e., the frequency is isolated and no large offset). We also prove that the ideal filter has several useful properties that can mush simplify the analysis (e.g., the function $I(t)$ is randomized, and with high probability, $I(t)$ commutes with $H(t)$).

By the Signal Equivalent Method, we can first prove an energy bound for the equivalent signal $\overline{z}(t)$, which follows from the Fourier-sparse signals' energy bounds (see Section 12.5). Then, it remains to show that the equivalent signals' energy bound can approximate the original filtered signal $z(t)$'s energy bound. We find that the approximation error comes from two sources: the observation noise $g(t)$ and the approximation error $\overline{z}(t) - z(t)$. The first part of the error is small due to the high SNR band condition (Eq. (12.2)). And the second part of the error is mitigated by the tail-bound for $G_{\sigma,b}^{(j)}(t)$ and the heavy-cluster condition (Eq. (12.1)). More specifically, the HashToBins procedure and the filter $G_{\sigma,b}^{(j)}(t)$ can bring some interference noise from other bins to $z(t)$, which is perfectly eliminated by the ideal filter $I(t)$ in the equivalent signal $\overline{z}(t)$. Hence, we need to bound this part of noise when we transfer back from the equivalent signal to the true filtered signal. The tail bound of $G_{\sigma,b}^{(j)}(t)$ ensures that adding small interference noise with frequencies far away from the center of the cluster will not drastically affect $z(t)$. However, by this argument, we can only bound the distance between $\overline{z}(t)$ and $z(t)$ by $\|x^*(t)\|_T$, which can be much larger than $\|z(t)\|_T$. Hence, we need to use the heavy-cluster assumption to ensure that $\|x^*(t)\|_T \lesssim \|z(t)\|_T$. Using these error-control techniques, we can prove that an energy bound for $\overline{z}(t)$ implies an energy bound for $z(t)$.

We give a comparison between ours and previous approaches for proving the

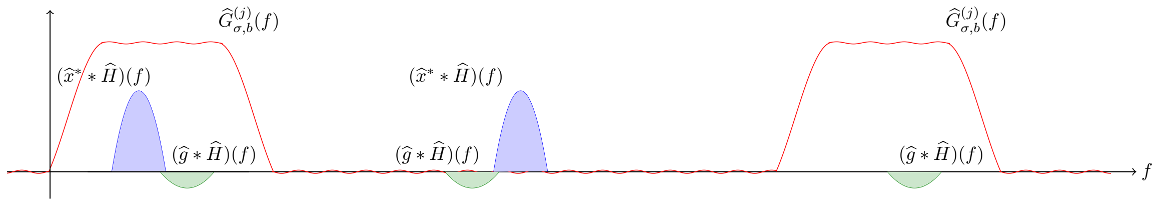


Figure 12.5: An illustration of a filtered noisy signal. $\widehat{G}^{(j)}_{\sigma,b}$ (the red curve) is the HashToBins filter for the j-th bin. The noise in the filtered signal comes from two parts: one is $\widehat{g}(f) * \widehat{H}(f)$ (the green signal), and another is the interference by signals outside the bin (the blue and green signals in the middle).

energy estimation guarantee. [CKPS16] considers $z(t)$ as a generic signal that satisfies the time and frequency domains concentration properties[8]. We exploit "finer" structure of $z(t)$ and obtain a stronger energy bound and reduce the number of samples required in norm preserving. [CP19b] also proves a similar property (but only for $(x \cdot H)(t)$). However, they assume that all the frequencies of $x^*(t)$ are contained in a small interval, making the task much easier. Our filtered signal $z(t)$ does not satisfy this condition due to the interference noise caused by the HashToBins procedure.

12.2.2.3 Time-domain concentration of filtered signals

The proof of Lemma 12.3 relies on an underlying assumption: the most of the energy of the filtered signal is contained in the observation window $[0, T]$. That is, we need the following lemma:

Lemma 12.5 (Informal version of Lemma 12.29)**.** *Let $j \in [B]$ be a bin that contains a heavy frequency. Let $z(t) = (x^* \cdot H) * G^{(j)}_{\sigma,b}$ be the filtered signal. Then, we have*

$$\int_{-\infty}^{+\infty} |z(t)|^2 \mathrm{d}t \le 1.35 \int_0^T |z(t)|^2 \mathrm{d}t.$$

A similar concentration property is also proved in [CKPS16], using a very strict requirement on the $H(t)$ filter that it decays at an exponential rate near the

[8]It means that most of the energy of $z(t)$ (i.e., $\|z(t)\|_2$) lies in $[0, T]$ and most of the energy of $\widehat{z}(f)$ lies in a poly$(k)/T$ length interval in frequency domain.

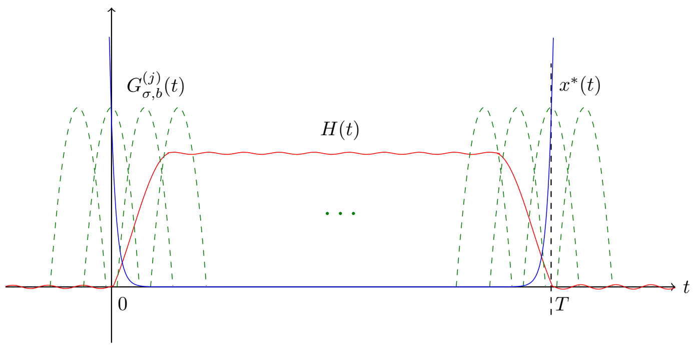


Figure 12.6: A bad case that may break the time domain concentration of $z(t) = (x \cdot H)(t) * G_{\sigma,b}^{(j)}(t)$ in $[0, T]$ when the filter decay slowly. $x^*(t)$ (in blue) is a k-Fourier-sparse signal. $G_{\sigma,b}^{(j)}$ (in green dashed) is the filter of frequency domain. $H(t)$ (in red) is the filter in time domain. On the one hand, since $x^*(t)$ is very small in $[T/\mathrm{poly}(k), T(1 - /\mathrm{poly}(k))]$, and $H(t)$ is very small in $[0, T/\mathrm{poly}(k)] \cup [T(1 - /\mathrm{poly}(k)), T]$, the filtered signal has very small energy within $[0, T]$. On the other hand, since the convolution with $G_{\sigma,b}^{(j)}(t)$ can bring some energy of $x^*(t)$ passing the boundary of $[0, T]$, and the signal $x^*(t)$ could be very large outside $[0, T]$, $(x \cdot H)(t) * G_{\sigma,b}^{(j)}(t)$ may contain very large energy in $\mathbb{R} \backslash [0, T]$. In this case, $\|z(t)\|_{L_2}^2 \gg \|z(t)\|_T^2$.

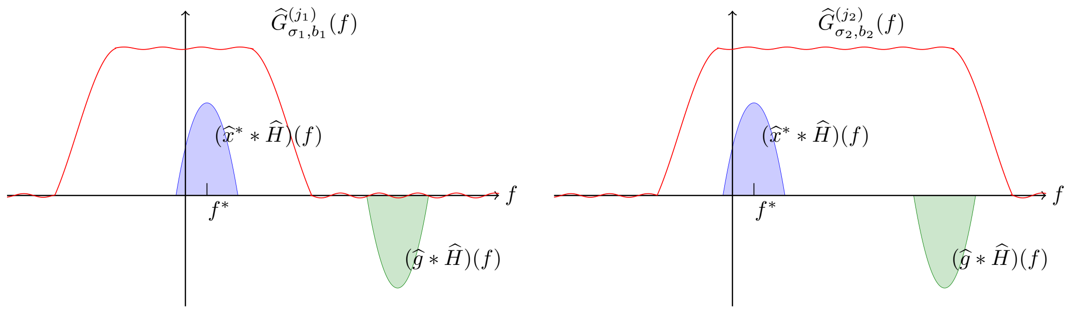
boundary. More specifically, they require that $H(t)$ is exponentially small not only outside the time duration $[0, T]$, but also in the shrinking boundary $[0, T/\mathrm{poly}(k)] \cup [T - T/\mathrm{poly}(k), T]$. The additional constraint allows them to show that $x(t)$'s energy near the boundary cannot "pass" the H filter, and the energy concentration of $z(t)$ easily follows. However, it also results in a large support of H in the frequency domain, which leads to a large error in the frequency estimation, and further causes large output sparsity and time/sample complexity of their Fourier interpolation algorithm.

We resolve this issue by changing the filter function to the one defined in [CP19b], which has much smaller support and thus saves time and sample complexities. However, it is exponentially small outside $[0, T]$, but only *polynomially* small near the boundary. To prove Lemma 12.5, we use our Signal Equivalent Method again. We construct an equivalent signal $\overline{z}(t) = (x^* \cdot H) * I(t) = (x^* * I) \cdot H(t)$, where $x^*(t) * I(t)$ is a Fourier-sparse signal. Then, by some finer analysis on the $H(t)$ filter (see Lemma 12.24), we can show that most of the energy of $x^*(t) * I(t)$ is preserved in $[0, T]$, i.e.,

$$\int_{-\infty}^{+\infty} |\overline{z}(t)|^2 \mathrm{d}t \leq 1.1 \int_0^T |\overline{z}(t)|^2 \mathrm{d}t.$$

Finally, by the approximation guarantee of Signal Equivalent Method, we get that the energy concentration of $\overline{z}(t)$ implies the energy concentration of $z(t)$.

12.2.3 Our techniques for Fourier interpolation


$\widehat{G}^{(j_1)}_{\sigma_1,b_1}(f)$
$(\widehat{x}^* * \widehat{H})(f)$
f^*
f
$(\widehat{g} * \widehat{H})(f)$
$\widehat{G}^{(j_2)}_{\sigma_2,b_2}(f)$
$(\widehat{x}^* * \widehat{H})(f)$
f^*
f
$(\widehat{g} * \widehat{H})(f)$

(a) High SNR, where $j_1 = h_{\sigma_1,b_1}(f^*)$ (b) Low SNR, where $j_2 = h_{\sigma_2,b_2}(f^*)$

Figure 12.7: The SNR of the same signal changes with different hash functions. In (a), the noise $g \cdot H$ is hashed outside the bin and suppressed by $G^{(j_1)}_{\sigma_1,b_1}$. Thus, this bin has high SNR. In (b), by a different hash function, the noise is hashed inside the bin and $G^{(j_2)}_{\sigma_2,b_2}$ preserves its energy. Thus, the SNR of the bin becomes very low, even if the signal doesn't change.

In this section, we discuss how to obtain a Fourier interpolation algorithm with improved efficiency and output sparsity (Theorem 12.1) based on our frequency estimation algorithm (Theorem 12.2).

We first remark that simply applying the original framework of Fourier Interpolation (e.g., [CKPS16]) and combining with an existing signal estimation algorithm is still not enough to improve the previous algorithm, since the frequency estimation algorithm has a low success probability and we cannot apply the success probability boosting trick in [CKPS16] to increase it to $1 - \rho$. More specifically, [CKPS16] first boosts the success probability of their frequency estimation algorithm by their merge-stage algorithm (which runs the frequency estimation algorithm for $R = \log(1/\rho)$

times, sorts all recovered frequencies, and picks every $R/2$-th entry of the sorted list), and then runs the signal estimation algorithm. It does not work here because our high SNR band condition makes frequency estimation and signal estimation "entangled". More specifically, whether a frequency is contained in a high SNR bin (which *needs* to be recovered) or not depends on the randomized hash function. However, if the outputs of multiple runs of the frequency estimation algorithm are mixed together, it is hard to justify which frequencies are necessary, since different runs use different hash functions, resulting in different high SNR bins. In other words, if we still use [CKPS16]'s boosting strategy, we cannot guarantee the final output of the frequency estimation satisfies the requirement of the signal estimation algorithm.

We propose a new Fourier Interpolation framework that boosts the success probability after the signal estimation step. That is, in each run of the constant success probability frequency estimation algorithm, we reconstruct the signal immediately. Let $y_1, \ldots, y_{R_p}$ denote the reconstructed signals of R_p runs. Then, we boost the total success probability by outputting the signal y_{j^*}:

$$j^* = \arg\min_{j \in [R_p]} \operatorname*{median}_{i \in [R_p]} \|y_j(t) - y_i(t)\|_T^2.$$

By Chernoff bound, there are more than a half of y_i's being good approximations of the ground-truth signal $x^*(t)$. Using the median trick, we can show that y_{j^*} satisfies the recovery guarantee with an exponentially small failure probability.

It remains to estimate the distance $\|y_j(t) - y_i(t)\|_T^2$ between different reconstructed signals. Naively, it takes $O(\widetilde{k}^2)$-time since y_1, y_2 are $\widetilde{k}$-Fourier sparse, and it is enough to obtain the time complexity of our Fourier interpolation algorithm in Theorem 12.1. We further propose an $\widetilde{O}(\widetilde{k} \cdot k)$-time approximation algorithm for estimating a Fourier-sparse signal's energy, which could be of independent interest. The main idea is to use $\|y_i(t) - y_j(t)\|_{S,w}^2$ to approximate $\|y_i(t) - y_j(t)\|_T^2$, where the sample set S and weights w are defined by the significant sample generation procedure in Section 12.2.2.1. We show that if we take $|S| = \widetilde{O}(\widetilde{k})$, we can achieve a constant approximation ratio in $\widetilde{O}(\widetilde{k} \cdot k)$ time. In addition, we prove that even if we use the approximated distances, the output signal y_{j^*} still satisfies the recovery guarantee of Fourier interpolation.

12.3 Organization

In Section 12.4, we define our notations in this chapter. In Section 12.5, we review several energy bounds for Fourier-sparse signal. In Section 12.6, we define and show several properties of the frequency domain filters $G_{\sigma,b}^{(j)}$. In Section 12.7, we review the HashToBins strategy and prove that bad events only happen with small probability. In Section 12.8, we define and show some properties of the time domain filter $H(t)$.

Based on the analysis of the filters, in Section 12.9, we study the ideal filter and develop the Signal Equivalent Method. In Section 12.10, we show that the filtered signal satisfies some concentration properties in both time and frequency domains.

Based on the Signal Equivalent Method and the concentration properties, in Section 12.11, we prove an energy bound for filtered Fourier-sparse signals. In Section 12.12, we further extend the energy bound for local-test signals. Then, in Section 12.13, we apply the energy bounds and describe how to use samples to empirically estimate the energy of filtered signals and local-test signals. In Section 12.14, we introduce our algorithm for generating significant samples. In Section 12.15, we use the significant samples to do frequency estimation for Fourier sparse signals. Finally, in Section 12.16, we combine our frequency estimation algorithm with a signal estimation procedure and boost the success probability of Fourier Interpolation.

Section 12.17 presents a flowchart of the key theorems/lemmas for our Fourier interpolation algorithm.

12.4 Preliminaries

For any positive integer n, we define $[n]$ to be the set $\{1, 2, \cdots, n\}$. We define $\mathbf{i}$ to be $\sqrt{-1}$. For a complex number $z = a + b\mathbf{i}$, we define $|z|$ to be the magnitude of z, i.e., $|z| = \sqrt{a^2 + b^2}$. For a function f, we use $\mathrm{supp}(f)$ to denote the support set of f. We use $f \lesssim g$ to denote that there exists a constant C such that $f \leq C \cdot g$. We use $f \eqsim g$ to denote that $f \lesssim g \lesssim f$. For any function f, we use $\mathrm{poly}(f)$ to denote $f^{O(1)}$, and $\widetilde{O}(f)$ to denote $f \cdot \mathrm{poly}\log(f)$. For an interval $U \subseteq \mathbb{R}$, we use $|U|$ to denote the size of the interval, and we use $\mathrm{Uniform}(U)$ to denote the uniform distribution over U.

We use ω to denote the exponent of matrix multiplication, i.e., n^ω denote the time of multiplying an $n \times n$ matrix with another $n \times n$ matrix. Currently $\omega \approx 2.373$ [Wil12, AW21].

For two functions f and g, we use $(f * g)(t) = \int_{-\infty}^{\infty} f(s)g(t-s)\mathrm{d}s$ to denote the convolution of two functions f and g. And we use f^{*l} to denote the l-fold convolution of f, i.e., $f^{*l}(t) = f(t) * f(t) * \cdots * f(t)$. For $a \in \mathbb{R}_+$, we use $\mathrm{rect}_a(t)$ to denote the box function with support set length a, i.e., $\mathrm{rect}_a(t) = \mathbf{1}_{[-a/2,a/2]}(t)$. For $a \in \mathbb{R}$, we use $\delta_a(f)$ to denote $\delta(f-a)$, where $\delta(f)$ is the Dirichlet function. We use $\mathrm{round}(x)$ to denote rounding $x \in \mathbb{R}$ to the nearest integer. For $x \in \mathbb{R}, y \in \mathbb{R}_+$, we use $x \pmod y$ to denote the smallest positive $z \in \mathbb{R}_+$ such that $z \in x + y\mathbb{Z}$.

We say $x(t)$ is k-Fourier-sparse if:

$$x(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}.$$

We define $\widehat{x}(f)$ to be the Fourier transform of $x(t)$:

$$\widehat{x}(f) = \int_{-\infty}^{\infty} x(t) e^{-2\pi \mathbf{i} f t} \mathrm{d}t.$$

We use $\mathcal{F}_{k,F}$ to denote the following family of signals:

$$\mathcal{F}_{k,F} := \Big\{ x(t) = \sum_{j=1}^{k} v_j \cdot e^{2\pi \mathbf{i} f_j t} \;\Big|\; f_j \in [-F, F], v_j \in \mathbb{C}\ \forall j \in [k] \Big\}.$$

Then, we define several norms for signal.

- For any discrete set $S \subseteq \mathbb{R}$, the *discrete norm* of x with respect to a set S is defined as

  $$\|x(t)\|_S^2 = \frac{1}{|S|} \sum_{t \in S} |x(t)|^2,$$

  and the *weighted discrete norm* with weights $w \in \mathbb{R}^S$ is defined as

  $$\|x(t)\|_{S,w}^2 = \sum_{t \in S} w_t |x(t)|^2.$$

- For any continuous interval $U \subset \mathbb{R}$, the continuous U-norm of x is defined as

  $$\|x(t)\|_U^2 = \frac{1}{|U|} \int_U |x(t)|^2 \mathrm{d}t.$$

- For any $T > 0$, the *continuous T-norm* is defined as

  $$\|x(t)\|_T^2 = \frac{1}{T} \int_0^T |x(t)|^2 \mathrm{d}t.$$

- Let D be a probability distribution over $\mathbb{R}$. The *continuous D-norm* is defined as

  $$\|x(t)\|_D^2 = \int_{-\infty}^{\infty} D(t)|x(t)|^2 \mathrm{d}t.$$

- The L_2-norm of $x(t)$ is defined as

  $$\|x(t)\|_{L_2}^2 = \int_{-\infty}^{\infty} |x(t)|^2 \mathrm{d}t.$$

Throughout this chapter, we assume that $x^*(t) \in \mathcal{F}_{k,F}$ is our ground-truth signal. And the observation signal is $x(t) = x^*(t) + g(t)$, where $g(t)$ is an arbitrary noise function. Furthermore, we assume that $x(t)$ can be observed at any point in $[0, T]$.

12.5 Energy Bounds of Fourier Sparse Signals

The energy bound of a function family $\mathcal{F}$ is the largest value achieved by a function $f \in \mathcal{F}$ normalized by its norm (total energy) $\|f\|_T$. It connects the extreme value and the average value of the functions in $\mathcal{F}$, and is very useful in analyzing the concentration property. In Chapter 11, we have already introduced several energy bounds for one-dimensional and multi-dimensional signals. For completeness, we restate some results for one-dimensional Fourier-sparse signals below.

In our setting, we take $\mathcal{F} = \mathcal{F}_{k,F}$ to be the set of F-band-limit, k-sparse Fourier signals. [Kós08] showed an energy bound that only depends on the sparsity k, without any dependence on the time point t, band-limit F, time duration T, frequency gap $\eta = \min_{i \neq j} |f_i - f_j|$:

Theorem 12.6 ([Kós08]). *For any $t \in [0, T]$,*

$$\sup_{x \in \mathcal{F}_{k,F}} \frac{|x(t)|^2}{\|x\|_T^2} \lesssim k^2.$$

The k^2 energy bound can be further improved if we only consider the functions' value at a fixed time point t:

Theorem 12.7 ([CP19a, BE06]). *Let $D :=$ Uniform$([-1,1])$. For any $t \in (-1,1)$,*

$$\sup_{x \in \mathcal{F}_{k,F}} \frac{|x(t)|^2}{\|x\|_D^2} \lesssim \frac{k}{1-|t|}.$$

12.6 Filter in Frequency Domain

Filtering is one of the most important techniques in sparse Fourier transform literature. In this section, we introduce the frequency domain filter function $\widehat{G}_{\sigma,b}^{(j)}(f)$, which is the key to implement the HashToBins strategy. We first review the the construction given by [CKPS16] with some different parameter settings and show some known properties (see Section 12.6.1). Then, we prove a new property of the filter functions: the frequency domain covering property (see Section 12.6.2).

12.6.1 Frequency domain filter construction

In this section we review the construction and several basic properties of the frequency domain filter $G^{(j)}_{\sigma,b}(t)$, $\widehat{G}^{(j)}_{\sigma,b}(f)$.

Definition 12.1 (*G*-filter's construction, [CKPS16])**.** Given $B > 1$, $\delta > 0$, $\alpha > 0$. Let $l := \Theta(\log(k/\delta))$. Define $G_{B,\delta,\alpha}(t)$ and its Fourier transform $\widehat{G_{B,\delta,\alpha}}(f)$ as follows:

$$\begin{aligned} G_{B,\delta,\alpha}(t) :=& \quad b_0 \cdot (\mathrm{rect}_{\frac{B}{(\alpha\pi)}}(t))^{\star l} \cdot \mathrm{sinc}(t\tfrac{\pi}{2B}), \\ \widehat{G_{B,\delta,\alpha}}(f) :=& \quad b_0 \cdot (\mathrm{sinc}(\tfrac{B}{\alpha\pi}f))^{l} * \mathrm{rect}_{\frac{\pi}{2B}}(f), \end{aligned}$$

where $b_0 = \Theta(B\sqrt{l}/\alpha)$ is the normalization factor such that $\widehat{G}(0) = 1$.

Definition 12.2 (Filter for bins)**.** Given $B > 1$, $\delta > 0$, $\alpha > 0$, let

$$\widehat{G}(f) := \widehat{G}_{B,\delta,\alpha}(2\pi(1-\alpha)f)$$

where $G_{B,\delta,\alpha}$ is defined in Definition 12.1. For any $\sigma > 0, b \in \mathbb{R}$ and $j \in [B]$, define

$$G^{(j)}_{\sigma,b}(t) := \frac{1}{\sigma}G(t/\sigma)e^{2\pi \mathbf{i} t(j/B-\sigma b)/\sigma},$$

and its Fourier transformation:

$$\widehat{G}^{(j)}_{\sigma,b}(f) = \sum_{i\in\mathbb{Z}} \widehat{G}(\sigma f + \sigma b - i - \frac{j}{B}).$$

Then, we provide several properties of G and $G^{(j)}_{\sigma,b}(t)$, which is proven by [CKPS16].

Lemma 12.8 (*G*-filter's properties, [CKPS16])**.** *Given $B > 1$, $\delta > 0$, $\alpha > 0$, let $G := G_{B,\delta,\alpha}(t)$ be defined in Definition 12.1. Then, G satisfies the following properties:*

$$\begin{aligned} \text{Property I}: &\quad \widehat{G}(f) \in [1-\delta/k, 1], \ \text{if } |f| \leq (1-\alpha)\frac{2\pi}{2B}. \\ \text{Property II}: &\quad \widehat{G}(f) \in [0, 1], \ \text{if } (1-\alpha)\frac{2\pi}{2B} \leq |f| \leq \frac{2\pi}{2B}. \\ \text{Property III}: &\quad \widehat{G}(f) \in [-\delta/k, \delta/k], \ \text{if } |f| > \frac{2\pi}{2B}. \\ \text{Property IV}: &\quad \mathrm{supp}(G(t)) \subset [\frac{l}{2}\cdot\frac{-B}{\pi\alpha}, \frac{l}{2}\cdot\frac{B}{\pi\alpha}]. \\ \text{Property V}: &\quad \max_t |G(t)| \lesssim \mathrm{poly}(B, l). \end{aligned}$$

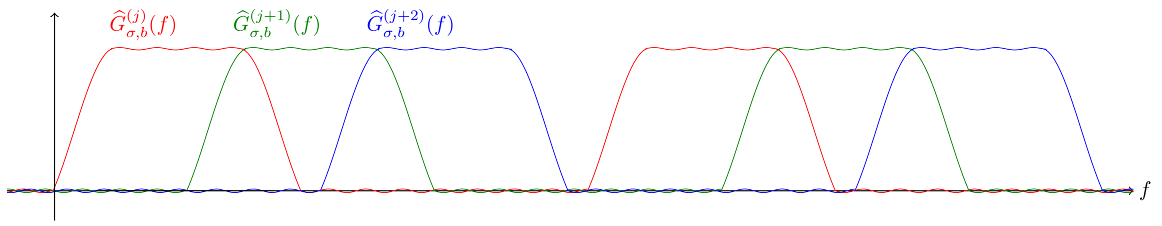


Figure 12.8: Filters with the frequency domain covering property. The red, green, and blue curves represent the filters $\widehat{G}^{(j)}_{\sigma,b}$, $\widehat{G}^{(j+1)}_{\sigma,b}$, and $\widehat{G}^{(j+2)}_{\sigma,b}$, respectively. The frequency domain covering property ensures that for each frequency $f \in \mathbb{R}$, there are *at least one but no more than two* filters satisfying $\widehat{G}^{(j)}_{\sigma,b}(f) \geq 1 - \delta$.

Lemma 12.9. *Let $G^{(j)}_{\sigma,b}(t)$ be defined in Definition 12.2. Let offset function*

$$o_{\sigma,b}(f) = |(\sigma f + \sigma b - \frac{j}{B}) - \frac{1}{2} \pmod 1| - \frac{1}{2}.$$

Then,

$$\begin{aligned}
\text{Property I}: &\quad \widehat{G}^{(b)}_{\sigma,b}(f) \in [1-\delta/k, 1], \text{ if } |o_{\sigma,b}(f)| \leq (1-\alpha)\frac{2\pi}{2B}.\\
\text{Property II}: &\quad \widehat{G}^{(b)}_{\sigma,b}(f) \in [0, 1], \text{ if } (1-\alpha)\frac{2\pi}{2B} \leq |o_{\sigma,b}(f)| \leq \frac{2\pi}{2B}.\\
\text{Property III}: &\quad \widehat{G}^{(b)}_{\sigma,b}(f) \in [-\delta/k, \delta/k], \text{ if } |o_{\sigma,b}(f)| > \frac{2\pi}{2B}.\\
\text{Property IV}: &\quad \operatorname{supp}(G^{(b)}_{\sigma,b}(t)) \subset [\frac{l}{2} \cdot \frac{-B}{\pi\alpha}, \frac{l}{2} \cdot \frac{B}{\pi\alpha}].\\
\text{Property V}: &\quad \max_t |G^{(b)}_{\sigma,b}(t)| \lesssim \operatorname{poly}(B, l).
\end{aligned}$$

12.6.2 Frequency domain covering

In this section, we show that the filter functions $\{\widehat{G}^{(j)}_{\sigma,b}\}_{j\in[B]}$ form a proper cover for the frequency domain. Roughly speaking, for any frequency $f \in \mathbb{R}$, we show that the sum of all filters' values (squared) at f is very close to one. This property is very important for our high SNR band assumption.

Lemma 12.10. *For any $f \in \mathbb{R}$, there exists at least one $j \in [B]$ such that*

$$\widehat{G}^{(j)}_{\sigma,b}(f) \geq 1 - \frac{\delta}{k}.$$

Proof. We first prove that the lemma holds for those f^* where there exists a $j \in [B]$ such that

$$f^* \in [-b + \frac{j}{\sigma B} - \frac{1}{2\sigma B}, -b + \frac{j}{\sigma B} + \frac{1}{2\sigma B}] + \frac{1}{\sigma}\mathbb{Z}.$$

For such f^*, we have

$$\begin{aligned} \widehat{G}^{(j)}_{\sigma,b}(f^*) = & \sum_{i \in \mathbb{Z}} \widehat{G}(\sigma f^* + \sigma b - i - \frac{j}{B}) \\ \geq & \widehat{G}((\sigma f^* + \sigma b - \frac{j}{B}) \mod 1) \\ \geq & 1 - \frac{\delta}{k}, \end{aligned}$$

where the first step follows from the definition of $\widehat{G}^{(j)}_{\sigma,b}(f^*)$, the second step is straight forward, the third step follows from

$$\sigma f^* + \sigma b - \frac{j}{B} \mod \frac{1}{\sigma} \in [-\frac{1}{2B}, \frac{1}{2B}]$$

and Lemma 12.9 Property I and Definition 12.2.

It remains to show that for an arbitrary $f \in \mathbb{R}$, the condition still holds. Let

$$j := \mathrm{round}((\sigma f + \sigma b \mod 1) \cdot B).$$

We have

$$j \in [(\sigma f + \sigma b \mod 1) \cdot B - \frac{1}{2}, (\sigma f + \sigma b \mod 1) \cdot B + \frac{1}{2}],$$

which implies that

$$j \in [(\sigma f + \sigma b) \cdot B - \frac{1}{2}, (\sigma f + \sigma b) \cdot B + \frac{1}{2}] + B\mathbb{Z}.$$

Thus,

$$f \in [-b + \frac{j}{\sigma B} - \frac{1}{2\sigma B}, -b + \frac{j}{\sigma B} + \frac{1}{2\sigma B}] + \frac{1}{\sigma}\mathbb{Z}.$$

The lemma is then proved. $\square$

Lemma 12.11. *For any $f \in \mathbb{R}$,*

$$\sum_{j=1}^{B} |\widehat{G}_{\sigma,b}^{(j)}(f)|^2 \eqsim 1.$$

Proof. By Lemma 12.10, we have that for any $f \in \mathbb{R}$, there exist at least a $j_0 \in [B]$ such that

$$\widehat{G}_{\sigma,b}^{(j_0)}(f) \geq \frac{1}{2}. \tag{12.5}$$

Moreover, we have that

$$B\frac{\delta}{k} = O(\delta) \leq 0.01. \tag{12.6}$$

where the first step follows from $B = O(k)$, the second step follows from $\delta = o(1) \leq 0.01$.

In the followings, we give lower and upper bounds for $\sum_{j=1}^{B} |\widehat{G}_{\sigma,b}^{(j)}(f)|^2$.

Lower bound:

$$\sum_{j=1}^{B} |\widehat{G}_{\sigma,b}^{(j)}(f)|^2 \geq |\widehat{G}_{\sigma,b}^{(j_0)}(f)|^2 \gtrsim 1.$$

where the first step follows from Lemma 12.9 Property I, II, and III, the second step follows from Eq. (12.5), the third step follows from Eq. (12.6) and $\delta/k \leq 1$.

Upper bound:

$$\sum_{j=1}^{B} (\widehat{G}_{\sigma,b}^{(j)}(f))^2 \leq 2 + B(\frac{\delta}{k})^2 \lesssim 1$$

where the first step follows from the definition of $\widehat{G}_{\sigma,b}^{(j)}(f)$, the second step follows from Eq. (12.6) and $\delta/k \leq 1$.

Combining them together, the lemma follows. $\square$

12.7 Hashing the Frequencies

In this section, we review the HashToBins strategy, which an important tool for Sparse Fourier Transform [HIKP12a, IKP14, PS15, CKPS16, Kap16, Kap17, JLS23]. Ideally, the HashToBins procedure randomly splits the frequency domain into B bins so that each bin contains at most one frequency. Then, the k-sparse Fourier reconstruction problem is reduced to a much easier one-sparse Fourier reconstruction problem.

We first describe the hashing strategy(see Section 12.7.1). However, there are two kinds of bad events such that the HashToBins procedure cannot work as good as we want: two frequencies are hashed to the same bin, or some frequency lies close to the boundary of a bin. We show that these bad events only happen with small probabilities (see Sections 12.7.2 and 12.7.3) .

12.7.1 HashToBins procedure

Here, we introduce the hash function and how to compute the resulting signal of the HashToBins procedure.

We first give the definition of the hashing function:

Definition 12.3 (Hash function, [CKPS16])**.** Let $\pi_{\sigma,b}(f) = \sigma(f+b) \pmod 1$ and $h_{\sigma,b}(f) = \mathrm{round}(\pi_{\sigma,b}(f) \cdot B)$ be the hash function that maps frequency $f \in [-F, F]$ into bins $\{0, \cdots, B-1\}$.

Intuitively, the j-th bin corresponding to f such that $\widehat{G}^{(j)}_{\sigma,b}(f) \geq 1 - \delta/k$. In general, we set $B = \Theta(k)$ and $\sigma \in [\frac{1}{B\Delta}, \frac{2}{B\Delta}]$ chosen uniformly at random, where $\Delta = k \cdot |\mathrm{supp}(\widehat{H}(f))|$.

Then, we show how to compute the HashToBins:

Lemma 12.12 (Lemma 6.9 in [CKPS16])**.** *Let $z_j(t) = x(t) * G^{(j)}_{\sigma,b}(t)$. Let $a := t/\sigma$*

Let $u \in \mathbb{C}^B$ and for $j \in [B]$,

$$u_j := \sum_{i \in \mathbb{Z}} x(\sigma(a - j - iB)) e^{-2\pi \mathbf{i} \sigma b (j+iB)} G(j + iB).$$

Then, we have that for all $j \in [B]$,

$$\widehat{u}_j = z_j(\sigma a).$$

Note that when we apply Lemma 12.12, we take $x(t) = x(t) \cdot H(t)$, where the latter $x(t)$ is the observable signal, the $H(t)$ is the filter of time domain (see Section 12.8).

12.7.2 Frequency isolation

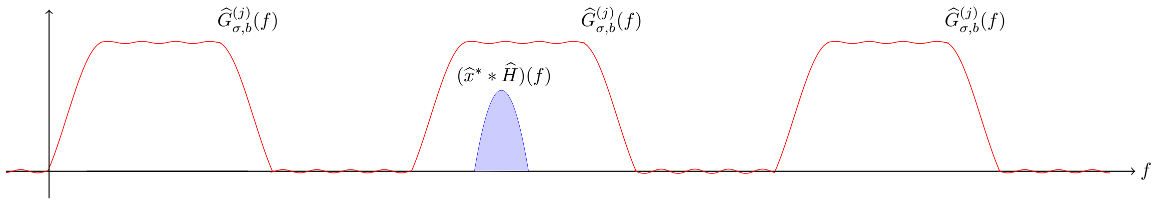

$\widehat{G}^{(j)}_{\sigma,b}(f)$
$\widehat{G}^{(j)}_{\sigma,b}(f)$
$\widehat{G}^{(j)}_{\sigma,b}(f)$
$(\widehat{x}^* * \widehat{H})(f)$
f

Figure 12.9: An example of the well-isolation event in the frequency domain. $\widehat{G}^{(j)}_{\sigma,b}$ (the red curve) is the filter of frequency domain for the j-th bin and $H(t)$ is the filter of time domain. $\widehat{x}^*(f) * \widehat{H}(f)$ (the blue curve) is the filtered ground-truth signal. Under the well-isolation event, there is one small interval that contains most of the energy. In other words, each bin only contains one-cluster of frequencies.

The goal of this section is to define and analyze the Frequency Isolation event. Frequency Isolation requires that the energy of the hashed signal in each bin is concentrated in a small band in the frequency domain. This condition is roughly equivalent to say that each bin only contains one cluster of frequencies. This condition is very useful in proving the concentration of the filtered signal in the frequency domain, which serves as one of the basic assumptions of our significant sample generation procedure.

We first restate Claim 12.13, which was proved in [CKPS16]. This claim states that if two frequencies are not close to each other, with a large probability, they are also not hashed into the same bin.

Claim 12.13 (Collision probability, [CKPS16])**.** *For any $\Delta_0 > 0$, let σ be a sample uniformly at random from $[\frac{1}{4B\Delta_0}, \frac{1}{2B\Delta_0}]$. Then, we have:*

1. *If $4\Delta_0 \leq |f^+ - f^-| < 2(B-1)\Delta_0$, then $\mathsf{Pr}[h_{\sigma,b}(f^+) = h_{\sigma,b}(f^-)] = 0$.*
2. *If $2(B-1)\Delta_0 \leq |f^+ - f^-|$, then $\mathsf{Pr}[h_{\sigma,b}(f^+) = h_{\sigma,b}(f^-)] \lesssim \frac{1}{B}$.*

We then recall the formal definition of the well-isolation event:

Definition 12.4 (Well-isolation condition)**.** We say that a frequency f^* is *well-isolated* under the hashing parameters (σ, b) if, for $j = h_{\sigma,b}(f^*)$, the hashed signal (in frequency domain) $\widehat{z}^{(j)}(f) := \widehat{x \cdot H}(f) \cdot \widehat{G}^{(j)}_{\sigma,b}(f)$ satisfies

$$\int_{\overline{I_{f^*}}} |\widehat{z}^{(j)}(f)|^2 df \lesssim \varepsilon \cdot T\mathcal{N}^2/k,$$

over the interval $\overline{I_{f^*}} = (-\infty, \infty) \setminus (f^* - \Delta, f^* + \Delta)$.

The following lemma shows the probability of the Frequency Isolation event under the randomized hashing functions:

Lemma 12.14 (Lemma 7.19 in [CKPS16])**.** *Let f^* be any frequency. Then f^* is well-isolated by hashing parameters (σ, b) with probability ≥ 0.9.*

12.7.3 Large offset event

Large offset event is another kind of bad event for the HashToBins procedure, which happens when a ground-truth frequency is hashed into the changing edge of the filter $G^{(j)}_{\sigma,b}$. The large offset event breaks the guarantee of our signal equivalent method, and thus affects the performance of our significant sample generation and frequency estimation. Fortunately, this bad event only happens with a small probability.

We first state a tool for analyzing the hashing procedure, which intuitively says that the modular of a random sampling from a long interval is almost uniformly distributed:

Lemma 12.15 ([PS15, CKPS16])**.** *For any $\widetilde{T}$, and $0 \leq \widetilde{\varepsilon}, \widetilde{\delta} \leq \widetilde{T}$, if we sample $\widetilde{\sigma}$ uniformly at random from $[A, 2A]$, then*

$$\frac{2\widetilde{\varepsilon}}{\widetilde{T}} - \frac{2\widetilde{\varepsilon}}{A} \leq \Pr\left[\widetilde{\sigma} \pmod{\widetilde{T}} \in [\widetilde{\delta} - \widetilde{\varepsilon}, \widetilde{\delta} + \widetilde{\varepsilon}\,]\right] \leq \frac{2\widetilde{\varepsilon}}{\widetilde{T}} + \frac{4\widetilde{\varepsilon}}{A}. \tag{12.7}$$

Then, we define the large offset event:

Definition 12.5 (Large offset event)**.** Given $\sigma \in \mathbb{R}_+, b \in \mathbb{R}$. Let $G_{\sigma,b}^{(j)}$ and δ be defined as in Definition 12.1. For any k-Fourier-sparse signal x, we say the *Large Offset event* happens, if for any $f \in \mathrm{supp}(\widehat{x \cdot H})$ and any $j \in [B]$,

$$\widehat{G}_{\sigma,b}^{(j)}(f) \in \left[\frac{\delta}{k}, 1 - \frac{\delta}{k}\right].$$

We analyze the probability of large offset event in the following lemma:

Lemma 12.16. *Let $\Delta_0 = O(\Delta)$, $\widehat{\sigma} = 1/\Delta_0$. Given $b = O(\max\{F, 1/\widehat{\sigma}\})$, suppose $\sigma \sim [0.5\widehat{\sigma}, \widehat{\sigma}]$ uniformly at random. Then, with probability at least 0.99, the Large Offset event does not happen.*

Furthermore, with probability at least 0.99, for any $j \in [k]$, for any $f \in f_j + \mathrm{supp}(\widehat{H})$, it holds that $\widehat{G}_{\sigma,b}^{(j)}(f) \notin [\delta/k, 1 - \delta/k]$.

Proof. Let α be defined as in Definition 12.1. Let $I_G := \{f \in \mathbb{R} \mid \widehat{G}_{\sigma,b}^{(j)}(f) \in [\delta/k, 1 - \delta/k]\}$. Following from Lemma 12.9 Property II, we have that $s_G := |I_G \pmod{1/\sigma}| \leq 10\alpha\Delta_0/B$.

Let $\delta_{f^*}(f)$ be the Dirichlet function at f^*. For any f_j with $j \in [k]$, let $I_{f_j} := \mathrm{supp}(\widehat{H} * \delta_{f_j})$. We also define

$$I'_{f_j} := \{f \in \mathbb{R} \mid [f - s_G, f + s_G] \cap I_{f_j} \neq \emptyset\}.$$

Since $\mathrm{supp}(\widehat{x \cdot H}) = \mathrm{supp}(\widehat{H} * \widehat{x}) \subseteq \bigcup_{j=1}^{k} \mathrm{supp}(\widehat{H} * \delta_{f_j})$, we know that the Large Offset event happens if

$$\Big(\bigcup_{j=1}^{k} I_{f_j} \Big) \cap I_G \neq \emptyset.$$

Thus, it suffices to bound $\mathsf{Pr}[(\cup_{j=1}^{k} I_{f_j}) \cap I_G \neq \emptyset]$.

First, for any $j \in [k]$, we have

$$|I'_{f_j}| \leq |I_{f_j}| + 2s_G \leq \Delta/B + 2s_G \leq O(\Delta/B) \tag{12.8}$$

where the first step follows from the definition of Δ, the second step follows from $s_G \leq 10\alpha\Delta_0/B$ and the setting of α.

We have that

$$\begin{aligned}
&\mathsf{Pr}\Big[\frac{1}{2B\sigma} + \frac{j}{B\sigma} - b \pmod{1/\sigma} \in I'_{f_j} \pmod{1/\sigma}\Big] \\
= {}& \mathsf{Pr}\Big[\frac{1}{2B} + \frac{j}{B} \pmod 1 \in \sigma b + \sigma I'_{f_j} \pmod 1\Big] \\
= {}& \mathsf{Pr}\Big[\sigma b + \sigma f_j \pmod 1 \in \frac{1}{2B} + \frac{j}{B} + \sigma[-|I'_{f_j}|/2, |I'_{f_j}|/2] \pmod 1\Big] \\
\leq {}& \mathsf{Pr}\Big[\sigma b + \sigma f_j \pmod 1 \in \frac{1}{2B} + \frac{j}{B} + \widehat{\sigma}[-|I'_{f_j}|/2, |I'_{f_j}|/2] \pmod 1\Big] \\
\leq {}& \widehat{\sigma}|I'_{f_j}| + \frac{2\widehat{\sigma}|I'_{f_j}|}{0.5\widehat{\sigma}b + 0.5\widehat{\sigma}f_j} \\
\leq {}& 2\widehat{\sigma}|I'_{f_j}| \\
\leq {}& 2\widehat{\sigma} \cdot O(\Delta/B) \\
\leq {}& O(1/B)
\end{aligned} \tag{12.9}$$

where the first steps are straightforward, the second step follows from the center of I'_{f_j} is f_j, the length of the interval I'_{f_j} is $|I'_{f_j}|$, and $a \in [c-b, c+b] \Rightarrow c \in [a-b, a+b]$, the third step follows from $\sigma \leq \widehat{\sigma}$, the forth step follows by applying Lemma 12.15

with the following parameters setting:

$$\begin{aligned}
\widetilde{T} &= 1, \\
\widetilde{\delta} &= \frac{1}{2B} + \frac{j}{B}, \\
\widetilde{\varepsilon} &= \widehat{\sigma}|I'_{f_j}|/2, \\
A &= 0.5\widehat{\sigma}b + 0.5\widehat{\sigma}f_j, \\
\widetilde{\sigma} &= \sigma b + \sigma f_j,
\end{aligned}$$

the fifth step follows from $0.5b \geq F \geq f_j$ and $0.5b\widehat{\sigma} \geq 1$, the sixth step follows from Eq. (12.8), the last step follows from the definition of $\widehat{\sigma}$.

Similarly, we have that

$$\Pr[-\frac{1}{2B\sigma} + \frac{j}{B\sigma} - b \pmod{1/\sigma} \in I'_{f_j} \pmod{1/\sigma}] \leq O(1/B) \tag{12.10}$$

Note that I_G is the edge of filter $G^{(j)}_{\sigma,b}$, under the meaning of module $1/\sigma$, the center of $G^{(j)}_{\sigma,b}$ is $\frac{j}{B\sigma} - b$, the length of $G^{(j)}_{\sigma,b}$ is $\frac{1}{B\sigma}$, the length of the edge is s_G. Moreover, I_{f_j} is an interval center at f_j and length $|\mathrm{supp}(\widehat{H})|$. We can judge whether two interval have intersect $I_{f_j} \cap I_G \neq \emptyset$ by moving the length of one interval to another and judging whether $-\frac{1}{2B\sigma} + \frac{j}{B\sigma} - b \pmod{1/\sigma}$, $\frac{1}{2B\sigma} + \frac{j}{B\sigma} - b \pmod{1/\sigma}$ (the end point of I_G) contains in I'_{f_j}. By combining Eq. (12.9) and Eq. (12.10), we have that

$$\Pr[I_{f_j} \cap I_G \neq \emptyset] \leq O(1/B) + O(1/B) = O(1/B). \tag{12.11}$$

Therefore, by a union bound over all $j \in [k]$, we get that

$$\Pr[(\cup_{j=1}^{k} I_{f_j}) \cap I_G \neq \emptyset] \leq \sum_{j=1}^{k} \Pr[I_{f_j} \cap I_G \neq \emptyset] \leq \sum_{j=1}^{k} O(1/B) \leq 0.01,$$

where the first step is by union bound, the second step follows from Eq. (12.11), and the last step follows from $B = O(k)$. By the definitions of I_{f_j} and I_G, it implies that with probability at least 0.99, for any $j \in [k]$, and any $f \in f_j + \mathrm{supp}(\widehat{H})$, $\widehat{G}^{(j)}_{\sigma,b}(f) \notin [\delta/k, 1 - \delta/k]$.

The proof of the lemma is then completed. $\square$

Lemma 12.17. *For $x^*(t)$ be a k-Fourier-sparse signal. For frequency $f^* \in \mathrm{supp}(\widehat{x}^*)$, let $j = h_{\sigma,b}(f^*)$ be the bin that f^* hashed into. If Large Offset event not happens, then for $f \in \mathrm{supp}(\widehat{x}^* * \widehat{H})$,*

$$\widehat{G}^{(j)}_{\sigma,b}(f) \in [1-\delta/k, 1]$$

Proof. Since Large Offset event not happens, for $f \in \mathbb{R}$,

$$\widehat{G}^{(j)}_{\sigma,b}(f) \geq 1-\delta/k \text{ or } \widehat{G}^{(j)}_{\sigma,b}(f) \leq \delta/k.$$

Since Large Offset event not happens and $j = h_{\sigma,b}(f^*)$, we have that for $f \in \mathrm{supp}(\widehat{x}^* * \widehat{H})$,

$$\widehat{G}^{(j)}_{\sigma,b}(f) \geq 1-\delta/k.$$

By Lemma 12.9 Property I, II, and III, we have that

$$\widehat{G}^{(j)}_{\sigma,b}(f) \in [1-\delta/k, 1].$$

$\square$

12.8 Filter in Time Domain

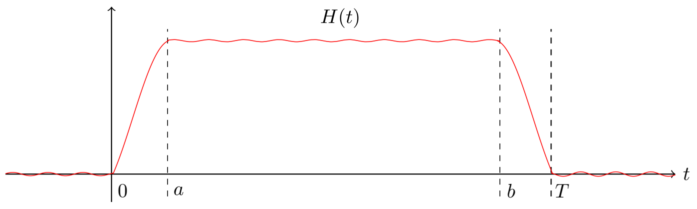


Figure 12.10: The filter $H(t)$ of time domain. We use decay region to refer $[0,a]$ and $[b,T]$. We use fluctuation region to refer $[a,b]$.

In this section, we discuss the filter $H(t)$ of time domain, which is an analogous of the ideal filter $\mathrm{rect}_T(t)$. In the Fourier interpolation problem, we only care about the time duration $[0,T]$. Thus, applying the filter $\mathrm{rect}_T(t)$ to the observation signal

$x(t)$ can cut-off the unobservable part and much simplify the analysis. Since $\mathrm{rect}_T(t)$ have an infinite band width, for efficient computation, we need to truncate $\mathrm{rect}_T(t)$'s frequency domain to a $\mathrm{poly}(k)/T$-length interval. However, the frequency truncation loses the high frequency components of the ideal filter $\mathrm{rect}_T(t)$, and the resulting filter $H(t)$ is no longer sharp around the boundary of $[0,T]$. More specifically, $H(t)$ is exponentially close to 1 within $[T/\mathrm{poly}(k), T(1-1/\mathrm{poly}(k))]$, and exponentially close to 0 outside $[0,T]$.

We first provide the construction of $H(t)$ in [CP19b] and review some known properties (see Section 12.8.1). Then, we discuss the normalization factor of the filter and provide a polynomial upper bound of it (see Section 12.8.2). This bound is crucial for our fluctuation bound of the $H(t)$. Next, we bound the fluctuation of $H(t)$ during a shrinking interval $[T/\mathrm{poly}(k), T(1-1/\mathrm{poly}(k))]$ and prove that $H(t)$ is exponentially close to 1 in that range (see Section 12.8.3). Furthermore, we prove that $H(t)$ preserve the energy of Fourier sparse signal in the duration $[0,T]$ (see Section 12.8.4).

12.8.1 Time domain filter construction

We first introduce an growth rate bound from [CP19b]. This theorem bound the growth of Fourier sparse signal outside of the duration $[0,T]$ by an exponent function of base t. This bound in this theorem is high related to the size of the support set of $\widehat{H}(f)$.

Theorem 12.18 ([CP19b]). *There exists $S = O(k^2 \log k)$ such that for any $|t| > T$ and $g(t) = \sum_{j=1}^{k} v_j \cdot e^{2\pi \mathbf{i} f_j t}$, $|g(t)|^2 \leq \mathrm{poly}(k) \cdot \mathop{\mathbb{E}}_{x \in [-T,T]}[|g(x)|^2] \cdot |\frac{t}{T}|^S$.*

The definition of the time domain filter $H(t)$ in [CP19b] is given in below. Intuitively, it uses some powers of $\mathrm{sinc}(t)$ to approximate $\delta_0(t) * \mathrm{rect}_1(t) = \mathrm{rect}_1(t)$, thus one can get finite band-limit and good approximation at the same time.

Definition 12.6 (Definition 4.1 in [CP19b])**.** Given an energy bound R satisfying

$$|x(t)|^2 \lesssim R\|x(t)\|_T^2, \quad \forall t \in [0,T] \text{ and } k\text{-Fourier sparse signal } x(t),$$

the growth rate S a power of two, $C \in 2\mathbb{Z}$, and $C_0 \in \pi\mathbb{Z}$,we define the filter function:

$$H_1(t) = s_0 \cdot \left(\mathrm{sinc}(C_0 R \cdot t)^{C \log R} \cdot \mathrm{sinc}\left(C_0 \cdot S \cdot t\right)^C \cdot \mathrm{sinc}\,(\frac{C_0 \cdot S}{2} \cdot t)^{2C} \cdots \mathrm{sinc}\left(C_0 \cdot t\right)^{C \cdot S} \right) * \mathrm{rect}_1(t),$$

where $s_0 \in \mathbb{R}^+$ is a parameter to normalize $H_1(0) = 1$. Its Fourier transform is as follows:

$$\widehat{H}_1(f) = s_0 \cdot \left(\mathrm{rect}_{C_0 R}(f)^{*C \log R} * \mathrm{rect}_{C_0 \cdot S}(f)^{*C} * \mathrm{rect}_{\frac{C_0 \cdot S}{2}}(f)^{*2C} * \cdots * \mathrm{rect}_{C_0}(f)^{*CS} \right) \cdot \mathrm{sinc}(f/2).$$

We then state some basic properties of the time domain filter in [CP19b]. The following lemma bounds the support size of $\widehat{H}_1(f)$:

Lemma 12.19 ([CP19b])**.** *Let $C_0 = \Theta(C)$. we have that*

$$|\mathrm{supp}(\widehat{H}_1(f))| = O(C^2 R \log R + C^2 S \log S).$$

The following theorem shows some time domain properties of the filter:

Theorem 12.20 (Theorem 4.2 in [CP19b])**.** *Let $R, S > 0$, let $C \in 2\mathbb{Z}$, $C_0 \in \pi\mathbb{Z}$, $C_0 = \Theta(C)$, and define $\alpha = (\frac{1}{2} + \frac{1.2}{\pi C_0 R})$. Consider any function x satisfying the following two conditions:*

1. $\sup_{t \in [-1,1]} \left[|x(t)|^2\right] \leq R \cdot \mathbb{E}_{t \in [-1,1]} \left[|x(t)|^2\right]$,
2. $\mathrm{poly}(R) \cdot \mathbb{E}_{t \in [-1,1]} \left[|x(t)|^2\right] \cdot |t|^S$ *for* $t \notin [-1,1]$,

Then, we have that the filter function $H(t) = H_1\big(\alpha t\big)$ satisfies

- *Part 1.* $\int_{-1}^{1} |x(t) \cdot H\big(t\big)|^2 \mathrm{d}t \geq 0.9 \int_{-1}^{1} |x(t)|^2 \mathrm{d}t$,
- *Part 2.* $\int_{-1}^{1} |x(t) \cdot H\big(t\big)|^2 \mathrm{d}t \geq 0.95 \int_{-\infty}^{\infty} |x(t) \cdot H\big(t\big)|^2 \mathrm{d}t$,

- *Part 3.* $|H(t)| \leq 1.01$ *for any* t.

Throughout this chapter, we denote $H(t)$ as the following re-scaling of $H_1(t)$:

Definition 12.7. Let $\alpha = (\frac{1}{2} + \frac{1.2}{\pi C_0 R})$. Let $H_1(t)$ be defined as in Definition 12.6. The filter $H(t)$ is defined as:

$$H(t) := H_1(\alpha t).$$

and setting $R = S = O(k^2)$, $R = 2^s$, where $s \in \mathbb{Z}_+$, $C = O(\log(1/\delta_1))$, $C \in 2\mathbb{Z}$, $C_0 = \Theta(C)$ and $C_0 \in \pi\mathbb{Z}$.

12.8.2 Normalization factor of the filter

The goal of this section is to prove Lemma 12.21, an upper-bound for the normalization factor s_0. This lemma will be used later to ensure that the scaling factor will not break the exponential small fluctuation of Section 12.8.3. We note that the same result has been proved in [CP19b], and we reprove it below for completeness.

Lemma 12.21 (Lemma 7.2 in [CP19b])**.** *It holds that*

$$s_0 \leq O(CR\sqrt{C \log R}).$$

Proof. We first have that,

$$\begin{aligned} H_1(t) &= s_0 \cdot (\mathrm{sinc}(C_0 R \cdot t)^{C \log R} \cdot \prod_{i=0}^{\log(S)} \mathrm{sinc}\,\big(\frac{C_0 \cdot S}{2^i} \cdot t\big)^{2^i \cdot C}) * \mathrm{rect}_1(t) \\ &= s_0 \cdot \int_{-\infty}^{\infty} \mathrm{sinc}(C_0 R \cdot \tau)^{C \log R} \cdot \prod_{i=0}^{\log(S)} \mathrm{sinc}\,\big(\frac{C_0 \cdot S}{2^i} \cdot \tau\big)^{2^i \cdot C} \cdot \mathrm{rect}_1(t-\tau) \mathrm{d}\tau \\ &= s_0 \cdot \int_{t-0.5}^{t+0.5} \mathrm{sinc}(C_0 R \cdot \tau)^{C \log R} \cdot \prod_{i=0}^{\log(S)} \mathrm{sinc}\,\big(\frac{C_0 \cdot S}{2^i} \cdot \tau\big)^{2^i \cdot C} \mathrm{d}\tau, \end{aligned}$$

where the first step follows from the definition of $H_1(t)$, the second step follows from the definition of the convolution, the third step follows from the definition of $\mathrm{rect}_2(t)$ function. Thus,

$$H_1(0) = s_0 \cdot \int_{-0.5}^{+0.5} \mathrm{sinc}(C_0 R \cdot \tau)^{C \log R} \cdot \prod_{i=0}^{\log(S)} \mathrm{sinc}\big(\frac{C_0 \cdot S}{2^i} \cdot \tau\big)^{2^i \cdot C} \mathrm{d}\tau$$

Let $U := (2C_0 \log R)^{-1/2}$. We have that

$$\begin{aligned}
&\int_{-0.5}^{+0.5} \mathrm{sinc}(C_0 R \cdot \tau)^{C \log R} \cdot \prod_{i=0}^{\log(S)} \mathrm{sinc}\big(\frac{C_0 \cdot S}{2^i} \cdot \tau\big)^{2^i \cdot C} \mathrm{d}\tau \\
\geq &\int_{-\frac{1}{\pi C_0 R}}^{+\frac{1}{\pi C_0 R}} \mathrm{sinc}(C_0 R \cdot \tau)^{C \log R} \cdot \prod_{i=0}^{\log(S)} \mathrm{sinc}\big(\frac{C_0 \cdot S}{2^i} \cdot \tau\big)^{2^i \cdot C} \mathrm{d}\tau \\
= &\frac{1}{\pi C_0 R} \int_{-1}^{+1} \mathrm{sinc}(\frac{\upsilon}{\pi})^{C \log R} \cdot \prod_{i=0}^{\log(S)} \mathrm{sinc}\big(\frac{\upsilon}{2^i \pi}\big)^{2^i \cdot C} \mathrm{d}\upsilon \\
\geq &\frac{1}{\pi C_0 R} \int_{-U}^{+U} \mathrm{sinc}(\frac{\upsilon}{\pi})^{C \log R} \cdot \prod_{i=0}^{\log(S)} \mathrm{sinc}\big(\frac{\upsilon}{2^i \pi}\big)^{2^i \cdot C} \mathrm{d}\upsilon \\
\geq &\frac{1}{\pi C_0 R} \int_{-U}^{+U} (1 - \frac{\upsilon^2}{6})^{C \log R} \cdot \prod_{i=0}^{\log(S)} (1 - \frac{\upsilon^2}{4^i 6})^{2^i \cdot C} \mathrm{d}\upsilon \\
\geq &\frac{1}{\pi C_0 R} \int_{-U}^{+U} (1 - C \log R \cdot \frac{\upsilon^2}{6} - \sum_{i=0}^{\log(S)} C \cdot \frac{\upsilon^2}{2^i 6}) \mathrm{d}\upsilon \\
\geq &\frac{1}{\pi C_0 R} \int_{-U}^{+U} (1 - C \log R \cdot \frac{\upsilon^2}{3}) \mathrm{d}\upsilon \\
= &\frac{1}{\pi C_0 R} (2U - 2C \log R \cdot \frac{U^3}{9}) \\
\geq &\frac{1}{\pi C_0 R \sqrt{2 C_0 \log R}},
\end{aligned} \tag{12.12}$$

where the first step follows from $R = O(k^2)$, $C_0 = O(\log(1/\delta_1))$, $0.5 \geq \frac{1}{\pi C_0 R}$, second step follows from changing the variable $\nu = \pi C R \cdot \tau$, the third step follows from $U < 1$, the forth step follows from Fact 12.22, the fifth step is follows from $(1-a)(1-b) \geq 1 - a - b$, the sixth step follows from $\log(R) > 2$, the seventh step is straight forward, the eighth step follows from setting $U = (2C_0 \log R)^{-1/2}$ and $C_0 = \Theta(C)$.

As a result, we have that

$$s_0 \leq H_1(0) \cdot \pi C R \sqrt{2C \log R} = \pi C R \sqrt{2C \log R}$$

where the first step follows from Eq. (12.12), the second step follows from $H_1(0) = 1$.

□

Fact 12.22. *For any $t \in \mathbb{R}$,*

$$1 - \frac{(\pi t)^2}{3!} \leq \mathrm{sinc}(t) \leq 1.$$

12.8.3 Fluctuation bound

The idea filter $\mathrm{rect}_T(t)$ has a constant value 1 in the interval $[0, T]$. Due to the frequency domain truncation in $H(t)$, it deviates from $\mathrm{rect}_T(t)$ with different magnitudes in different regions. In this section, we prove the Lemma 12.23, which shows that $H(t)$ is fluctuating near 1 in the "interior" of $[0, T]$ (i.e., $[0 + \frac{T}{\mathrm{poly}(k)}, T - \frac{T}{\mathrm{poly}(k)}]$). It serves as an important tool for analyzing the error in our signal equivalent method.

Lemma 12.23. *For filter $H_1(t)$ defined in Definition 12.6 with the parameters $C = \log(1/\delta_1)$, $C_0 = \Theta(C)$, $R = S$, and $S = 2^s$ (where $s \in \mathbb{Z}_+$), $C_0 \in \pi\mathbb{Z}$, we have that*

$$H_1(t) \in [1 - \delta_1, 1], \forall |t| < 0.5 - \frac{\pi}{C_0 R}.$$

Moreover, $H(t) \in [1 - \delta_1, 1]$ for any $t \in [\frac{T}{2} - \alpha^{-1}(\frac{1}{2} - \frac{\pi}{C_0 R})\frac{T}{2}, \frac{T}{2} + \alpha^{-1}(\frac{1}{2} - \frac{\pi}{C_0 R})\frac{T}{2}]$.

Proof. The proof consists of two parts: upper bound and lower bound. For the upper bound, the idea is to compare the value of $H(t)$ with $H(0)$ by analyzing the gradient of $H(t)$. And the lower bound follows from directly estimating the integral of the product of sinc functions.

Upper bound: We have that $H_1(0) = 1$ by definition. We will show $H_1(t) \leq 1$ by proving $H_1(t)$ is monotonically decreasing in t.

We have that,

$$\begin{aligned} H_1(t) &= s_0 \cdot (\mathrm{sinc}(C_0 R \cdot t)^{C \log R} \cdot \prod_{i=0}^{\log(S)} \mathrm{sinc}\big(\frac{C_0 \cdot S}{2^i} \cdot t\big)^{2^i \cdot C}) * \mathrm{rect}_1(t) \\ &= s_0 \cdot \int_{-\infty}^{\infty} \mathrm{sinc}(C_0 R \cdot \tau)^{C \log R} \cdot \prod_{i=0}^{\log(S)} \mathrm{sinc}\big(\frac{C_0 \cdot S}{2^i} \cdot \tau\big)^{2^i \cdot C} \cdot \mathrm{rect}_1(t - \tau) \mathrm{d}\tau \\ &= s_0 \cdot \int_{t-0.5}^{t+0.5} \mathrm{sinc}(C_0 R \cdot \tau)^{C \log R} \cdot \prod_{i=0}^{\log(S)} \mathrm{sinc}\big(\frac{C_0 \cdot S}{2^i} \cdot \tau\big)^{2^i \cdot C} \mathrm{d}\tau, \end{aligned} \tag{12.13}$$

where the first step follows from the definition of $H_1(t)$, the second step follows from the definition of the convolution, the third step follows from the definition of $\mathrm{rect}_1(t)$ function.

Since $C \log R \in 2\mathbb{Z}$, $2^i \cdot C \in 2\mathbb{Z}$, we have that

$$\mathrm{sinc}(C_0 R \cdot \tau)^{C \log R} \geq 0,$$

and

$$\mathrm{sinc}\big(\frac{C_0 \cdot S}{2^i} \cdot \tau\big)^{2^i \cdot C} \geq 0.$$

Moreover, by setting $C_0 = \pi \mathbb{Z}$, we have that

$$2 \mod \frac{2\pi}{C_0} = 0. \tag{12.14}$$

By Eq. (12.14), we have that

$$\begin{aligned} \sin\big(\frac{C_0 \cdot S}{2^i} \cdot (t + 0.5)\big) &= \sin\big(\frac{C_0 \cdot S}{2^i} \cdot (t - 0.5)\big), \quad \forall i \in \{0, \cdots, \log(S)\}, \quad \text{and} \\ \sin(C_0 R \cdot (t + 0.5)) &= \sin(C_0 R \cdot (t - 0.5)). \end{aligned}$$

Furthermore, for any $t > 0$,

$$\begin{aligned} \big(\frac{C_0 \cdot S}{2^i} \cdot (t + 0.5)\big)^{-1} &\leq \big(\frac{C_0 \cdot S}{2^i} \cdot (t - 0.5)\big)^{-1}, \forall i \in \{0, \cdots, \log(S)\}, \quad \text{and} \\ (C_0 R \cdot (t + 0.5))^{-1} &\leq (C_0 R \cdot (t - 0.5))^{-1}. \end{aligned}$$

Thus,

$$\begin{aligned}|\operatorname{sinc}\big(\frac{C_0 \cdot S}{2^i} \cdot (t+0.5)\big)| \leq |\operatorname{sinc}\big(\frac{C_0 \cdot S}{2^i} \cdot (t-0.5)\big)|, \quad \forall i \in \{0, \cdots, \log(S)\}, \quad \text{and} \\ |\operatorname{sinc}(C_0 R \cdot (t+0.5))| \leq |\operatorname{sinc}(C_0 R \cdot (t-0.5))|. \end{aligned} \tag{12.15}$$

Then, we have that

$$\begin{aligned}\frac{H_1'(t)}{s_0} = &\ \operatorname{sinc}(C_0 R \cdot (t+0.5))^{C \log R} \cdot \prod_{i=0}^{\log(S)} \operatorname{sinc}\big(\frac{C_0 \cdot S}{2^i} \cdot (t+0.5)\big)^{2^i \cdot C} \\ &- \operatorname{sinc}(C_0 R \cdot (t-0.5))^{C \log R} \cdot \prod_{i=0}^{\log(S)} \operatorname{sinc}\big(\frac{C_0 \cdot S}{2^i} \cdot (t-0.5)\big)^{2^i \cdot C} \\ &< 0,\end{aligned}$$

where the first step is straight forward, the second step follows from Eq. (12.15).

Thus, $H_1(t) < H_1(0) = 1$ for any $t > 0$.

Similarly, we also have that $H_1(t) < H_1(0) = 1$ for any $t \leq 0$ since $H_1(t)$ is symmetric with respect to t.

Lower bound: We have that, for any $|t| < 0.5 - \frac{\pi}{C_0 R}$,

$$\begin{aligned}
& \int_{-\infty}^{t-0.5} \mathrm{sinc}(C_0 R \cdot \tau)^{C \log R} \cdot \prod_{i=0}^{\log(S)} \mathrm{sinc}\,\big(\frac{C_0 \cdot S}{2^i} \cdot \tau\big)^{2^i \cdot C} \mathrm{d}\tau \\
= & \int_{0.5-t}^{\infty} \mathrm{sinc}(C_0 R \cdot \tau)^{C \log(R)} \cdot \prod_{i=0}^{\log(S)} \mathrm{sinc}\,\big(\frac{C_0 \cdot S}{2^i} \cdot \tau\big)^{2^i \cdot C} \mathrm{d}\tau \\
\leq & \int_{0.5-t}^{\infty} \mathrm{sinc}(C_0 R \cdot \tau)^{C \log(R)} \mathrm{d}\tau \\
\leq & \int_{0.5-t}^{\infty} (C_0 R \cdot \tau)^{-C \log(R)} \mathrm{d}\tau \\
\leq & \int_{\frac{\pi}{C_0 R}}^{\infty} (C_0 R \cdot \tau)^{-C \log(R)} \mathrm{d}\tau \\
= & \frac{1}{C_0 R} \int_{\pi}^{\infty} \upsilon^{-C \log(R)} \mathrm{d}\upsilon \\
= & \frac{1}{C_0 R} \frac{1}{C \log(R) - 1} \pi^{-C \log(R) + 1} \\
\lesssim & \frac{1}{C_0^2 R \log(R)} \delta_1,
\end{aligned} \tag{12.16}$$

where the first step is straight forward, the second step follows from $\mathrm{sinc}(x) \leq 1$, the third step is follows from $\mathrm{sinc}(x) \leq 1/x$, the forth step follows follows from $0.5 - t \geq \frac{\pi}{C_0 R}$, the fifth step follows from $\upsilon := C R \tau$, the sixth step is straight forward, the seventh step follows from $\log(R) > 1$, $C \geq \log(1/\delta_1)$.

Hence, for any $t > 0$,

$$\begin{aligned}
H_1(t) &= s_0 \cdot \int_{t-0.5}^{t+0.5} \mathrm{sinc}(C_0 R \cdot \tau)^{C \log R} \cdot \prod_{i=0}^{\log(S)} \mathrm{sinc}\,\big(\frac{C_0 \cdot S}{2^i} \cdot \tau\big)^{2^i \cdot C} \mathrm{d}\tau \\
&= H_1(0) + s_0 \cdot \int_{1}^{t+0.5} \mathrm{sinc}(C_0 R \cdot \tau)^{C \log R} \cdot \prod_{i=0}^{\log(S)} \mathrm{sinc}\,\big(\frac{C_0 \cdot S}{2^i} \cdot \tau\big)^{2^i \cdot C} \mathrm{d}\tau \\
&\qquad - s_0 \cdot \int_{-1}^{t-0.5} \mathrm{sinc}(C_0 R \cdot \tau)^{C \log R} \cdot \prod_{i=0}^{\log(S)} \mathrm{sinc}\,\big(\frac{C_0 \cdot S}{2^i} \cdot \tau\big)^{2^i \cdot C} \mathrm{d}\tau \\
&\geq H_1(0) - s_0 \cdot \int_{-\infty}^{t-0.5} \mathrm{sinc}(C_0 R \cdot \tau)^{C \log R} \cdot \prod_{i=0}^{\log(S)} \mathrm{sinc}\,\big(\frac{C_0 \cdot S}{2^i} \cdot \tau\big)^{2^i \cdot C} \mathrm{d}\tau \\
&\geq H_1(0) - s_0 O(\frac{1}{C_0^2 R \log(R)})\delta_1 \\
&= 1 - s_0 O(\frac{1}{C_0^2 R \log(R)})\delta_1 \\
&\geq 1 - O(\delta_1),
\end{aligned}$$

where the first step follows from Eq. (12.13), the second step is straight forward, the third step follows from

$$\mathrm{sinc}(C_0 R \cdot \tau)^{C \log R} \cdot \prod_{i=0}^{\log(S)} \mathrm{sinc}\,\big(\frac{C_0 \cdot S}{2^i} \cdot \tau\big)^{2^i \cdot C} \geq 0, \quad \forall \tau \in \mathbb{R}$$

the forth step follows from Eq. (12.16), the fifth step follows from $H_1(0) = 1$, the sixth step follows from Lemma 12.21.

By re-scaling δ_1, we get that $H(t) \geq 1 - \delta_1$ for any $|t| < 0.5 - \frac{\pi}{C_0 R}$.

The lemma then follows from the upper and lower bounds.

□

12.8.4 Energy preserving of the time domain filter

In this section, we show the properties of $H(t)$ that we use in the rest of the chapter.

We first prove Lemma 12.24, which summarizes the results in above sections and prove the energy preserving property of $H(t)$.

Lemma 12.24. *The filter function* $(H(t), \widehat{H}(f))$ *has the following properties:*

$$\begin{aligned}
\text{Property I}: \quad & |H(t)| \leq 1.01, \ \forall t \in \mathbb{R} \\
\text{Property II}: \quad & 1 - \delta_1 \leq H(t) \leq 1, \ \forall |t| < \alpha^{-1}(\frac{1}{2} - \frac{\pi}{CR}) \\
\text{Property III}: \quad & |\mathrm{supp}(\widehat{H}(f))| \leq O(k^2 \log^2(k) \log^2(1/\delta_1)) \\
\text{Property IV}: \quad & \int_{-\infty}^{+\infty} \left| x^*(t) \cdot H(t) \cdot (1 - \mathrm{rect}_2(t)) \right|^2 \mathrm{d}t < 0.1 \int_{-\infty}^{+\infty} |x^*(t) \cdot \mathrm{rect}_2(t)|^2 \mathrm{d}t \\
\text{Property V}: \quad & \int_{-\infty}^{+\infty} |x^*(t) \cdot H(t) \cdot \mathrm{rect}_2(t)|^2 \mathrm{d}t \in [0.9, 1.1] \cdot \int_{-\infty}^{+\infty} |x^*(t) \cdot \mathrm{rect}_2(t)|^2 \mathrm{d}t
\end{aligned}$$

Proof. We prove each of the five properties in below.

Property I: By Theorem 12.20 Part 3, we have that

$$|H(t)| \leq 1.01.$$

Property II: By Lemma 12.23, we have that

$$1 - \delta_1 \leq H(t) \leq 1, \ \forall |t| < \alpha^{-1}(\frac{1}{2} - \frac{\pi}{CR}).$$

Property III: By the k-Fourier-sparse signals' energy bound (Theorem 12.6), we have that

$$R = O(k^2).$$

By Theorem 12.18, we have that

$$S = O(k^2 \log k).$$

Then, by Lemma 12.19, we have that

$$\begin{aligned}
|\mathrm{supp}(\widehat{H}(f))| &= C^2 R \log R + C^2 S \log S \\
&= O(\log(1/\delta_1))^2 \cdot O(k^2 \log k \log(k^2 \log k)) \\
&= O(k^2 \log^2 k \log^2(1/\delta_1)).
\end{aligned}$$

Property IV: We have that

$$\begin{aligned}
&\int_{-\infty}^{+\infty} \left|x^*(t) \cdot H(t) \cdot (1 - \mathrm{rect}_1(t))\right|^2 \mathrm{d}t \\
= &\int_{-\infty}^{+\infty} \left|x^*(t) \cdot H(t)\right|^2 \mathrm{d}t - \int_{-1}^{1} \left|x^*(t) \cdot H(t)\right|^2 \mathrm{d}t \\
\leq &\ 0.06 \int_{-1}^{+1} \left|x^*(t) \cdot H(t)\right|^2 \mathrm{d}t \\
\leq &\ 0.1 \int_{-1}^{+1} \left|x^*(t)\right|^2 \mathrm{d}t \\
= &\ 0.1 \int_{-\infty}^{+\infty} |x^*(t) \cdot \mathrm{rect}_1(t)|^2 \mathrm{d}t,
\end{aligned}$$

where the first step is straight forward, the second step follows from Theorem 12.20 Part 2, the third step follows from Theorem 12.20 Part 3, the forth step is straight forward.

Property V: We first prove the upper bound:

$$\begin{aligned}
&\int_{-\infty}^{+\infty} |x^*(t) \cdot H(t) \cdot \mathrm{rect}_1(t)|^2 \mathrm{d}t \\
= &\int_{-1}^{+1} |x^*(t) \cdot H(t)|^2 \mathrm{d}t \\
\leq &\ 1.1 \int_{-1}^{+1} |x^*(t)|^2 \mathrm{d}t \\
= &\ 1.1 \int_{-\infty}^{+\infty} |x^*(t) \cdot \mathrm{rect}_1(t)|^2 \mathrm{d}t,
\end{aligned}$$

where the first step is straight forward, the second step follows from Theorem 12.20 Part 3, the third step is straight forward.

Then, we prove the lower bound:

$$\begin{aligned}
&\int_{-\infty}^{+\infty} |x^*(t) \cdot H(t) \cdot \mathrm{rect}_1(t)|^2 \mathrm{d}t \\
= &\int_{-1}^{+1} |x^*(t) \cdot H(t)|^2 \mathrm{d}t \\
\geq &\ 0.9 \cdot \int_{-1}^{+1} |x^*(t)|^2 \mathrm{d}t \\
= &\ 0.9 \cdot \int_{-\infty}^{+\infty} |x^*(t) \cdot \mathrm{rect}_1(t)|^2 \mathrm{d}t
\end{aligned}$$

where the first step is straight forward, the second step follows from Theorem 12.20 Part 1, the third step is straight forward.

□

The following lemma bounds the length of the fluctuation region (where $H(t)$ is close to 1) in the time domain.

Lemma 12.25. *Let* $\Delta = k|\mathrm{supp}(\widehat{H}(f))|$, $\beta = O(1/\Delta)$, $L = \frac{T}{2} - \alpha^{-1}(\frac{1}{2} - \frac{\pi}{C_0 R})\frac{T}{2}, R = \frac{T}{2} + \alpha^{-1}(\frac{1}{2} - \frac{\pi}{C_0 R})\frac{T}{2} - \beta$, *we have that*

$$T - k^2(T + L - R) \eqsim T,$$

and

$$R - L \eqsim T.$$

Proof. Let $U := [L, R]$. By Lemma 12.23, we have that for any $t_0 \in U$,

$$H(t) > 1 - \delta_1, \forall t \in [t_0, t_0 + \beta].$$

We have that

$$\begin{aligned}
R - L &= |U| \\
&= ((\frac{1}{2} + \frac{1.2}{\pi CR})^{-1} \cdot (\frac{1}{2} - \frac{\pi}{CR}) - \beta) \cdot T \\
&\eqsim T,
\end{aligned}$$

where the first step follows from the definition of L, R, the second step follows from Lemma 12.24 Property II, the third step follows from $\Delta, CR \gg 1$.

We have that

$$\begin{aligned} T + L - R &= T - |U| \\ &= (1 - (\frac{1}{2} + \frac{1.2}{\pi CR})^{-1} \cdot (\frac{1}{2} - \frac{\pi}{CR})) \cdot T + \beta T \\ &= \frac{2\pi + 2.4/\pi}{CR + 2.4/\pi} \cdot T + \beta T, \end{aligned} \tag{12.17}$$

where the first step follows from the definition of L, R, the second step follows from Lemma 12.24 Property II, the third step is straight forward.

Then, we have that

$$T - k^2(T + L - R) = T - k^2 \cdot (\frac{2\pi + 2.4/\pi}{CR + 2.4/\pi} + \beta) \cdot T \eqsim T,$$

where the first step follows from Eq. (12.17), the second step follows from $C = O(\log(1/\delta_1))$, $R = k^2$, $k^2\beta < 1/k$.

□

12.9 Ideal Filter Approximation

As we discussed in previous sections, the filtered signal $z^{(j)}(t) = (x \cdot H) * G_{\sigma,b}^{(j)}(t)$ is the signal in the j-th bin by the HashToBins procedure. In this section, we consider an approximation of the frequency domain filter $G_{\sigma,b}^{(j)}$ by the *ideal filter* $I_{\sigma,b}^{(j)}(t)$ defined by its Fourier transform:

$$\widehat{I}_{\sigma,b}^{(j)}(f) := \begin{cases} 1, & \widehat{G}_{\sigma,b}^{(j)}(f) > 1 - \delta_1 \\ 0, & \text{otherwise} \end{cases} \tag{12.18}$$

Intuitively, $I_{\sigma,b}^{(j)}$ is "denoising" the frequency domain filter $\widehat{G}_{\sigma,b}^{(j)}$ in the sense that it rounds the heavy Fourier coefficients of $\widehat{G}_{\sigma,b}^{(j)}$ to 1 and rounds the remaining Fourier coefficients to 0. The main purpose of this section is to show that $(x \cdot H) * I_{\sigma,b}^{(j)}$ is a good approximation of $z^{(j)}$. For simplicity, we will use I to denote $I_{\sigma,b}^{(j)}$ when σ, b, j are clear from context.

We first show a commuting property of the ideal filter (see Section 12.9.1). Then, we derive the approximation error bound of the ideally filtered signals (see Section 12.9.2).

12.9.1 Swap the order of filtering

We first prove a good property of the ideal filter that $I_{\sigma,b}^{(j)}$ "commutes" with the time domain filter H with high probability over the random hashing function.

Lemma 12.26. *Let δ_1 be the δ defined in Lemma 12.9.Let H be defined as in Definition 12.7, $G_{\sigma,b}^{(j)}$ be defined as in Definition 12.2. Let the ideal filter $I = I_{\sigma,b}^{(j)}$ be defined as in Eq. (12.18).*

Then, for any $x \in \mathcal{F}_{k,F}$, with probability 0.9 over the choice of (σ, b), for any $j \in [B]$,

$$(x \cdot H) * I(t) = (x * I)(t) \cdot H(t) \quad \forall t \in \mathbb{R}.$$

Proof. By Fourier transformation, we have

$$(x \cdot H) * I(t) = \int_{-\infty}^{\infty} (\widehat{x} * \widehat{H})(f) \cdot \widehat{I}(f) \cdot \exp(2\pi \mathbf{i} f t) \mathrm{d} f.$$

We will show that $(\widehat{x} * \widehat{H})(f) \cdot \widehat{I}(f) = ((\widehat{x} \cdot \widehat{I}) * \widehat{H})(f)$ with high probability.

On the one hand,

$$\begin{aligned}
(\widehat{x} * \widehat{H})(f) \cdot \widehat{I}(f) &= \sum_{j=1}^{k} v_j \cdot (\delta_{f_j} * \widehat{H})(f) \cdot \widehat{I}(f) \\
&= \sum_{j=1}^{k} v_j \cdot \widehat{H}(f - f_j) \cdot \widehat{I}(f) \\
&= \sum_{j=1}^{k} v_j \cdot \widehat{H}(f - f_j) \cdot \mathbf{1}_{f \in \mathrm{supp}(\widehat{I})},
\end{aligned}$$

where the first step follows from $\widehat{x}(f) = \sum_{j=1}^k v_j \cdot \delta_{f_j}(f)$, the second step follows from the convolution property of Delta function. By Lemma 12.16, we get that with probability at least 0.9, for any $j \in [k]$ and any $f \in \mathrm{supp}(\widehat{H}) + f_j$, either $\widehat{G}^{(j)}_{\sigma,b} < \delta_1$ or $\widehat{G}^{(j)}_{\sigma,b} > 1 - \delta_1$. In other words, either $f_j + \mathrm{supp}(\widehat{H}) \subseteq \mathrm{supp}(\widehat{I})$ or $f_j + \mathrm{supp}(\widehat{H}) \cap \mathrm{supp}(\widehat{I}) = \emptyset$. Since $0 \in \mathrm{supp}(\widehat{H})$, we get that for any $f \in f_j + \mathrm{supp}(\widehat{H})$,

$$f \in \mathrm{supp}(\widehat{I}) \iff f_j \in \mathrm{supp}(\widehat{I}).$$

Hence, we have

$$(\widehat{x} * \widehat{H})(f) \cdot \widehat{I}(f) = \sum_{j \in [k] : f_j \in \mathrm{supp}(\widehat{I})} v_j \cdot \widehat{H}(f - f_j).$$

On the other hand,

$$\begin{aligned}(\widehat{x} \cdot \widehat{I}) * \widehat{H}(f) &= \sum_{j \in [k] : f_j \in \mathrm{supp}(\widehat{I})} v_j \cdot \delta_{f_j} * \widehat{H}(f) \\ &= \sum_{j \in [k] : f_j \in \mathrm{supp}(\widehat{I})} v_j \cdot \widehat{H}(f - f_j) \\ &= (\widehat{x} * \widehat{H})(f) \cdot \widehat{I}(f).\end{aligned}$$

Therefore,

$$\begin{aligned}(x \cdot H) * I(t) &= \int_{-\infty}^{\infty} (\widehat{x} * \widehat{H})(f) \cdot \widehat{I}(f) \cdot \exp(2\pi \mathbf{i} f t) \mathrm{d}f \\ &= \int_{-\infty}^{\infty} (\widehat{x} \cdot \widehat{I}) * \widehat{H}(f) \cdot \exp(2\pi \mathbf{i} f t) \mathrm{d}f \\ &= (x * I)(t) \cdot H(t),\end{aligned}$$

where the last step follows from the definition of Fourier transform.

The lemma is then proved.

□

12.9.2 Approximation error bounds

We analyze the approximation error due to replacing $\widehat{G}_{\sigma,b}^{(j)}$ with the ideal filter $I_{\sigma,b}^{(j)}$ defined by Eq. (12.18). The following lemma gives a point-wise error bound.

Lemma 12.27. *Let δ_1 be defined as in Lemma 12.9. Let H be defined as in Definition 12.7, $G_{\sigma,b}^{(j)}$ be defined as in Definition 12.2.*

For any $x \in \mathcal{F}_{k,F}$, we have that with probability 0.9, for any $j \in [B]$,

$$|(x \cdot H) * G_{\sigma,b}^{(j)}(t) - (x \cdot H) * I(t)| \lesssim \delta_1 \sqrt{T|S|} \cdot \|x(t)\|_T \quad \forall t \in \mathbb{R}.$$

Proof. Let $S := \operatorname{supp}(\widehat{x} * \widehat{H})$ be defined as the support set of $\widehat{x} * \widehat{H}$. Then $|S| \leq \Delta$.

We have that

$$\begin{aligned}
& |(x \cdot H) * G_{\sigma,b}^{(j)}(t) - (x \cdot H) * I(t)| \quad (12.19)\\
= & |(x \cdot H) * (G_{\sigma,b}^{(j)} - I)(t)| \\
= & \Big| \int_{-\infty}^{\infty} (\widehat{x} * \widehat{H})(f) \cdot (\widehat{G}_{\sigma,b}^{(j)} - \widehat{I})(f) \cdot e^{2\pi \mathbf{i} f t} \mathrm{d}f \Big| \\
\leq & \int_{-\infty}^{\infty} |(\widehat{x} * \widehat{H})(f) \cdot (\widehat{G}_{\sigma,b}^{(j)} - I)(f)| \mathrm{d}f \\
= & \int_S |(\widehat{x} * \widehat{H})(f) \cdot (\widehat{G}_{\sigma,b}^{(j)} - I)(f)| \mathrm{d}f \\
\leq & \int_S |(\widehat{x} * \widehat{H})(f) \cdot \delta_1| \mathrm{d}f \\
\leq & \delta_1 \sqrt{|S|} \cdot \sqrt{\int_{-\infty}^{\infty} |(\widehat{x} * \widehat{H})(f)|^2 \mathrm{d}f} \\
= & \delta_1 \sqrt{|S|} \cdot \sqrt{\int_{-\infty}^{\infty} |(x \cdot H)(t)|^2 \mathrm{d}t} \\
\lesssim & \delta_1 \sqrt{T|S|} \cdot \|(x \cdot H)(t)\|_T \\
\lesssim & \delta_1 \sqrt{T|S|} \cdot \|x(t)\|_T \quad (12.20)
\end{aligned}$$

where the first step is straight forward, the second step follows from the definition of Fourier transform, the third step follows from triangle equality, the forth step follows from the definition of S. For the fifth step, by Lemma 12.16 that with probability at least 0.9, the Large Offset event does not happen (i.e., for any $f \in S$, either $\widehat{G}^{(j)}_{\sigma,b}(f) < \delta_1$ or $\widehat{G}^{(j)}_{\sigma,b}(f) > 1 - \delta_1$). Then, by Lemma 12.9, we know that $-\delta_1 \leq \widehat{G}^{(j)}_{\sigma,b}(f) \leq 1$. Thus, we get that $|(\widehat{G}^{(j)}_{\sigma,b} - \widehat{I})(f)| \leq \delta_1$. The sixth step follows from Cauchy–Schwarz inequality, the seventh step follows from Parseval's theorem, the eighth step follows from Lemma 12.24 Property IV and V, the last step follows from Lemma 12.24 Property V. $\square$

The following lemma gives a T-norm bound for the approximation error.

Lemma 12.28. *Let δ_1 be defined as in Lemma 12.9. Let H be defined as in Definition 12.7, $G^{(j)}_{\sigma,b}$ be defined as in Definition 12.2.*

Then, for any $x \in \mathcal{F}_{k,F}$, with probability 0.9, for any $j \in [B]$,

$$\int_{-\infty}^{\infty} |(x \cdot H) * I(t) - (x \cdot H) * G^{(j)}_{\sigma,b}(t)|^2 \mathrm{d}t \lesssim \delta_1^2 T \|x(t)\|_T^2.$$

In particular,

$$\|(x \cdot H) * I(t) - (x \cdot H) * G^{(j)}_{\sigma,b}(t)\|_T \lesssim \delta_1 \|x(t)\|_T.$$

Proof. Let $S := \mathrm{supp}(\widehat{x} * \widehat{H})$ be defined as the support set of $\widehat{x} * \widehat{H}$.

We have that

$$\begin{aligned}
& T\|(x\cdot H) * I(t) - (x\cdot H) * G_{\sigma,b}^{(j)}(t)\|_T^2 \\
= & \int_0^T |(x\cdot H) * I(t) - (x\cdot H) * G_{\sigma,b}^{(j)}(t)|^2 \mathrm{d}t \\
\leq & \int_{-\infty}^{\infty} |(x\cdot H) * I(t) - (x\cdot H) * G_{\sigma,b}^{(j)}(t)|^2 \mathrm{d}t \\
\leq & \int_{-\infty}^{\infty} |(\widehat{x} * \widehat{H})(f) \cdot (\widehat{I}(f) - \widehat{G}_{\sigma,b}^{(j)}(f))|^2 \mathrm{d}f \\
= & \int_S |(\widehat{x} * \widehat{H})(f) \cdot (I(f) - \widehat{G}_{\sigma,b}^{(j)}(f))|^2 \mathrm{d}f \\
\leq & \int_S |(\widehat{x} * \widehat{H})(f) \cdot \delta_1|^2 \mathrm{d}f \\
\leq & \int_{-\infty}^{\infty} |(\widehat{x} * \widehat{H})(f) \cdot \delta_1|^2 \mathrm{d}f \\
= & \delta_1^2 \int_{-\infty}^{\infty} |(x\cdot H)(t)|^2 \mathrm{d}t \\
\lesssim & \delta_1^2 T\|x(t)\|_T^2
\end{aligned}$$

where the first step follows from the definition of the norm, the second step is straight forward, the third step follows from Parseval's theorem, the forth step follows from the definition of S, the fifth step follows from Lemma 12.16 and Lemma 12.9, the sixth step is straight forward, the seventh step follows from Parseval's theorem, the eighth step follows from Lemma 12.24 Property IV and Property V.

$\square$

12.10 Concentration Property of the Filtered Signal

Recall that a frequency f^* is *heavy* if it satisfies the following condition:

$$\int_{f^*-\Delta_h}^{f^*+\Delta_h} |\widehat{x^* \cdot H}(f)|^2 \mathrm{d}f \geq T\mathcal{N}^2/k.$$

In this section, we consider the filtered signal in a hashing bin that contains a heavy frequency; that is, $z(t) = (x^* \cdot H) * \widehat{G}_{\sigma,b}^{(j)}$ where $j = h_{\sigma,b}(f^*)$ is the index of the bin containing f^*. We will prove that $z(t)$ form a one-cluster signal around f^*, which means that in the frequency domain most energy are concentrated around f^*, and in the time domain, most energy are contained in the observation window $[0, T]$. The formal definition are given as follows:

Definition 12.8 ((ε, Δ)-one-cluster signal)**.** We say that a signal $z(t)$ is an (ε, Δ)-one-cluster signal around f_0 if and only if $z(t)$ and $\widehat{z}(f)$ satisfy the following two properties:

$$\begin{aligned} \text{Property I} &: \int_{f_0-\Delta}^{f_0+\Delta} |\widehat{z}(f)|^2 \mathrm{d}f \geq (1-\varepsilon) \int_{-\infty}^{+\infty} |\widehat{z}(f)|^2 \mathrm{d}f \\ \text{Property II} &: \int_0^T |z(t)|^2 \mathrm{d}t \geq (1-\varepsilon) \int_{-\infty}^{+\infty} |z(t)|^2 \mathrm{d}t. \end{aligned}$$

We first prove the energy preservation in the time domain:

Lemma 12.29 (Time domain energy preservation)**.** *Let $\Delta_h = |\mathrm{supp}(\widehat{H})|$. Let f^* satisfy*

$$\int_{f^*-\Delta_h}^{f^*+\Delta_h} |\widehat{x^* \cdot H}(f)|^2 \mathrm{d}f \geq T\mathcal{N}^2/k$$

and $\widehat{z} = \widehat{x^* \cdot H} \cdot \widehat{G}_{\sigma,b}^{(j)}$ where $j = h_{\sigma,b}(f^*)$. Suppose the Large Offset event does not happen. Then, we have that,

$$\int_{-\infty}^{+\infty} |z(t)|^2 \mathrm{d}t \leq 1.35 \int_0^T |z(t)|^2 \mathrm{d}t.$$

Proof. Let $I(f)$ be the ideal filter defined by Eq. (12.18).

We first have

$$\begin{aligned} \|z(t)\|_{L_2} &= \|(x^* \cdot H)(t) * G_{\sigma,b}^{(j)}(t)\|_{L_2} \\ &\leq \|(x^* \cdot H)(t) * I(t)\|_{L_2} + \|(x^* \cdot H)(t) * (I - G_{\sigma,b}^{(j)})(t)\|_{L_2}, \end{aligned}$$

where the second step follows from triangle inequality.

Then, we bound the two terms separately.

For the first term, by Lemma 12.26, if the Large Offset event does not happen, we have that

$$(x^* \cdot H) * I(t) = (x^* * I)(t) \cdot H(t). \tag{12.21}$$

It implies that

$$\|(x^* \cdot H)(t) * I(t)\|_{L_2} = \|(x^* * I)(t) \cdot H(t)\|_{L_2}$$

Let $y(t) := (x^* * I)(t)$. It's easy to see that $y(y)$ is k-Fourier-sparse. Then, we have

$$\begin{aligned}
& \int_{-\infty}^{\infty} |y(t) \cdot H(t)|^2 \mathrm{d}t \\
= & \int_0^T |y(t) \cdot H(t)|^2 \mathrm{d}t + \int_{[-\infty,\infty]\setminus[0,T]} |y(t) \cdot H(t)|^2 \mathrm{d}t \\
\leq & \int_0^T |y(t) \cdot H(t)|^2 \mathrm{d}t + 0.1 \int_0^T |y(t)|^2 \mathrm{d}t \\
\leq & \, 1.1 \int_0^T |y(t)|^2 \mathrm{d}t \\
\leq & \, 1.3 \int_0^T |y(t) \cdot H(t)|^2 \mathrm{d}t,
\end{aligned} \tag{12.22}$$

where the first step is straight forward, the second step follows from Lemma 12.24 Property IV, the third step follows from Lemma 12.24 Property V, the forth step follows from Lemma 12.24 Property V. Hence,

$$\|(x^* \cdot H)(t) * I(t)\|_{L_2} = \|y(t) \cdot H(t)\|_{L_2} \leq \sqrt{1.3T} \cdot \|(x^* * I)(t) \cdot H(t)\|_T.$$

By Eq. (12.21) again, we can swap the order of I and H and obtain:

$$\|(x^* \cdot H)(t) * I(t)\|_{L_2} \leq \sqrt{1.3T} \cdot \|(x^* \cdot H)(t) * I(t)\|_T.$$

For the second term, by Lemma 12.28, we have that

$$\|(x^* \cdot H) * I(t) - (x^* \cdot H) * G_{\sigma,b}^{(j)}(t)\|_{L_2} \lesssim \delta_1 \sqrt{T} \|x^*(t)\|_T. \tag{12.23}$$

Therefore, we get that

$$\begin{aligned}
\|z(t)\|_{L_2} &\leq \sqrt{1.3T} \cdot \|(x^* \cdot H)(t) * I(t)\|_T + O(\delta_1 \sqrt{T} \|x^*(t)\|_T) \\
&\leq \sqrt{1.3T} \cdot \|(x^* \cdot H) * G_{\sigma,b}^{(j)}(t)\|_T + \sqrt{1.3T} \cdot \|(x^* \cdot H) * (I - G_{\sigma,b}^{(j)})(t)\|_T + O(\delta_1 \sqrt{T} \|x^*(t)\|_T) \\
&\leq \sqrt{1.3T} \cdot \|z(t)\|_T + O(\delta_1 \sqrt{T} \|x^*(t)\|_T),
\end{aligned}$$

where the second step follows from triangle inequality, and the last step follows from Lemma 12.28 again.

We claim that the second term can be bounded by $o(1) \cdot \|z(t)\|_T$. Indeed, we have

$$\begin{aligned}
& \int_{-\infty}^{\infty} |(x^* \cdot H) * G_{\sigma,b}^{(j)}(t)|^2 \mathrm{d}t \\
= & \int_{-\infty}^{\infty} |(\widehat{x}^* * \widehat{H}) \cdot \widehat{G}_{\sigma,b}^{(j)}(f)|^2 \mathrm{d}f \\
\geq & \int_{f^*-\Delta_h}^{f^*+\Delta_h} |(\widehat{x}^* * \widehat{H}) \cdot \widehat{G}_{\sigma,b}^{(j)}(f)|^2 \mathrm{d}f \\
\gtrsim & \int_{f^*-\Delta_h}^{f^*+\Delta_h} |(\widehat{x}^* * \widehat{H})(f)|^2 \mathrm{d}f \\
\gtrsim & \int_{f^*-\Delta_h}^{f^*+\Delta_h} |(\widehat{x}^* * \widehat{H})(f)|^2 \mathrm{d}f \\
\geq & \frac{T\delta \|x^*\|_T^2}{k}
\end{aligned} \tag{12.24}$$

where the first step follows from Parseval's theorem, the second step is straightforward, the third step follows from Lemma 12.16 and Lemma 12.9 Property I, the forth step follows from our assumption that there exists a heavy frequency f^* hashing into the j-th bin, the fifth step follows from f^* satisfying

$$\int_{f^*-\Delta}^{f^*+\Delta} |\widehat{x^* \cdot H}(f)|^2 \mathrm{d}f \geq T\mathcal{N}^2/k.$$

Thus, $\|x^*\|_T \leq O(\sqrt{k/(T\delta)})\|z(t)\|_{L_2}$ and we have

$$O(\delta_1 \sqrt{T} \|x^*\|_T) = O\Big(\delta_1 \sqrt{\frac{k}{\delta}} \|z(t)\|_{L_2}\Big) \leq O\Big(\sqrt{\frac{\delta}{k}}\Big) \|z(t)\|_{L_2} = o(1) \cdot \|z(t)\|_{L_2},$$

where the second step follows from $\delta_1 \leq \delta/k$.

Finally, we have

$$\|z(t)\|_{L_2} \leq \sqrt{1.3T} \cdot \|z(t)\|_T + o(1) \cdot \|z(t)\|_{L_2},$$

which implies that

$$\int_{-\infty}^{+\infty} |z(t)|^2 \mathrm{d}t \leq 1.35 \int_0^T |z(t)|^2 \mathrm{d}t.$$

The lemma is then proved.

$\square$

We next show the frequency domain energy concentration in the following lemma. Together with Lemma 12.29, we conclude that $z(t)$ is a one-cluster signal.

Lemma 12.30 (Frequency domain energy concentration)**.** *Let x^* be a k-Fourier-sparse signal. Let $f^* \in [-F, F]$ satisfy the following property:*

$$\int_{f^*-\Delta_h}^{f^*+\Delta_h} |\widehat{x^* \cdot H}(f)|^2 \mathrm{d}f \geq T\mathcal{N}^2/k. \tag{12.25}$$

Let σ, b be the parameter of the hashing function. Suppose that Large Offset event not happened and f^* is well-isolated. Let $j = h_{\sigma,b}(f^*)$ be the bucket that f^* maps to under the hash such that $z = (x^* \cdot H) * G_{\sigma,b}^{(j)}$ and $\widehat{z} = \widehat{x^* \cdot H} \cdot \widehat{G}_{\sigma,b}^{(j)}$. Then, we have

$$\int_{f^*-\Delta}^{f^*+\Delta} |\widehat{z}(f)|^2 \mathrm{d}f \geq 0.7 \int_{-\infty}^{+\infty} |\widehat{z}(f)|^2 \mathrm{d}f.$$

Furthermore, $z(t)$ is a $(0.3, \Delta)$-one-cluster signal around f^*.

Proof. Define region $I_{f^*} = (f^*-\Delta, f^*+\Delta)$ with the complement $\overline{I_{f^*}} = (-\infty, \infty) \backslash I_{f^*}$. We have that

$$\int_{I_{f^*}} |\widehat{z}(f)|^2 \mathrm{d}f \geq (1 - \delta/k) \int_{I_{f^*}} |\widehat{x^* \cdot H}(f)|^2 \mathrm{d}f \geq (1 - o(1))T\mathcal{N}^2/k$$

where the first step follows from Lemma 12.17, the second step follows from Eq. (12.25).

On the other hand, f^* is well-isolated. Thus, by the definition of well-isolation (Definition 12.4), we have that

$$\int_{\overline{I_{f^*}}} |\widehat{z}(f)|^2 \mathrm{d}f \lesssim \varepsilon \cdot T\mathcal{N}^2/k \leq 0.1T\mathcal{N}^2/k.$$

Combining them together, we get that

$$\int_{f_0-\Delta}^{f_0+\Delta} |\widehat{z}(f)|^2 \mathrm{d}f \geq 0.7 \int_{-\infty}^{+\infty} |\widehat{z}(f)|^2 \mathrm{d}f$$

For the furthermore part, Lemma 12.29 implies that

$$\int_0^T |z(t)|^2 \mathrm{d}t \geq (1-0.3) \int_{-\infty}^{+\infty} |z(t)|^2 \mathrm{d}t.$$

Hence, by Definition 12.8, $z(t)$ is a $(0.3, \Delta)$-one-cluster. $\square$

12.11 Energy Bound for Filtered Fourier Sparse Signals

In this section, we prove an energy bound for the filtered signals $(x \cdot H) * G_{\sigma,b}^{(j)}$, which upper bounds the magnitude of any such signal at a point t by its energy in the time duration $[0, T]$. We first prove an energy bound for untruncated ideally filtered signals (see Section 12.11.1). Then, we prove an energy bound for filtered signals (see Section 12.11.2). In addition, we prove a technical claim (see Section 12.11.3).

12.11.1 Energy bound for untruncated ideally filtered signals

In Section 12.9, we show that the ideally filtered signal $(x \cdot H) * I_{\sigma,b}^{(j)}$, where $I_{\sigma,b}^{(j)}$ defined as Eq. (12.18) is the ideal filter, is close to the true filtered signal. Here, we further simplify the signal by ignoring the truncation filter $H(t)$, and prove an energy bound for the signals of the form $(x * I_{\sigma,b}^{(j)})(t)$:

Lemma 12.31. *Let H be defined as in Definition 12.7, $G_{\sigma,b}^{(j)}$ be defined as in Definition 12.2 and the corresponding ideal filter $I = I_{\sigma,b}^{(b)}$ be defined as in Eq. (12.18). Let $D(t) :=$ Uniform$([-1, 1])$.*

For any $x \in \mathcal{F}_{k,F}$, we have that with probability 0.6*, for any $t \in (-1,1)$*

$$|(x * I)(t)|^2 \lesssim \min\Big\{\frac{k}{1-|t|}, k^2\Big\} \cdot \|(x * I)(t)\|_D^2$$

Proof. Since $\widehat{(x * I)}(f) = \widehat{x} \cdot \widehat{I}(f)$ and x is k-Fourier-sparse, we know that $(x * \widehat{I})(t)$ is also a k-Fourier-sparse signal.

On the one hand, by the k-Fourier-sparse signal's location-dependent energy bound (Theorem 12.7), we have

$$|(x * I)(t)|^2 \lesssim \frac{k}{1-|t|}\|(x * I)(t)\|_D^2 \tag{12.26}$$

On the other hand, by the location-independent energy bound (Theorem 12.6), we have that

$$|(x * I)(t)|^2 \lesssim k^2\|(x * I)(t)\|_D^2 \tag{12.27}$$

Combine Eq. (12.26) and Eq. (12.27) together, we prove the lemma:

$$|(x * I)(t)|^2 \lesssim \min\Big\{\frac{k}{1-|t|}, k^2\Big\} \cdot \|(x * I)(t)\|_D^2.$$

□

12.11.2 Energy bound for filtered signals

Based on Lemma 12.31, we can relate the magnitude of the filtered signal with its own energy plus the original Fourier-sparse signal's energy.

Lemma 12.32. *Let H be defined as in Definition 12.7, $G_{\sigma,b}^{(j)}$ be defined as in Definition 12.2 and the corresponding ideal filter $I = I_{\sigma,b}^{(b)}$ be defined as in Eq. (12.18).*

For any $x \in \mathcal{F}_{k,F}$, $j \in [B]$, and (σ, b) such that Large Offset event does not happen, let $z(t) = (x \cdot H) * G_{\sigma,b}^{(j)}(t)$. It holds that:

$$|z(t)|^2 \lesssim \min\Big\{\frac{k \cdot H(t)}{1-|2t/T-1|}, k^2\Big\} \cdot \|z(t)\|_T^2 + \delta_1\|x(t)\|_T^2 \quad \forall t \in (-1,1).$$

Proof. Let $S := \mathrm{supp}(\widehat{x} * \widehat{H})$ be defined as the support set of $\widehat{x} * \widehat{H}$. Then $|S| \leq \Delta$.

First, by the ideally untruncated filtered signal's energy bound (Lemma 12.31), we have

$$\begin{aligned} |(x * I)(t) \cdot H(t)|^2 &\lesssim H^2(t) \cdot \min\Big\{ \frac{k}{1 - |2t/T - 1|}, k^2 \Big\} \cdot \|(x * I)(t)\|_T^2 \\ &\lesssim \min\Big\{ \frac{k \cdot H(t)}{1 - |2t/T - 1|}, k^2 \Big\} \cdot \|(x * I)(t)\|_T^2, \end{aligned} \tag{12.28}$$

where the second step follows from $H(t) \lesssim 1$ (Lemma 12.24 Property I, II).

Then, we bound the magnitude of the ideal filtered signal as follows:

$$\begin{aligned} |(x \cdot H) * I(t)|^2 &= |(x * I)(t) \cdot H(t)|^2 \\ &\lesssim \min\Big\{ \frac{k \cdot H(t)}{1 - |2t/T - 1|}, k^2 \Big\} \cdot \|(x * I)(t)\|_T^2 \\ &\lesssim \min\Big\{ \frac{k \cdot H(t)}{1 - |2t/T - 1|}, k^2 \Big\} \cdot (\|(x \cdot H) * G_{\sigma,b}^{(j)}(t)\|_T^2 + \delta_1^2 \|x(t)\|_T^2) \\ &\lesssim \min\Big\{ \frac{k \cdot H(t)}{1 - |2t/T - 1|}, k^2 \Big\} \cdot \|(x \cdot H) * G_{\sigma,b}^{(j)}(t)\|_T^2 + \delta_1 \|x(t)\|_T^2 \end{aligned} \tag{12.29}$$

where the first step follows from Lemma 12.26, the second step follows from Eq. (12.28), the third step follows from Claim 12.35, the forth step follows from $k^2 \delta_1 \leq 1$.

Next, we consider the difference between the signals filtered by $G_{\sigma,b}^{(j)}(t)$ and $I(t)$:

$$\begin{aligned} |(x \cdot H) * G_{\sigma,b}^{(j)}(t) - (x \cdot H) * I(t)|^2 &\leq \delta_1^2 T |S| \cdot \|x(t)\|_T^2 \\ &\leq \delta_1 \cdot \|x(t)\|_T^2 \end{aligned} \tag{12.30}$$

where the first step follows from Lemma 12.27, the second step follows from $\delta_1 T |S| \leq 1$.

Finally, we have that

$$\begin{aligned}|(x \cdot H) * G_{\sigma,b}^{(j)}(t)|^2 &\leq 2|(x \cdot H) * I(t)|^2 + 2|(x \cdot H) * G_{\sigma,b}^{(j)}(t) - (x \cdot H) * I(t)|^2 \\ &\lesssim |(x \cdot H) * I(t)|^2 + \delta_1 \|x(t)\|_T^2 \\ &\lesssim \min\Big\{ \frac{k \cdot H(t)}{1 - |2t/T - 1|}, k^2 \Big\} \cdot \|(x \cdot H) * \widehat{G}_{\sigma,b}^{(j)}(t)\|_T^2 + \delta_1 \|x(t)\|_T^2,\end{aligned}$$

where the first step follows from $(a+b)^2 \leq 2a^2 + 2b^2$, the second step follows from Eq. (12.30), the third step follows from Eq. (12.29).

The lemma is then proved.

□

The energy bound in Lemma 12.32 not only depends on $\|z(t)\|_T$, but also on $\|x(t)\|_T$. The following lemma show that assuming the filtered signal contains a heavy frequency, $\|x(t)\|_T$ can be upper bounded by $\|z(t)\|_T$.

Lemma 12.33. *Given $k \in \mathbb{Z}_+, F \in \mathbb{R}_+$. Let H be defined as in Definition 12.7, $G_{\sigma,b}^{(j)}$ be defined as in Definition 12.2. Let $x \in \mathcal{F}_{k,F}$ be any k-Fourier sparse signal. For $j \in [B]$ such that there exists a f^* satisfying: $j = h_{\sigma,b}(f^*)$ and*

$$\int_{f^*-\Delta_h}^{f^*+\Delta_h} |\widehat{x \cdot H}(f)|^2 \mathrm{d}f \geq T\mathcal{N}^2/k, \tag{12.31}$$

where $\mathcal{N}^2 \geq \delta \|x\|_T^2$ and $\Delta_h = |\mathrm{supp}(\widehat{H})|$.

For any (σ, b) that Large Offset event does not happen, we have that

$$\|(x \cdot H) * G_{\sigma,b}^{(j)}(t)\|_T^2 \gtrsim \frac{\delta \|x\|_T^2}{k}.$$

Proof. We have that

$$\begin{aligned}
T\|(x\cdot H)*G_{\sigma,b}^{(j)}(t)\|_T^2 &= \int_0^T |(x\cdot H)*G_{\sigma,b}^{(j)}(t)|^2 \mathrm{d}t \\
&\gtrsim \int_{-\infty}^{\infty} |(x\cdot H)*G_{\sigma,b}^{(j)}(t)|^2 \mathrm{d}t \\
&= \int_{-\infty}^{\infty} |(\widehat{x}*\widehat{H})\cdot \widehat{G}_{\sigma,b}^{(j)}(f)|^2 \mathrm{d}f \\
&\geq \int_{f^*-\Delta_h}^{f^*+\Delta_h} |(\widehat{x}*\widehat{H})\cdot \widehat{G}_{\sigma,b}^{(j)}(f)|^2 \mathrm{d}f \\
&\gtrsim \int_{f^*-\Delta_h}^{f^*+\Delta_h} |(\widehat{x}*\widehat{H})(f)|^2 \mathrm{d}f \\
&\geq \frac{T\delta\|x\|_T^2}{k}
\end{aligned}$$

where the first step follows from the definition of norm, the second step follows from Lemma 12.30, the third step follows from Parseval's theorem, the forth step is straight forward, the fifth step follows from Lemma 12.17, the sixth step follows from Eq. (12.31).

□

Lemma 12.32 and Lemma 12.33 implies the following energy bound:

Corollary 12.34 (Energy bound for filtered signals)**.** *Given $k\in\mathbb{N}$ and $F\in\mathbb{R}_+$. Let $x\in\mathcal{F}_{k,F}$. Let H be defined as in Definition 12.7, $G_{\sigma,b}^{(j)}$ be defined as in Definition 12.2 with (σ,b) such that Large Offset event does not happen.*

For any $j\in[B]$, suppose there exists an f^* with $j=h_{\sigma,b}(f^*)$ satisfying:

$$\int_{f^*-\Delta}^{f^*+\Delta} |\widehat{x\cdot H}(f)|^2 \mathrm{d}f \geq T\mathcal{N}^2/k,$$

where $\mathcal{N}^2\geq\delta\|x\|_T^2$. Then, for $z(t)=(x\cdot H)*G_{\sigma,b}^{(j)}(t)$, it holds that:

$$|z(t)|^2 \lesssim \min\Big\{\frac{k\cdot H(t)+\delta}{1-|2t/T-1|}, k^2\Big\}\cdot\|z(t)\|_D^2 \quad \forall t\in(0,T).$$

Proof. We have that

$$\begin{aligned}
|z(t)|^2 &\lesssim \min\Big\{\frac{k\cdot H(t)}{1-|2t/T-1|}, k^2\Big\}\cdot\|z(t)\|_T^2 + \delta_1\|x(t)\|_T^2\\
&\lesssim \min\Big\{\frac{k\cdot H(t)}{1-|2t/T-1|}, k^2\Big\}\cdot\|z(t)\|_T^2 + \delta^2k^{-1}\|x(t)\|_T^2\\
&\lesssim \min\Big\{\frac{k\cdot H(t)}{1-|2t/T-1|}, k^2\Big\}\cdot\|z(t)\|_T^2 + \delta\|(x\cdot H)*G_{\sigma,b}^{(j)}(t)\|_T^2\\
&\lesssim \min\Big\{\frac{k\cdot H(t)+\delta}{1-|2t/T-1|}, k^2\Big\}\cdot\|z(t)\|_T^2
\end{aligned}$$

where the first step follows from Lemma 12.32, the second step follows from $\delta_1 \le \delta^2k^{-1}$, the third step follows from Lemma 12.33, the forth step is straight forward.

□

12.11.3 Technical claim

Claim 12.35. *Given $k \in \mathbb{Z}_+, F \in \mathbb{R}_+$. Let δ_1 be defined as the δ of Lemma 12.9.Let H be defined as in Definition 12.7, $G_{\sigma,b}^{(j)}$ be defined as in Definition 12.2, and $I = I_{\sigma,b}^{(j)}$ be the ideal filter defined by Eq. (12.18).*

Then, for any $x \in \mathcal{F}_{k,F}$ and $j \in [B]$, with probability 0.6 over (σ, b), we have that

$$\|(x*I)(t)\|_T^2 \lesssim \|(x\cdot H)*\widehat{G}_{\sigma,b}^{(j)}(t)\|_T^2 + \delta_1^2\|x(t)\|_T^2.$$

Proof. We have that

$$\begin{aligned}
\|(x*I)(t)\|_T^2 &\lesssim \|(x*I)(t)\cdot H\|_T^2\\
&= \|(x\cdot H)*I(t)\|_T^2\\
&\le 2\|(x\cdot H)*G_{\sigma,b}^{(j)}(t)\|_T^2 + 2\|(x\cdot H)*G_{\sigma,b}^{(j)}(t) - (x\cdot H)*I(t)\|_T^2\\
&\lesssim \|(x\cdot H)*G_{\sigma,b}^{(j)}(t)\|_T^2 + \delta_1^2\|x(t)\|_T^2
\end{aligned}$$

where the first step follows from $(x*I)(t)$ is a k-Fourier-sparse signal and Lemma 12.24 Property V, the second step follows from Lemma 12.26 conditioning on Large

Offset event not happening, the third step follows from $(a+b)^2 \leq 2a^2 + 2b^2$, the forth step follows from Lemma 12.28. $\square$

12.12 Local-Test Signal

Recall that the filtered signal in the j-th bin of the HashToBins procedure can be written as $z(t) = (x \cdot H) * G_{\sigma,b}^{(j)}(t)$. The next step of the frequency estimation algorithm is to extract a significant frequency from $z(t)$ by considering a so-called *local-test signal*:

$$d_z(t) := z(t)e^{2\pi f_0 \beta} - z(t+\beta), \tag{12.32}$$

where $f_0 \in \mathrm{supp}(\widehat{x}^*)$, and $j = h_{\sigma,b}(f_0)$, where $\beta \in \mathbb{R}_+$ is a parameter such that $\beta \leq O(1/\Delta)$ with $\Delta = O(k \cdot |\mathrm{supp}(\widehat{H})|)$.

In this section, we will study some properties of $d_z(t)$ and its ideal versions (see Section 12.12.1 and Section 12.12.2) and derive an energy bound for it (See Section 12.12.3).

12.12.1 Ideal local-test signal

In previous section, we've shown that ideal filter $I_{\sigma,b}^{(j)}$ can be used to approximate $G_{\sigma,b}^{(j)}$ such that the ideally filtered signal is close to the true filtered signal. We will show that under the ideal filter approximation, the *ideal local-test signal* is also close to the true local-test signal. More formally, we define the ideal filtered signal and the ideal local-test signal as follows:

$$\begin{aligned} z_I(t) &:= (x \cdot H) * I(t), \\ d_{I,z}(t) &:= z_I(t)e^{2\pi \mathbf{i} f_0 \beta} - z_I(t+\beta), \end{aligned} \tag{12.33}$$

The following lemma bounds the point-wise distance between $d_z(t)$ and $d_{I,z}(t)$.

Lemma 12.36. *Let δ_1 be defined as in Lemma 12.9. Let H be defined as in Definition 12.7, $G_{\sigma,b}^{(j)}$ be defined as in Definition 12.2 and $I = I_{\sigma,b}^{(j)}$ be the corresponding ideal filter as in Eq.* (12.18)*.*

For any $x \in \mathcal{F}_{k,F}$ and (σ, b) such that Large Offset event does not happen, for any $j \in [B]$, let $z(t) = (x \cdot H) * G_{\sigma,b}^{(j)}(t)$, $d_z(t)$ be defined as Eq. (12.32), $z_I(t)$ and $d_{z,I}(t)$ be defined as Eq. (12.33).

Then, we have

$$|d_z(t) - d_{I,z}(t)| \lesssim \delta_1 \sqrt{T|S|} \cdot \|x(t)\|_T \quad \forall t \in \mathbb{R}.$$

Proof.

$$\begin{aligned}
|d_z(t) - d_{I,z}(t)| &= |z(t)e^{2\pi \mathbf{i} f_0 \beta} - z_I(t)e^{2\pi \mathbf{i} f_0 \beta} - (z(t+\beta) - z_I(t+\beta))| \\
&\leq |z(t)e^{2\pi \mathbf{i} f_0 \beta} - z_I(t)e^{2\pi \mathbf{i} f_0 \beta}| + |z(t+\beta) - z_I(t+\beta)| \\
&= |z(t) - z_I(t)| + |z(t+\beta) - z_I(t+\beta)| \\
&\lesssim \delta_1 \sqrt{T|S|} \cdot \|x(t)\|_T,
\end{aligned}$$

where the first step follows from the definition of $d_z(t)$ and $d_{I,z}(t)$, the second step follows from triangle inequality, the third step follows from $|e^{2\pi \mathbf{i} f_0 \beta}| = 1$, the forth step follows from Lemma 12.27. □

The following lemma bounds the L_2-distance between $d_z(t)$ and $d_{z,I}(t)$.

Lemma 12.37. *Let δ_1 be defined as in Lemma 12.9. Let H be defined as in Definition 12.7, $G_{\sigma,b}^{(j)}$ be defined as in Definition 12.2 and $I = I_{\sigma,b}^{(j)}$ be the corresponding ideal filter as in Eq. (12.18).*

For any $x \in \mathcal{F}_{k,F}$ and (σ, b) such that Large Offset event does not happen, for any $j \in [B]$, let $z(t) = (x \cdot H) * G_{\sigma,b}^{(j)}(t)$, $d_z(t)$ be defined as Eq. (12.32), $z_I(t)$ and $d_{z,I}(t)$ be defined as Eq. (12.33). Then,

$$\int_{-\infty}^{\infty} |d_{I,z}(t) - d_z(t)|^2 \mathrm{dt} \lesssim \delta_1^2 T \|x(t)\|_T^2$$

Proof. We first have that,

$$\begin{aligned}
&\int_{-\infty}^{\infty} |z_I(t)e^{2\pi \mathbf{i} f_0 \beta} - z(t)e^{2\pi \mathbf{i} f_0 \beta}|^2 \mathrm{d}t \\
= &\int_{-\infty}^{\infty} |z_I(t) - z(t)|^2 \mathrm{d}t \\
\leq &\ \delta_1^2 T \|x(t)\|_T^2, \qquad (12.34)
\end{aligned}$$

where the first step follows from $|e^{2\pi \mathbf{i} f_0 \beta}| = 1$, the second step follows from Lemma 12.28.

Then, we complete the proof as follows:

$$\begin{aligned}
&\int_{-\infty}^{\infty} |d_{I,z}(t) - d_z(t)|^2 \mathrm{d}t \\
= &\int_{-\infty}^{\infty} |z_I(t)e^{2\pi \mathbf{i} f_0 \beta} - z(t)e^{2\pi \mathbf{i} f_0 \beta} - (z_I(t+\beta) - z(t+\beta))|^2 \mathrm{d}t \\
\leq &\ 2\int_{-\infty}^{\infty} |z_I(t)e^{2\pi \mathbf{i} f_0 \beta} - z(t)e^{2\pi \mathbf{i} f_0 \beta}|^2 \mathrm{d}t + 2\int_{-\infty}^{\infty} |z_I(t+\beta) - z(t+\beta)|^2 \mathrm{d}t \\
\lesssim &\ \delta_1^2 T \|x(t)\|_T^2 + \int_{-\infty}^{\infty} |z_I(t+\beta) - z(t+\beta)|^2 \mathrm{d}t \\
\lesssim &\ \delta_1^2 T \|x(t)\|_T^2
\end{aligned}$$

where the first step follows from the definition of $d_{I,z}(t)$ and $d_z(t)$, the second step follows from $(a+b)^2 \leq 2a^2 + 2b^2$, the third step follows from Eq. (12.34), the forth step follows from Lemma 12.28.

□

12.12.2 Ideal post-truncated local-test signal

It is still difficult to directly study the energy bound for $d_{z,I}(t)$. In this section, we further simplify the ideally filtered signal by removing the H filter and consider the untruncted ideally filtered signal $(x * I)(t)$. Then, in the local-test signal, we perform a post-truncation. More specifically, the untruncated ideally filtered signal and the *ideal post-truncated local-test signal* are defined as follows:

$$\begin{aligned} x_I(t) &:= (x * I)(t), \\ d_{I,x}(t) &:= x_I(t) \cdot H(t) \cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t+\beta) \cdot H(t+\beta). \end{aligned} \tag{12.35}$$

Intuitively, $d_{I,x}(t)$ can be viewed as swapping the order of the I and H filters in $d_{I,z}(t)$.

The following lemma shows that $d_{I,z}(t)$ and $d_{I,x}(t)$ are actually the same!

Lemma 12.38. *Let δ_1 be defined as in Lemma 12.9. Let H be defined as in Definition 12.7, $G_{\sigma,b}^{(j)}$ be defined as in Definition 12.2 and $I = I_{\sigma,b}^{(j)}$ be the corresponding ideal filter as in Eq. (12.18).*

For any $x \in \mathcal{F}_{k,F}$, and (σ, b) such that Large Offset event does not happen, let $z_I(t)$ and $d_{z,I}(t)$ be defined as Eq. (12.33), $x_I(t)$ and $d_{x,I}(t)$ be defined as Eq. (12.35).

Then, we have

$$d_{I,z}(t) = d_{I,x}(t) \quad \forall t \in \mathbb{R}.$$

Proof. We have that

$$\begin{aligned} d_{I,z}(t) &= z_I(t) \cdot e^{2\pi \mathbf{i} f_0 \beta} - z_I(t+\beta) \\ &= x_I(t) \cdot H(t) \cdot e^{2\pi \mathbf{i} f_0 \beta} - z_I(t+\beta) \\ &= x_I(t) \cdot H(t) \cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t+\beta) \cdot H(t+\beta) \\ &= d_{I,x}(t), \end{aligned}$$

where the first step follows from the definition of $d_{I,z}(t)$, the second step follows from Lemma 12.26, the third step follows from Lemma 12.26, the last step follows from the definition of $d_{I,x}(t)$. $\square$

The structure of $d_{I,x}(t)$ makes it easy to study its magnitude at any "good point":

Lemma 12.39. *Let H be defined as in Definition 12.7, $G_{\sigma,b}^{(j)}$ be defined as in Definition 12.2 and $I = I_{\sigma,b}^{(j)}$ be the corresponding ideal filter as in Eq. (12.18). Let $U := \{t_0 \in \mathbb{R} \mid H(t) > 1 - \delta_1 \ \forall t \in [t_0, t_0 + \beta]\}$.*

For any $x \in \mathcal{F}_{k,F}$, and (σ, b) such that Large Offset event does not happen, let $x_I(t), d_{x,I}(t)$ be defined as Eq. (12.35). Then, we have

$$|d_{I,x}(t)| \lesssim \left|x_I(t) \cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t+\beta)\right| + \delta_1 k \|x_I(t)\|_T \quad \forall t \in U.$$

Proof. First, for any $t \in U$,

$$\begin{aligned} |x_I(t) \cdot H(t) \cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t) \cdot e^{2\pi \mathbf{i} f_0 \beta}| &= |x_I(t) \cdot H(t) - x_I(t)| \\ &= |x_I(t)| \cdot |1 - H(t)| \\ &\leq \delta_1 |x_I(t)| \\ &\lesssim \delta_1 k \|x_I(t)\|_T \end{aligned} \tag{12.36}$$

where the first step follows from $|e^{2\pi \mathbf{i} f_0 \beta}| = 1$, the second step is straight forward, the third step follows from $H(t) \leq 1$ (Lemma 12.24 Property I, II) and $\forall t \in U, H(t) > 1 - \delta_1$, and the last step follows from Lemma 12.31.

Second, for any $t \in U$,

$$\begin{aligned} |x_I(t+\beta) - x_I(t+\beta) \cdot H(t+\beta)| &= |x_I(t+\beta)| \cdot |1 - H(t+\beta)| \\ &\leq \delta_1 |x_I(t+\beta)| \\ &\lesssim \delta_1 k \|x_I(t)\|_T \end{aligned} \tag{12.37}$$

where the first step is straight forward, the second step follows from $H(t) \leq 1$ (Lemma 12.24 Property I, II) and $\forall t \in U, H(t+\beta) > 1 - \delta_1$, the last step follows from Lemma 12.31.

Combining them together, we have that for any $t \in U$,

$$\begin{aligned}
|d_{I,x}(t)| &= |x_I(t) \cdot H(t) \cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t+\beta) \cdot H(t+\beta)| \\
&\leq |x_I(t) \cdot H(t) \cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t) \cdot e^{2\pi \mathbf{i} f_0 \beta}| + |x_I(t) \cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t+\beta)| \\
&\quad + |x_I(t+\beta) - x_I(t+\beta) \cdot H(t+\beta)| \\
&\lesssim |x_I(t) \cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t+\beta)| + |x_I(t+\beta) - x_I(t+\beta) \cdot H(t+\beta)| + \delta_1 k \|x_I(t)\|_T \\
&\lesssim |x_I(t) \cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t+\beta)| + \delta_1 k \|x_I(t)\|_T
\end{aligned}$$

where the first step follows from the definition of $d_{I,x}(t)$, the second step follows from triangle inequality, the third step follows Eq. (12.36), the forth step follows from Eq. (12.37).

□

Furthermore, we can show that the ideal post-truncated local-test signal is close to the ideal local-test signal without truncation on most of "good points".

Lemma 12.40. *Let H be defined as in Definition 12.7, $G_{\sigma,b}^{(j)}$ be defined as in Definition 12.2 and $I = I_{\sigma,b}^{(j)}$ be the corresponding ideal filter as in Eq. (12.18). Let $U := \{t_0 \in \mathbb{R} \mid H(t) > 1 - \delta_1, \forall t \in [t_0, t_0 + \beta]\}$. Let $D_U(t) := \mathrm{Uniform}(U)$ and $D_{U+\beta}(t) := \mathrm{Uniform}(U + \beta)$.*

For any $x \in \mathcal{F}_{k,F}$, and (σ, b) such that Large Offset event does not happen, let $x_I(t), d_{x,I}(t)$ be defined as Eq. (12.35). Then, we have

$$\|d_{I,x}(t) - (x_I(t) \cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t+\beta))\|_{D_U} \lesssim \delta_1 \|x_I(t)\|_T.$$

Proof. First,

$$\begin{aligned}
\|x_I(t) \cdot H(t) \cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t) \cdot e^{2\pi \mathbf{i} f_0 \beta}\|_{D_U} &= \|x_I(t) \cdot H(t) - x_I(t)\|_{D_U} \\
&\leq \max_{t \in U}\{|1 - H(t)|\} \cdot \|x_I(t)\|_{D_U} \\
&\leq \delta_1 \cdot \|x_I(t)\|_{D_U} \\
&\lesssim \delta_1 \cdot \sqrt{\frac{T}{|U|}} \|x_I(t)\|_T \\
&\lesssim \delta_1 \cdot \|x_I(t)\|_T \qquad (12.38)
\end{aligned}$$

where the first step follows from $|e^{2\pi \mathbf{i} f_0 \beta}| = 1$, the second step is straight forward, the third step follows from $H(t) \leq 1$ (Lemma 12.24 Property I, II) and $\forall t \in U, H(t) > 1 - \delta_1$, the forth step follows from the definition of the norm

$$\|x(t)\|_{D_U}^2 = \frac{1}{|U|} \int_U |x(t)|^2 \mathrm{d}t \leq \frac{1}{|U|} \int_{[0,T]} |x(t)|^2 \mathrm{d}t = \frac{T}{|U|} \|x(t)\|_T^2,$$

and the last step follows from Lemma 12.31.

Second,

$$\begin{aligned}
\|x_I(t+\beta) - x_I(t+\beta) \cdot H(t+\beta)\|_{D_U} &\leq \max_{t \in U}\{|1 - H(t+\beta)|\} \cdot \|x_I(t+\beta)\|_{D_U} \\
&\leq \delta_1 \cdot \|x_I(t)\|_{D_{U+\beta}} \\
&\lesssim \delta_1 \cdot \frac{1}{|U+\beta|} \|x_I(t)\|_{D_1} \\
&\lesssim \delta_1 \cdot \|x_I(t)\|_{D_1} \qquad (12.39)
\end{aligned}$$

where the first step is straight forward, the second step follows from $H(t) \leq 1$ (Lemma 12.24 Property I, II) and $\forall t \in U, H(t+\beta) > 1 - \delta_1$, the third step follows from the definition of the norm, the forth step follows from $|U + \beta| = |U| \gtrsim 1$.

Then, we have that,

$$\begin{aligned}
&\|d_{I,x}(t) - (x_I(t) \cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t+\beta))\|_{D_U} \\
=& \|x_I(t) \cdot H(t) \cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t+\beta) \cdot H(t+\beta) - (x_I(t) \cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t+\beta))\|_{D_U} \\
\leq& \|x_I(t) \cdot H(t) \cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t) \cdot e^{2\pi \mathbf{i} f_0 \beta}\|_{D_U} + \|x_I(t+\beta) \cdot H(t+\beta) - x_I(t+\beta)\|_{D_U} \\
\lesssim& \delta_1 \cdot \|x_I(t)\|_{D_1} + \|x_I(t+\beta) \cdot H(t+\beta) - x_I(t+\beta)\|_{D_U} \\
\lesssim& \delta_1 \cdot \|x_I(t)\|_{D_1}
\end{aligned}$$

where the first step follows from the definition of $d_{I,x}(t)$, the second step follows from triangle inequality, the third step follows from Eq. (12.38), the forth step follows from Eq. (12.39).

$\square$

12.12.3 Energy bound for local-test signals

In this section, we prove the following lemma, which gives an energy bound for local-test signals.

Lemma 12.41 (Energy bound for local-test signals)**.** *Let H be defined as in Definition 12.7, $G_{\sigma,b}^{(j)}$ be defined as in Definition 12.2. Let U, D_U be defined as in Lemma 12.40.*

For any $x \in \mathcal{F}_{k,F}$, and (σ, b) such that Large Offset event does not happen, let $z(t) = (x \cdot H) * G_{\sigma,b}^{(j)}(t)$ and $d_z(t)$ be defined as Eq. (12.32). Then, we have

$$|d_z(t)|^2 \lesssim \min\Big\{\frac{k}{1 - |2t/T - 1|}, k^2\Big\} \cdot \|d_z(t)\|_{D_U}^2 + \delta_1 \|x(t)\|_T^2 \quad \forall t \in U.$$

Proof. Let $I = I_{\sigma,b}^{(j)}$ be the corresponding ideal filter as in Eq. (12.18). Let $S := \mathrm{supp}(\widehat{x} * \widehat{H})$ be the support set of $\widehat{x} * \widehat{H}$. We have $|S| \leq \Delta$.

Let $z_I(t)$, $d_{z,I}(t)$ be defined as in Lemma 12.36 and $x_I(t)$, $d_{I,x}(t)$ be defined as in Lemma 12.38.

Before proving the energy bound for $|d_z(t)|$, we first consider the signal $x_I(t) \cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t+\beta)$. By Fourier transformation, we know that its Fourier coefficient of a frequency f is:

$$\widehat{x}_I(f) e^{2\pi \mathbf{i} f_0 \beta} - \widehat{x}_I(f) e^{2\pi \mathbf{i} f \beta} = \widehat{x}(f) \cdot \widehat{I}(f) e^{2\pi \mathbf{i} f_0 \beta} - \widehat{x}(f) \cdot \widehat{I}(f) e^{2\pi \mathbf{i} f \beta}$$

Thus, $x_I(t) \cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t+\beta)$ is at most k-Fourier-sparse.

Let $[L, R] := U$. By Fourier-sparse signals' energy bound (Theorem 12.7 and Theorem 12.6), we have

$$\begin{aligned} |x_I(t) \cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t+\beta)|^2 &\lesssim \min\Big\{ \frac{k}{\min\{R-t, t-L\}}, k^2 \Big\} \cdot \|x_I(t) \cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t+\beta)\|_{D_U}^2 \\ &\lesssim \min\{ \frac{k}{1 - |2t/T - 1|}, k^2 \} \cdot \|x_I(t) \cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t+\beta)\|_{D_U}^2 \end{aligned} \tag{12.40}$$

where the first step follows from applying Theorem 12.7 with $x(t) = x(Tt/2 + T/2)$ and applying Theorem 12.6 with $T = |U|, x(t) = x(t+L)$, the second step follows from $[-1 + 0.5/k, 1 - 0.5/k] \subseteq [L, R]$, which implies that $k(\min\{R-t, t-L\})^{-1} \lesssim k(1 - |2t/T - 1|)^{-1}$ for any $|t| \in [L + 1/k, R - 1/k]$. Moreover, for any $|t| \in [L, L + 1/k] \cup [R - 1/k, R]$, $k^2 \lesssim k(1 - |2t/T - 1|)^{-1}$.

The RHS can be upper bounded by:

$$\begin{aligned} \|x_I(t) \cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t+\beta)\|_{D_U}^2 &\le 2\|d_{I,x}(t)\|_{D_U}^2 + 2\|d_{I,x}(t) - (x_I(t) \cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t+\beta))\|_{D_U}^2 \\ &\lesssim \|d_{I,x}(t)\|_{D_U}^2 + \delta_1^2 \|x_I(t)\|_T^2 \\ &= \|d_{I,z}(t)\|_{D_U}^2 + \delta_1^2 \|x_I(t)\|_T^2 \\ &\lesssim \|d_z(t)\|_{D_U}^2 + \|d_{I,z}(t) - d_z(t)\|_{D_U}^2 + \delta_1^2 \|x_I(t)\|_T^2 \\ &\lesssim \|d_z(t)\|_{D_U}^2 + \|d_{I,z}(t) - d_z(t)\|_{D_U}^2 + \delta_1^2 \|x(t)\|_T^2 \end{aligned} \tag{12.41}$$

where the first step follows from $(a+b)^2 \le 2a^2 + 2b^2$, the second step follows from Lemma 12.40, the third step follows from Lemma 12.38, the forth step follows from

$(a+b)^2 \leq 2a^2 + 2b^2$, the last step follows from Claim 12.42. For the second term, we have that

$$\begin{aligned}
& \|d_{I,z}(t) - d_z(t)\|_{D_U}^2 \\
\lesssim & \frac{1}{|U|}\int_U |d_{I,z}(t) - d_z(t)|^2 \mathrm{d}t \\
\lesssim & \frac{1}{|U|}\int_{-\infty}^{\infty} |d_{I,z}(t) - d_z(t)|^2 \mathrm{d}t \\
\lesssim & \frac{1}{|U|}\delta_1^2 \|x(t)\|_T^2 \\
\lesssim & \delta_1^2 \|x(t)\|_T^2
\end{aligned}$$

where the first step follows from the definition of the norm, second step is straight forward, the third step follows from Lemma 12.37 with appropriate scaling, the forth step follows from $|U| \gtrsim 1$. Hence,

$$\|x_I(t)\cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t+\beta)\|_{D_U}^2 \lesssim \|d_z(t)\|_{D_U}^2 + \delta_1^2 \|x(t)\|_T^2. \tag{12.42}$$

Therefore, we have that

$$\begin{aligned}
|d_{I,z}(t)|^2 = & |d_{I,x}(t)|^2 \\
\lesssim & (|x_I(t)\cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t+\beta)| + \delta_1 k \|x_I(t)\|_T)^2 \\
\lesssim & |x_I(t)\cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t+\beta)|^2 + \delta_1^2 k^2 \|x_I(t)\|_T^2 \\
\lesssim & |x_I(t)\cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t+\beta)|^2 + \delta_1 \|x_I(t)\|_T^2 \\
\lesssim & \min\{\frac{k}{1-|2t/T-1|}, k^2\} \cdot \|x_I(t)\cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t+\beta)\|_{D_U}^2 + \delta_1 \|x_I(t)\|_T^2 \\
\lesssim & \min\{\frac{k}{1-|2t/T-1|}, k^2\} \cdot \|x_I(t)\cdot e^{2\pi \mathbf{i} f_0 \beta} - x_I(t+\beta)\|_{D_U}^2 + \delta_1 \|x(t)\|_T^2 \\
\lesssim & \min\{\frac{k}{1-|2t/T-1|}, k^2\} \cdot (\|d_z(t)\|_{D_U}^2 + \delta_1^2 \|x(t)\|_T^2) + \delta_1 \|x(t)\|_T^2 \\
\lesssim & \min\{\frac{k}{1-|2t/T-1|}, k^2\} \cdot \|d_z(t)\|_{D_U}^2 + \delta_1 \|x(t)\|_T^2,
\end{aligned} \tag{12.43}$$

where the first step follows from Lemma 12.38, the second step follows from Lemma 12.39, the third step follows from $(a+b)^2 \leq 2a^2 + 2b^2$, the forth step follows from

$\delta_1 k^2 \leq 1$, the fifth step follows from Eq. (12.40), the six step follows from Claim 12.42, the seventh step follows from Eq. (12.42), the last step follows from $\delta_1 k^2 \lesssim 1$.

Finally, we have

$$\begin{aligned}|d_z(t)|^2 &\leq 2|d_z(t) - d_{I,z}(t)|^2 + 2|d_{I,z}(t)|^2\\ &\leq 2\delta_1^2 T|S| \cdot \|x(t)\|_T^2 + 2|d_{I,z}(t)|^2\\ &\leq 2\delta_1 \cdot \|x(t)\|_T^2 + 2|d_{I,z}(t)|^2\\ &\lesssim \delta_1 \cdot \|x(t)\|_T^2 + \min\{\frac{k}{1-|2t/T-1|}, k^2\} \cdot \|d_z(t)\|_{D_U}^2\end{aligned}$$

where the first step follows from $(a+b)^2 \leq 2a^2 + 2b^2$, the second step follows from Lemma 12.36, the third step follows from $\delta_1 T|S| \leq 1$, the forth step follows from Eq. (12.43).

The lemma is then proved.

□

Claim 12.42 (Energy Reduction by Ideal Filter). *Let H be defined as in Definition 12.7, $G_{\sigma,b}^{(j)}$ be defined as in Definition 12.2 and $I = I_{\sigma,b}^{(j)}$ be the corresponding ideal filter as in Eq.* (12.18)*.*

For any $x \in \mathcal{F}_{k,F}$, for any (σ, b) such that Large Offset event does not happen, then we have

$$\|(x * I)(t)\|_T \lesssim \|x(t)\|_T$$

Proof. Let $S = \mathrm{supp}(\widehat{x} * \widehat{H})$. We have that

$$\begin{aligned}
T\|(x * I)(t)\|_T &\lesssim T\|(x * I)(t) \cdot H(t)\|_T \\
&= \int_0^T |(x * I)(t) \cdot H(t)|^2 \mathrm{d}t \\
&\leq \int_{-\infty}^{\infty} |(x * I)(t) \cdot H(t)|^2 \mathrm{d}t \\
&= \int_{-\infty}^{\infty} |(\widehat{x} \cdot \widehat{I})(f) * \widehat{H}(f)|^2 \mathrm{d}f \\
&= \int_S |(\widehat{x} \cdot \widehat{I})(f) * \widehat{H}(f)|^2 \mathrm{d}f \\
&= \int_S |\widehat{x}(f) * \widehat{H}(f)|^2 \mathrm{d}f \\
&\leq \int_{-\infty}^{\infty} |\widehat{x}(f) * \widehat{H}(f)|^2 \mathrm{d}f \\
&= \int_{-\infty}^{\infty} |x \cdot H(t)|^2 \mathrm{d}t \\
&\lesssim \int_0^T |x(t)|^2 \mathrm{d}t \\
&= T\|x(t)\|_T^2
\end{aligned}$$

where the first step follows from Lemma 12.24 Property V, the second step follows from the definition of the norm, the third step is straight forward, the forth step follows from Parseval's theorem, the fifth and sixth steps follow from Large Offset event not happening, the seventh step is straight forward, the eighth step follows from Parseval's theorem, the ninth step follows from Lemma 12.24 Property IV and VI, the last step follows from the definition of the norm.

□

12.13 Empirical Energy Estimation

The goal of this section is to show how to estimate a signal's energy using a few samples. We start with a general sampling and reweighing method (see Section 12.13.1). Then, combining with the energy bounds derived in previous section,

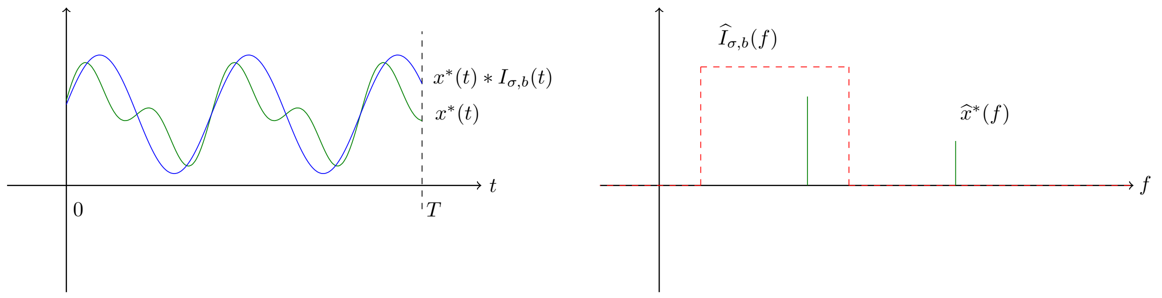


Figure 12.11: An illustration of the energy reduction by the ideal filter. $I_{\sigma,b}$ is the ideal filter and $x^*(t)$ is a Fourier sparse signal. The energy of $x^*(t)$ in duration $[0,T]$ is reduced by applying the ideal filter, i.e., $\|x^* * I_{\sigma,b}(t)\|_T \lesssim \|x^*(t)\|_T$.

we obtain sample-efficient energy estimation methods for Fourier-sparse signals and filtered signals (see Section 12.13.2). We further extend our methods to estimate the energy of filtered signals and local-test signals within a *sub-interval* in the time duration (see Section 12.13.3). Finally, we prove several technical lemmas (see Section 12.13.4).

Throughout this section, for the convenience, we use a slightly different notation for the T-norm:

$$\|z\|_T^2 := \frac{1}{2T}\int_{-T}^{T} |z(t)|^2 \mathrm{d}t.$$

This results of using this T-norm is equivalent with the result of the norm taking on $[0,T]$, since we can always re-scaling the signal and transform the result into the new T-norm result.

12.13.1 Sampling and reweighing

In this section, we provide a generic sample-efficient method for estimating the energy of any function using discrete samples with proper weights.

Lemma 12.43. *Let $k \in \mathbb{N}_+$ and D be a probability distribution such that $\int_{-T}^{T} D(t)\mathrm{d}t = 1$. For any $\varepsilon, \rho \in (0,1)$ and function $z : \mathbb{R} \to \mathbb{C}$, let $S_D = \{t_1, \cdots, t_s\}$ be a set of* *i.i.d. samples from D of size*

$$s \geq \Big(\max_{t\in[-T,T]} \frac{|z(t)|^2}{D(t)}\Big) \cdot O\Big(\frac{\log(1/\rho)}{\varepsilon^2 T \|z(t)\|_T^2}\Big).$$

Let the weight vector $w \in \mathbb{R}^s$ be defined by $w_i := 1/(2TsD(t_i))$ for $i \in [s]$.

Then with probability at least $1-\rho$, we have

$$(1-\varepsilon)\|z(t)\|_T^2 \leq \|z(t)\|_{S_D,w}^2 \leq (1+\varepsilon)\|z(t)\|_T^2,$$

where $\|z\|_T^2 := \frac{1}{2T}\int_{-T}^{T} |z(t)|^2 \mathrm{d}t$.

Proof. Let $M := \max_{t\in[-T,T]} \frac{|z(t)|^2}{D(t)}$. Let $z_D(t) := \frac{1}{M}\frac{|z(t)|^2}{D(t)}$. By applying Chernoff bound (Lemma A.2) for the random variables $z_D(t_1), \ldots, z_D(t_s)$, we get that,

$$\Pr_{t_i \sim D}\Big[\Big|\sum_{i=1}^{s} z_D(t_i) - \mu\Big| \leq \varepsilon\mu\Big] \geq 1 - 2\exp(-\varepsilon^2\mu/3), \tag{12.44}$$

where $\mu := \sum_{i=1}^{s} \mathbb{E}_{t_i\sim D}[z_D(t_i)] = s \cdot \mathbb{E}_{t\sim D}[z_D(t)]$.

We first consider the expectation:

$$\begin{aligned}\mathbb{E}_{t\sim D}[z_D(t)] = & \int_{-T}^{T} D(t) \cdot \frac{1}{M}\frac{|z(t)|^2}{D(t)}\mathrm{d}t \\ = & \frac{1}{M}\int_{-T}^{T} |z(t)|^2 \mathrm{d}t \\ = & \frac{2T}{M}\|z(t)\|_T^2\end{aligned}$$

where the first step follows from the definition of expectation, the second step is straightforward, the third step follows from the definition of the norm. Thus,

$$\mu = s \cdot \mathbb{E}_{t\sim D}[z_D(t)] = \frac{2Ts}{M}\|z(t)\|_T^2. \tag{12.45}$$

Then, we consider the sum of samples:

$$\begin{aligned}\sum_{i=1}^{s} z_D(t_i) = & \sum_{i=1}^{s} \frac{1}{M}\frac{|z(t_i)|^2}{D(t_i)} \\ = & \sum_{i=1}^{s} \frac{2w_i Ts}{M}|z(t_i)|^2 \\ = & \frac{2Ts}{M}\|z(t)\|_{S_D,w}^2 \end{aligned} \tag{12.46}$$

where the first step follows from the definition of z_D, the second step follows from the definition of w_i, the last step follows from the definition of the norm.

Putting Eqs. (12.44) - (12.46) together, we get that with probability at least $1 - 2\exp(-\varepsilon^2\mu/3)$,

$$\Big|\frac{2Ts}{M}\|z(t)\|_{S_D,w}^2 - \frac{2Ts}{M}\|z(t)\|_T^2\Big| \leq \varepsilon \cdot \frac{2Ts}{M}\|z(t)\|_T^2,$$

which can be simplified at follows:

$$|\|z(t)\|_{S_D,w}^2 - \|z(t)\|_T^2| \leq \varepsilon \cdot \|z(t)\|_T^2.$$

Finally, we need the success probability to be at least $1 - \rho$, which requires that:

$$\begin{aligned} 1 - 2\exp\Big(-\frac{\varepsilon^2}{3}\frac{2Ts}{M}\|z(t)\|_T^2\Big) &= 1 - 2\exp\Big(-\frac{\varepsilon^2}{3}\frac{2Ts}{\cdot\max_{t\in[-T,T]}\{|z(t)|^2/D(t)\}}\|z(t)\|_T^2\Big) \\ &\geq 1 - \rho. \end{aligned}$$

Hence, we need the sample complexity s to be at least

$$s \geq \Big(\max_{t\in[-T,T]} \frac{|z(t)|^2}{D(t)}\Big) \cdot O\Big(\frac{\log(1/\rho)}{\varepsilon^2 T\|z(t)\|_T^2}\Big).$$

□

12.13.2 Energy estimation for Fourier-sparse signals and filtered signals

The goal of this section is to apply Lemma 12.43 for Fourier-sparse signals and filtered signals.

The following lemma defines the sampling distribution:

Lemma 12.44. *For $k \in \mathbb{N}_+$, define a probability distribution D as follows:*

$$D(t) := \begin{cases} c \cdot (1 - |t/T|)^{-1}T^{-1}, & \text{for } |t| \leq T(1 - 1/k) \\ c \cdot kT^{-1}, & \text{for } |t| \in [T(1 - 1/k), T] \end{cases} \tag{12.47}$$

where $c = \Theta(\log(k)^{-1})$ is a normalization factor such that $\int_{-T}^{T} D(t)\mathrm{d}t = 1$. Then, D is well-defined.

Proof. We justify that D can be normalized with $c = \Theta(\log(k)^{-1}))$. By the condition $\int_{-T}^{T} D(t)\mathrm{d}t = 1$, we have

$$2\int_0^{T(1-1/k)} \frac{c}{(1-|t/T|)T}\mathrm{d}t + 2\int_{T(1-1/k)}^{T} c\cdot\frac{k}{T}\mathrm{d}t = 1,$$

which implies that

$$\begin{aligned} c^{-1} &= 2\int_0^{T(1-1/k)} \frac{1}{(1-|t/T|)T}\mathrm{d}t + 2\int_{T(1-1/k)}^{T} \frac{k}{T}\mathrm{d}t \\ &\eqsim \log(k) + 1 \\ &= \Theta(\log(k)). \end{aligned}$$

Thus, we get that $c = \Theta(\log(k)^{-1})$.

$\square$

The following lemma gives the sampling complexity for estimating the energy of a Fourier-sparse signal. The main idea is to apply the energy bounds in Section 12.5.

Lemma 12.45 (Energy estimation for Fourier-sparse signals)**.** *Let D be the probability distribution defined as Eq. (12.47). Let $x \in \mathcal{F}_{k,F}$. For any $\varepsilon, \rho \in (0,1)$, let $S_D = \{t_1, \cdots, t_s\}$ be a set of i.i.d. samples from $D(t)$ of size $s \geq O(\varepsilon^{-2}k\log(k)\log(1/\rho))$. Let the weight vector $w \in \mathbb{R}^s$ be defined by $w_i := 1/(2TsD(t_i))$ for $i \in [s]$.*

Then with probability at least $1-\rho$, we have

$$(1-\varepsilon)\|x(t)\|_T^2 \leq \|x(t)\|_{S_D,w}^2 \leq (1+\varepsilon)\|x(t)\|_T^2.$$

Proof. By applying Lemma 12.43, we have that the desired result satisfy when

$$s \geq \Big(\max_{t\in[-T,T]} \frac{|x(t)|^2}{D(t)}\Big) \cdot O\Big(\frac{\log(1/\rho)}{\varepsilon^2 T\|x(t)\|_T^2}\Big).$$

By Fourier-sparse signals' energy bound (Theorem 12.6 and Theorem 12.7 with $x(t) = x(T\cdot t)$), we have that

$$|x(t)|^2 \lesssim \min\Big\{\frac{k}{1-|t/T|}, k^2\Big\} \cdot \|x(t)\|_T^2 \quad \forall t \in [-T,T]. \tag{12.48}$$

Thus,

$$\begin{aligned}
& \max_{t\in[-T,T]} \frac{|x(t)|^2}{D(t)} \\
\lesssim\ & \max_{t\in[-T,T]} \min\Big\{\frac{k}{1-|t/T|}, k^2\Big\} \cdot \frac{\|x(t)\|_T^2}{D(t)} \\
\lesssim\ & \max_{t\in[-T,T]} \min\Big\{\frac{k}{1-|t/T|}\frac{T(1-|t/T|)}{c}, k^2\frac{T}{ck}\Big\} \cdot \|x(t)\|_T^2 \\
=\ & kT\|x(t)\|_T^2/c \\
\simeq\ & k\log(k)T\|x(t)\|_T^2, && (12.49)
\end{aligned}$$

where the first step follows from Eq. (12.48), the second step follows from the definition of $D(t)$, the third step is straight forward, the forth step follows from $c = \Theta(\log(k)^{-1})$.

Hence, we get that

$$s \geq O(k\log(k)T\|x(t)\|_T^2) \cdot O\Big(\frac{\log(1/\rho)}{\varepsilon^2 T\|x(t)\|_T^2}\Big) = O(\varepsilon^{-2}k\log(k)\log(1/\rho)).$$

The lemma is then proved.

$\square$

Using the energy bound for filtered signals, we immediately get the following lemma.

Lemma 12.46 (Energy estimation for filtered signals)**.** *Let D be the probability distribution defined as Eq. (12.47). Let $x \in \mathcal{F}_{k,F}$. Let H be defined as in Definition 12.7. Let $G_{\sigma,b}^{(j)}$ be defined as in Definition 12.2. Let $j \in [B]$ satisfying that there exists an f^* with $h_{\sigma,b}(f^*) = j$ such that:*

$$\int_{f^*-\Delta_h}^{f^*+\Delta_h} |\widehat{x\cdot H}(f)|^2 \mathrm{d}f \geq T\mathcal{N}^2/k,$$

where $\mathcal{N}^2 \geq \delta\|x\|_T^2$. Let $z(t) := (x\cdot H) * G_{\sigma,b}^{(j)}(t)$ be the filtered signal.

For any $\varepsilon, \rho \in (0,1)$, let $S_D = \{t_1, \cdots, t_s\}$ be a set of i.i.d. samples from $D(t)$ of size $s \geq O(\varepsilon^{-2} k \log(k) \log(1/\rho))$. Let the weight vector $w \in \mathbb{R}^s$ be defined by $w_i := 1/(2TsD(t_i))$ for $i \in [s]$.

Then when Large Offset event not happens, with probability at least $1-\rho$, we have

$$(1-\varepsilon)\|z(t)\|_T^2 \leq \|z(t)\|_{S_D,w}^2 \leq (1+\varepsilon)\|z(t)\|_T^2.$$

Proof. By applying Lemma 12.43, we have that the desired result requires that

$$s \geq \Big(\max_{t\in[-T,T]} \frac{|z(t)|^2}{D(t)}\Big) \cdot O\Big(\frac{\log(1/\rho)}{\varepsilon^2 T \|z(t)\|_T^2}\Big).$$

By the filtered signals' energy bound (Corollary 12.34), we have that

$$\begin{aligned} |z(t)|^2 &\lesssim \min\Big\{\frac{k \cdot H(t) + \delta}{1 - |t/T|}, k^2\Big\} \cdot \|z(t)\|_T^2 \\ &\lesssim \min\Big\{\frac{k}{1-|t/T|}, k^2\Big\} \cdot \|z(t)\|_T^2. \end{aligned} \tag{12.50}$$

where the second step follows from $H(t) \lesssim 1$ (Lemma 12.24 Property I, II). Then, we get that

$$\begin{aligned} & \max_{t\in[-T,T]} \frac{|z(t)|^2}{D(t)} \\ \lesssim\ & \max_{t\in[-T,T]} \min\Big\{\frac{k}{1-|t/T|}, k^2\Big\} \cdot \|z(t)\|_T^2 \\ \lesssim\ & \max_{t\in[-T,T]} \min\Big\{\frac{k}{1-|t/T|}\frac{T(1-|t/T|)}{c}, k^2\frac{T}{ck}\Big\} \cdot \|z(t)\|_T^2 \\ =\ & kT\|z(t)\|_T^2/c \\ \simeq\ & k\log(k)T\|z(t)\|_T^2, \end{aligned} \tag{12.51}$$

where the first step follows from Eq. (12.50), the second step follows from the definition of $D(t)$, the third step is straight forward, the forth step follows from $c = \Theta(\log(k)^{-1})$.

As a result,

$$s \geq O(k\log(k)T\|z(t)\|_T^2) \cdot O\Big(\frac{\log(1/\rho)}{\varepsilon^2 T\|z(t)\|_T^2}\Big) = O(\varepsilon^{-2}k\log(k)\log(1/\rho)).$$

The lemma is then proved.

$\square$

12.13.3 Partial energy estimation for filtered signals and local-test signals

In this section, we consider a variant version of energy estimation problem, which we are given a sub-interval $U \subseteq [-T, T]$ and we only want to estimate the energy within this interval.

The following lemma gives the sampling distribution with respect to U.

Lemma 12.47. *Let $U = [L, R]$ such that $[-T(1-1/k), T(1-1/k)] \subseteq U \subseteq [-T, T]$. For $k \in \mathbb{N}_+$, define a probability distribution D_U as follows:*

$$D_U(t) := \begin{cases} c \cdot (1 - |t/T|)^{-1} T^{-1}, & \textit{for } |t| \leq T(1-1/k) \wedge t \in U \\ c \cdot k T^{-1}, & \textit{for } |t| \in [T(1-1/k), T] \wedge t \in U \end{cases} \tag{12.52}$$

where $c = \Theta(\log(k)^{-1})$ is a normalization factor such that $\int_{-T}^{T} D_U(t) \mathrm{d}t = 1$. Then, D_U is well-defined.

Proof. We compute the normalization factor of D_U in below. The condition that $\int_{-T}^{T} D_U(t) \mathrm{d}t = 1$ requires that

$$2 \int_0^{T(1-1/k)} \frac{c}{(1 - |t/T|)T} \mathrm{d}t + \int_{T(1-1/k)}^{R} c \cdot \frac{k}{T} \mathrm{d}t + \int_{L}^{-T(1-1/k)} c \cdot \frac{k}{T} \mathrm{d}t = 1,$$

which implies that

$$\begin{aligned} c^{-1} = {} & 2 \int_0^{T(1-1/k)} \frac{1}{(1 - |t/T|)T} \mathrm{d}t + \int_{T(1-1/k)}^{R} \frac{k}{T} \mathrm{d}t + \int_{L}^{-T(1-1/k)} \frac{k}{T} \mathrm{d}t \\ \lesssim {} & \log(k) + 1 \\ = {} & \Theta(\log(k)). \end{aligned}$$

where the second step follows from $R \leq T$ and $L \geq -T$.

Thus, we get that $c = \Theta(\log(k)^{-1})$. $\square$

Similar to Lemma 12.46, we have a sample-efficient approach for estimating the partial energy of a filtered signal.

Lemma 12.48 (Partial energy estimation for filtered signals)**.** *Let $U = [L, R]$ be such that $[-T(1-1/k), T(1-1/k)] \subseteq U$. For $k \in \mathbb{N}_+$, let D_U be the probability distribution defined as Eq.* (12.52)*.*

Let $x \in \mathcal{F}_{k,F}$. Let H be defined as in Definition 12.7, $G_{\sigma,b}^{(j)}$ be defined as in Definition 12.2 with (σ, b) such that Large Offset event does not happen. For any $j \in [B]$, suppose there exists an f^* with $j = h_{\sigma,b}(f^*)$ satisfying:

$$\int_{f^*-\Delta_h}^{f^*+\Delta_h} |\widehat{x \cdot H}(f)|^2 \mathrm{d}f \geq T\mathcal{N}^2/k,$$

where $\mathcal{N}^2 \geq \delta\|x\|_T^2$. Let $z(t) = (x \cdot H) * G_{\sigma,b}^{(j)}(t)$ be the filtered signal.

For any $\varepsilon, \rho \in (0, 1)$, let $S_{D_U} = \{t_1, \cdots, t_s\}$ be a set of i.i.d. samples from D_U of size $s \geq O(\varepsilon^{-2} k \log(k) \log(1/\rho))$. Let the weight vector $w \in \mathbb{R}^s$ be defined by $w_i := 2/(TsD_U(t_i))$ for $i \in [s]$.

Then when Large Offset event not happens, with probability at least $1 - \rho$, we have

$$(1 - \varepsilon)\|z\|_U^2 \leq \|z\|_{S_{D_U},w}^2 \leq (1 + \varepsilon)\|z\|_U^2,$$

where $\|z\|_U^2 := \frac{1}{R-L} \cdot \int_L^R |z(t)|^2 \mathrm{d}t$.

Proof. By applying Lemma 12.43, we have that the desired result requires that

$$s \geq \Big(\max_{t \in U} \frac{|z(t)|^2}{D_U(t)}\Big) \cdot O\Big(\frac{\log(1/\rho)}{\varepsilon^2 T \|z(t)\|_T^2}\Big).$$

The first term can be upper bounded as follows:

$$\begin{aligned}
\max_{t\in U} \frac{|z(t)|^2}{D_U(t)} &\lesssim \max_{t\in U} \min\Big\{\frac{k}{1-|t/T|}, k^2\Big\} \cdot \frac{\|z(t)\|_T^2}{D_U(t)} \\
&\lesssim \max_{t\in U} \min\Big\{\frac{k}{1-|t/T|}\frac{T(1-|t/T|)}{c}, k^2\frac{T}{ck}\Big\} \cdot \|z(t)\|_T^2 \\
&= kT\|z(t)\|_T^2/c \\
&\lesssim k\log(k)T\|z(t)\|_T^2 \\
&\lesssim k\log(k)\frac{R-L}{2T-k^2(2T+L-R)} \cdot T\|z(t)\|_U^2 \\
&\le k\log(k) \cdot T\|z(t)\|_U^2, \qquad (12.53)
\end{aligned}$$

where the first step follows from Eq. (12.50), the second step follows from the definition of $D_U(t)$, the third step is straight forward, the forth step follows from $c = \Theta(\log(k)^{-1})$, the fifth step follows from Lemma 12.51, the sixth step follows from $R - L \le 2T - k^2(2T + L - R)$.

Therefore, the sample complexity s should be at least:

$$s \ge O(k\log(k) \cdot T\|z(t)\|_U^2) \cdot O\Big(\frac{\log(1/\rho)}{\varepsilon^2 T\|z(t)\|_T^2}\Big) = O(\varepsilon^{-2}k\log(k)\log(1/\rho)).$$

The proof of the lemma is then completed.

$\square$

Recall that in Section 12.12, we study the local-test signal $d_z(t) = z(t)e^{2\pi\mathbf{i}f_0\beta} - z(t+\beta)$. The following lemma gives a way to estimate the partial energy of a local-test signal. It can be proved by the same strategy with the energy bound in Lemma 12.41.

Lemma 12.49 (Partial energy estimation for local-test signals)**.** *Let $x \in \mathcal{F}_{k,F}$. Let H be defined as in Definition 12.7, $G_{\sigma,b}^{(j)}$ be defined as in Definition 12.2 with (σ, b) such that Large Offset event does not happen. Let $z(t) = (x \cdot H) * G_{\sigma,b}^{(j)}(t)$ be the filtered signal. Let $U := \{t_0 \in \mathbb{R} \mid H(t) > 1 - \delta_1, \forall t \in [t_0, t_0 + \beta]\}$. Let D_U be the probability distribution defined as Eq. (12.52). Let $D_H(t) :=$* Uniform$(\{t \in \mathbb{R} \mid H(t) > 1 - \delta_1\})$.

For any $\varepsilon, \rho \in (0,1)$, let $S_{D_U} = \{t_1, \cdots, t_s\}$ be a set of i.i.d. samples from D_U of size $s \geq O(k \log(k) \log(1/\rho))$. Let the weight vector $w \in \mathbb{R}^s$ be defined by $w_i := 2/(Ts D_U(t_i))$ for $i \in [s]$.

Let $d_z(t) = z(t)e^{2\pi \mathbf{i} f_0 \beta} - z(t+\beta)$ be the local-test signal. Then, with probability at least $1-\rho$, we have

$$\|d_z(t)\|_{S_{D_U},w}^2 \leq 2\|d_z(t)\|_U^2 + \sqrt{\delta_1}\|x(t)\|_T \cdot \|d_z(t)\|_U.$$

Proof. By Lemma 12.43, we have that when

$$s \geq \Big(\max_{t\in U} \frac{|d_z(t)|^2}{D_U(t)}\Big) \cdot O\Big(\frac{\log(1/\rho)}{\xi^2 |U| \cdot \|d_z(t)\|_U^2}\Big),$$

the following result holds with probability at least $1-\rho$,

$$\|d_z(t)\|_{S_{D_U},w}^2 \in (1 \pm \xi)\|d_z(t)\|_U^2, \tag{12.54}$$

where ξ is a parameter to be chosen later.

By the energy bound for local-test signals (Lemma 12.41), we have that for any $t \in U$,

$$|d_z(t)|^2 \lesssim \min\Big\{\frac{k}{1-|t/T|}, k^2\Big\} \cdot \|d_z(t)\|_U^2 + \delta_1 \|x(t)\|_T^2. \tag{12.55}$$

Then, we get that

$$\begin{aligned}
&\max_{t\in U} \frac{|d_z(t)|^2}{D_U(t)} \\
\lesssim\ & \max_{t\in U} \min\Big\{\frac{k}{1-|t/T|}, k^2\Big\} \cdot \frac{\|d_z(t)\|_U^2 + \delta_1\|x(t)\|_T^2}{D_U(t)} \\
\lesssim\ & \max_{t\in U} \min\Big\{\frac{k}{1-|t/T|}\frac{T(1-|t/T|)}{c}, k^2\frac{T}{ck}\Big\} \cdot (\|d_z(t)\|_U^2 + \delta_1\|x(t)\|_T^2) \\
=\ & kTc^{-1} \cdot (\|d_z(t)\|_U^2 + \delta_1\|x(t)\|_T^2) \\
\simeq\ & k\log(k)T \cdot (\|d_z(t)\|_{D_U}^2 + \delta_1\|x(t)\|_{D_1}^2),
\end{aligned} \tag{12.56}$$

where the first step follows from Eq. (12.55), the second step follows from the definition of $D_U(t)$, the third step is straight forward, the forth step follows from $c = \Theta(\log(k)^{-1})$.

As a result, the sample complexity is

$$\begin{aligned}
s &\geq k\log(k)T \cdot (\|d_z(t)\|_{D_U}^2 + \delta_1\|x(t)\|_{D_1}^2) \cdot O\Big(\frac{\log(1/\rho)}{\xi^2|U| \cdot \|d_z(t)\|_U^2}\Big) \\
&\simeq \xi^{-2} \cdot k\log(k) \cdot (1 + \frac{\delta_1\|x(t)\|_T^2}{\|d_z(t)\|_U^2}) \cdot \log(1/\rho) \\
&= k\log(k) \cdot \log(1/\rho),
\end{aligned}$$

where the first step follows from Eq. (12.56), the second step follows from $|U| \gtrsim T$, the third step follows by taking ξ to be such that

$$\xi^{-2}(1 + \frac{\delta_1\|x(t)\|_T^2}{\|d_z(t)\|_U^2}) \simeq 1.$$

It remains to bound the estimation error. We have that

$$\begin{aligned}
\|d_z(t)\|_{S_D,w}^2 &\leq (1+\xi)\|d_z(t)\|_U^2 \\
&\simeq \Big(1 + \sqrt{1 + \frac{\delta_1\|x(t)\|_{D_1}^2}{\|d_z(t)\|_U^2}}\Big)\|d_z(t)\|_U^2 \\
&\leq \Big(2 + \frac{\sqrt{\delta_1}\|x(t)\|_T}{\|d_z(t)\|_U}\Big)\|d_z(t)\|_U^2 \\
&\leq 2\|d_z(t)\|_U^2 + \sqrt{\delta_1}\|x(t)\|_T \cdot \|d_z(t)\|_U
\end{aligned}$$

where the first step follows from Eq. (12.54), the second step follows from the setting of ε, the third step follows from $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$, the forth step is straight forward.

The lemma is then proved. $\square$

12.13.4 Technical lemmas

We prove two technical lemmas in this section.

The following lemma bounds the energy of a Fourier-sparse signal within time duration $[L,R] \subseteq [-T,T]$ by its total energy.

Lemma 12.50 (Partial energy of Fourier-sparse signal)**.** *Given $k \in \mathbb{Z}_+, F \in \mathbb{R}_+$. For any $x \in \mathcal{F}_{k,F}$, $[L,R] \subseteq [-T(1-O(\frac{1}{k^2})), T(1-O(\frac{1}{k^2}))]$, we have that,*

$$\frac{2T - k^2(2T + L - R)}{R-L}\|x(t)\|_T^2 \lesssim \frac{1}{R-L}\int_L^R |x(t)|^2 \mathrm{d}t \leq \frac{2T}{R-L}\|x(t)\|_T^2.$$

Proof. For the upper bound, we have that

$$\frac{1}{R-L}\int_L^R |x(t)|^2 \mathrm{d}t \leq \frac{1}{R-L}\int_{-T}^T |x(t)|^2 \mathrm{d}t \leq \frac{2T}{R-L}\|x(t)\|_T^2,$$

where the first step is straight forward, the second step follows from the definition of the norm.

For the lower bound, we have that

$$\begin{aligned}\int_L^R |x(t)|^2 \mathrm{d}t =& \int_{-T}^T |x(t)|^2 \mathrm{d}t - \int_{-T}^L |x(t)|^2 \mathrm{d}t - \int_R^T |x(t)|^2 \mathrm{d}t \\ &\geq 2T\|x(t)\|_T^2 - (L+T)\cdot \max_{t\in[-T,L]}|x(t)|^2 - (T-R)\cdot \max_{t\in[R,T]}|x(t)|^2 \\ &\gtrsim 2T\|x(t)\|_T^2 - (L+T)\cdot k^2\|x(t)\|_T^2 - (T-R)\cdot k^2\|x(t)\|_T^2 \\ &= (2T - k^2(2T+L-R))\|x(t)\|_T^2,\end{aligned}$$

where the first step is straight forward, the second step follows from the definition of the norm, the third step follows from Theorem 12.6, the forth step is straight forward.

□

By replacing the energy bound for Fourier-sparse signals with the energy bound for filtered signals (Corollary 12.34), we obtain the following lemma:

Lemma 12.51 (Partial energy of filtered signal)**.** *Given $k \in \mathbb{N}$ and $F \in \mathbb{R}_+$. Let $x \in \mathcal{F}_{k,F}$. Let H be defined as in Definition 12.7, $G_{\sigma,b}^{(j)}$ be defined as in Definition 12.2 with (σ, b) such that Large Offset event does not happen.*

For any $j \in [B]$, suppose there exists an f^* with $j = h_{\sigma,b}(f^*)$ satisfying:

$$\int_{f^*-\Delta_h}^{f^*+\Delta_h} |\widehat{x\cdot H}(f)|^2 \mathrm{d}f \geq T\mathcal{N}^2/k,$$

where $\mathcal{N}^2 \geq \delta\|x\|_T^2$. Then, for $z(t) = (x\cdot H) * G_{\sigma,b}^{(j)}(t)$, we have that

$$\frac{2T - k^2(2T+L-R)}{R-L}\|z(t)\|_T^2 \lesssim \frac{1}{R-L}\int_L^R |z(t)|^2 \mathrm{d}t \leq \frac{2T}{R-L}\|z(t)\|_T^2.$$

Proof. For the upper bound, we have that

$$\frac{1}{R-L}\int_L^R |z(t)|^2 \mathrm{d}t \leq \frac{1}{R-L}\int_{-T}^T |z(t)|^2 \mathrm{d}t \leq \frac{2T}{R-L}\|z(t)\|_T^2,$$

where the first step is straight forward, the second step follows from the definition of the norm.

For the lower bound, we have that

$$\begin{aligned}\int_L^R |z(t)|^2 \mathrm{d}t &= \int_{-T}^T |z(t)|^2 \mathrm{d}t - \int_{-T}^L |z(t)|^2 \mathrm{d}t - \int_R^T |z(t)|^2 \mathrm{d}t \\ &\geq T\|z(t)\|_T^2 - (L+T)\cdot \max_{t\in[0,L]}|z(t)|^2 - (T-R)\cdot \max_{t\in[R,T]}|z(t)|^2 \\ &\gtrsim T\|z(t)\|_T^2 - (L+T)\cdot k^2\|z(t)\|_T^2 - (T-R)k^2\|z(t)\|_T^2 \\ &= (2T - k^2(2T+L-R))\|z(t)\|_T^2\end{aligned}$$

where the first step is straight forward, the second step follows from the definition of the norm, the third step follows from Corollary 12.34, the forth step is straight forward.

□

12.14 Generate Significant Samples

In this section, we show our significant sample generation procedure for noisy signals. Recall that we use $x^*(t)$ to denote the ground-truth k-Fourier-sparse signal and $x(t) = x^*(t) + g(t)$ to denote the observation signal. We first generalize the energy estimation method in previous section to the noisy signals (see Section 12.14.1). Then, we give our significant sample generation algorithm for a single bin (see Section 12.14.2). Next, we show how to adapt our significant sample generation algorithm for multiple bins (see Section 12.14.3). In addition, we provide some technical claims (see Section 12.14.4).

12.14.1 Energy estimation for noisy signals

In this section, we generalize our methods in Section 12.13 to estimate the (partial) energy of the true observing signals, which contains some noise.

In the following lemma, we show that the energy of the filtered signal $z(t)$ can be estimated with a few samples, assuming it contains a small fraction of noise.

Lemma 12.52. *Let $x^* \in \mathcal{F}_{k,F}$ be the ground-truth signal and $x(t) = x^*(t) + g(t)$ be the noisy observation signal. Let H be defined as in Definition 12.7, $G_{\sigma,b}^{(j)}$ be defined as in Definition 12.2 with (σ, b) such that Large Offset event does not happen. For any $j \in [B]$, suppose there exists an f_0 with $j = h_{\sigma,b}(f_0)$ satisfying:*

$$\int_{f_0-\Delta_h}^{f_0+\Delta_h} |\widehat{x^* \cdot H}(f)|^2 \mathrm{d}f \geq T\mathcal{N}^2/k,$$

where $\mathcal{N}^2 \geq \delta\|x^*\|_T^2$. Let $z^*(t) := (x^* \cdot H) * G_{\sigma,b}^{(j)}(t)$ and $z(t) = (x \cdot H) * G_{\sigma,b}^{(j)}(t)$. Let $g_z(t) := z(t) - z^*(t)$. Let $U = \{t_0 \in \mathbb{R} \mid H(t) > 1 - \delta_1 \ \forall t \in [t_0, t_0 + \beta]\}$. Suppose that $\|g_z(t)\|_T^2 \leq c\|z^*(t)\|_U^2$, where $c \in (0, 0.001)$ is a small universal constant.

For $s \geq O(k \log(k) \log(1/\rho))$, let $S_{D_U} = \{t_1, \ldots, t_s\}$ be a set of i.i.d. samples from the distribution D_U defined as Eq. (12.52). Let the weights $w_i = 1/(TsD_U(t_i))$ for $i \in [s]$.

Then, with probability at least 0.85*,*

$$\|z(t)\|_{S_{D_U},w}^2 \geq (0.2 - 20c) \cdot \|z^*(t)\|_U^2$$

Proof. We consider the expectation of $\|g_z(t)\|_{S_U,w}^2$ first.

$$\begin{aligned}
\mathbb{E}\Big[\sum_{j=1}^s w_i |g_z(t_j)|^2\Big] =&~ \sum_{j=1}^s \mathbb{E}_{t_j \sim D_U}[w_i |g_z(t_j)|^2] \\
=&~ \sum_{j=1}^s \mathbb{E}_{t_j \sim D_U}\Big[\frac{1}{TsD_U(t_i)} |g_z(t_j)|^2\Big] \\
\leq &~ \mathbb{E}_{t \sim D_U}\Big[\frac{1}{TD_U(t)} |g_z(t)|^2\Big] \\
\leq &~ \int_U \frac{1}{T} |g_z(t)|^2 \mathrm{d}t \\
\leq &~ \frac{1}{T} \int_0^T |g_z(t)|^2 \mathrm{d}t \\
\leq &~ \|g_z(t)\|_T^2
\end{aligned} \tag{12.57}$$

where the first step is straight forward, the second step follows from the definition of w_i, the third step is straight forward, the forth step follows from the definition of expectation, the fifth step follows from $U \subseteq [0, T]$, the sixth step follows from the definition of the norm.

By Eq. (12.57) and Markov inequality, we have that with probability at least 0.9,

$$\sum_{j=1}^s w_i |g_z(t_j)|^2 \leq 20 \|g_z(t)\|_T^2. \tag{12.58}$$

Then, we have that

$$\begin{aligned}
\sum_{j=1}^s w_i |z(t_j)|^2 \geq &~ 0.5 \sum_{j=1}^s w_i |z^*(t_j)|^2 - \sum_{j=1}^s w_i |g_z(t_j)|^2 \\
\geq &~ 0.5 \sum_{j=1}^s w_i |z^*(t_j)|^2 - 20 \|g_z(t)\|_T^2 \\
\geq &~ 0.2 \|z^*(t)\|_U^2 - 20 \|g_z(t)\|_T^2 \\
\geq &~ (0.2 - 20c) \cdot \|z^*(t)\|_U^2
\end{aligned}$$

where the first step follows from $(a+b)^2 \geq 0.5a^2 - b^2$, the second step follows from Eq. (12.58), the third step follows from Lemma 12.48, the forth step follows from $\|g_z(t)\|_T^2 \leq c\|z^*(t)\|_U^2$.

The total success probability follows from a union bound: $0.9 - \rho > 0.85$.

The lemma is then proved. $\square$

The following lemma shows how to estimate the energy of a noisy local-test signal.

Lemma 12.53. *Let $x^* \in \mathcal{F}_{k,F}$ be the ground-truth signal and $x(t) = x^*(t) + g(t)$ be the noisy observation signal. Let H be defined as in Definition 12.7, $G_{\sigma,b}^{(j)}$ be defined as in Definition 12.2 with (σ, b) such that Large Offset event does not happen. For any $j \in [B]$, suppose there exists an f_0 with $j = h_{\sigma,b}(f_0)$ satisfying:*

$$\int_{f_0-\Delta_h}^{f_0+\Delta_h} |\widehat{x^* \cdot H}(f)|^2 \mathrm{d}f \geq T\mathcal{N}^2/k,$$

where $\mathcal{N}^2 \geq \delta\|x^*\|_T^2$. Let $z^*(t) := (x^* \cdot H) * G_{\sigma,b}^{(j)}(t)$ and $z(t) = (x \cdot H) * G_{\sigma,b}^{(j)}(t)$. Let $g_z(t) := z(t) - z^*(t)$. Let $U = \{t_0 \in \mathbb{R} \mid H(t) > 1 - \delta_1 \ \forall t \in [t_0, t_0 + \beta]\}$. Suppose that $\|g_z(t)\|_T^2 \leq c\|z^*(t)\|_U^2$, where $c \in (0, 0.001)$ is a small universal constant.

For $s \geq O(k\log(k)\log(1/\rho))$, let $S_{D_U} = \{t_1, \ldots, t_s\}$ be a set of i.i.d. samples from the distribution D_U defined as Eq. (12.52). Let $w_i = 1/(TsD_U(t_i))$ for $i \in [s]$.

Then, with probability at least 0.85,

$$\|d_z(t)\|_{S_{D_U},w}^2 \lesssim (c + \sqrt{\gamma^2 + \delta_1}) \cdot \|z^*(t)\|_U^2,$$

Proof. We first consider the expectation of $\|g_z(t)e^{2\pi i f_0 \beta} - g_z(t+\beta)\|_{S_{D_U},w}^2$. We have that

$$\begin{aligned}
&\mathbb{E}\Big[\sum_{i=1}^{s} w_i |g_z(t_i)e^{2\pi i f_0 \beta} - g_z(t_i+\beta)|^2\Big] \\
= & \sum_{i=1}^{s} \mathbb{E}_{t_i \sim D_U}[w_i |g_z(t_i)e^{2\pi i f_0 \beta} - g_z(t_i+\beta)|^2] \\
= & \sum_{i=1}^{s} \mathbb{E}_{t_i \sim D_U}\Big[\frac{1}{TsD_U(t_i)} |g_z(t_i)e^{2\pi i f_0 \beta} - g_z(t_i+\beta)|^2\Big] \\
= & \ \mathbb{E}_{t \sim D_U}\Big[\frac{1}{TD_U(t)} |g_z(t)e^{2\pi i f_0 \beta} - g_z(t+\beta)|^2\Big] \\
= & \int_U \frac{1}{T} |g_z(t)e^{2\pi i f_0 \beta} - g_z(t+\beta)|^2 \mathrm{d}t \\
\leq & \ \frac{4}{T}\int_U (|g_z(t)|^2 + |g_z(t+\beta)|^2)\mathrm{d}t \\
\leq & \ \frac{8}{T}\int_0^T |g_z(t)|^2 \mathrm{d}t \\
\leq & \ 10\|g_z(t)\|_T^2,
\end{aligned} \tag{12.59}$$

where the first step is straight forward, the second step follows from the definition of w_i, the third step is straight forward, the forth step follows from the definition of expectation, the fifth step follows from $(a+b)^2 \leq 2a^2 + 2b^2$, the sixth step follows from $U \subseteq [0,T]$ and $U + \beta \subseteq [0,T]$, the seventh step follows from the definition of the norm.

By Eq. (12.59) and Markov inequality, we have that with probability at least 0.9,

$$\sum_{i=1}^{s} w_i |g_z(t_i)e^{2\pi i f_0 \beta} - g_z(t_i+\beta)|^2 \leq 100\|g_z(t)\|_T^2. \tag{12.60}$$

We have that

$$\begin{aligned}
&\sum_{i=1}^{s} w_i |z(t_i) e^{2\pi i f_0 \beta} - z(t_i+\beta)|^2 \\
\leq &\sum_{i=1}^{s} (2w_i |z^*(t_i) e^{2\pi i f_0 \beta} - z^*(t_i+\beta)|^2 + 2w_i |g_z(t_i) e^{2\pi i f_0 \beta} - g_z(t_i+\beta)|^2) \\
\leq &\, 200 \|g_z(t)\|_T^2 + \sum_{i=1}^{s} 2w_i |z^*(t_i) e^{2\pi i f_0 \beta} - z^*(t_i+\beta)|^2 \\
\leq &\, 200 \|g_z(t)\|_T^2 + 4\|z^*(t+\beta) - e^{2\pi \mathbf{i} f_0 \beta} \cdot z^*(t)\|_U^2 + 2\sqrt{\delta_1} \|x(t)\|_T \|z^*(t+\beta) - e^{2\pi \mathbf{i} f_0 \beta} \cdot z^*(t)\|_U \\
\lesssim &\, (200c + 4\gamma^2 + 4\delta_1) \|z^*(t)\|_U^2 + 2\sqrt{\delta_1(\gamma^2+\delta_1)} \|x^*(t)\|_T \|z^*(t)\|_U \\
\lesssim &\, (c + \gamma^2 + \delta_1) \|z^*(t)\|_U^2 + \sqrt{\delta_1(\gamma^2+\delta_1)\frac{k}{\delta}} \|z^*(t)\|_T \|z^*(t)\|_U \\
\lesssim &\, ((c + \gamma^2 + \delta_1) + \sqrt{\delta_1(\gamma^2+\delta_1)\frac{k}{\delta}\frac{R-L}{T-k^2(T+L-R)}}) \|z^*(t)\|_U^2 \\
\lesssim &\, (c + \sqrt{\gamma^2+\delta_1}) \cdot \|z^*(t)\|_U^2
\end{aligned}$$

where the first step follows from $(a+b)^2 \leq 2a^2 + 2b^2$, the second step follows from Eq. (12.60), the third step follows from the partial energy estimation for local-test signal (Lemma 12.49) which holds with probability $1-\rho$, the forth step follows from the $\|g_z(t)\|_T^2 \leq c\|z^*(t)\|_U^2$ and Claim 12.59, the fifth step follows from Lemma 12.33, the sixth step follows from $[L, R] := U$ and Lemma 12.51, the seventh step follows form $R - L \lesssim T - k^2(T+L-R)$, $\delta_1 \delta^{-1} k \lesssim 1$.

The total success probability follows from a union bound $0.9 - \rho > 0.85$.

The lemma is then proved. □

12.14.2 Significant sample generation for a single bin

Recall that we define a sample $t \in [0, T]$ is *significant* if the magnitude of the local-test signal at t is small, i.e., $|d_z(t)| \leq O(|z(t)|)$. The following lemma shows that a significant sample can be efficiently generated, provided that the filtered noisy signal does not contain too much noise.

Lemma 12.54 (Generate Significant samples for filtered noisy signals)**.** *Let $x^* \in \mathcal{F}_{k,F}$ be the ground-truth signal and $x(t) = x^*(t) + g(t)$ be the noisy observation signal. Let H be defined as in Definition 12.7, $G_{\sigma,b}^{(j)}$ be defined as in Definition 12.2 with (σ, b) such that Large Offset event does not happen. For any $j \in [B]$, suppose there exists an f_0 with $j = h_{\sigma,b}(f_0)$ satisfying:*

$$\int_{f_0-\Delta_h}^{f_0+\Delta_h} |\widehat{x^* \cdot H}(f)|^2 \mathrm{d}f \geq T\mathcal{N}^2/k,$$

where $\mathcal{N}^2 \geq \delta \|x^*\|_T^2$. Let $z^*(t) := (x^* \cdot H) * G_{\sigma,b}^{(j)}(t)$ and $z(t) = (x \cdot H) * G_{\sigma,b}^{(j)}(t)$. Let $g_z(t) := z(t) - z^*(t)$. Let $U = \{t_0 \in \mathbb{R} \mid H(t) > 1 - \delta_1 \ \forall t \in [t_0, t_0 + \beta]\}$. Suppose that $\|g_z(t)\|_T^2 \leq c\|z^*(t)\|_U^2$, where $c \in (0, 0.001)$ is a small universal constant.

Then, there is an algorithm that takes $O(k \log(k))$ samples in z, runs in $O(k \log(k))$ time, and output an $\alpha \in U$ such that with probability at least 0.6,

$$|z(\alpha + \beta) - z(\alpha)e^{2\pi \mathbf{i} f_0 \beta}|^2 \leq O(c + \sqrt{\gamma^2 + \delta_1})|z(\alpha)|^2 \leq 0.01|z(\alpha)|^2.$$

Proof. The output α is sample in two steps:

1. For $s \geq O(k \log(k))$, generate s i.i.d. samples $S_{D_U} = \{t_1, \ldots, t_s\}$ bfrom the distribution D_U defined as Eq. (12.52). Let $w_i = 1/(TsD_U(t_i))$ for $i \in [s]$ be the weights.

2. Define a probability distribution D_S such that

$$D_S(t_i) := \frac{w_i |z(t_i)|^2}{\sum_{i \in [s]} w_i |z(t_i)|^2} \quad \forall i \in [s]. \tag{12.61}$$

   And sample α according to D_S.

The sample and time complexities of this procedure are straightforward. It remains to prove that α satisfies the significance requirement stated in the lemma.

By Lemma 12.52, we have that with probability at least 0.85,

$$\sum_{j=1}^{s} w_i |z(t_j)|^2 \geq (0.2 - 20c) \cdot \|z^*(t)\|_U^2 \tag{12.62}$$

By Lemma 12.53, we have that with probability at least 0.85,

$$\sum_{i=1}^{s} w_i |z(t_i) e^{2\pi i f_0 \beta} - z(t_i + \beta)|^2 \lesssim (c + \sqrt{\gamma^2 + \delta_1}) \cdot \|z^*(t)\|_U^2 \tag{12.63}$$

Thus, with probability at least 0.7,

$$\begin{aligned} & \frac{\sum_{i=1}^{s} w_i |z(t_i) e^{2\pi i f_0 \beta} - z(t_i + \beta)|^2}{\sum_{j=1}^{s} w_i |z(t_j)|^2} \\ \leq & \frac{O(c + \sqrt{\gamma^2 + \delta_1}) \cdot \|z^*(t)\|_U^2}{\sum_{j=1}^{s} w_i |z(t_j)|^2} \\ \leq & \frac{O(c + \sqrt{\gamma^2 + \delta_1}) \cdot \|z^*(t)\|_U^2}{(0.2 - 20c) \cdot \|z^*(t)\|_U^2} \\ = & O(c + \sqrt{\gamma^2 + \delta_1}) \end{aligned} \tag{12.64}$$

where the first step follows from Eq. (12.63), the second step follows from Eq. (12.62), the third step is straight forward.

For a random sample $\alpha \sim D_S$, we bound the following expectation:

$$\begin{aligned} & \mathbb{E}_{\alpha \sim D_S} \left[\frac{|z(\alpha) e^{2\pi i f_0 \beta} - z(\alpha + \beta)|^2}{|z(\alpha)|^2} \right] \\ = & \sum_{i=1}^{s} \frac{w_i |z(t_i)|^2}{\sum_{j=1}^{s} w_j |z(t_j)|^2} \cdot \frac{|z(t_i) e^{2\pi i f_0 \beta} - z(t_i + \beta)|^2}{|z(t_i)|^2} \\ = & \frac{\sum_{i=1}^{s} w_i |z(t_i) e^{2\pi i f_0 \beta} - z(t_i + \beta)|^2}{\sum_{j=1}^{s} w_j |z(t_j)|^2} \\ \leq & O(c + \sqrt{\gamma^2 + \delta_1}), \end{aligned}$$

where the first step follows from the definition of D_m, the second step is straightforward, the third step follows from Eq. (12.64).

Thus by Markov inequality, with probability 0.9,

$$\frac{|z(\alpha) e^{2\pi i f_0 \beta} - z(\alpha + \beta)|^2}{|z(\alpha)|^2} \leq \frac{O(c + \sqrt{\gamma^2 + \delta_1})}{0.1} = O(c + \sqrt{\gamma^2 + \delta_1}).$$

The success probability follows from a union bound. And the second inequality follows from the range of the parameters c, γ, δ_1. □

12.14.3 Significant sample generation for multiple bins

In this section, we present our significant sample generation procedure that simultaneously works for all "good bins".

Algorithm 65 Generate Significant Samples

```
procedure GenerateSignificantSamples($z$)
    $B \leftarrow O(k)$
    $U \leftarrow \{t_0 \in \mathbb{R} | H(t) > 1-\delta_1, \forall t \in [t_0, t_0+\beta]\}$
    $D_z \leftarrow \begin{cases} c \cdot (1-|t/T|)^{-1} T^{-1}, & \text{for } |t| \leq T(1-1/k) \wedge t \in U \\ c \cdot k T^{-1}, & \text{for } |t| \in [T(1-1/k), T] \wedge t \in U \end{cases}$
    $S \leftarrow O(k \log(k))$ i.i.d. samples from $D_z$
    for $t_i \in S$ do
        for $j \in [B]$ do
            $a \leftarrow t_i/\sigma$
            $u_j \leftarrow \sum_{i \in \mathbb{Z}} x \cdot H(\sigma(a-j-iB)) e^{-2\pi \mathbf{i} \sigma b(j+iB)} G(j+iB)$    ▷ $u \in \mathbb{R}^B$
            $u_j^\beta \leftarrow \sum_{i \in \mathbb{Z}} x \cdot H(\sigma(a+\beta-j-iB)) e^{-2\pi \mathbf{i} \sigma b(j+iB)} G(j+iB)$    ▷ $u^\beta \in \mathbb{R}^B$
        end for
        $\widehat{u} = \mathrm{FFT}(u)$
        $\widehat{u^\beta} = \mathrm{FFT}(u^\beta)$
        for $j \in [B]$ do
            $z_j(t_i) \leftarrow \widehat{u}_j$
            $z_j(t_i+\beta) \leftarrow \widehat{u}_j^\beta$
        end for
    end for
    $w_i \leftarrow D_z(t_i), \forall t_i \in S$
    $W \leftarrow \sum_{t_i \in S} w_i |z_j(t_i)|^2$
    $Z_{j,1} \leftarrow 0$, $Z_{j,2} \leftarrow 0$    ▷ $Z \in \mathbb{C}^{B \times 2}$
    for $j \in [B]$ do
        $D_S(t_i) \leftarrow w_i |z_j(t_i)|^2 / W, \forall t_i \in S$
        Sample $t_i \sim D_S$    ▷ $\alpha \in \mathbb{R}^B$
        $Z_{j,1} \leftarrow z_j(t_i)$, $Z_{j,2} \leftarrow z_j(t_i+\beta)$
    end for
    return $Z$
end procedure
```

We first prove the correctness of Algorithm 65.

Lemma 12.55 (Generate significant samples for different bins simultaneously). *Let*

$x^* \in \mathcal{F}_{k,F}$ be the ground-truth signal and $x(t) = x^*(t) + g(t)$ be the noisy observation signal. Let H be defined as in Definition 12.7, $G_{\sigma,b}^{(j)}$ be defined as in Definition 12.2 with (σ, b) such that Large Offset event does not happen. Let $U = \{t_0 \in \mathbb{R} \mid H(t) > 1 - \delta_1 \ \forall t \in [t_0, t_0 + \beta]\}$.

For $j \in [B]$, let $z_j^*(t) := (x^* \cdot H) * G_{\sigma,b}^{(j)}(t)$ and $z_j(t) = (x \cdot H) * G_{\sigma,b}^{(j)}(t)$. Let $g_j(t) := z_j(t) - z_j^*(t)$. Let

$$S_{g1} := \left\{ j \in [B] \mid \|g_j(t)\|_T^2 \leq c \|z_j^*(t)\|_U^2 \right\},$$

where $c \in (0, 0.001)$ is a small universal constant. Let

$$S_{g2} := \left\{ j \in [B] \ \middle| \ \exists f_0, h_{\sigma,b}(f_0) = j \text{ and } \int_{f_0 - \Delta_h}^{f_0 + \Delta_h} |\widehat{x^* \cdot H}(f)|^2 \mathrm{d}f \geq T\mathcal{N}^2 / k \right\},$$

where $\mathcal{N}^2 \geq \delta \|x^*\|_T^2$. Let $S_g = S_{g1} \cap S_{g2}$.

There is a Procedure GenerateSignificantSamples *(Algorithm 65) that takes $O(k^2 \log^2(k/\delta_1))$ samples in x, runs in $O(k^2 \log^3(k/\delta_1))$ time, and for each $j \in S_g$, output α_j such that with probability at least 0.6,*

$$|z_j(\alpha_j + \beta) - z_j(\alpha_j) e^{2\pi \mathbf{i} f_0 \beta}|^2 \leq 0.01 |z_j(\alpha_j)|^2 \quad \forall j \in S_g.$$

Proof. For $k \in \mathbb{N}_+$, define a probability distribution $D(t)$ as follows:

$$D(t) := \begin{cases} c \cdot (1 - |t/T|)^{-1} T^{-1}, & \text{for } |t| \leq T(1 - 1/k) \wedge t \in U \\ c \cdot k T^{-1}, & \text{for } |t| \in [T(1 - 1/k), T] \wedge t \in U \end{cases}$$

where $c = \Theta(\log(k)^{-1})$ is a normalization factor such that $\int_{-T}^{T} D(t) \mathrm{d}t = 1$. For any $\varepsilon, \rho \in (0, 1)$, let $S_D = \{t_1, \cdots, t_s\}$ be a set of i.i.d. samples from $D(t)$ of size $s \geq O(k \log(k) \log(1/\rho))$. Let the weight vector $w \in \mathbb{R}^s$ be defined by $w_i := 1/(T s D(t_i))$ for $i \in [s]$.

Suppose all the bins can access the same set of time points S_D. Then, by Lemma 12.54, for any $j \in S_g$ with probability 0.6, we have that,

$$|z_j(\alpha + \beta) - z_j(\alpha) e^{2\pi \mathbf{i} f_0 \beta}|^2 \leq 0.01 |z_j(\alpha)|^2.$$

Then, we show that the value of $z_j(t), j = 1, \cdots, B$ of same set of time points S_D can be compute by accessing a same set of time points in $x(t)$. By Lemma 12.12 with setting $a = \alpha/\sigma$, we have that

$$z_j(\alpha) = \widehat{u}_j,$$

which is computed by the algorithm.

As a result, for each $j \in S_g$,

$$|z_j(\alpha_j + \beta) - z_j(\alpha_j)e^{2\pi \mathbf{i} f_0 \beta}|^2 \leq 0.01|z_j(\alpha_j)|^2 \quad \forall j \in S_g,$$

holds with probably 0.6.

□

We compute the time and sample complexities of Algorithm 65 in the following two lemmas.

Lemma 12.56 (Running time of Procedure GenerateSignificantSamples in Algorithm 65)**.** *Procedure* GenerateSignificantSamples *in Algorithm 65 runs in $O(k^2 \log(k) \log(k/\delta_1))$ times.*

Proof. In each call of Procedure GenerateSignificantSamples in Algorithm 65,

- In line 5, taking $|S|$ samples runs $O(|S|)$ times.
- In line 6, the for loop repeats $|S|$ times,
  - In line 7, the for loops repeats B times and j iterate from 1 to B, in each loop line 9 and line 10, computing the summation runs in $|\{j + iB | i \in \mathbb{Z} \wedge j + iB \in \mathrm{supp}(G)\}|$ times.
  - In line 12 and 13, running Fast Fourier Transform algorithm takes $O(B \log(B))$ time, where B is the length of the vector u and u^β.
  - In line 14, the for loop repeats B times, each loop runs in $O(1)$ times.

- In line 19, assigning w_i runs in $|S|$ times.
- In line 20, computing $\sum_{t_i \in S} w_i |z_j(t_i)|^2$ runs in $|S|$ times.
- In line 22, the for loop repeats B times, each loop runs in $O(1)$ times.

Notice that

$$\sum_{j \in [B]} |\{j + iB | i \in \mathbb{Z} \wedge j + iB \in \mathrm{supp}(G)\}| \leq |\mathrm{supp}(G)|.$$

In the algorithm, we set the parameters:

$$B = \ O(k), \quad \text{and} \quad |S| = \ k \log(k). \tag{12.65}$$

Thus,

$$|\mathrm{supp}(G)| = O(lB/\alpha) = k \log(k/\delta_1), \tag{12.66}$$

where the first step follows from Lemma 12.9 Property IV, the second step follows from $\alpha \precapprox 1$ and $l = \Theta(\log(k/\delta_1))$.

Therefore, the time complexity in total is

$$\begin{aligned}
& O(O(|S|) + |S| \cdot (|\mathrm{supp}(G)| + O(B \log(B)) + B \cdot O(1)) + |S| + |S| + B \cdot O(1)) \\
\leq\ & O(|S| \cdot (|\mathrm{supp}(G)| + B \log(B))) \\
\leq\ & O(k \log(k) \cdot (|\mathrm{supp}(G)| + k \log(k))) \\
\leq\ & O(k \log(k) \cdot (k \log(k/\delta_1) + k \log(k))) \\
\leq\ & O(k^2 \log(k) \log(k/\delta_1)),
\end{aligned}$$

where the first step is straightforward, the second step follows from Eq. (12.65), the third step follows from Eq. (12.66), the forth step is straight forward. □

Lemma 12.57 (Sample complexity of Procedure GenerateSignificantSamples in Algorithm 65)**.** *Procedure* GenerateSignificantSamples *in Algorithm 65 takes* $O(k^2 \log(k) \log(k/\delta_1))$ *samples.*

Proof. In each call of Procedure GenerateSignificantSamples in Algorithm 65,

- In line 6, the for loop repeats $|S|$ times,
- In line 7, the for loops repeats B times and j iterate from 1 to B, in each loop line 9 and line 10, computing the summation takes $O(|\{\sigma(a-j-iB)|i \in \mathbb{Z} \wedge j+iB \in \mathrm{supp}(G)\}|)$ samples.

Following from the setting in the algorithm, we have that

$$|S| = k\log(k). \tag{12.67}$$

Thus,

$$|\mathrm{supp}(G)| = O(lB/\alpha) = k\log(k/\delta_1). \tag{12.68}$$

where the first step follows from Lemma 12.9 Property IV, the second step follows from $\alpha \eqsim 1$ and $l = \Theta(\log(k/\delta_1))$.

So, the samples complexity of Procedure GenerateSignificantSamples in Algorithm 65 is

$$\begin{aligned}
&|S| \cdot \sum_{j\in[B]} O(|\{\sigma(a-j-iB) \mid i \in \mathbb{Z} \wedge j+iB \in \mathrm{supp}(G)\}|) \\
\le\ & O(|S| \cdot |\mathrm{supp}(G)|) \\
\le\ & O(|S| \cdot k\log(k/\delta_1)) \\
\le\ & O(k\log(k) \cdot k\log(k/\delta_1)) \\
=\ & O(k^2\log(k)\log(k/\delta_1))
\end{aligned}$$

where the first step is straight forward, the second step follows from Eq. (12.68), the third step follows from Eq. (12.67), the forth step is straight forward. □

12.14.4 Technical claims

We prove two technical claims in below about the local-test signals' energy reduction.

Claim 12.58 (Energy decay of local-test signals)**.** *Let $x^* \in \mathcal{F}_{k,F}$. For any (σ, b) such that Large Offset event does not happen and any $j \in [B]$, suppose there exists an f_0 with $j = h_{\sigma,b}(f_0)$ satisfying: well-isolation conditions and*

$$\int_{f_0-\Delta_h}^{f_0+\Delta_h} |\widehat{x^* \cdot H}(f)|^2 \mathrm{d}f \geq T\mathcal{N}^2/k,$$

where $\mathcal{N}^2 \geq \delta\|x^*\|_T^2$. Let $z(t) = (x^* \cdot H) * G_{\sigma,b}^{(j)}(t)$ be the filtered signal.

For $\beta \leq \gamma/\Delta_0$ with $\Delta_0 = O(\Delta)$, let $d_z(t) = z(t)e^{2\pi\mathbf{i}f_0\beta} - z(t+\beta)$ be the local-test signal. We have that

$$\|d_z(t)\|_T^2 \lesssim (\gamma^2 + \delta_1) \cdot \|z(t)\|_T^2.$$

Proof. Let $S := \mathrm{supp}(\widehat{x}^* * \widehat{H})$ be the support set of $\widehat{x}^* * \widehat{H}$. Let

$$V := \{f \in \mathbb{R} \mid \widehat{G}_{\sigma,b}^{(j)}(f) \geq 1 - \delta_1\}.$$

Note that $\|d_z(t)\|_T^2$ can be expressed as follows (ignoring the $\frac{1}{T}$ factor):

$$\begin{aligned}
& \int_0^T |z(t+\beta) - e^{2\pi\mathbf{i}f_0\beta} \cdot z(t)|^2 \mathrm{d}t \\
\leq & \int_{-\infty}^{\infty} |z(t+\beta) - e^{2\pi\mathbf{i}f_0\beta} \cdot z(t)|^2 \mathrm{d}t \\
= & \int_{-\infty}^{\infty} |\widehat{z}(f)e^{2\pi\mathbf{i}f\beta} - e^{2\pi\mathbf{i}f_0\beta} \cdot \widehat{z}(f)|^2 \mathrm{d}f \\
= & \int_{-\infty}^{\infty} |\widehat{z}(f)|^2 \cdot |e^{2\pi\mathbf{i}f\beta} - e^{2\pi\mathbf{i}f_0\beta}|^2 \mathrm{d}f \\
= & \int_{S} |\widehat{z}(f)|^2 \cdot |e^{2\pi\mathbf{i}f\beta} - e^{2\pi\mathbf{i}f_0\beta}|^2 \mathrm{d}f \\
= & \int_{S\cap V} |\widehat{z}(f)|^2 \cdot |e^{2\pi\mathbf{i}f\beta} - e^{2\pi\mathbf{i}f_0\beta}|^2 \mathrm{d}f + \int_{S\setminus V} |\widehat{z}(f)|^2 \cdot |e^{2\pi\mathbf{i}f\beta} - e^{2\pi\mathbf{i}f_0\beta}|^2 \mathrm{d}f,
\end{aligned}$$

where the first step is straight forward, the second step follows from Parseval's theorem, the third step is straight forward, the forth step follows from the assumption that Large Offset event does not happen, the fifth step is straight forward.

Then, for the first term, we have that

$$\begin{aligned}\int_{S\cap V} |\widehat{z}(f)|^2 \cdot |e^{2\pi \mathbf{i} f\beta} - e^{2\pi\mathbf{i} f_0\beta}|^2 \mathrm{d}f \leq & \int_{f_0-\Delta_0}^{f_0+\Delta_0} |\widehat{z}(f)|^2 \cdot |e^{2\pi \mathbf{i} f\beta} - e^{2\pi\mathbf{i} f_0\beta}|^2 \mathrm{d}f \\ \lesssim & \int_{f_0-\Delta_0}^{f_0+\Delta_0} |\widehat{z}(f)|^2 \cdot \gamma^2 \mathrm{d}f \\ \leq & \gamma^2 \cdot \int_{-\infty}^{\infty} |\widehat{z}(f)|^2 \mathrm{d}f \\ \leq & \gamma^2 \cdot \int_{-\infty}^{\infty} |z(t)|^2 \mathrm{d}t \\ \leq & \gamma^2 \cdot \int_{0}^{T} |z(t)|^2 \mathrm{d}t \end{aligned} \quad (12.69)$$

where the first step follows from $S\cap V \subset [f_0-\Delta_0, f_0+\Delta_0]$ by Claim 12.13, the second step follows from

$$|e^{2\pi\mathbf{i} f\beta} - e^{2\pi\mathbf{i} f_0\beta}| \leq 4\pi\beta|f-f_0| \leq 4\pi\beta\Delta_{h0} \lesssim \gamma,$$

the third step is straight forward, the forth step follows from Parseval's theorem, the fifth step follows from Lemma 12.30.

For the second term, we have that

$$\begin{aligned}\int_{S\setminus V} |\widehat{z}(f)|^2 \cdot |e^{2\pi \mathbf{i} f\beta} - e^{2\pi\mathbf{i} f_0\beta}|^2 \mathrm{d}f = & \int_{S\setminus V} |(\widehat{x}^* * \widehat{H})(f)|^2 \cdot |\widehat{G}_{\sigma,b}^{(j)}(f)|^2 \cdot |e^{2\pi \mathbf{i} f\beta} - e^{2\pi\mathbf{i} f_0\beta}|^2 \mathrm{d}f \\ \leq & \int_{S\setminus V} |(\widehat{x}^* * \widehat{H})(f)|^2 \cdot \delta_1^2 \cdot |e^{2\pi \mathbf{i} f\beta} - e^{2\pi\mathbf{i} f_0\beta}|^2 \mathrm{d}f \\ \lesssim & \int_{S\setminus V} |(\widehat{x}^* * \widehat{H})(f)|^2 \cdot \delta_1^2 \mathrm{d}f \\ \leq & \int_{-\infty}^{\infty} |(\widehat{x}^* * \widehat{H})(f)|^2 \cdot \delta_1^2 \mathrm{d}f \\ = & \delta_1^2 \cdot \int_{-\infty}^{\infty} |(x^* \cdot H)(t)|^2 \mathrm{d}t \\ \lesssim & \delta_1^2 \cdot \int_{0}^{T} |x^*(t)|^2 \mathrm{d}t \end{aligned} \quad (12.70)$$

where the first step follows from the definition of z, the second step follows from the definition of V, the third step follows from $|e^{2\pi\mathbf{i}f\beta} - e^{2\pi\mathbf{i}f_0\beta}|^2 \lesssim 1$, the forth step is straight forward, the fifth step follows from Parseval's theorem, the sixth step follows from Lemma 12.24 Property IV and V.

Putting them together, we get that

$$\begin{aligned}\int_0^T |z(t+\beta) - e^{2\pi\mathbf{i}f_0\beta} \cdot z(t)|^2 \mathrm{d}t &\lesssim \gamma^2 \cdot \int_0^T |z(t)|^2 \mathrm{d}t + \delta_1^2 \cdot \int_0^T |x(t)|^2 \mathrm{d}t \\ &\lesssim (\gamma^2 + \delta_1^2 \delta^{-1} k) \cdot \int_0^T |z(t)|^2 \mathrm{d}t \\ &\lesssim (\gamma^2 + \delta_1) \cdot \int_0^T |z(t)|^2 \mathrm{d}t,\end{aligned}$$

where the first step follows from Eq. (12.69) and Eq. (12.70), the second step follows from Lemma 12.33, the last step follows from $\delta_1 \delta^{-1} k \lesssim 1$.

The proof of the lemma is then completed. $\square$

Similar result also holds for the partial energy:

Claim 12.59 (Partial energy decay of local-test signals)**.** *Let $x^*(t)$ be a k-Fourier-sparse signal. Let $z(t) := (x^* \cdot H) * G_{\sigma,b}^{(j)}(t)$ and $d_z(t) = z(t+\beta) - e^{2\pi\mathbf{i}f_0\beta} \cdot z(t)$. Let $U = \{t_0 \in \mathbb{R} \mid H(t) > 1 - \delta_1 \ \forall t \in [t_0, t_0 + \beta]\} =: [L, R]$. For $\beta \leq \gamma/\Delta_0$ with $\Delta_0 = O(\Delta)$, we have that*

$$\int_L^R |d_z(t)|^2 \mathrm{d}t \lesssim (\gamma^2 + \delta_1) \cdot \int_L^R |z(t)|^2 \mathrm{d}t.$$

Proof. We have that

$$\begin{aligned}\int_L^R |z(t+\beta) - e^{2\pi\mathbf{i}f_0\beta} \cdot z(t)|^2 \mathrm{d}t &\leq \int_0^T |z(t+\beta) - e^{2\pi\mathbf{i}f_0\beta} \cdot z(t)|^2 \mathrm{d}t \\ &\leq (\gamma^2 + \delta_1) \int_0^T |z(t)|^2 \mathrm{d}t \\ &\leq (\gamma^2 + \delta_1) \frac{T}{T - k^2(T + L - R)} \int_L^R |z(t)|^2 \mathrm{d}t \\ &\lesssim (\gamma^2 + \delta_1) \int_L^R |z(t)|^2 \mathrm{d}t,\end{aligned}$$

where the first step is straight forward, the second step follows from Claim 12.58, the third follows from Lemma 12.51 with time duration changed from $[-T, T]$ to $[0, T]$, the forth step follows from $T - k^2(T + L - R) \gtrsim T$. $\square$

12.15 Frequency Estimation

We introduce our improved frequency estimation algorithm in this section. We first show that given significant samples, we are able to estimate a specific target frequency (see Section 12.15.1). Then, we show to generalize it to simultaneously estimate frequencies for multiple bins and give our main frequency estimation algorithm (see Section 12.15.2). Next, we prove several technical claims on the votes distribution in the ArySearch procedure (see Section 12.15.3).

12.15.1 Frequency estimation via significant samples

In this section, we show an algorithm such that for a target frequency f_0, it can use several significant samples to estimate it with high accuracy. The main idea is as follows: for a significant sample α, since $|z(\alpha + \beta) - z(\alpha)e^{2\pi \mathbf{i} f_0 \beta}|$ is very small, the angle of $\frac{z(\alpha+\beta)}{z(\alpha)}$ will be close to $2\pi f_0 \beta$. That is,

$$\arg\left(\frac{z(\alpha + \beta)}{z(\alpha)}\right) \approxeq 2\pi f_0 \beta \pmod{2\pi}.$$

Solving the congruence equation gives that

$$f_0 \approx \frac{1}{2\pi\beta}\left(\arg\left(\frac{z(\alpha + \beta)}{z(\alpha)}\right) + 2\pi s\right)$$

for some unknown $s \in \mathbb{Z}$.

To find the unknown s, we use the same strategy as in [PS15]: perform a D-round searching procedure to narrow the possible range of f_0. More specifically, at the beginning, the possible range of f_0 (frequency interval) is $[-F, F]$. And after D rounds, f_0 is located in a frequency interval of length $O(\Delta)$, resulting in an estimate with Δ accuracy.

For $d \in [D]$, consider the d-th round of searching, where the frequency interval is:

$$[\mathsf{left}_d, \mathsf{left}_d + \mathsf{len}_d].$$

We equally partition the frequency interval into num parts and do a num-ary search. We generate R significant samples, and for each sample α, we enumerate all possible s and compute

$$\frac{1}{2\pi\beta}\Big(\arg\Big(\frac{z(\alpha+\beta)}{z(\alpha)}\Big) + 2\pi s\Big).$$

Then, we find which part this quantity falls in and add a vote to that part. For robustness, we also add votes to that part's left and right neighbors. In the end, the frequency interval for the next round is the part with more than $R/2$ votes. It is easy to see that at most 5 parts can be selected in the new frequency interval. Hence, we have

$$\mathsf{len}_{d+1} \leq \frac{\mathsf{len}_d}{\mathsf{num}/5}, \tag{12.71}$$

i.e., the length of the possible range of f_0 decays at a constant rate.

More formally, we have the following lemma:

Lemma 12.60 (significant sample to frequency estimation)**.** *Suppose that there is an algorithm* GetSignificantSample *that*

- *takes $z(t), \beta$ as input where $\beta \leq O(1/\Delta)$,*
- *takes $\mathcal{S}$ samples in $z(t)$,*
- *runs in $\mathcal{T}$ time,*
- *outputs an α such that with probability* 0.9*,*

$$|z(\alpha+\beta) - z(\alpha)e^{2\pi \mathbf{i} f_0 \beta}|^2 \leq 0.0001|z(\alpha)|^2.$$

Then, there is an Procedure FrequencyEstimationZ *in Algorithm 66 that*

- *takes* $O(\log(FT)\log(\log(FT))\log(1/\rho_1)\mathcal{S})$ *samples,*
- *runs in* $O(\log(FT)\log(\log(FT))\log(1/\rho_1)\mathcal{T})$ *times,*
- *and outputs* $\widetilde{f}_0$ *such that with probability at least* $1-\rho_1$,

$$|\widetilde{f}_0 - f_0| \lesssim \Delta.$$

Proof. We prove the correctness, time/sample complexity of Algorithm 66 in below.

Correctness: We first compute the value of D, the number of rounds needed for the searching procedure. Note that the GetSignificantSample procedure requires that $\beta \leq O(1/\Delta)$. In our algorithm, we take $\beta_d = O(\mathsf{num}/\mathsf{len}_d)$ for the d-th round. Hence, for the last round $d = D$, we have that,

$$O(\mathsf{num}/\mathsf{len}_D) = O(1/\Delta) \implies \mathsf{len}_D \geq \mathsf{num}\Delta.$$

Then, by Eq. (12.71) and $\mathsf{len}_1 = F$, we get that

$$D = \log_{\mathsf{num}}\left(\frac{FT}{\mathsf{num}(T\Delta)}\right) \lesssim \log(FT)/\log(\mathsf{num}). \tag{12.72}$$

Then, we calculate the success probability. For $d \in [D]$, by Claim 12.67, with probability at least $1 - O(c+\rho)^{R/6}$, the true part containing f_0 and its left and right neighbor will get R votes in total, and the other far away parts will get at most $R/2$ votes. In this case, the new frequency interval will contain the true part, and we consider this round being success. Since the search procedure takes D rounds, by a union bound, all rounds will succeed with probability at least

$$1 - D \cdot O(c+\rho)^{R/6} \geq 1 - \frac{\log(FT)}{\log(\mathsf{num})} \cdot O(c+\rho)^{R/6} \geq 1 - \rho_1$$

where the first step follows from Eq. (12.72), the second step follows from

$$R \geq O\Big(\frac{\log(\log(FT)/\rho_1)}{\log(1/(c+\rho))}\Big) \geq O\Big(\frac{\log(\log(FT)/(\rho_1\log(\mathsf{num})))}{\log(1/(c+\rho))}\Big).$$

Therefore, with probability at least $1-\rho$, the final frequency interval of length $O(\Delta)$ will contain the target frequency f_0, which means that the output $\widetilde{f}_0$ satisfies $|\widetilde{f}_0 - f_0| \lesssim \Delta$. And the correctness of the algorithm is proved.

Time complexity: We show that Procedure FrequencyEstimationZ in Algorithm 66 runs in $O(\log(FT) \cdot \log(\log(FT)/\rho_1))$ times.

In each call of the Procedure FrequencyEstimationZ in Algorithm 66,

- The for-loop repeats D times.
- In each loop, line 6 call Procedure ArySearch.

Then, in the d-th call of the Procedure ArySearch,

- In line 12, the for-loop repeats R times.
- In line 13, the Procedure GetSignificantSample is called.
- In line 14, the for-loop repeats $\beta_d \mathsf{len}_d + O(1)$ times.
- In line 16, the for-loop repeats num times.

Thus, the total time complexity is dominated by:

$$D \cdot R \cdot (\beta_d \mathsf{len}_d + O(1)) \cdot \mathsf{num} + D \cdot R \cdot \mathcal{T}.$$

By the parameter settings in Algorithm 66, we have that

$$\begin{aligned}
\mathsf{num} &= O(1), \\
D &= O(\log(\frac{FT}{\Delta})/\log(\mathsf{num})), \\
R &= O(\log(\frac{\log(FT)}{\rho_1 \log(\mathsf{num})})), \\
\beta_d &= O(\frac{\mathsf{num}}{\mathsf{len}_d}),
\end{aligned}$$

In particular, we have

$$D = O(\log(\frac{FT}{\Delta})/\log(\mathsf{num})) \leq O(\log(\frac{FT}{\Delta})) \leq O(\log(FT)),$$

where the first step follows from the setting of D, the second step follows from $\mathsf{num} = O(1)$, the third step follows from $\Delta = \mathrm{poly}(k)$.

Hence, the total time complexity of Algorithm 66 is

$$\begin{aligned} & O(D \cdot R \cdot (\beta_d \mathsf{len}_d + O(1)) \cdot \mathsf{num}) + D \cdot R \cdot \mathcal{T} \\ = & O(D \cdot R \cdot (O(\mathsf{num}) + O(1)) \cdot \mathsf{num}) + D \cdot R \cdot \mathcal{T} \\ = & O(D \cdot R \cdot \mathcal{T}) \\ = & O(\log(FT) \cdot \log(\log(FT)/\rho_1) \cdot \mathcal{T}), \end{aligned}$$

where the first step follows from $\beta_d = O(\frac{\mathsf{num}}{\mathsf{len}_d})$, the second step follows from $\mathsf{num} = O(1)$, the third step follows from the choices of D and R.

Sample complexity: Each call of the Procedure GETSIGNIFICANTSAMPLE takes $\mathcal{S}$ samples, and it is called DR times. Thus, the total sample complexity of Algorithm 66 is

$$DR \cdot \mathcal{S} = O(\log(FT) \cdot \log(\log(FT)/\rho_1) \cdot \mathcal{S}).$$

The proof of the lemma is completed.

$\square$

12.15.2 Simultaneously estimate frequencies for different bins

Combining the significant sample generation procedure discussed in Section 12.14 with Algorithm 66, we obtain the frequency estimation algorithm that improves the algorithms in [PS15] and [CKPS16].

Algorithm 66 Frequency Estimation of the Filtered Signal

```
1: procedure FrequencyEstimationZ($x, H, G_{\sigma,b}^{(j)}$)
2:     $\mathsf{num} \leftarrow O(1)$, $D \leftarrow O(\log(\frac{FT}{\Delta})/\log(\mathsf{num}))$, $R \leftarrow O(\log(\frac{\log(FT)}{\rho_1 \log(\mathsf{num})}))$
3:     $\mathsf{left}_1 \leftarrow -F$, $\mathsf{len}_1 \leftarrow 2F$
4:     for $d \in [D]$ do
5:         $\mathsf{len}_d \leftarrow 5\frac{\mathsf{len}_{d-1}}{\mathsf{num}}$
6:         $\mathsf{left}_{d+1} \leftarrow$ ArySearch($x, H, G_{\sigma,b}^{(j)}, F, T, \Delta, \mathsf{left}_d, \mathsf{len}_d, \mathsf{num}$)
7:     end for
8:     return $\mathsf{left}_D$
9: end procedure
10: procedure ArySearch($x, H, G_{\sigma,b}^{(j)}, F, T, \Delta, \mathsf{left}_i, \mathsf{len}_i, \mathsf{num}$)
11:     Let $v \in \mathbb{Z}_+^{\mathsf{num}}$ and $v_q \leftarrow 0$ for $q \in [\mathsf{num}]$
12:     for $r = 1 \to R$ do
13:         $z(\alpha+\beta), z(\alpha) \leftarrow$ GetSignificantSample($x, H, G_{\sigma,b}^{(j)}, r, d$)
14:         for $s \in [\beta\mathsf{left}_d - 10, \beta(\mathsf{left}_d + \mathsf{len}_d) + 10] \cap \mathbb{Z}$ do
15:             $\widetilde{f} = \frac{1}{2\pi\beta}(\arg(\frac{z(\alpha+\beta)}{z(\alpha)}) + 2\pi s)$
16:             for $q \in [\mathsf{num}]$ do
17:                 if $\widetilde{f} \in [\mathsf{left}_d + (q-1)\mathsf{len}_d/\mathsf{num}, \mathsf{left}_d + q\mathsf{len}_d/\mathsf{num}]$ then
18:                     $v_q \leftarrow v_q + 1$
19:                 end if
20:             end for
21:         end for
22:     end for
23:     for $q \in [\mathsf{num}]$ do
24:         if $v_q + v_{q+1} + v_{q+2} \geq R/2$ then
25:             $\mathsf{left}_{d+1} \leftarrow \mathsf{left}_d + (q-1)\mathsf{len}_d/\mathsf{num}$
26:             return $\mathsf{left}_{d+1}$
27:         end if
28:     end for
29:     return $\emptyset$
30: end procedure
```

Theorem 12.61 (Better frequency estimation algorithm). *Let $x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$ and $x(t) = x^*(t) + g(t)$ be the observation signal where $g(t)$ is arbitrary noise. Let $\Delta_h := O(|\mathrm{supp}(\widehat{H})|)$, $\Delta := O(k \cdot \Delta_h)$ and $\mathcal{N}^2 := \|g(t)\|_T^2 + \delta\|x^*(t)\|_T^2$. Let H be defined as in Definition 12.7, $G_{\sigma,b}^{(j)}$ be defined as in Definition 12.2 with (σ, b) such that Large*

Algorithm 67 Pre-computation of the Significant Samples

```
1: procedure SAMPLINGSIGNIFICANTSAMPLE($x, H, G_{\sigma,b}^{(j)}, F, T, \Delta, \mathsf{num}, D, R$)
2:     $\mathsf{len}_1 \leftarrow 2F$, $\mathcal{L} \in \mathbb{C}^{D\times R\times B\times 2}$
3:     for $d \in [D]$ do
4:         $\mathsf{len}_d = 5\frac{\mathsf{len}_{d-1}}{\mathsf{num}}$
5:         $\widehat{\beta} \leftarrow O(\frac{\mathsf{num}}{\mathsf{len}_d})$
6:         for $r \in [R]$ do
7:             Sample $\beta \in \mathrm{Uniform}([\frac{1}{2}\widehat{\beta}, \widehat{\beta}])$
8:             $Z \leftarrow$ GENERATESIGNIFICANTSAMPLES($x, H, G$)        ▷ $Z \in \mathbb{C}^{B\times 2}$, see Algorithm 65
9:             for $j \in [B]$ do
10:                $\mathcal{L}_{d,r,j,1} \leftarrow Z_{j,1}$        ▷ $Z_{j,1} = z^{(j)}(\alpha + \beta)$
11:                $\mathcal{L}_{d,r,j,2} \leftarrow Z_{j,2}$        ▷ $Z_{j,2} = z^{(j)}(\alpha)$
12:            end for
13:        end for
14:    end for
15:    return $\mathcal{L}$
16: end procedure
17: procedure GETSIGNIFICANTSAMPLE($\mathcal{L}, d, r, j$)
18:    return $(\mathcal{L}_{d,r,j,1}, \mathcal{L}_{d,r,j,2})$
19: end procedure
```

Algorithm 68 Frequency Estimation

```
1: procedure FrequencyEstimationX($x$)
2:     $\mathcal{L} \leftarrow$ SamplingSignificantSample($x$)
3:     for $j \leftarrow 1, \cdots, B$ do
4:         $\widetilde{f}_j \leftarrow$ FrequencyEstimationZ($x, H, G_{\sigma,b}^{(j)}$)    $\triangleright$ $z^{(j)} = (x \cdot H) * G_{\sigma,b}^{(j)}(t)$
5:     end for
6:     $L \leftarrow \{\widetilde{f}_1, \cdots, \widetilde{f}_B\}$
7:     return $L$
8: end procedure
```

Offset event does not happen. Let $U = \{t_0 \in \mathbb{R} \mid H(t) > 1 - \delta_1 \ \forall t \in [t_0, t_0 + \beta]\}$.

For $j \in [B]$, let $z_j^*(t) := (x^* \cdot H) * G_{\sigma,b}^{(j)}(t)$ and $z_j(t) = (x \cdot H) * G_{\sigma,b}^{(j)}(t)$. Let $g_j(t) := z_j(t) - z_j^*(t)$. Let

$$S_{g1} = \{j \in [B] \mid \|g_j(t)\|_T^2 \le c\|z_j^*(t)\|_U^2\},$$

where $c \in (0, 0.001)$ is a small universal constant. Let

$$S_{g2} = \left\{ j \in [B] \ \middle| \ \exists f_0, h_{\sigma,b}(f_0) = j \text{ and } \int_{f_0 - \Delta_h}^{f_0 + \Delta_h} |\widehat{x^* \cdot H}(f)|^2 \mathrm{d}f \ge T\mathcal{N}^2/k \right\}.$$

Let $S_g = S_{g1} \cap S_{g2}$. Let $S_f = \{f_i \mid \exists j \in S_g : h_{\sigma,b}(f_i) = j \ \forall i \in [k]\}$.

There is a Procedure FrequencyEstimationX *in Algorithm 68 such that:*

- *takes $k^2 \log(1/\delta) \log(FT)$ samples,*
- *runs in $k^2 \log(1/\delta) \log^2(FT)$ time,*
- *returns a set L of $O(k)$ frequencies such that with probability $1 - \rho_0$, for any $f \in S_f$, there exists an $\widetilde{f} \in L$ satisfying*

$$|f - \widetilde{f}| \lesssim \Delta.$$

Proof. We prove the correctness, time complexity, and sample complexity of Algorithm 68 in below.

Correctness: By Lemma 12.55, we know that the Procedure GenerateSignificantSamples in Algorithm 65 takes $\mathcal{S} = O(k^2 \log^2(k/\delta_1))$ samples in x, runs in $\mathcal{T} = O(k^2 \log^3(k/\delta_1))$ time, and for each $j \in S_g$, and outputs α_j such that for each $j \in S_g$ with probability 0.6,

$$|z_j(\alpha_j + \beta) - z_j(\alpha_j) e^{2\pi \mathbf{i} f_0 \beta}|^2 \leq 0.01 |z_j(\alpha_j)|^2,$$

where f_0 satisfies

$$\int_{f_0-\Delta}^{f_0+\Delta} |\widehat{x^* \cdot H}(f)|^2 \mathrm{d}f \geq T\mathcal{N}^2/k, \tag{12.73}$$

and $j = h_{\sigma,b}(f_0)$.

In the line 4, we call the algorithm FrequencyEstimationZ$(x, H, G_{\sigma,b}^{(j)})$. By Lemma 12.60, FrequencyEstimationZ$(x, H, G_{\sigma,b}^{(j)})$ output $\widetilde{f}$ for each $f_j \in S_f$ such that with probability at least $1 - \rho_1$

$$|\widetilde{f} - f_j| \lesssim \Delta.$$

As a result, for all the $f \in S_f$, there is a $\widetilde{f} \in L$ such that

$$|\widetilde{f} - f| \lesssim \Delta$$

holds with probability at least

$$1 - B\rho_1 \geq 1 - \rho_0.$$

Time complexity: We show that the Procedure FrequencyEstimationX in Algorithm 68 runs in

$$O(k^2 \log(k) \log(k/\delta_1) \log(FT) \log(\log(FT)/\rho_1))$$

time.

In each call of the Procedure FrequencyEstimationX in Algorithm 68,

- Line 2 call Procedure SamplingSignificantSample, which runs in

$$O(k^2 \log(k) \log(k/\delta_1) \log(FT) \log(\log(FT)/\rho_1))$$

  time by Lemma 12.62.

- The for-loop repeats B times:

  - In each loop, line 4 call Procedure FrequencyEstimationZ, which runs in

    $$O(\log(FT) \cdot \log(\log(FT)/\rho_1))$$

    time by Lemma 12.60.

Thus, the total time complexity is

$$\begin{aligned}
&O(k^2 \log(k) \log(k/\delta_1) \log(FT) \log(\log(FT)/\rho_1)) + B \cdot O(\log(FT) \cdot \log(\log(FT)/\rho_1)) \\
\leq\ &O(k^2 \log(k) \log(k/\delta_1) \log(FT) \log(\log(FT)/\rho_1)) + O(k) \cdot O(\log(FT) \cdot \log(\log(FT)/\rho_1)) \\
\leq\ &O(k^2 \log(k) \log(k/\delta_1) \log(FT) \log(\log(FT)/\rho_1)),
\end{aligned}$$

where the first step follows from $B = O(k)$, the second step is straight forward.

Sample complexity: We show that the Procedure FrequencyEstimationX in Algorithm 68 takes

$$O(k^2 \log(k) \log(k/\delta_1) \log(FT) \log(\log(FT)/\rho_1))$$

samples.

In each call of the Procedure FrequencyEstimationX in Algorithm 68, Line 2 call Procedure SamplingSignificantSample, which takes $O(k^2 \log(k) \log(k/\delta_1) \log(FT) \log(\log(FT$ samples by Lemma 12.63.

So, the sample complexity of Procedure FrequencyEstimationX in Algorithm 68 is

$$O(k^2 \log(k) \log(k/\delta_1) \log(FT) \log(\log(FT)/\rho_1)).$$

□

The following two lemmas shows the time complexity and sample complexity of the significant sample generation procedure in Algorithm 67.

Lemma 12.62 (Running time of Procedure SAMPLINGSIGNIFICANTSAMPLE in Algorithm 67)**.** *Procedure* SAMPLINGSIGNIFICANTSAMPLE *in Algorithm 67 runs in*

$$O(k^2 \log(k) \log(k/\delta_1) \log(FT) \log(\log(FT)/\rho_1))$$

times.

Proof. In each call of Procedure SAMPLINGSIGNIFICANTSAMPLE in Algorithm 67, in line 3, the for loop repeats D times, in line 6, the for loops repeats R times,

- In line 8, by Lemma 12.56, each call of Procedure GENERATESIGNIFICANTSAMPLES takes $O(k^2 \log(k) \log(k/\delta_1))$ times.
- In line 9, the for loop repeats B times, each iteration runs in $O(1)$ times.

Following from the setting in the algorithm, we have that

$$\begin{aligned}
\mathsf{num} &= O(1), \\
D &= O(\log(\frac{FT}{\Delta})/\log(\mathsf{num})), \\
R &= O(\log(\frac{\log(FT)}{\rho_1 \log(\mathsf{num})})), \\
B &= O(k).
\end{aligned} \tag{12.74}$$

We have that

$$D = O(\log(\frac{FT}{\Delta})/\log(\mathsf{num})) \leq O(\log(\frac{FT}{\Delta})) \leq O(\log(FT)), \tag{12.75}$$

where the first step follows from the setting of D, the second step follows from $\mathsf{num} = O(1)$, the third step follows from $\Delta = \mathrm{poly}(k)$.

We also have that

$$R = O(\log(\frac{\log(FT)}{\rho_1 \log(\mathsf{num})})) \leq O(\log(\log(FT)/\rho_1)), \tag{12.76}$$

where the first step follows from the setting of R, the second step follows from $\mathsf{num} = O(1)$.

So, the time complexity of Procedure SamplingSignificantSample in Algorithm 67 is

$$\begin{aligned}
& D \cdot R \cdot (O(k^2 \log(k) \log(k/\delta_1)) + B \cdot O(1)) \\
\leq & O(\log(FT)) \cdot R \cdot (O(k^2 \log(k) \log(k/\delta_1)) + B \cdot O(1)) \\
\leq & O(\log(FT)) \cdot O(\log(\log(FT)/\rho_1)) \cdot (O(k^2 \log(k) \log(k/\delta_1)) + B \cdot O(1)) \\
\leq & O(\log(FT)) \cdot O(\log(\log(FT)/\rho_1)) \cdot (O(k^2 \log(k) \log(k/\delta_1)) + O(k) \cdot O(1)) \\
= & O(k^2 \log(k) \log(k/\delta_1) \log(FT) \log(\log(FT)/\rho_1)),
\end{aligned}$$

where the first step follows from Eq. (12.75), the second step follows from Eq. (12.76), the third step follows from Eq. (12.74), the forth step is straightforward.

□

Lemma 12.63 (Sample complexity of Procedure SamplingSignificantSample in Algorithm 67)**.** *Procedure* SamplingSignificantSample *in Algorithm 67 takes*

$$O(k^2 \log(k) \log(k/\delta_1) \log(FT) \log(\log(FT)/\rho_1))$$

samples.

Proof. In each call of Procedure SamplingSignificantSample in Algorithm 67, In line 3, the for loop repeats D times, in line 6, the for loops repeats R times,

- In line 8, by Lemma 12.56, each call of Procedure GenerateSignificantSamples takes $O(k^2 \log(k) \log(k/\delta_1))$ samples.

Following from the setting in the algorithm, we have that

$$\begin{aligned}
\mathsf{num} &= O(1),\\
D &= O(\log(\frac{FT}{\Delta})/\log(\mathsf{num})),\\
R &= O(\log(\frac{\log(FT)}{\rho_1 \log(\mathsf{num})})).
\end{aligned}$$

We have that

$$D = O(\log(\frac{FT}{\Delta})/\log(\mathsf{num})) \leq O(\log(\frac{FT}{\Delta})) \leq O(\log(FT)), \tag{12.77}$$

where the first step follows from the setting of D, the second step follows from $\mathsf{num} = O(1)$, the third step follows from $\Delta = \mathrm{poly}(k)$.

We also have that

$$R = O(\log(\frac{\log(FT)}{\rho_1 \log(\mathsf{num})})) \leq O(\log(\log(FT)/\rho_1)), \tag{12.78}$$

where the first step follows from the setting of R, the second step follows from $\mathsf{num} = O(1)$.

So, the sample complexity of Procedure SamplingSignificantSample in Algorithm 67 is

$$\begin{aligned}
& D \cdot R \cdot O(k^2 \log(k) \log(k/\delta_1))\\
\leq\ & O(\log(FT)) \cdot R \cdot O(k^2 \log(k) \log(k/\delta_1))\\
\leq\ & O(\log(FT)) \cdot O(\log(\log(FT)/\rho_1)) \cdot O(k^2 \log(k) \log(k/\delta_1))\\
=\ & O(k^2 \log(k) \log(k/\delta_1) \log(FT) \log(\log(FT)/\rho_1)),
\end{aligned}$$

where the first step follows from Eq. (12.77), the second step follows from Eq. (12.78), the third step is straightforward.

□

12.15.3 Vote distributions in ArySearch

In this section, we prove several claims on the distributions of votes when we perform the num-ary search on the frequency interval.

We first consider a single voter, i.e., one significant sample. The following claim shows that, if f_0 is in the q-th part, then this part or its left neighbor or its right neighbor will get at least one vote.

Claim 12.64. *For* $\mathsf{len} \in \mathbb{R}_+, \mathsf{num} \in \mathbb{Z}_+, q \in [1, \mathsf{num}]$*, let* $f_0 \in [\mathsf{left} + (q-1)\frac{\mathsf{len}}{\mathsf{num}}, \mathsf{left} + q\frac{\mathsf{len}}{\mathsf{num}}]$*. For any* $\beta \in [\frac{c}{2} \cdot \frac{\mathsf{num}}{\mathsf{len}}, c \cdot \frac{\mathsf{num}}{\mathsf{len}}]$ *with constant* $c \in (0, 0.01)$*, for any constant* $\varepsilon \in (0, 0.01 \cdot c^2)$*, let* $\alpha \in \mathbb{R}$ *such that*

$$|z(\alpha+\beta) - z(\alpha)e^{2\pi \mathbf{i} f_0 \beta}|^2 \le \varepsilon |z(\alpha)|^2.$$

Let

$$\Theta = \Big\{ \frac{1}{2\pi\beta}(\arg(\frac{z(\alpha+\beta)}{z(\alpha)}) + 2\pi s) \ \Big| \ s \in [\beta\mathsf{left} - 10, \beta(\mathsf{left}+\mathsf{len}) + 10] \cap \mathbb{Z} \Big\}.$$

Then, we have that

$$\Big| \Theta \cap \Big[\mathsf{left} + (q-2)\frac{\mathsf{len}}{\mathsf{num}}, \mathsf{left} + (q+1)\frac{\mathsf{len}}{\mathsf{num}}\Big) \Big| = 1.$$

Proof. We have that

$$\Big| \frac{z(\alpha+\beta)}{z(\alpha)} - e^{2\pi \mathbf{i} f_0 \beta} \Big| \le \sqrt{\varepsilon}.$$

Since $|e^{2\pi \mathbf{i} f_0 \beta}| = 1$ and $\sin(x) \approx x$ for small x, it indicates that

$$\Big\| \arg(\frac{z(\alpha+\beta)}{z(\alpha)}) - 2\pi f_0 \beta \Big\|_{\bigcirc} \lesssim \sqrt{\varepsilon}, \tag{12.79}$$

where $\|a\|_{\bigcirc} = \min_{x \in \mathbb{Z}} |a + 2\pi x|$. Thus, Eq. (12.79) can be rewritten as:

$$\min_{x \in \mathbb{Z}} \Big| \arg(\frac{z(\alpha+\beta)}{z(\alpha)}) - 2\pi f_0 \beta + 2\pi x \Big| \lesssim \sqrt{\varepsilon}. \tag{12.80}$$

Let s_0 be defined as

$$s_0 := \arg\min_{x \in \mathbb{Z}} \ \arg(\frac{z(\alpha+\beta)}{z(\alpha)}) - 2\pi f_0 \beta + 2\pi x.$$

We first show that s_0 falls in interval in the definition of Θ. We have that

$$\begin{aligned} |2\pi s_0 - 2\pi f_0 \beta| &\le \left| \arg(\frac{z(\alpha+\beta)}{z(\alpha)}) - 2\pi f_0\beta + 2\pi s_0 \right| + \left| \arg(\frac{z(\alpha+\beta)}{z(\alpha)}) \right| \\ &\le \left| \arg(\frac{z(\alpha+\beta)}{z(\alpha)}) - 2\pi f_0\beta + 2\pi s_0 \right| + 2\pi \\ &\le 2\pi + O(\sqrt{\varepsilon}) \end{aligned} \tag{12.81}$$

where the first step follows from the triangle inequality, the second step follows from $|\arg(\frac{z(\alpha+\beta)}{z(\alpha)})| \le 2\pi$, the third step follows from Eq. (12.80).

As a result, s_0 has the following upper bound:

$$\begin{aligned} s_0 &\le f_0\beta + 1 + O(\sqrt{\varepsilon}) \\ &\le \beta(\mathsf{left} + \mathsf{len}) + 1 + O(\sqrt{\varepsilon}) \\ &\le \beta(\mathsf{left} + \mathsf{len}) + 2 \end{aligned}$$

where the first step follows from Eq. (12.81), the second step follows from $f_0 \in [\mathsf{left} + (q-1)\frac{\mathsf{len}}{\mathsf{num}}, \mathsf{left} + q\frac{\mathsf{len}}{\mathsf{num}}] \subseteq [\mathsf{left}, \mathsf{left} + \mathsf{len}]$, the third step follows from the setting of ε.

Also, s_0 has the following lower bound:

$$\begin{aligned} s_0 &\ge f_0\beta - 1 - O(\sqrt{\varepsilon}) \\ &\ge \beta\mathsf{left} - 1 - O(\sqrt{\varepsilon}) \\ &\ge \beta\mathsf{left} - 2 \end{aligned}$$

where the first step follows from Eq. (12.81), the second step follows from $f_0 \in [\mathsf{left}, \mathsf{left} + \mathsf{len}]$, the third step follows from the setting of ε.

Combining the lower and upper bounds of s_0 together, and by the definition of the set Θ, we know that

$$\frac{1}{2\pi\beta}\Big(\arg(\frac{z(\alpha+\beta)}{z(\alpha)})+2\pi s_0\Big)\in\Theta.$$

Then, we show that

$$\mathsf{left}+(q-2)\frac{\mathsf{len}}{\mathsf{num}}\leq\frac{1}{2\pi\beta}\Big(\arg(\frac{z(\alpha+\beta)}{z(\alpha)})+2\pi s_0\Big)<\mathsf{left}+(q+1)\frac{\mathsf{len}}{\mathsf{num}}.$$

Eq. (12.80) also implies that

$$\Big|\frac{1}{2\pi\beta}(\arg(\frac{z(\alpha+\beta)}{z(\alpha)})+2\pi s_0)-f_0\Big|\leq O(\frac{\sqrt{\varepsilon}}{\beta}). \tag{12.82}$$

Then, we have the following upper bound:

$$\begin{aligned}\frac{1}{2\pi\beta}(\arg(\frac{z(\alpha+\beta)}{z(\alpha)})+2\pi s_0)&\leq f_0+O(\frac{\sqrt{\varepsilon}}{\beta})\\&\leq\mathsf{left}+q\frac{\mathsf{len}}{\mathsf{num}}+O(\frac{\sqrt{\varepsilon}}{\beta})\\&\leq\mathsf{left}+q\frac{\mathsf{len}}{\mathsf{num}}+O(\frac{2\sqrt{\varepsilon}}{c}\frac{\mathsf{len}}{\mathsf{num}})\\&\leq\mathsf{left}+(q+1)\frac{\mathsf{len}}{\mathsf{num}},\end{aligned}$$

where the first step follows from Eq. (12.82), the second step follows from $f_0\in[\mathsf{left}+(q-1)\mathsf{len}/\mathsf{num},\mathsf{left}+q\mathsf{len}/\mathsf{num}]$, the third step follows from $\beta\in[c\frac{\mathsf{num}}{2\mathsf{len}},c\frac{\mathsf{num}}{\mathsf{len}}]$, the forth step follows from $O(\sqrt{\varepsilon}/c)\leq 1$.

We also have the following lower bound:

$$\begin{aligned}\frac{1}{2\pi\beta}(\arg(\frac{z(\alpha+\beta)}{z(\alpha)})+2\pi s_0)&\geq f_0-O(\frac{\sqrt{\varepsilon}}{\beta})\\&\geq\mathsf{left}+(q-1)\frac{\mathsf{len}}{\mathsf{num}}-O(\frac{\sqrt{\varepsilon}}{\beta})\\&\geq\mathsf{left}+(q-1)\frac{\mathsf{len}}{\mathsf{num}}-O(\frac{\sqrt{\varepsilon}}{c}\frac{\mathsf{len}}{\mathsf{num}})\\&>\mathsf{left}+(q-2)\frac{\mathsf{len}}{\mathsf{num}},\end{aligned}$$

where the first step follows from Eq. (12.82), the second step follows from $f_0 \in [\mathsf{left} + (q-1)\mathsf{len}/\mathsf{num}, \mathsf{left} + q\mathsf{len}/\mathsf{num}]$, the third step follows from $\beta \in [c\frac{\mathsf{num}}{2\mathsf{len}}, c\frac{\mathsf{num}}{\mathsf{len}}]$, the forth step follows from $O(\sqrt{\varepsilon}/c) < 1$.

Moreover, since

$$\frac{1}{\beta} \geq \frac{\mathsf{len}}{\mathsf{num}},$$

we have that there is at most 1 element in the intersection

$$\left|\Theta \cap \left[\mathsf{left} + (q-2)\frac{\mathsf{len}}{\mathsf{num}}, \mathsf{left} + (q+1)\frac{\mathsf{len}}{\mathsf{num}}\right)\right| \leq 1.$$

The lemma then follows. □

The following claim shows that for those parts far away from the true part containing f_0, they will get no vote.

Claim 12.65. *For* $\mathsf{len} \in \mathbb{R}_+, \mathsf{num} \in \mathbb{Z}_+, q \in [1, \mathsf{num}]$, *let* $f_0 \in [\mathsf{left} + (q-1)\frac{\mathsf{len}}{\mathsf{num}}, \mathsf{left} + q\frac{\mathsf{len}}{\mathsf{num}}]$. *Let* $\beta \sim$ *Uniform*$([\frac{c}{2} \cdot \frac{\mathsf{num}}{\mathsf{len}}, c \cdot \frac{\mathsf{num}}{\mathsf{len}}])$ *with constant* $c \in (0, 0.01)$. *For any constant* $\varepsilon \in (0, 0.01 \cdot c^2)$, *let* $\alpha \in \mathbb{R}$ *such that*

$$|z(\alpha+\beta) - z(\alpha)e^{2\pi \mathbf{i} f_0 \beta}|^2 \leq \varepsilon |z(\alpha)|^2.$$

Let

$$\Theta = \left\{ \frac{1}{2\pi\beta}(\arg(\frac{z(\alpha+\beta)}{z(\alpha)}) + 2\pi s) \;\middle|\; s \in [\beta\mathsf{left} - 10, \beta(\mathsf{left} + \mathsf{len}) + 10] \cap \mathbb{Z} \right\}.$$

Then, we have that for any $q' \in [0, \mathsf{num} - 1]$ *such that* $|q - q'| > 1$, *with probability at least* $1 - O(c)$,

$$\Theta \cap \left[\mathsf{left} + (q'-1)\frac{\mathsf{len}}{\mathsf{num}}, \mathsf{left} + q'\frac{\mathsf{len}}{\mathsf{num}}\right] = \emptyset.$$

Proof. Let s_0 be defined as

$$s_0 := \arg\min_{x \in \mathbb{Z}} \; \arg(\frac{z(\alpha+\beta)}{z(\alpha)}) - 2\pi f_0 \beta + 2\pi x.$$

By Claim 12.64, we have that

$$\frac{1}{2\pi\beta}(\arg(\frac{z(\alpha+\beta)}{z(\alpha)}) + 2\pi s_0) \in \Big[\mathsf{left} + (q-2)\frac{\mathsf{len}}{\mathsf{num}}, \mathsf{left} + (q+1)\frac{\mathsf{len}}{\mathsf{num}}\Big). \tag{12.83}$$

Then, we discuss two cases based on the range of q'.

Case 1: $1 < |q - q'| < 1/(4c)$**.**

For the ease of discussion, suppose $1 < q' - q < 1/(4c)$. We have that

$$\begin{aligned}\mathsf{left} + (q'-1)\frac{\mathsf{len}}{\mathsf{num}} &\geq \mathsf{left} + (q+1)\frac{\mathsf{len}}{\mathsf{num}} \\ &> \frac{1}{2\pi\beta}(\arg(\frac{z(\alpha+\beta)}{z(\alpha)}) + 2\pi s_0),\end{aligned}$$

where the first step follows from $q, q' \in \mathbb{Z}$, and the second step follows from Eq. (12.83).

Moreover, we also have that

$$\begin{aligned}\mathsf{left} + q'\frac{\mathsf{len}}{\mathsf{num}} &\leq \mathsf{left} + (q + \frac{1}{4c} - 1)\frac{\mathsf{len}}{\mathsf{num}} \\ &\leq \frac{1}{2\pi\beta}(\arg(\frac{z(\alpha+\beta)}{z(\alpha)}) + 2\pi s_0) + (\frac{1}{4c}+1)\frac{\mathsf{len}}{\mathsf{num}} \\ &< \frac{1}{2\pi\beta}(\arg(\frac{z(\alpha+\beta)}{z(\alpha)}) + 2\pi s_0) + \frac{1}{\beta} \\ &\leq \frac{1}{2\pi\beta}(\arg(\frac{z(\alpha+\beta)}{z(\alpha)}) + 2\pi(s_0+1)),\end{aligned}$$

where the second step follows from Eq. (12.83), the third step follows from $(\frac{1}{4c} + 1)\mathsf{len}/\mathsf{num} < 1/\beta$.

Hence, we get that $\Big[\mathsf{left} + (q'-1)\frac{\mathsf{len}}{\mathsf{num}}, \mathsf{left} + q'\frac{\mathsf{len}}{\mathsf{num}}\Big]$ is contained in the following interval:

$$\Big(\frac{1}{2\pi\beta}(\arg(\frac{z(\alpha+\beta)}{z(\alpha)}) + 2\pi s_0), \frac{1}{2\pi\beta}(\arg(\frac{z(\alpha+\beta)}{z(\alpha)}) + 2\pi(s_0+1))\Big).$$

Since s_0 and $s_0 + 1$ are two consecutive integers, by the definition of Θ, there is no element of Θ in this open interval. Hence, we know that in this case,

$$\Theta \cap \Big[\mathsf{left} + (q'-1)\frac{\mathsf{len}}{\mathsf{num}}, \mathsf{left} + q'\frac{\mathsf{len}}{\mathsf{num}}\Big] = \emptyset.$$

Case 2: $|q - q'| \geq 1/(4c)$.

For the ease of discussion, suppose that $q' - q \geq 1/(4c)$. We have that,

$$c(q' - q) \geq \frac{1}{4}. \tag{12.84}$$

Moreover, we have that

$$\arg(\frac{z(\alpha+\beta)}{z(\alpha)}) - 2\pi\beta\left(\mathsf{left} + (q - \frac{1}{2})\frac{\mathsf{len}}{\mathsf{num}}\right) \pmod{2\pi} \in \left[-\frac{3\pi\beta\mathsf{len}}{\mathsf{num}}, \frac{3\pi\beta\mathsf{len}}{\mathsf{num}}\right], \tag{12.85}$$

which follows from Eq. (12.83).

We also have that,

$$\beta\frac{\mathsf{len}}{\mathsf{num}} \leq c. \tag{12.86}$$

Then, we have that,

$$\begin{aligned}
&\Pr\left[2\pi\beta(\mathsf{left} + (q' - \frac{1}{2})\frac{\mathsf{len}}{\mathsf{num}}) - \arg(\frac{z(\alpha+\beta)}{z(\alpha)}) \pmod{2\pi} \in [-\frac{\pi\beta\mathsf{len}}{\mathsf{num}}, \frac{\pi\beta\mathsf{len}}{\mathsf{num}}]\right] \\
\leq{}& \Pr\left[2\pi\beta(q' - q)\frac{\mathsf{len}}{\mathsf{num}} \pmod{2\pi} \in [-\frac{4\pi\beta\mathsf{len}}{\mathsf{num}}, \frac{4\pi\beta\mathsf{len}}{\mathsf{num}}]\right] \\
\leq{}& \Pr\left[2\pi\beta(q' - q)\frac{\mathsf{len}}{\mathsf{num}} \pmod{2\pi} \in [-4\pi c, +4\pi c]\right] \\
\leq{}& 4c + \frac{16}{q' - q} \\
\leq{}& 100c
\end{aligned} \tag{12.87}$$

where the first step follows from Eq. (12.85), the second step follows from Eq. (12.86), the third step follows from Lemma 12.15 with the following parameters:

$$\begin{aligned}
\widetilde{T} &= 2\pi, \\
\widetilde{\sigma} &= 2\pi\beta(q' - q)\frac{\mathsf{len}}{\mathsf{num}}, \\
\widetilde{\varepsilon} &= 4\pi c, \\
\widetilde{\delta} &= 0, \\
A &= \pi c(q' - q),
\end{aligned}$$

the forth step follows from Eq. (12.84).

By Eq. (12.87), we have that

$$\mathsf{Pr}\Big[\exists s_0 \in \mathbb{Z}, \frac{1}{2\pi\beta}(\arg(\frac{z(\alpha+\beta)}{z(\alpha)}) + 2\pi s_0) \in \Big[\mathsf{left} + (q'-1)\frac{\mathsf{len}}{\mathsf{num}}, \mathsf{left} + q'\frac{\mathsf{len}}{\mathsf{num}}\Big]\Big] \leq 100c$$

As a result, we know that in this case, with probability at least $1 - O(c)$,

$$\Theta \cap \Big[\mathsf{left} + (q'-1)\frac{\mathsf{len}}{\mathsf{num}}, \mathsf{left} + q'\frac{\mathsf{len}}{\mathsf{num}}\Big] = \emptyset.$$

$\square$

Then, we consider R independent voters, i.e., R significant samples $\alpha_1, \ldots, \alpha_R$. The following claim shows that the true part and its left and right neighbors will get at least R votes. Meanwhile, those parts far away from the true part will get at most $R/2$ votes with high probability.

Claim 12.66. *For For* $\mathsf{len} \in \mathbb{R}_+, \mathsf{num} \in \mathbb{Z}_+, q \in [1, \mathsf{num}]$, *let* $f_0 \in [\mathsf{left} + (q-1)\frac{\mathsf{len}}{\mathsf{num}}, \mathsf{left} + q\frac{\mathsf{len}}{\mathsf{num}}]$. *Let* $\beta \sim$ *Uniform*$([\frac{c}{2} \cdot \frac{\mathsf{num}}{\mathsf{len}}, c \cdot \frac{\mathsf{num}}{\mathsf{len}}])$ *with constant* $c \in (0, 0.01)$. *For any constant* $\varepsilon \in (0, 0.01 \cdot c^2)$, *Let* $\alpha_1, \cdots, \alpha_R \in \mathbb{R}$ *such that for any* $i \in [R]$,

$$|z(\alpha_i + \beta) - z(\alpha_i)e^{2\pi \mathbf{i} f_0 \beta}|^2 \leq \varepsilon |z(\alpha_i)|^2.$$

For any $i \in [R]$, *let*

$$\Theta_i = \Big\{\frac{1}{2\pi\beta}(\arg(\frac{z(\alpha_i+\beta)}{z(\alpha_i)}) + 2\pi s) \;\Big|\; s \in [\beta\mathsf{left} - 10, \beta(\mathsf{left} + \mathsf{len}) + 10] \cap \mathbb{Z}\Big\}.$$

Then, it holds that:

1. $$\sum_{i=1}^{R} \Big|\Theta_i \cap \Big[\mathsf{left} + (q-2)\frac{\mathsf{len}}{\mathsf{num}}, \mathsf{left} + (q+1)\frac{\mathsf{len}}{\mathsf{num}}\Big]\Big| \geq R.$$

2. *For any* $|q' - q| \geq 3$, *with probability at least* $1 - O(c)^{R/6}$,

$$\sum_{i=1}^{R} \Big|\Theta_i \cap \Big[\mathsf{left} + (q'-2)\frac{\mathsf{len}}{\mathsf{num}}, \mathsf{left} + (q'+1)\frac{\mathsf{len}}{\mathsf{num}}\Big]\Big| \leq \frac{R}{2}.$$

Proof. **Part 1.**

By applying Claim 12.64, we have that,

$$\Big|\Theta_i \cap \Big[\mathsf{left} + (q-2)\frac{\mathsf{len}}{\mathsf{num}}, \mathsf{left} + (q+1)\frac{\mathsf{len}}{\mathsf{num}}\Big]\Big| \geq 1,$$

which implies that

$$\sum_{i=1}^{R} \Big|\Theta_i \cap \Big[\mathsf{left} + (q-2)\frac{\mathsf{len}}{\mathsf{num}}, \mathsf{left} + (q+1)\frac{\mathsf{len}}{\mathsf{num}}\Big]\Big| \geq R.$$

Part 2. By applying Claim 12.65, we have that, for any $|q - q'| > 1$, with probability at most $O(c)$,

$$\Big|\Theta_i \cap \Big[\mathsf{left} + (q'-1)\frac{\mathsf{len}}{\mathsf{num}}, \mathsf{left} + q'\frac{\mathsf{len}}{\mathsf{num}}\Big]\Big| \geq 1.$$

By the setting of our parameter $\frac{1}{\beta} \geq \frac{\mathsf{len}}{\mathsf{num}}$, thus

$$\Big|\Theta_i \cap \Big[\mathsf{left} + (q'-1)\frac{\mathsf{len}}{\mathsf{num}}, \mathsf{left} + q'\frac{\mathsf{len}}{\mathsf{num}}\Big]\Big| = 1.$$

Then, for any $|q - q'| \geq 3$, by a union bound over $q'-1$, q', and $q'+1$, with probability at most $O(c)$,

$$3 \geq \Big|\Theta_i \cap \Big[\mathsf{left} + (q'-2)\frac{\mathsf{len}}{\mathsf{num}}, \mathsf{left} + (q'+1)\frac{\mathsf{len}}{\mathsf{num}}\Big]\Big| \geq 1.$$

Then, we have that

$$\begin{aligned}
&\Pr\Big[\sum_{i=1}^{R} \Big|\Theta_i \cap \Big[\mathsf{left} + (q'-2)\frac{\mathsf{len}}{\mathsf{num}}, \mathsf{left} + (q'+1)\frac{\mathsf{len}}{\mathsf{num}}\Big]\Big| \geq \frac{R}{2}\Big] \\
\leq &\binom{R}{R/6} O(c)^{R/6} \\
\leq &(\frac{eR}{R/6})^{R/6} O(c)^{R/6} \\
\leq &O(c)^{R/6}
\end{aligned}$$

where the first step follows from there should be at least $0.5R/3 = R/6$ different $i \in [R]$ satisfying $|\Theta_i \cap [\mathsf{left} + (q'-2)\mathsf{len}/\mathsf{num}, \mathsf{left} + (q'+1)\mathsf{len}/\mathsf{num}]| \geq 1$,the second step follows from $\binom{n}{k} \leq (\frac{en}{k})^k$, the third step is straight forward.

The lemma is then proved. $\square$

Finally, we consider probabilistic voters, that is, for each sample α_i, with probability $1-\rho$, it is significant. The following claim shows the votes distribution in this case.

Claim 12.67. *For* $\mathsf{len} \in \mathbb{R}_+, \mathsf{num} \in \mathbb{Z}_+, q \in [1, \mathsf{num}]$, *let* $f_0 \in [\mathsf{left} + (q-1)\frac{\mathsf{len}}{\mathsf{num}}, \mathsf{left} + q\frac{\mathsf{len}}{\mathsf{num}}]$. *Let* $\beta \sim$ *Uniform*$([\frac{c}{2} \cdot \frac{\mathsf{num}}{\mathsf{len}}, c \cdot \frac{\mathsf{num}}{\mathsf{len}}])$ *with* $c = \Theta(1) \in (0, 0.01)$, $\varepsilon = \Theta(1) \in (0, 0.01 \cdot c^2)$, *let* $\alpha_1, \cdots, \alpha_R \in \mathbb{R}$ *such that for any* $i \in [R]$ *with probability at least* $1-\rho$,

$$|z(\alpha_i + \beta) - z(\alpha_i)e^{2\pi \mathbf{i} f_0 \beta}|^2 \leq \varepsilon |z(\alpha_i)|^2.$$

For any $i \in [R]$, *let*

$$\Theta_i = \Big\{ \frac{1}{2\pi\beta}(\arg(\frac{z(\alpha_i+\beta)}{z(\alpha_i)}) + 2\pi s) \ \Big| \ s \in [\beta\mathsf{left} - 10, \beta(\mathsf{left} + \mathsf{len}) + 10] \cap \mathbb{Z} \Big\}.$$

Then, it holds that

1. *With probability at least* $1 - O(\rho)^{R/3}$,
$$\sum_{i=1}^{R} \left| \Theta_i \cap \left[\mathsf{left} + (q-2)\frac{\mathsf{len}}{\mathsf{num}}, \mathsf{left} + (q+1)\frac{\mathsf{len}}{\mathsf{num}}\right] \right| \geq \frac{2R}{3}.$$

2. *For any* $|q' - q| \geq 3$, *with probability at least* $1 - O(c+\rho)^{R/6}$,
$$\sum_{i=1}^{R} \left| \Theta_i \cap \left[\mathsf{left} + (q'-2)\frac{\mathsf{len}}{\mathsf{num}}, \mathsf{left} + (q'+1)\frac{\mathsf{len}}{\mathsf{num}}\right] \right| \leq \frac{R}{2}.$$

Proof. **Part 1.**

By applying Claim 12.64, we have that with probability at most ρ,

$$\left| \Theta_i \cap \left[\mathsf{left} + (q-2)\frac{\mathsf{len}}{\mathsf{num}}, \mathsf{left} + (q+1)\frac{\mathsf{len}}{\mathsf{num}}\right] \right| = 0,$$

then we have that,

$$\begin{aligned}
&\Pr\Big[\sum_{i=1}^{R}\Big|\Theta_i \cap [\mathsf{left}+(q-2)\frac{\mathsf{len}}{\mathsf{num}}, \mathsf{left}+(q+1)\frac{\mathsf{len}}{\mathsf{num}}]\Big| \leq \frac{R}{3}\Big] \\
\leq &\binom{R}{R/3} O(\rho)^{R/3} \\
\leq &O(\frac{eR}{R/3})^{R/3} O(\rho)^{R/3} \\
\leq &O(\rho)^{R/3}
\end{aligned}$$

where the first step follows from $|\Theta_i \cap [\mathsf{left}+(q-2)\mathsf{len}/\mathsf{num}, \mathsf{left}+(q+1)\mathsf{len}/\mathsf{num}]| = 0$ or 1 by our parameter setting $1/\beta > 3\mathsf{len}/\mathsf{num}$ and there should be at least $R/3$ different $i \in [R]$ satisfying $|\Theta_i \cap [\mathsf{left}+(q-2)\mathsf{len}/\mathsf{num}, \mathsf{left}+(q+1)\mathsf{len}/\mathsf{num}]| = 0$, the second step follows from $\binom{n}{k} \leq (\frac{en}{k})^k$, the third step is straight forward.

Part 2.

By applying Claim 12.65, we have that, for any $|q-q'| > 1$, with probability at most $O(c)+\rho$,

$$\Big|\Theta_i \cap \Big[\mathsf{left}+(q'-1)\frac{\mathsf{len}}{\mathsf{num}}, \mathsf{left}+q'\frac{\mathsf{len}}{\mathsf{num}}\Big]\Big| = 1,$$

where the probability follows from a union bound over the success of Claim 12.65 and α_i being significant.

Thus, for any $|q-q'| \geq 3$, by a union bound, with probability at most $3((1-\rho)O(c)+\rho) = O(c+\rho)$,

$$3 \geq \Big|\Theta_i \cap \Big[\mathsf{left}+(q'-2)\frac{\mathsf{len}}{\mathsf{num}}, \mathsf{left}+(q'+1)\frac{\mathsf{len}}{\mathsf{num}}\Big]\Big| \geq 1.$$

Then, we have that

$$\Pr\left[\sum_{i=1}^{R}\left|\Theta_i \cap [\mathsf{left} + (q'-2)\mathsf{len}/\mathsf{num}, \mathsf{left} + (q'+1)\mathsf{len}/\mathsf{num}]\right| \geq \frac{R}{2}\right]$$
$$\leq \binom{R}{R/6} O(c+\rho)^{R/6}$$
$$\leq (\frac{eR}{R/6})^{R/6} O(c+\rho)^{R/6}$$
$$\leq O(c+\rho)^{R/6}$$

where the first step follows from there should be at least $0.5R/3 = R/6$ different $i \in [R]$ satisfying $|\Theta_i \cap [\mathsf{left} + (q'-2)\mathsf{len}/\mathsf{num}, \mathsf{left} + (q'+1)\mathsf{len}/\mathsf{num}]| \geq 1$, the second step follows from $\binom{n}{k} \leq (\frac{en}{k})^k$, the third step is straight forward.

$\square$

12.16 Signal Reconstruction

In this section, we wrap up all technical tools developed in previous sections and present our main result: a Fourier interpolation algorithm with improved time complexity, sample complexity, and output sparsity.

This section consists of two parts. The first part is devoted to the signal estimation. We first provide some tools that are useful for signal estimation (see Section 12.16.1). Then, we formally define the heavy clusters and show their approximation property (see Section 12.16.2). Next, we give a Fourier set query algorithm, which is a component in signal estimation (see Section 12.16.3). We further show that it suffices to only reconstruct the signals in the bins satisfying the high SNR band condition (see Section 12.16.4).

The second part focuses on the Fourier interpolation algorithm. Combining the frequency estimation algorithm in Section 12.15 with the signal estimation method we just developed, we obtain a Fourier interpolation algorithm with a constant success probability (see Section 12.16.5). Then, we introduce the min-of-median signal estimator used to boost the success probability (see Section 12.16.6). Finally, we prove our main theorem that gives a Fourier interpolation algorithm with high success probability (see Section 12.16.7).

12.16.1 Preliminary

We provide some technical tools in this section.

The following two lemma shows that Fourier-polynomial mixed signals and Fourier-sparse signals can approximate each other.

Lemma 12.68 ([CKPS16]). *For any $\Delta > 0$, $\delta > 0$, for any $n_1, \ldots, n_k \in \mathbb{Z}_{\geq 0}$ with $\sum_{j\in[k]} n_j = k$, let*

$$x^*(t) = \sum_{j\in[k]} e^{2\pi \mathbf{i} f_j t} \sum_{i=1}^{n_j} v_{j,i} e^{2\pi \mathbf{i} f'_{j,i} t},$$

where $|f'_{j,i}| \leq \Delta$ for each $j \in [k], i \in [n_j]$. There exist k polynomials $P_j(t)$ for $j \in [k]$ of degree at most

$$d = O(T\Delta + k^3 \log k + k \log(1/\delta))$$

such that

$$\Big\| \sum_{j\in[k]} e^{2\pi \mathbf{i} f_j t} P_j(t) - x^*(t) \Big\|_T^2 \leq \delta \|x^*(t)\|_T^2.$$

Lemma 12.69 ([CKPS16, Lemma 8.8]). *For any degree-d polynomial $Q(t) = \sum\limits_{j=0}^{d} c_j t^j$, any $T > 0$ and any $\varepsilon > 0$, there always exist $\gamma > 0$ and*

$$x^*(t) = \sum_{j=1}^{d+1} \alpha_j e^{2\pi \mathbf{i} (\gamma j) t}$$

such that

$$|x^*(t) - Q(t)| \leq \varepsilon \quad \forall t \in [0, T].$$

Algorithm 69 Multipoint evaluation of a polynomial

```
1: procedure PolynomialEvaluation($P, t$)                                  ▷ Fact 12.70
2:     return $(P(t_1), P(t_2), \cdots, P(t_d))$                           ▷ $t \in \mathbb{C}^d$
3: end procedure
4: procedure MixedPolynomialEvaluation($\sum_{j=1}^{k} P_j(t)\exp(2\pi \mathbf{i} f_j t), t$)
5:     for $j \in [k]$ do
6:         $v_j \leftarrow (P_j(t_1), P_j(t_2), \cdots, P_j(t_d))$          ▷ $t \in \mathbb{C}^d$
7:     end for
8:     return $(\sum_{j=1}^{k} v_{j,1}\exp(2\pi \mathbf{i} f_j t_1), \sum_{j=1}^{k} v_{j,2}\exp(2\pi \mathbf{i} f_j t_2), \cdots, \sum_{j=1}^{k} v_{j,3}\exp(2\pi \mathbf{i} f_j t_3))$
9: end procedure
```

The following fact shows an efficient method multi-point evaluation of a polynomial.

Fact 12.70 ([VZGG99, Chapter 10]). *Given a degree-d polynomial $P(t)$, and a set of d locations $\{t_1, t_2, \cdots, t_d\}$. There exists an algorithm that takes $O(d\log^2 d\log\log d)$ time to output the evaluations $\{P(t_1), P(t_2), \cdots, P(t_d)\}$.*

The following lemma shows the time complexity of evaluating a mixed polynomial.

Lemma 12.71 (Time complexity of Algorithm 69). *Procedure* MixedPolynomialEvaluation *in Algorithm 69 runs*

$$O\Big(\sum_{j=1}^{k} \max\{d, \deg(P_j)\}\log^3(\max\{d, \deg(P_j)\})\Big)$$

time.

Proof. Procedure MixedPolynomialEvaluation in Algorithm 69 consists of the following steps:

- In line 5, the for loop repeats k times.
- In line 6, multipoint evaluation of a polynomial takes $d_j \log^c(d_j)$ times by Fact 12.70, where $d_j = \max\{d, \deg(P_j)\}$.

Hence, the total time complexity is

$$\sum_{j=1}^{k} O(d_j \log^3(d_j)) = O\Big(\sum_{j=1}^{k} \max\{d, \deg(P_j)\} \log^3(\max\{d, \deg(P_j)\})\Big).$$

□

12.16.2 Heavy cluster

In this section, we formally define the heavy clusters and show that using "heavy frequencies" only yields a good approximation of the ground-truth signal.

Definition 12.9 (Heavy cluster)**.** Let $x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$ and $\mathcal{N} > 0$. Let the filter H be defined as in Lemma 12.24. Let $\Delta_h = |\mathrm{supp}(\widehat{H})|$. We say a frequency f^* belongs to an $\mathcal{N}$-*heavy cluster* if and only if

$$\int_{f^*-\Delta_h}^{f^*+\Delta_h} |\widehat{H \cdot x^*}(f)|^2 \mathrm{d}f \geq T \cdot \mathcal{N}^2/k.$$

Claim 12.72. *Given* $x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$ *and any* $\mathcal{N} > 0$*. For the set of heavy frequencies:*

$$S^* = \left\{ j \in [k] \middle| \int_{f_j-\Delta_h}^{f_j+\Delta_h} |\widehat{H \cdot x^*}(f)|^2 \mathrm{d}f \geq T \cdot \mathcal{N}^2/k \right\},$$

and the signal $x_{S^*}(t) = \sum_{j \in S^*} v_j e^{2\pi \mathbf{i} f_j t}$*, it holds that*

$$\|x_{S^*} - x^*\|_T^2 \lesssim \mathcal{N}^2.$$

Proof. Let $x_{\overline{S^*}}(t) = \sum_{j \in [k] \setminus S^*} v_j e^{2\pi \mathbf{i} f_j t}$. Then $\|x^* - x_{S^*}\|_T^2 = \|x_{\overline{S^*}}\|_T^2$.

Then, we have that

$$\begin{aligned}
T\|x_{\overline{S^*}}(t)\|_T^2 &= \int_0^T |x_{\overline{S^*}}(t)|^2 \mathrm{d}t \\
&\lesssim \int_0^T |x_{\overline{S^*}}(t) \cdot H(t)|^2 \mathrm{d}t \\
&\leq \int_{-\infty}^{\infty} |x_{\overline{S^*}}(t) \cdot H(t)|^2 \mathrm{d}t \\
&= \int_{-\infty}^{\infty} |\widehat{x}_{\overline{S^*}}(f) * \widehat{H}(f)|^2 \mathrm{d}f \\
&\leq \sum_{j\in[k]\backslash S^*} \int_{f_j-\Delta_h}^{f_j+\Delta_h} |\widehat{x}_{\overline{S^*}}(f) * \widehat{H}(f)|^2 \mathrm{d}f \\
&\leq \sum_{j\in[k]\backslash S^*} T\mathcal{N}^2/k \\
&\leq T\mathcal{N}^2,
\end{aligned}$$

where the first step follows from the definition of the norm, the second step follows from Lemma 12.24 Property V, the third step is straight forward, the forth step follows from Parseval's theorem, the fifth step follows from the definition of $x_{\overline{S^*}}(t)$, the sixth step follows from the definition of heavy frequency, the seventh step is straightforward.

□

12.16.3 Fourier set query

In this section, we present a Fourier set query algorithm such that for a Fourier-polynomial mixed signal, given all of its frequencies, the algorithm can reconstruct the signal very efficiently.

Lemma 12.73. *For $j \in [k]$, given a d_j-degree polynomial $P_j(t)$ and a frequency f_j. Let $x_S(t) = \sum_{j=1}^k P_j(t)\exp(2\pi\mathbf{i}f_jt)$. Given observations of the form $x(t) := x_S(t) + g(t)$ for arbitrary noise $g(t)$ in time duration $t \in [0,T]$. Let $D := \sum_{j=1}^k d_j$.*

Then, there is an algorithm (Procedure SignalEstimation *in Algorithm 70) such that*

- *takes $O(D\log(D))$ samples from $x(t)$,*
- *runs $O(D^\omega \log(D))$ time,*
- *outputs $y(t) = \sum_{j=1}^k P_j'(t)\exp(2\pi \mathbf{i} f_j t)$ with d-degree polynomial $P_j'(t)$, such that with probability at least 0.99, we have*

$$\|y - x_S\|_T^2 \lesssim \|g\|_T^2.$$

Proof. By Lemma 12.69, we have that, for all $t \in [0,T]$, there exist D-Fourier-sparse signals $y_1(t)$ and $x_{S,1}(t)$

$$|y(t) - y_1(t)| \le \varepsilon_1, \tag{12.88}$$

and

$$|x_S(t) - x_{S,1}(t)| \le \varepsilon_1. \tag{12.89}$$

Then, we have that

$$\begin{aligned}\|y(t) - x_S(t)\|_T^2 &\lesssim \|y(t) - y_1(t)\|_T^2 + \|x_S(t) - x_{S,1}(t)\|_T^2 + \|y_1(t) - x_{S,1}(t)\|_T^2 \\ &\lesssim 2\varepsilon_1 + \|y_1(t) - x_{S,1}(t)\|_T^2 \\ &\lesssim \|y_1(t) - x_{S,1}(t)\|_T^2 \end{aligned} \tag{12.90}$$

where the first step follows from $(a+b)^2 \le 2a^2 + 2b^2$, the second step follows from Eq. (12.88) and Eq. (12.89), the third step follows from $\varepsilon_1 \lesssim \|y_1(t) - x_{S,1}(t)\|_T^2$.

We also have that

$$\begin{aligned}\|y_1(t) - x_{S,1}(t)\|_{S,w}^2 &\lesssim \|y_1(t) - y(t)\|_{S,w}^2 + \|x_{S,1}(t) - x_S(t)\|_{S,w}^2 + \|y(t) - x_S(t)\|_{S,w}^2 \\ &\lesssim 2\varepsilon_1 + \|y(t) - x_S(t)\|_{S,w}^2 \\ &\lesssim \|y(t) - x_S(t)\|_{S,w}^2 \end{aligned} \tag{12.91}$$

where the first step follows from $(a+b)^2 \le 2a^2 + 2b^2$, the second step follows from Eq. (12.88) and Eq. (12.89), the third step follows from $\varepsilon_1 \lesssim \|y(t) - x_S(t)\|_{S,w}^2$.

By the definition of $y(t)$ in line 17 in Procedure SignalEstimation of Algorithm 70, we have that

$$\|y(t) - x(t)\|_{S,w}^2 \leq \|x_S(t) - x(t)\|_{S,w}^2 \tag{12.92}$$

We have that

$$\begin{aligned}
\mathbb{E}[\|x - x_S\|_{S,w}^2] =& \ \mathbb{E}\Big[\sum_{i\in[|S|]} w_i|x(t_i) - x_S(t_i)|^2\Big] \\
=& \ \mathbb{E}\Big[\sum_{i\in[|S|]} \frac{1}{2T|S|D(t)}|x(t_i) - x_S(t_i)|^2\Big] \\
=& \sum_{i\in[|S|]} \mathbb{E}_{t_i\sim D(t)}\Big[\frac{1}{2T|S|D(t)}|x(t_i) - x_S(t_i)|^2\Big] \\
=& \ |S| \cdot \int_{-T}^{T} D(t)\frac{1}{2T|S|D(t)}|x(t) - x_S(t)|^2 \mathrm{d}t \\
=& \int_{-T}^{T} \frac{1}{2T}|x(t) - x_S(t)|^2 \mathrm{d}t \\
=& \ \|x(t) - x_S(t)\|_T^2
\end{aligned} \tag{12.93}$$

where the first step follows from the definition of the norm, the second step follows from the definition of w_i, the third step is straightforward, the forth follows from the definition of expectation, the fifth step follows from the definition of the norm.

We have that

$$\begin{aligned}
\|y - x_S\|_T^2 \lesssim& \ \|y_1 - x_{S,1}\|_T^2 \\
\lesssim& \ \|y_1 - x_{S,1}\|_{S,w}^2 \\
\lesssim& \ \|y - x_S\|_{S,w}^2 \\
\lesssim& \ \|y - x\|_{S,w}^2 + \|x - x_S\|_{S,w}^2 \\
\lesssim& \ \|x - x_S\|_{S,w}^2 \\
\lesssim& \ \|x - x_S\|_T^2,
\end{aligned}$$

where the first step follows from Eq. (12.90), the second step follows from Lemma 12.45, the third step follows from Eq. (12.91), the forth step follows from $(a + b)^2 \leq$

$2a^2+2b^2$, the fifth step follows from Eq. (12.92), the sixth step follows from Eq. (12.93) by Markov inequality with probability at least 0.99.

$\square$

12.16.4 High signal-to-noise ratio band approximation

The goal of this section is to prove the following lemma, which roughly states that for the heavy frequencies, it suffices to only reconstruct those in the bins with high SNRs.

Lemma 12.74. *Let $x^*(t) = \sum_{j=1}^k v_j e^{2\pi \mathbf{i} f_j t}$ be the ground-truth signal and $x(t) = x^*(t) + g(t)$ be the noisy observation signal. Let H be defined as in Definition 12.7, $G_{\sigma,b}^{(j)}$ be defined as in Definition 12.2 with (σ, b) such that Large Offset event does not happen. Let $U := \{t_0 \in \mathbb{R} \mid H(t) > 1-\delta_1\ \forall t \in [t_0, t_0+\beta]\}$. Let*

$$S := \Big\{ j \in [k] \ \Big|\ \int_{f_j-\Delta}^{f_j+\Delta} |\widehat{H \cdot x^*}(f)|^2 \mathrm{d}f \geq T\mathcal{N}^2/k \Big\},$$

and $x_S(t) = \sum_{j\in S} v_j e^{2\pi \mathbf{i} f_j t}$.

For $j \in [B]$, let $z_j^*(t) := (x^* \cdot H) * G_{\sigma,b}^{(j)}(t)$ and $z_j(t) = (x \cdot H) * G_{\sigma,b}^{(j)}(t)$. Let $g_j(t) := z_j(t) - z_j^*(t)$. Let

$$S_{g1} := \big\{ j \in [B] \mid \|g_j(t)\|_T^2 \leq c \|z_j^*(t)\|_U^2 \big\}, \tag{12.94}$$

where $c \in (0, 0.001)$ is a small universal constant. Let

$$S_{g2} := \left\{ j \in [B] \ \middle|\ \exists f_0 \in \{f_1, \dots, f_k\}, \text{ and } h_{\sigma,b}(f_0) = j, \text{ and } \int_{f_0-\Delta}^{f_0+\Delta} |\widehat{x^* \cdot H}(f)|^2 \mathrm{d}f \geq T\mathcal{N}^2/k \right\}.$$

Let $S_g = S_{g1} \cap S_{g2}$. Let $S_f := \{j \in [k] \mid h_{\sigma,b}(f_j) \in S_g\} \cap S$ and $x_{S_f}(t) := \sum\limits_{j\in S_f} v_j e^{2\pi \mathbf{i} f_j t}$.

Then, we have

$$\|x_{S_f}(t) - x_S(t)\|_T^2 \lesssim \|g(t)\|_T^2.$$

Proof. By the definition of S and S_f, we have that

$$S_f \subseteq S.$$

Let $[L, R] := U$. We have that for any $f \in S \backslash S_f$, $j = h_{\sigma,b}(f)$,

$$\begin{aligned} \|(g(t) \cdot H(t)) * G^{(j)}_{\sigma,b}(t)\|^2_T &\geq c\|(x^*(t) \cdot H(t)) * G^{(j)}_{\sigma,b}(t)\|^2_U \\ &\geq c\frac{T - k^2(T + L - R)}{R - L}\|(x^*(t) \cdot H(t)) * G^{(j)}_{\sigma,b}(t)\|^2_T \\ &\geq O(c) \cdot \|(x^*(t) \cdot H(t)) * G^{(j)}_{\sigma,b}(t)\|^2_T, \end{aligned} \tag{12.95}$$

where the first step follows from Eq. (12.94), the second step follows from Lemma 12.51, the third step follows from the Lemma 12.25.

Let $\mathcal{T} = S \backslash S_f$. And for $j \in [B]$, let

$$\mathcal{T}_j := \begin{cases} \{i \in S \mid h_{\sigma,b}(f_i) = j\}, & \forall j \in [B] \backslash S_g, \\ \emptyset, & \text{otherwise.} \end{cases}$$

It is easy to see that

$$\mathcal{T} = \bigcup_{i=1}^{B} \mathcal{T}_i.$$

Moreover, by Lemma 12.9 Property I and III, the definition of $\mathcal{T}_j$ and $\widehat{G}^{(j)}_{\sigma,b}(f)$, and the Large Offset event not happening, we have that for any $f \in \mathrm{supp}(\widehat{x}_{\mathcal{T}_j} * \widehat{H})$,

$$\widehat{G}^{(j)}_{\sigma,b}(f) \geq 1 - \frac{\delta}{k}, \tag{12.96}$$

where $x_{\mathcal{T}_j} = \sum_{i \in \mathcal{T}_j} v_i e^{2\pi \mathbf{i} f_i t}$ and $\widehat{x}_{\mathcal{T}_j}$ is its Fourier transform.

Then, we have that

$$\begin{aligned}
& T\|(x^*(t)\cdot H(t)) * G_{\sigma,b}^{(j)}(t)\|_T^2 \\
= & \int_0^T |(x^*(t)\cdot H(t)) * G_{\sigma,b}^{(j)}(t)|^2 \mathrm{d}t \\
\gtrsim & \int_{-\infty}^{\infty} |(x^*(t)\cdot H(t)) * G_{\sigma,b}^{(j)}(t)|^2 \mathrm{d}t \\
= & \int_{-\infty}^{\infty} |(\widehat{x}^*(f) * \widehat{H}(f))\cdot \widehat{G}_{\sigma,b}^{(j)}(f)|^2 \mathrm{d}f \\
= & \int_{-\infty}^{\infty} |(\widehat{x}_{\mathcal{T}_j}(f) * \widehat{H}(f))\cdot \widehat{G}_{\sigma,b}^{(j)}(f)|^2 \mathrm{d}f + \int_{-\infty}^{\infty} |(\widehat{x}_{[k]\setminus\mathcal{T}_j}(f) * \widehat{H}(f))\cdot \widehat{G}_{\sigma,b}^{(j)}(f)|^2 \mathrm{d}f \\
\geq & \int_{-\infty}^{\infty} |(\widehat{x}_{\mathcal{T}_j}(f) * \widehat{H}(f))\cdot \widehat{G}_{\sigma,b}^{(j)}(f)|^2 \mathrm{d}f \\
\gtrsim & \int_{-\infty}^{\infty} |\widehat{x}_{\mathcal{T}_j}(f) * \widehat{H}(f)|^2 \mathrm{d}f
\end{aligned} \tag{12.97}$$

where the first step follows from the definition of the norm, the second step follows from Lemma 12.30, third step follows from Parseval's theorem, the forth step follows from the Large Offset event not happening and the definition of $\mathcal{T}_j$, the fifth step is straight forward, the sixth step follows from Eq. (12.96).

Thus, we have that

$$\begin{aligned}
& T\|x_{S_f}(t) - x_S(t)\|_T^2 \\
= & T\|x_{\mathcal{T}}(t)\|_T^2 \\
\lesssim & T\|x_{\mathcal{T}}(t) \cdot H(t)\|_T^2 \\
= & \int_0^T |x_{\mathcal{T}}(t) \cdot H(t)|^2 \mathrm{d}t \\
\leq & \int_{-\infty}^{\infty} |x_{\mathcal{T}}(t) \cdot H(t)|^2 \mathrm{d}t \\
= & \int_{-\infty}^{\infty} |\widehat{x}_{\mathcal{T}}(f) * \widehat{H}(f)|^2 \mathrm{d}f \\
= & \sum_{j=1}^{B} \int_{-\infty}^{\infty} |\widehat{x}_{\mathcal{T}_j}(f) * \widehat{H}(f)|^2 \mathrm{d}f \\
\lesssim & \sum_{j \in [B] \backslash S_g} T\|(x^*(t) \cdot H(t)) * G_{\sigma,b}^{(j)}(t)\|_T^2 \\
\lesssim & \sum_{j \in [B] \backslash S_g} T\|(g(t) \cdot H(t)) * G_{\sigma,b}^{(j)}(t)\|_T^2
\end{aligned} \tag{12.98}$$

where the first step follows from the definition of $\mathcal{T}$, the second step follows from $x_{\mathcal{T}}$ is a k-Fourier-sparse signal and Lemma 12.24 Property V, the third step follows from the definition of the norm, the forth step is straight forward, the fifth step follows from Parseval's theorem, the sixth step follows from the definition of $\mathcal{T}_j$ and the Large Offset event not happened, the seventh step follows from Eq. (12.97), the eighth step follows from Eq. (12.95).

Eq. (12.98) can be upper bounded by the summation over all bins, which can be further upper bounded as follows:

$$\begin{aligned}
& \sum_{j\in[B]} T \cdot \|(g(t)\cdot H(t)) * G_{\sigma,b}^{(j)}(t)\|_T^2 \\
= & \sum_{j\in[B]} \int_0^T |(g(t)\cdot H(t)) * G_{\sigma,b}^{(j)}(t)|^2 \mathrm{d}t \\
\leq & \sum_{j\in[B]} \int_{-\infty}^{\infty} |(g(t)\cdot H(t)) * G_{\sigma,b}^{(j)}(t)|^2 \mathrm{d}t \\
\leq & \sum_{j\in[B]} \int_{-\infty}^{\infty} |(\widehat{g}(f) * \widehat{H}(f)) \cdot \widehat{G}_{\sigma,b}^{(j)}(f)|^2 \mathrm{d}f \\
= & \int_{-\infty}^{\infty} |(\widehat{g}(f) * \widehat{H}(f))|^2 \cdot \sum_{j\in[B]} |\widehat{G}_{\sigma,b}^{(j)}(f)|^2 \mathrm{d}f \\
\lesssim & \int_{-\infty}^{\infty} |(\widehat{g}(f) * \widehat{H}(f))|^2 \mathrm{d}f \\
= & \int_{-\infty}^{\infty} |(g(t)\cdot H(t))|^2 \mathrm{d}t \\
= & \int_0^T |(g(t)\cdot H(t))|^2 \mathrm{d}t \\
\lesssim & \int_0^T |g(t)|^2 \mathrm{d}t \\
= & T\|g(t)\|_T^2
\end{aligned} \tag{12.99}$$

where the first step follows from the definition of the norm, the second step is straightforward, the third step follows from Parseval's theorem, the forth step is straightforward, the fifth step follows from Lemma 12.11, the sixth step follows from Parseval's theorem, the seventh step follows from $g(t) = 0, \forall t \in \mathbb{R}\backslash[0,T]$, the eighth step follows from Lemma 12.24 Property I, II, the ninth step follows from the definition of the norm.

Therefore, we get that

$$\begin{aligned}
& T\|x_{S_f}(t) - x_S(t)\|_T^2 \\
\lesssim & \sum_{j \in [B] \setminus S_g} T\|(g(t) \cdot H(t)) * G_{\sigma,b}^{(j)}(t)\|_T^2 \\
\leq & \sum_{j \in [B]} T\|(g(t) \cdot H(t)) * G_{\sigma,b}^{(j)}(t)\|_T^2 \\
\lesssim & T\|g(t)\|_T^2,
\end{aligned}$$

where the first step follows from Eq. (12.98), the second step is straight forward, the third step follows from Eq. (12.99).

The lemma is then proved.

□

12.16.5 Fourier interpolation with constant success probability

In this section, we give an algorithm for Fourier interpolation by combining our frequency estimation algorithm with a signal estimation algorithm. However, it only succeeds with a constant probability.

Theorem 12.75. *Let $x(t) = x^*(t) + g(t)$, where $x^*(t) \in \mathcal{F}_{k,F}$ and $g(t)$ is arbitrary noise. Given samples of x over $[0, T]$, there is an algorithm (Procedure* CONSTANTPROBFOURIERINTERPOLATION *in Algorithm 71) that uses*

$$O(k^4 \log^3(k) \log^2(1/\delta_1) \log(\log(1/\delta_1)) \log(FT) \log(\log(FT)))$$

samples, runs in

$$O(k^{4\omega} \log^{2\omega+1}(k) \log^{2\omega}(1/\delta_1) \log(\log(1/\delta_1)) \log(FT) \log(\log(FT)))$$

time, and outputs an $O(k^4 \log^4(k/\delta))$-Fourier-sparse signal $y(t)$ such that with probability at least 0.6,

$$\|y - x^*\|_T \lesssim \|g\|_T + \delta\|x^*\|_T.$$

Proof. Let $\mathcal{N}^2 := \|g(t)\|_T^2 + \delta\|x^*(t)\|_T^2$ be the noisy level of the observation signal.

Heavy-clusters approximation. Let S be the set of heavy frequencies:

$$S = \Big\{ j \in [k] \Big| \int_{f_j - \Delta_h}^{f_j + \Delta_h} |\widehat{H \cdot x^*}(f)|^2 \mathrm{d} f \geq T \cdot \mathcal{N}^2 / k \Big\},$$

where $\Delta_h = |\mathrm{supp}(\widehat{H})|$, and let $x_S(t) = \sum_{j \in S} v_j e^{2\pi \mathbf{i} f_j t}$. By Claim 12.72, we have

$$\|x_S - x^*\|_T \lesssim \mathcal{N}, \tag{12.100}$$

which implies that it suffices to reconstruct x_S, instead of x^*.

Frequency estimation. Conditioning on Large Offset event not happening, which holds with probability at least 0.6 by Lemma 12.16, let $S_f \subseteq S$ be defined as in Lemma 12.74 and $x_{S_f}(t) = \sum_{j \in S_f} v_j e^{2\pi \mathbf{i} f_j t}$. By Lemma 12.74, we have

$$\|x_{S_f}(t) - x_S(t)\|_T^2 \lesssim \|g(t)\|_T^2. \tag{12.101}$$

Furthermore, by Theorem 12.61, there is an algorithm that outputs a set of frequencies $L \subset \mathbb{R}$ of size B such that with probability at least $1 - 2^{-\Omega(k)}$, for any $j \in S_f$, there exists an $\widetilde{f} \in L$ such that,

$$|f_j - \widetilde{f}| \lesssim \Delta.$$

Fourier-polynomial mixed signal approximation. We define a map $p : \mathbb{R} \to L$ as follows:

$$p(f) := \arg\min_{\widetilde{f} \in L} |f - \widetilde{f}| \quad \forall f \in \mathbb{R}.$$

Then, $x_{S_f}(t)$ can be expressed as

$$\begin{aligned} x_{S_f}(t) = & \sum_{j \in S_f} v_j e^{2\pi \mathbf{i} f_j t} \\ = & \sum_{j \in S_f} v_j e^{2\pi \mathbf{i} \cdot p(f_j) t} \cdot e^{2\pi \mathbf{i} \cdot (f_j - p(f_j)) t} \\ = & \sum_{\widetilde{f} \in L} e^{2\pi \mathbf{i} \widetilde{f} t} \cdot \sum_{j \in S_f :\ p(f_j) = \widetilde{f}} v_j e^{2\pi \mathbf{i} (f_j - \widetilde{f}) t}, \end{aligned}$$

where the first step follows from the definition of x_{S_f}, the last step follows from interchanging the summations.

For each $\widetilde{f}_i \in L$, by Lemma 12.68 with $x^* = x_{S_f}$, there exists a degree $d = O(T\Delta + k^3 \log k + k \log 1/\delta)$ polynomial $P_i(t)$ such that,

$$\Big\|x_{S_f}(t) - \sum_{\widetilde{f}_i \in L} e^{2\pi \mathbf{i} \widetilde{f}_i t} P_i(t)\Big\|_T \leq \sqrt{\delta} \|x_{S_f}(t)\|_T \tag{12.102}$$

Reconstructing the polynomials. Define the following function family:

$$\mathcal{F} := \mathrm{span}\Big\{e^{2\pi \mathbf{i} \widetilde{f} t} \cdot t^j \mid \widetilde{f} \in L, j \in \{0, 1, \ldots, d\}\Big\}.$$

Note that $\sum_{\widetilde{f}_i \in L} e^{2\pi \mathbf{i} \widetilde{f}_i t} P_i(t) \in \mathcal{F}$.

Let $D := d \cdot |L|$. By Lemma 12.73, there is an algorithm that runs in $O(\varepsilon^{-1} D^\omega \log^3(D) \log(1/\rho))$-time using $O(\varepsilon^{-1} D \log^3(D) \log(1/\rho))$ samples, and outputs $y'(t) \in \mathcal{F}$ such that, with probability $1 - \rho$,

$$\Big\|y'(t) - \sum_{\widetilde{f}_i \in L} e^{2\pi \mathbf{i} \widetilde{f}_i t} P_i(t)\Big\|_T \leq (1 + \varepsilon)\Big\|x(t) - \sum_{\widetilde{f}_i \in L} e^{2\pi \mathbf{i} \widetilde{f}_i t} P_i(t)\Big\|_T \tag{12.103}$$

Thus, we have that

$$\begin{aligned}\Big\|y'(t) - \sum_{\widetilde{f}_i \in L} e^{2\pi \mathbf{i} \widetilde{f}_i t} P_i(t)\Big\|_T &\lesssim \Big\|\sum_{\widetilde{f}_i \in L} e^{2\pi \mathbf{i} \widetilde{f}_i t} P_i(t) - x^*(t)\Big\|_T + \|x(t) - x^*(t)\|_T \\ &= \Big\|\sum_{\widetilde{f}_i \in L} e^{2\pi \mathbf{i} \widetilde{f}_i t} P_i(t) - x^*(t)\Big\|_T + \|g(t)\|_T, \end{aligned} \tag{12.104}$$

where the first step follows from triangle inequality, the second step follows from the definition of $g(t)$.

For the first term, we have that

$$\begin{aligned}
\Big\| \sum_{\widetilde{f}_i \in L} e^{2\pi \mathbf{i} \widetilde{f}_i t} P_i(t) - x^*(t) \Big\|_T &\lesssim \Big\| \sum_{\widetilde{f}_i \in L} e^{2\pi \mathbf{i} \widetilde{f}_i t} P_i(t) - x_{S_f}(t) \Big\|_T + \|x_{S_f}(t) - x^*(t)\|_T \\
&\lesssim \sqrt{\delta} \|x_{S_f}(t)\|_T + \|x_{S_f}(t) - x^*(t)\|_T \\
&\leq \sqrt{\delta} (\|x_{S_f}(t) - x^*(t)\|_T + \|x^*(t)\|_T) + \|x_{S_f}(t) - x^*(t)\|_T \\
&\lesssim \|x_{S_f}(t) - x^*(t)\|_T + \sqrt{\delta} \|x^*(t)\|_T \\
&\leq \|x_{S_f}(t) - x_S(t)\|_T + \|x_S(t) - x^*(t)\|_T + \sqrt{\delta} \|x^*(t)\|_T \\
&\lesssim \|x_{S_f}(t) - x_S(t)\|_T + \mathcal{N} + \sqrt{\delta} \|x^*(t)\|_T \\
&\lesssim \|g(t)\|_T + \mathcal{N} + \sqrt{\delta} \|x^*(t)\|_T, \qquad (12.105)
\end{aligned}$$

where the first step follows from triangle inequality, the second step follows from Eq. (12.102), the third step follows from triangle inequality, the forth step follows is straightforward, the fifth step follows from triangle inequality, the sixth step follows from Eq. (12.100), and the last step follows from Eq. (12.101).

Hence, we get that

$$\begin{aligned}
\Big\| y'(t) - \sum_{\widetilde{f}_i \in L} e^{2\pi \mathbf{i} \widetilde{f}_i t} P_i(t) \Big\|_T &\leq \Big\| \sum_{\widetilde{f}_i \in L} e^{2\pi \mathbf{i} \widetilde{f}_i t} P_i(t) - x^*(t) \Big\|_T + \|g(t)\|_T \\
&\lesssim \|g(t)\|_T + \mathcal{N} + \sqrt{\delta} \|x^*(t)\|_T \qquad (12.106)
\end{aligned}$$

where the first step follows from Eq. (12.104), the second step follows from Eq. (12.105).

Transforming back to Fourier-sparse signal. By Lemma 12.69, we have that there is a $O(kd)$-Fourier-sparse signal $y(t)$, such that

$$\|y(t) - y'(t)\|_T \leq \delta' \qquad (12.107)$$

where $\delta' > 0$ is any positive real number. Thus, $y(t)$ can be arbitrarily close to $y'(t)$. Moreover, the sparsity of $y(t)$ is

$$O(kd) = O(k \cdot (T\Delta + k^3 \log k + k \log 1/\delta)) = O(k^4 \log^4(k/\delta)),$$

which follows from Lemma 12.24 Property III:

$$\Delta = k\Delta_h = k|\mathrm{supp}(\widehat{H})| = O(k^3 \log^2(k) \log^2(1/\delta_1)/T).$$

Moreover, we take

$$\mathcal{N} = \sqrt{\|g\|_T^2 + \delta\|x^*\|_T^2} \leq \|g\|_T + \sqrt{\delta}\|x^*\|_T. \tag{12.108}$$

Therefore, the total approximation error can be bounded as follows:

$$\begin{aligned}
&\|y(t) - x^*(t)\|_T \\
\leq\ & \|y(t) - y'(t)\|_T + \Big\|y'(t) - \sum_{\widetilde{f}_i \in L} e^{2\pi \mathbf{i} \widetilde{f}_i t} P_i(t)\Big\|_T + \Big\|\sum_{\widetilde{f}_i \in L} e^{2\pi \mathbf{i} \widetilde{f}_i t} P_i(t) - x^*(t)\Big\|_T \\
\lesssim\ & \Big\|y'(t) - \sum_{\widetilde{f}_i \in L} e^{2\pi \mathbf{i} \widetilde{f}_i t} P_i(t)\Big\|_T + \Big\|\sum_{\widetilde{f}_i \in L} e^{2\pi \mathbf{i} \widetilde{f}_i t} P_i(t) - x^*(t)\Big\|_T \\
\lesssim\ & \Big\|y'(t) - \sum_{\widetilde{f}_i \in L} e^{2\pi \mathbf{i} \widetilde{f}_i t} P_i(t)\Big\|_T + \mathcal{N} + \|g(t)\|_T + \sqrt{\delta}\|x^*(t)\|_T \\
\lesssim\ & \mathcal{N} + \|g(t)\|_T + \sqrt{\delta}\|x^*(t)\|_T \\
\lesssim\ & \|g(t)\|_T + \sqrt{\delta}\|x^*(t)\|_T,
\end{aligned} \tag{12.109}$$

where the first step follows from triangle inequality, the second step follows from Eq. (12.107), the third step follows from Eq. (12.105), the forth step follows from Eq. (12.106), the fifth step follows from $\mathcal{N} = \sqrt{\|g\|_T^2 + \delta\|x^*\|_T^2} \leq \|g\|_T + \sqrt{\delta}\|x^*\|_T$.

The correctness then follows by re-scaling δ.

The running time of the algorithm follows from Lemma 12.76, and the sample complexity follows from Lemma 12.77.

The theorem is then proved.

$\square$

Lemma 12.76 (Running time of Algorithm 71)**.** *Procedure* ConstantProbFourierInterpolation *in Algorithm 71 runs in*

$$O(k^{4\omega} \log^{2\omega+1}(k) \log^{2\omega}(1/\delta_1) \log(\log(1/\delta_1)) \log(FT) \log(\log(FT)))$$

times.

Proof. Procedure ConstantProbFourierInterpolation in Algorithm 71 consists of the following two steps:

- Line 2 calls Procedure FrequencyEstimationX. By Theorem 12.61, it runs in
$$O(k^2 \log(k) \log(k/\delta_1) \log(FT) \log(\log(FT)))$$
time.
- Line 3 calls Procedure SignalEstimation. By Lemma 12.73, it runs in
$$O(\varepsilon^{-1} D^{\omega} \log(D) \log(1/\rho))$$
time, where ε, ρ are set to be universal constants and $D = B \cdot d$.

Following from the setting in the algorithm, we have that
$$\begin{aligned} B &= O(k), \\ d &= O(\Delta T + k^3 \log k + k \log 1/\delta). \end{aligned}$$

By Lemma 12.24 Property III, we have that
$$\Delta = k\Delta_h = k|\mathrm{supp}(\widehat{H}(f))| = O(k^3 \log^2(k) \log^2(1/\delta_1)/T).$$

As a result, we have that
$$D = B \cdot d = O(k^4 \log^2(k) \log^2(1/\delta_1)) \tag{12.110}$$

Thus, the time complexity of Procedure ConstantProbFourierInterpolation in Algorithm 71 is
$$\begin{aligned} & O(k^2 \log(k) \log(k/\delta_1) \log(FT) \log(\log(FT)/\rho_1)) + O(\varepsilon^{-1} D^{\omega} \log(D) \log(1/\rho)) \\ \leq\ & O(k^2 \log(k) \log(k/\delta_1) \log(FT) \log(\log(FT)/\rho_1)) \\ & \quad + O(\varepsilon^{-1} (k^4 \log^2(k) \log^2(1/\delta_1))^{\omega} \log(k^4 \log^2(k) \log^2(1/\delta_1)) \log(1/\rho)) \\ \leq\ & O(k^{4\omega} \log^{2\omega+1}(k) \log^{2\omega}(1/\delta_1) \log(\log(1/\delta_1)) \log(FT) \log(\log(FT))) \end{aligned}$$

where the first step follows from Eq. (12.110), the second step follows from $\varepsilon = O(1), \rho = O(1), \rho_1 = O(1)$.

□

Lemma 12.77 (Sample complexity of Algorithm 71)**.** *Procedure* CONSTANTPROBFOURIERINTERPOLATION *in Algorithm 71 takes*

$$O(k^4 \log^3(k) \log^2(1/\delta_1) \log(\log(1/\delta_1)) \log(FT) \log(\log(FT)))$$

samples.

Proof. The sample complexity of each steps of Procedure CONSTANTPROBFOURIERINTERPOLATION in Algorithm 71 is as follows:

- Line 2 calls Procedure FREQUENCYESTIMATIONX. By Theorem 12.61, it takes

  $$O(k^2 \log(k) \log(k/\delta_1) \log(FT) \log(\log(FT)))$$

  samples.

- Line 3 calls Procedure SIGNALESTIMATION. By Lemma 12.73, it takes in

  $$O(\varepsilon^{-1} D \log(D) \log(1/\rho))$$

  samples, where ε, ρ are set to be a universal constant and $D = B \cdot d$.

  By Eq. (12.110), we have

  $$D = O(k^4 \log^2(k) \log^2(1/\delta_1)).$$

Thus, the sample complexity of Procedure CONSTANTPROBFOURIERINTERPOLATION in Algorithm 71 is

$$\begin{aligned}
& O(k^2 \log(k) \log(k/\delta_1) \log(FT) \log(\log(FT)/\rho_1)) + O(\varepsilon^{-1} D \log(D) \log(1/\rho)) \\
\leq\ & O(k^2 \log(k) \log(k/\delta_1) \log(FT) \log(\log(FT)/\rho_1)) \\
& \quad + O(\varepsilon^{-1} (k^4 \log^2(k) \log^2(1/\delta_1)) \log(k^4 \log^2(k) \log^2(1/\delta_1)) \log(1/\rho)) \\
\leq\ & O(k^4 \log^3(k) \log^2(1/\delta_1) \log(\log(1/\delta_1)) \log(FT) \log(\log(FT)))
\end{aligned}$$

where the first step follows from Eq. (12.110), the second step follows from $\varepsilon = O(1), \rho = O(1), \rho_1 = O(1)$.

□

12.16.6 Min-of-medians signal estimator

In this section, we propose a "min-of-medians" estimator for signals that can exponentially boost the success probability.

Lemma 12.78. *Let $R_p \in \mathbb{N}$. For each $i \in [R_p]$, let $y_i(t)$ be a signal independently sampled from some distribution such that with probability at least* 0.9*,*

$$\|y_i(t) - x^*(t)\|_T^2 \lesssim \|g(t)\|_T^2.$$

Let $y(t) := y_{j^*}(t)$ where

$$j^* := \arg\min_{j \in [R_p]} \operatorname*{median}_{i \in [R_p]} \|y_j(t) - y_i(t)\|_T^2.$$

Then, with probability at least $1 - 2^{-\Omega(R_p)}$,

$$\|y(t) - x^*(t)\|_T^2 \lesssim \|g(t)\|_T^2.$$

Proof. Let $S = \{i \mid \|y_i(t) - x^*(t)\|_T^2 \lesssim \|g(t)\|_T^2\}$. By the Chernoff bound, we have that

$$\Pr[|S| \geq 3/4R_p] \geq 1 - 2^{-\Omega(R_p)}.$$

For the ease of discussion, we suppose $|S| \geq 3/4R_p$ holds in the following proof.

Fix any $j \in S$. Then, for any $q \in S$, we have that

$$\|y_j(t) - y_q(t)\|_T \leq \|y_j(t) - x^*(t)\|_T + \|x^*(t) - y_q(t)\|_T \lesssim \|g(t)\|_T^2, \tag{12.111}$$

where the first step follows from triangle inequality, the second step follows from the definition of S.

In other words, there are at least $|S| \geq (3/4)R_p$ elements such that Eq. (12.111) holds. By the definition of median, we get that

$$\underset{i\in[R_p]}{\mathrm{median}} \|y_j(t) - y_i(t)\|_T^2 \lesssim \|g(t)\|_T^2. \tag{12.112}$$

By definition of $y(t)$, we have that,

$$\underset{i\in[R_p]}{\mathrm{median}} \|y(t) - y_i(t)\|_T^2 \leq \underset{i\in[R_p]}{\mathrm{median}} \|y_j(t) - y_i(t)\|_T^2 \lesssim \|g(t)\|_T^2, \tag{12.113}$$

where the first step follows from the definition of $y(t)$, the second step follows from Eq. (12.112).

By the definition of median, we know that there are $R_p/2$ elements $r \in [R_p]$ such that

$$\|y(t) - y_r(t)\|_T^2 \leq \underset{i\in[R_p]}{\mathrm{median}} \|y(t) - y_i(t)\|_T^2 \lesssim \|g(t)\|_T^2, \tag{12.114}$$

where the last step follows from Eq. (12.113). Since $|S| \geq (3/4)R_p > (1/2)R_p$, there must exists an $r \in S$ such that Eq. (12.114) holds.

As a result, we have that

$$\begin{aligned}\|y(t) - x^*(t)\|_T^2 &\leq \|y(t) - y_r(t)\|_T^2 + \|y_r(t) - x^*(t)\|_T^2 \\ &\lesssim \|g(t)\|_T^2 + \|y_r(t) - x^*(t)\|_T^2 \\ &\lesssim \|g(t)\|_T^2,\end{aligned}$$

where the first step follows from triangle inequality, the second step follows from Eq. (12.113), the third step follows from the definition of S.

The lemma is then proved. $\square$

One potential issue in applying the min-of-median signal estimator is that, we may not be able to compute the distances $\|y_i(t) - y_j(t)\|_T^2$ exactly, but we can only estimate then with high accuracy. Therefore, we show that our estimator is robust with respect to approximated distances.

We first show a fact about the approximation of min and median.

Fact 12.79. *Let $x_1, \cdots, x_n \in \mathbb{R}_+$, and $y_1, \cdots, y_n \in \mathbb{R}_+$ such that for any $i \in [n]$, $y_i \in [\alpha \cdot x_i, \beta \cdot x_i]$. Then, we have:*

- $\min_{i\in[n]} y_i \in \left[\alpha \cdot \min_{i\in[n]} x_i, \beta \cdot \min_{i\in[n]} x_i\right]$.
- $\operatorname{median}_{i\in[n]} y_i \in \left[\alpha \cdot \operatorname{median}_{i\in[n]} x_i, \beta \cdot \operatorname{median}_{i\in[n]} x_i\right]$.

Proof. **Part 1:** Let $i^* = \arg\min_{i\in[n]} y_i$. We have that

$$y_{i^*} \geq \alpha \cdot x_{i^*} \geq \alpha \cdot \min_{i\in[n]} x_i.$$

Let $j^* = \arg\min_{j\in[n]} x_j$. We have that

$$\min_{j\in[n]} y_j \leq y_{j^*} \leq \beta \cdot x_{j^*} = \beta \cdot \min_{j\in[n]} x_j,$$

Hence,

$$\min_{i\in[n]} y_i \in \left[\alpha \cdot \min_{i\in[n]} x_i, \beta \cdot \min_{i\in[n]} x_i\right].$$

Part 2: For any $x_j \leq \operatorname{median}_{i\in[n]} x_i$, we have that

$$y_j \leq \beta \cdot x_j \leq \beta \cdot \operatorname{median}_{i\in[n]} x_i.$$

Thus,

$$|\{j \in [n] \mid y_j \leq \beta \cdot \operatorname{median}_{i\in[n]} x_i\}| \geq n/2,$$

which implies that

$$\operatorname{median}_{i\in[n]} y_i \leq \beta \cdot \operatorname{median}_{i\in[n]} x_i.$$

For any $x_j \geq \operatorname*{median}_{i\in[n]} x_i$, we have that

$$y_j \geq \alpha \cdot x_j \geq \alpha \cdot \operatorname*{median}_{i\in[n]} x_i.$$

Thus,

$$|\{j \in [n] \mid y_j \geq \alpha \cdot \operatorname*{median}_{i\in[n]} x_i\}| \geq n/2,$$

which implies that

$$\operatorname*{median}_{i\in[n]} y_i \geq \alpha \cdot \operatorname*{median}_{i\in[n]} x_i.$$

As a result,

$$\operatorname*{median}_{i\in[n]} y_i \in \Big[\alpha \cdot \operatorname*{median}_{i\in[n]} x_i, \beta \cdot \operatorname*{median}_{i\in[n]} x_i\Big].$$

□

The following lemma shows that our min-and-median estimator can still exponentially boost the success probability given access to approximated distances.

Lemma 12.80 (Robust min-of-median signal estimator)**.** *Let $R_p \in \mathbb{N}$. For each $i \in [R_p]$, let $y_i(t)$ be a signal independently sampled from some distribution such that with probability at least* 0.9*,*

$$\|y_i(t) - x^*(t)\|_T^2 \lesssim \|g(t)\|_T^2.$$

Given $d \in \mathbb{R}_+^{R_p \times R_p}$ such that for any $i, j \in [R_p]$,

$$d_{i,j} \in \Big[\alpha \cdot \|y_i(t) - y_j(t)\|_T^2, \beta \cdot \|y_i(t) - y_j(t)\|_T^2\Big].$$

Let $y(t) := y_{j^*}(t)$ where

$$j^* := \arg\min_{j\in[R_p]} \operatorname*{median}_{i\in[R_p]} d_{j,i}.$$

Then, we have that, with probability at least $1 - 2^{-\Omega(R_p)}$,

$$\|y(t) - x^*(t)\|_T^2 \lesssim \frac{\beta}{\alpha}\|g(t)\|_T^2.$$

Proof. Let $S = \{i \mid \|y_i(t) - x^*(t)\|_T^2 \lesssim \|g(t)\|_T^2\}$. By the Chernoff bound, we have that

$$\Pr[|S| \geq 3/4R_p] \geq 1 - 2^{-\Omega(R_p)}.$$

For the ease of discussion, we suppose $|S| \geq 3/4R_p$ holds in the following proof.

Fix any $i^* \in S$. For any $q \in S$, we have that

$$\|y_{i^*}(t) - y_q(t)\|_T \leq \|y_{i^*}(t) - x^*(t)\|_T + \|x^*(t) - y_q(t)\|_T \lesssim \|g(t)\|_T^2, \tag{12.115}$$

where the first step follows from triangle inequality, the second step follows from the definition of S.

By the definition of median, since $|S| > R_p/2$, we know that

$$\operatorname*{median}_{i \in [R_p]} \|y_{i^*}(t) - y_i(t)\|_T^2 \lesssim \|g(t)\|_T^2. \tag{12.116}$$

Then, we have that

$$\begin{aligned} \operatorname*{median}_{i \in [R_p]} d_{j^*,i} &\leq \operatorname*{median}_{i \in [R_p]} d_{i^*,i} \\ &\leq \beta \cdot \operatorname*{median}_{i \in [R_p]} \|y_{i^*}(t) - y_i(t)\|_T^2 \\ &\lesssim \beta \cdot \|g(t)\|_T^2, \end{aligned} \tag{12.117}$$

where the first step follows from the definition of j^*, the second step follows from Fact 12.79, the third step follows from Eq. (12.116).

Since $|S| > R_p/2$, by the definition of median, there must exists an $r \in S$ such that

$$d_{j^*,r} \leq \operatorname*{median}_{i \in [R_p]} d_{j^*,i} \lesssim \beta \cdot \|g(t)\|_T^2. \tag{12.118}$$

As a result, we have that

$$\begin{aligned}\|y(t)-x^*(t)\|_T^2 &\leq \|y(t)-y_r(t)\|_T^2 + \|y_r(t)-x^*(t)\|_T^2 \\ &\leq \frac{1}{\alpha} v_{j^*,r} + \|y_r(t)-x^*(t)\|_T^2 \\ &\lesssim \frac{\beta}{\alpha}\|g(t)\|_T^2 + \|y_r(t)-x^*(t)\|_T^2 \\ &\lesssim \frac{\beta}{\alpha}\|g(t)\|_T^2,\end{aligned}$$

where the first step follows from triangle inequality, the second step follows from the definition of d, the third step follows from Eq. (12.118), the forth step follows from Eq. (12.117), the fifth step follows from $r \in S$.

The proof of the lemma is then completed. $\square$

12.16.7 Main algorithm for Fourier interpolation

In this section, we present our main theorem—a time and sample efficient Fourier interpolation algorithm with high success probability. The pseudocode is given in Algorithm 71.

Theorem 12.81 (Main Fourier interpolation algorithm). *Let $x(t) = x^*(t) + g(t)$, where x^* is k-Fourier-sparse signal with frequencies in $[-F, F]$. Given samples of x over $[0, T]$, there is an algorithm (Procedure* HighProbFourierInterpolation*) uses*

$$O(k^4 \log^6(k/\delta)\log(FT)\log(\log(FT))\log(1/\rho))$$

samples, runs in

$$O(k^{4\omega} \log^{4\omega+2}(k/\delta)\log(FT)\log(\log(FT))\log^5(1/\rho))$$

time, and outputs an $O(k^4\log^4(k/\delta))$-Fourier-sparse signal $y(t)$ such that with probability at least $1-\rho$,

$$\|y - x^*\|_T \lesssim \|g\|_T + \delta\|x^*\|_T.$$

Algorithm 70 Signal estimation algorithm

```
procedure WeightedSketch($m, k, T$)
    $c \leftarrow \Theta(\log(k)^{-1})$
    $D(t)$ is defined as follows:
        $$D(t) \leftarrow \begin{cases} c \cdot (1 - |t/T|)^{-1} T^{-1}, & \text{for } |t| \leq T(1 - 1/k), \\ c \cdot k T^{-1}, & \text{for } |t| \in [T(1 - 1/k), T]. \end{cases}$$
    $S_0 \leftarrow m$ i.i.d. samples from $D$
    for $t \in S_0$ do
        $w_t \leftarrow \frac{1}{2T \cdot |S_0| \cdot D(t)}$
    end for
    Set a new distribution $D'(t) \leftarrow w_t / \sum_{t' \in S_0} w_{t'}$ for all $t \in S_0$
    return $D'$
end procedure
procedure SignalEstimation($x, F, T, L$)
    $\{f_1, f_2, \cdots, f_B\} \leftarrow L$                                  ▷ $L \in \mathbb{R}^B$
    $d \leftarrow O(\Delta T + k^3 \log k + k \log 1/\delta)$
    $s, \{t_1, t_2, \cdots, t_s\}, w \leftarrow$ WeightedSketch($O(Bd \log(Bd)), Bd, T$)   ▷ $w \in \mathbb{R}^s$
    $A_{i, B \cdot j_2 + j_1} \leftarrow t_i^{j_2} \cdot \exp(2\pi \mathbf{i} f_{j_1} t_i)$, $A \in \mathbb{C}^{s \times B}$
    $b \leftarrow (x(t_1), x(t_2), \cdots, x(t_s))^\top$
    Solving the following weighted linear regression
        $$v' \leftarrow \arg\min_{v' \in \mathbb{C}^{Bd}} \| \sqrt{w} \circ (Av' - b) \|_2.$$
    return $y(t) \leftarrow \sum_{j_1=1}^{B} \sum_{j_2=1}^{d} v'_{B \cdot j_2 + j_1} \cdot t^{j_2} \cdot \exp(2\pi \mathbf{i} f'_{j_1} t)$.
end procedure
```

Proof. We first prove the correctness of the algorithm.

Let

$$D(t) := \begin{cases} c \cdot (1 - |t/T|)^{-1} T^{-1}, & \text{for } |t| \leq T(1 - 1/k), \\ c \cdot k T^{-1}, & \text{for } |t| \in [T(1 - 1/k), T]. \end{cases}$$

Let $y_1(t), \cdots, y_{R_p}(t)$ be the outputs of R_p independent runs of Procedure ConstantProbFourierInterpolation in Algorithm 71. By Theorem 12.75, we have

Algorithm 71 Fourier-sparse signal interpolation

```
1: procedure ConstantProbFourierInterpolation($x, H, G, T, F$)
2:     $L \leftarrow$ FrequencyEstimationX($x, H, G, T, F$)                    ▷ $L \in \mathbb{R}^B$
3:     $y(t) \leftarrow$ SignalEstimation($x, \varepsilon, k, F, T, L$)
4:     return $y(t)$
5: end procedure
6: procedure HighProbFourierInterpolation($x, H, G, T, F$)
7:     $R_p \leftarrow \log(1/\rho)$
8:     for $i \in [R_p]$ do
9:         $y_i(t) \leftarrow$ ConstantProbFourierInterpolation($x, H, G, T, F$)
10:    end for
11:    $y(t) \leftarrow$ MergeSignal($y_1(t), y_2(t), \cdots, y_{R_p}(t)$)
12:    return $y(t)$
13: end procedure
14: procedure MergeSignal($y_1(t), y_2(t), \cdots, y_{R_p}(t)$)
15:    $d \leftarrow O(\Delta T + k^3 \log k + k \log 1/\delta)$
16:    for $i \in [R_p]$ do
17:        for $j \in [R_p]$ do
18:            $s, \{t_1, t_2, \cdots, t_s\}, w \leftarrow$ WeightedSketch($O(Bd \log(Bd) \log(R_p^2/\rho)), 2 \cdot Bd, T$)
19:                                                                            ▷ $w \in \mathbb{R}^s$
20:            $S \leftarrow \{t_1, t_2, \cdots, t_s\}$
21:            $Y \leftarrow$ MixedPolynomialEvaluation($y_i - y_j, S$)              ▷ $Y \in \mathbb{C}^s$
22:            $\|y_i(t) - y_j(t)\|_{S,w}^2 \leftarrow \sum_{l=1}^{s} w_l \cdot |Y_l|^2$
23:        end for
24:        $\mathsf{med}_i \leftarrow \mathrm{median}_{j \in [R_p]} \{\|y_i - y_j\|_{S,w}^2\}$
25:    end for
26:    $i^* \leftarrow \arg\min_{i \in [R_p]} \{\mathsf{med}_i\}$
27:    return $y_{i^*}$
28: end procedure
```

that for any $j \in [R_p]$ with probability at least 0.9,

$$\|y_j(t) - x^*(t)\|_T^2 \lesssim \|g(t)\|_T^2.$$

Let $S = \{t_1, \ldots, t_s\}$ be $s = O(k \log(k) \log(R_p^2/\rho))$ i.i.d. samples from $D(t)$, and let $w_i = 1/(TsD(t_i))$ for $i \in [s]$. By Lemma 12.45, for any $i, j \in [R_p]$, with probability at least $1 - \rho/R_p^2$,

$$\|y_i(t) - y_j(t)\|_{S,w}^2 \in [1/2, 3/2] \cdot \|y_i(t) - y_j(t)\|_T^2.$$

Let

$$j^* = \arg\min_{j \in [R_p]} \operatorname*{median}_{i \in [R_p]} \|y_j(t) - y_i(t)\|_T^2,$$

and let $y(t) := y_{j^*}(t)$.

By Lemma 12.80, we have that with probability at least $1 - 2^{-\Omega(R_p)}$,

$$\|y_j(t) - x^*(t)\|_T^2 \lesssim \frac{3/2}{1/2} \|g(t)\|_T^2 \eqsim \|g(t)\|_T^2.$$

By setting $R_p = \log(1/\rho)$, we get the desired result. The correctness is then proved.

The time complexity follows from Lemma 12.84. And the sample complexity follows from Lemma 12.85.

The proof of the theorem is completed. □

In the remaining of this section, we prove the time and sample complexities of Procedure HighProbFourierInterpolation in Algorithm 71.

The following two lemmas show the time complexity of Procedure MergeSignal in Algorithm 71, which is used to boost the success probability of Fourier interpolation algorithm.

Lemma 12.82 (Time complexity of Procedure WeightedSketch in Algorithm 70)**.** *Procedure* WeightedSketch *in Algorithm 70 runs in*

$$O(\varepsilon^{-2} k \log(k) \log(1/\rho))$$

time.

Proof. Procedure WeightedSketch contains the following steps:

- In line 4, sampling S_0 takes $O(\varepsilon^{-2}k\log(k)\log(1/\rho))$ times.
- In line 5, the for loop repeats $|S_0|$ times, and each takes $O(1)$ times.

  Following from the setting in the algorithm, we have that

  $$|S_0| = O(\varepsilon^{-2}k\log(k)\log(1/\rho)).$$

  So, the time complexity of Procedure WEIGHTEDSKETCH in Algorithm 70 is

  $$O(\varepsilon^{-2}k\log(k)\log(1/\rho)) + |S_0|\cdot O(1) = O(\varepsilon^{-2}k\log(k)\log(1/\rho)).$$

$\square$

Lemma 12.83 (Time complexity of Procedure MERGESIGNAL in Algorithm 71)**.** *Procedure* MERGESIGNAL *in Algorithm 71 runs in*

$$O(k^5\log^6(k)\log^4(1/\delta_1)\log^5(1/\rho))$$

time.

Proof. In each call of the Procedure MERGESIGNAL in Algorithm 71, the for loop (Line 16) repeats R_p times, each consisting of the following steps:

- In Line 17, the for loop repeats R_p times and each iteration has the following steps:
  - Line 19 calls Procedure WEIGHTEDSKETCH. By Lemma 12.82, it runs in

    $$O(Bd\log(Bd)\log(R_p^2/\rho))$$

    time.
  - Line 21 calls Procedure MIXEDPOLYNOMIALEVALUATION. By Lemma 12.71, it runs in

    $$O\Big(\sum_{j=1}^{k}\max\{d',\deg(P_j)\}\log^3(\max\{d',\deg(P_j)\})\Big)$$

    time, where $d' = O(Bd\log(Bd)\log(R_p^2/\rho))$ and $\deg(P_j) = d$.

- Line 26 computes the median in $R_p \log(R_p)$ time.

Following from the parameter setting in the algorithm, we have that

$$\begin{aligned} B &= O(k), \\ d &= O(\Delta T + k^3 \log k + k \log 1/\delta). \end{aligned}$$

By Lemma 12.24 Property III, we have that

$$\Delta = k\Delta_h = k|\mathrm{supp}(\widehat{H}(f))|/T = O(k^3 \log^2(k) \log^2(1/\delta_1)/T).$$

As a result, we have that

$$B \cdot d = O(k^4 \log^2(k) \log^2(1/\delta_1)) \tag{12.119}$$

Moreover, we have that

$$\begin{aligned} \max\{d', \deg(P_j)\} &= O(Bd \log(Bd) \log(R_p^2/\rho)) \\ &= O(k^4 \log^3(k) \log^2(1/\delta_1) \log(\log(1/\delta_1)) \log(1/\rho) \log(\log(1/\rho))) \\ &\leq O(k^4 \log^3(k) \log^3(1/\delta_1) \log^2(1/\rho)). \end{aligned} \tag{12.120}$$

where the first step follows from the definition of d' and $\deg(P_j)$, the second step follows from Eq. (12.119), the third step is straight forward.

So, the time complexity of Procedure MERGESIGNAL in Algorithm 71 is

$$\begin{aligned} & R_p^2 \cdot \left(O(Bd \log(Bd) \log(R_p^2/\rho)) + O\Big(\sum_{j=1}^k \max\{d', \deg(P_j)\} \log^3(\max\{d', \deg(P_j)\}) \Big) \right) \\ & + R_p \cdot O(R_p \log(R_p)) \\ & \leq O\Big(R_p^2 \sum_{j=1}^k \max\{d', \deg(P_j)\} \log^3(\max\{d', \deg(P_j)\}) \Big) \\ & \leq O(\log^2(1/\rho) \cdot k \cdot (k^4 \log^3(k) \log^3(1/\delta_1) \log^2(1/\rho)) \log^3(k \log(1/\delta_1) \log(1/\rho))) \\ & \leq O(k^5 \log^6(k) \log^4(1/\delta_1) \log^5(1/\rho)), \end{aligned}$$

where the first step follows from Eq. (12.119), the second step follows from Eq. (12.120), the third step follows from Eq. (12.120), the forth step is straight forward.

□

The following two lemmas show the time complexity and sample complexity of our main algorithm.

Lemma 12.84 (Time complexity of the main algorithm)**.** *Procedure* HighProbFourierInterpolation *in Algorithm 71 runs in*

$$O(k^{4\omega} \log^{4\omega+2}(k/\delta) \log(FT) \log(\log(FT)) \log^5(1/\rho))$$

times.

Proof. Procedure HighProbFourierInterpolation in Algorithm 71 consists of the following steps:

- In Line 8, the for loop repeats R_p times with the following step:
  - Line 9 calls Procedure ConstantProbFourierInterpolation. By Lemma 12.76, it runs in

    $$O(k^{4\omega} \log^{2\omega+1}(k) \log^{2\omega}(1/\delta_1) \log(\log(1/\delta_1)) \log(FT) \log(\log(FT)))$$

    time.
- Line 11 calls Procedure MergeSignal. By Lemma 12.83, it runs in

  $$O(k^5 \log^6(k) \log^4(1/\delta_1) \log^5(1/\rho))$$

  time.

Following from the setting in the algorithm, we have that

$$\delta_1 = \delta/\mathrm{poly}(k). \tag{12.121}$$

So, the time complexity of Procedure HIGHPROBFOURIERINTERPOLATION in Algorithm 71 in Algorithm 71 is

$$\begin{aligned}
&R_p \cdot O\left(k^{4\omega}\log^{2\omega+1}(k)\log^{2\omega}(1/\delta_1)\log(\log(1/\delta_1))\log(FT)\log(\log(FT))\right)\\
&+O(k^5\log^{c+3}(k)\log^4(1/\delta_1)\log^5(1/\rho))\\
=\;& O(k^{4\omega}\log^{2\omega+1}(k)\log^{2\omega}(1/\delta_1)\log(\log(1/\delta_1))\log(FT)\log(\log(FT))\log^5(1/\rho))\\
\le\;& O(k^{4\omega}\log^{4\omega+2}(k/\delta)\log(FT)\log(\log(FT))\log^5(1/\rho)),
\end{aligned}$$

where the first step follows from $R_p = \log(1/\rho)$, the second step follows from Eq. (12.121).

□

Lemma 12.85 (Sample complexity of the main algorithm)**.** *Procedure* HIGHPROB-FOURIERINTERPOLATION *in Algorithm 71 takes*

$$O(k^4\log^6(k/\delta)\log(FT)\log(\log(FT))\log(1/\rho))$$

samples.

Proof. Procedure Procedure HIGHPROBFOURIERINTERPOLATION in Algorithm 71 consists of the following steps:

- In Line 8, the for loop repeats R_p times:
  - Line 9 calls Procedure CONSTANTPROBFOURIERINTERPOLATION. By Lemma 12.77, it takes

    $$O(k^4\log^3(k)\log^2(1/\delta_1)\log(\log(1/\delta_1))\log(FT)\log(\log(FT)))$$

    samples.

  The remaining steps do not use any new sample.

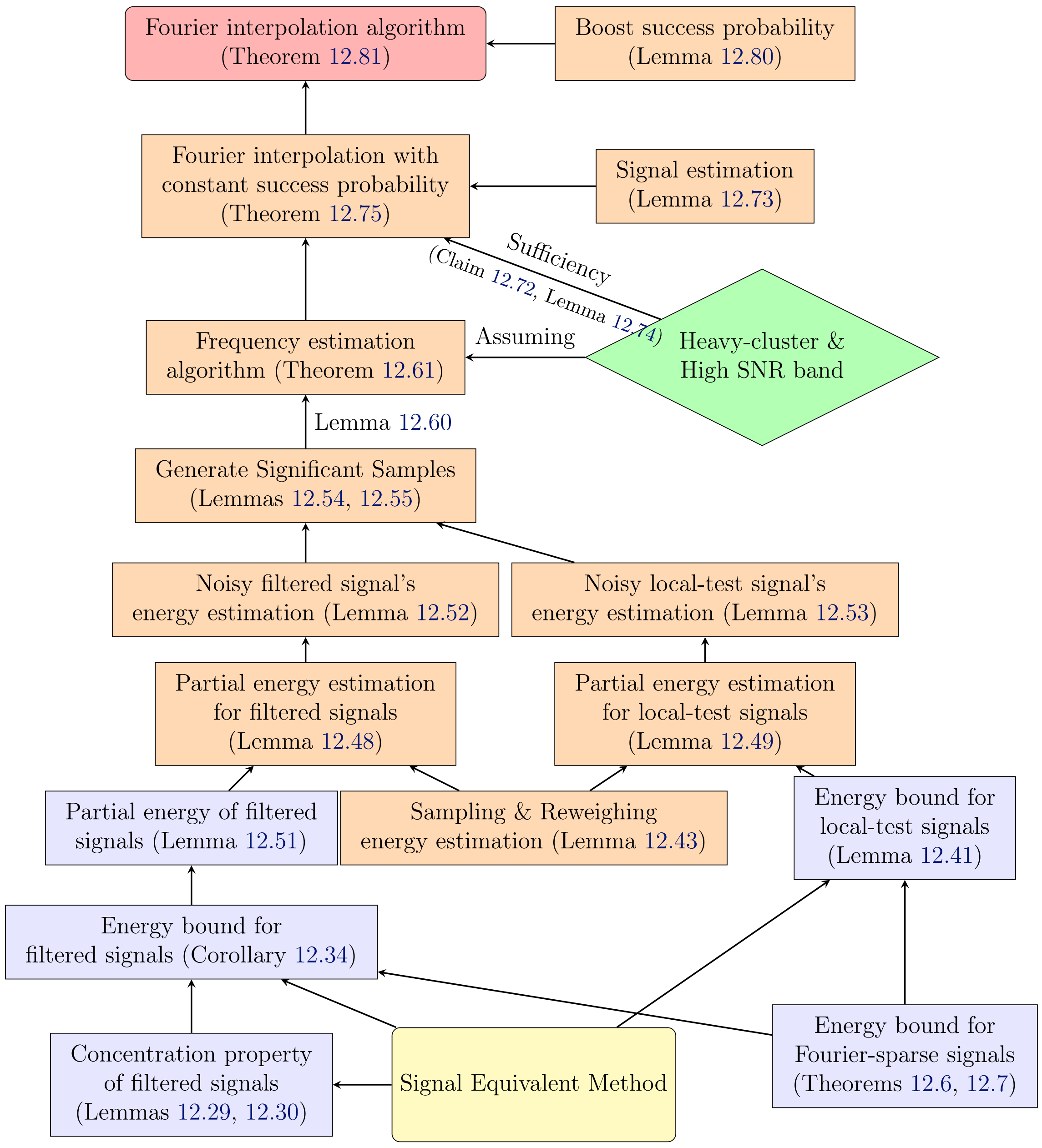
Thus, the total sample complexity is

$$\begin{aligned}
& R_p \cdot O(k^4 \log^3(k) \log^2(1/\delta_1) \log(\log(1/\delta_1)) \log(FT) \log(\log(FT))) \\
\leq\ & O(k^4 \log^3(k) \log^2(1/\delta_1) \log(\log(1/\delta_1)) \log(FT) \log(\log(FT)) \log(1/\rho)) \\
\leq\ & O(k^4 \log^6(k/\delta) \log(FT) \log(\log(FT)) \log(1/\rho)),
\end{aligned}$$

where the first step follows from $R_p = \log(1/\rho)$, the second step follows from Eq. (12.121).

$\square$

12.17 Structure of Our Fourier Interpolation Algorithm



Fourier interpolation algorithm (Theorem 12.81)
Boost success probability (Lemma 12.80)
Fourier interpolation with constant success probability (Theorem 12.75)
Signal estimation (Lemma 12.73)
Sufficiency
(Claim 12.72, Lemma 12.74)
Frequency estimation algorithm (Theorem 12.61)
Assuming
Heavy-cluster & High SNR band
Lemma 12.60
Generate Significant Samples (Lemmas 12.54, 12.55)
Noisy filtered signal's energy estimation (Lemma 12.52)
Noisy local-test signal's energy estimation (Lemma 12.53)
Partial energy estimation for filtered signals (Lemma 12.48)
Partial energy estimation for local-test signals (Lemma 12.49)
Partial energy of filtered signals (Lemma 12.51)
Sampling & Reweighing energy estimation (Lemma 12.43)
Energy bound for local-test signals (Lemma 12.41)
Energy bound for filtered signals (Corollary 12.34)
Concentration property of filtered signals (Lemmas 12.29, 12.30)
Signal Equivalent Method
Energy bound for Fourier-sparse signals (Theorems 12.6, 12.7)

Chapter 13: Distance Oracles for Any Symmetric Norm

13.1 Introduction

Estimating and detecting similarities in datasets is a fundamental problem in computer science, and a basic subroutine in most industry-scale ML applications, from optimization [CMJF$^+$20, CLP$^+$21, XSS21] and reinforcement learning [SSX21], to discrepancy theory [SXZ22] and covariance estimation [Val12, Alm19], Kernel SVMs [CS09, SSSSC11], compression and clustering [IRW17, MMR19], to mention a few. Such applications often need to quickly compute distances of online (test) points to a subset of points in the input data set (e.g., the training data) for transfer-learning and classification. These applications have motivated the notion of *distance oracles* (DO) [Pel00, GPPR04, WP11]: In this problem, the goal is to preprocess a dataset of n input points $X = (x_1, x_2, \dots, x_n)$ in some d-dimensional metric space, into a small-space data structure which, given a query vector q and a subset $\mathsf{S} \subseteq [n]$, can quickly estimate all the distances $\mathsf{d}(q, x_i)$ of q to S (note that the problem of estimating a *single* distance $\mathsf{d}(q, x_i)$ is not interesting in $\mathbb{R}^d$, as this can be trivially done in $O(d)$ time, which is necessary to merely read the query q). The most well-studied case (in both theory and practice) is when the metric space is in fact a *normed* space, i.e., the data points $\{x_i\}_{i \in [n]} \in \mathbb{R}^d$ are endowed with some predefined norm $\| \cdot \|$, and the goal is to quickly estimate $\|x_i - q\|$ simultaneously for all i, i.e., in time $\ll nd$ which is the trivial query time. Distance oracles can therefore be viewed as generalizing *matrix-vector* multiplication: for the inner-product distance function $\langle x_i, q \rangle$, the query asks to approximate $X \cdot q$ in $\ll nd$ time.

For the most popular distance metrics—the Euclidean distance (ℓ_2-norm) and Manhattan distance (ℓ_1-norm)—classic dimension-reduction (sketching) provide very efficient distance oracles [JL84, AC09, LDFU13]. However, in many real-world problems, these metrics do not adequately capture similarities between data points, and a long line of work has demonstrated that more complex (possibly *learnable*) metrics can lead to substantially better prediction and data compression [DKJ$^+$07]. In particular, many works over the last decade have been dedicated to extending various optimization problems beyond Euclidean/Manhattan distances, for example in (kernel) linear regression [SWY$^+$19], approximate nearest neighbor [ANN$^+$17, ANN$^+$18], sampling [LSV18], matrix column subset selection [SWZ19], and statistical queries [LNRW19].

For ℓ_p norms, the DO problem is well-understood [BYJKS04],where the standard tool for constructing the data structure is via randomized *linear sketching* : The basic idea is to reduce the dimension (d) of the data points by applying some *sketching matrix* $\Phi \in \mathbb{R}^{m \times d}$ ($m \ll d$) to each data point x_i and store the sketch $\Phi x_i \in \mathbb{R}^m$. For a query point q, linearity then allows to estimate the distance from $\Phi(q - x_i)$. The seminal works of [JL84, AMS99, CCF04, TZ12] developed polylogarithmic-size sketching methods for the ℓ_2-norm, which was extended, in a long line of work, to any ℓ_p norm with $0 < p < 2$ [Ind06, KNPW11, CN20]. For $p > 2$, the sketch-size (d) becomes *polynomial*, yet still sublinear in d [SS02, BYJKS04].

In this chapter, we consider general *symmetric norms*, which generalize ℓ_p norms. More formally, a norm norm $l : \mathbb{R}^d \to \mathbb{R}$ is symmetric if, for all $x \in \mathbb{R}^d$ and every $d \times d$ permutation matrix P, it satisfies $l(Px) = l(x)$ and also $l(|x|) = l(x)$, where $|x|$ is the coordinate-wise absolute value of x (for a broader introduction, see [Bha97] Chapter IV). Important special cases of symmetric norms are ℓ_p norms and *Orlicz norms* [ALS$^+$18], which naturally arise in harmonic analysis and model data with sub-gaussian properties. Other practical examples of symmetric norms include top-k norms, max-mix of ℓ_p norms, sum-mix of ℓ_p norms, the k-support norm [AFS12] and the box-norm [MPS14].

Several recent works have studied dimension-reduction (sketching) for special cases of symmetric norms such as the Orlicz norm [BBC$^+$17, SWY$^+$19, ALS$^+$18], for various numerical linear algebra primitives [SWZ19]. These sketching techniques are quite ad-hoc and are carefully tailored to the norm in question. It is therefore natural to ask whether *symmetry alone* is enough to guarantee dimensionality-reduction for symmetric similarity search, in other words

Is there an efficient $(1+\epsilon)$*-distance oracle for general symmetric norms?*

By "efficient", we mean small space and preprocessing time (ideally $\sim nd$), fast query time ($\ll n|S|$ for a query $(q, S \subseteq [n])$) and ideally supporting dynamic updates to x_i's in $\widetilde{O}(d)$ time. Indeed, most ML applications involve rapidly-changing *dynamic datasets*, and it is becoming increasingly clear that static data structures do not adequately capture the requirements of real-world applications [JKDG08, CMJF+20, CLP+21]. As such, it is desirable to design a *dynamic distance oracle* which has both small update time (t_u) for adding/removing a point $x_i \in \mathbb{R}^d$, and small query time (t_q) for distance estimation. We remark that most known DOs are dynamic by nature (as they rely on linear sketching techniques), but for general metrics (e.g., graph shortest-path or the ℓ_∞ norm) this is much less obvious, and indeed *linearity* of encoding/decoding will be a key challenge in our data structure (see next section). The problem is formally defined as follows:

Definition 13.1 (Symmetric-Norm Distance Oracles). Let $\|\cdot\|_{\mathrm{sym}}$ denote the symmetric norm. We need to design a data structure that efficiently supports any sequence of the following operations:

- $\textsc{Init}(\{x_1, x_2, \cdots, x_n\} \subset \mathbb{R}^d, \epsilon \in (0,1), \delta \in (0,1))$. The data structure takes n data points $\{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$, an accuracy parameter ϵ and a failure probability δ as input.

- $\textsc{UpdateX}(z \in \mathbb{R}^d, i \in [n])$. Update the data structure with the i-th new data point z.

- $\textsc{EstPair}(i, j \in [n])$ Outputs a number pair such that $(1-\epsilon)\|x_i - x_j\|_{\mathrm{sym}} \leq$ pair $\leq (1+\epsilon) \cdot \|x_i - x_j\|_{\mathrm{sym}}$ with probability at least $1-\delta$.

- Query($q \in \mathbb{R}^d$). Outputs a vector dist $\in \mathbb{R}^n$ such that $\forall i \in [n], (1-\epsilon)\|q - x_i\|_{\mathrm{sym}} \leq \mathrm{dst}_i \leq (1+\epsilon)\|q - x_i\|_{\mathrm{sym}}$. with probability at least $1-\delta$.

where $\|x\|_{\mathrm{sym}}$ is the symmetric norm of vector x.

This problem can be viewed as an *online* version of the (approximate) *closest-pair* problem [Val12], which asks to find the closest pair of points among an *offline batch* of data points $X = x_1, \ldots, x_n \in \mathbb{R}^d$, or equivalently, the smallest entry of the *covariance matrix* $XX^\top$. One major (theoretical) advantage of the offline case is that it enables the use of fast matrix-multiplication (FMM) to speed-up the computation of the covariance matrix [Val12, AWY14, AWY18, Alm19] (i.e., sub-linear amortized per query). By contrast, in the online setting such speedups are conjectured to be impossible [HKNS15, LW17].

Notations. For any positive integer n, we use $[n]$ to denote $\{1, 2, \cdots, n\}$. For any function f, we use $\widetilde{O}(f)$ to denote $f \cdot poly(\log f)$. We use $\Pr[]$ to denote probability. We use $\mathbb{E}[]$ to denote expectation. We use both $l(\cdot)$ and $\|\cdot\|_{\mathrm{sym}}$ to denote the symmetric norm. We use $\|\cdot\|_2$ to denote the entry-wise ℓ_2 norm. We define a tail notation which is very standard in sparse recover/compressed sensing literature. For any given vector $x \in \mathbb{R}^d$ and an integer k, we use $x_{\overline{[k]}}$ or $x_{\mathrm{tail}(k)}$ to denote the vector that without (zeroing out) top-k largest entries (in absolute). For a vector x, we use $x^\top$ to denote the transpose of x. For a matrix A, we use $A^\top$ to denote the transpose of A. We use $\mathbf{1}_n$ denote a length-n vector where every entry 1. We use $\mathbf{0}_n$ to denote a length-n vector where every entry is 0.

13.1.1 Our results

Two important complexity measures of (symmetric) norms, which capture their "intrinsic dimensionality", are the *concentration modulus* (mc) and *maximum modulus* (mmc) parameters. We now define these quantities along the lines of [BBC+17, SWY+19].

Definition 13.2 (Modulus of concentration (mc))**.** Let $X \in \mathbb{R}^n$ be uniformly distributed on S^{n-1}, the l_2 unit sphere. The median of a symmetric norm l is the (unique) value M_l such that $\Pr[l(X) \geq M_l] \geq 1/2$ and $\Pr[l(X) \leq M_l] \geq 1/2$. Similarly, $\mathfrak{b}_l$ denotes the maximum value of $l(x)$ over $x \in S^{n-1}$. We call the ratio

$$\mathrm{mc}(l) := \mathfrak{b}_l / M_l$$

the modulus of concentration of the norm l.

For every $k \in [n]$, the norm l induces a norm $l^{(k)}$ on $\mathbb{R}^k$ by setting

$$l^{(k)}((x_1, x_2, \ldots, x_k)) := l((x_1, x_2, \ldots, x_k, \ldots, 0, \ldots, 0))$$

Definition 13.3 (mmc)**.** Define the maximum modulus of concentration of the norm l as

$$\mathrm{mmc}(l) := \max_{k \in [n]} \mathrm{mc}(l^{(k)}) = \max_{k \in [n]} \frac{\mathfrak{b}_{l^{(k)}}}{M_{l^{(k)}}}$$

Next, we present a few examples (in Table 13.1) for different norm l's $\mathrm{mmc}(l)$.

Norm l	$\mathrm{mmc}(l)$
$\ell_p (p \leq 2)$	$\Theta(1)$
$\ell_p (p > 2)$	$\Theta(d^{1/2-1/p})$
top-k norms	$\widetilde{\Theta}(\sqrt{d/k})$
k-support norms and the box-norm	$O(\log d)$
max-mix and sum-mix of ℓ_1 and ℓ_2	$O(1)$
Orlicz norm $\|\cdot\|_G$	$O(\sqrt{C_G \log d})$

Table 13.1: Examples of $\mathrm{mmc}(l)$, where max-mix of ℓ_1 and ℓ_2 is defined as $\max\{\|x\|_2, c\|x\|_1\}$ for a real number c, sum-mix of ℓ_1 and ℓ_2 is defined as $\|x\|_2 + c\|x\|_1$ for a real number c, and C_G of Orlicz norm is defined as the number that for all $0 < x < y$, $G(y)/G(x) \leq C_G(y/x)^2$.

We are now ready to state our main result:

Theorem 13.1 (Main result, informal version of Theorem 13.19)**.** *Let $\|\cdot\|_l$ be any symmetric norm on $\mathbb{R}^d$. There is a data structure for the online symmetric-norm Distance Oracle problem (Definition 13.1), which uses $n(d + \mathrm{mmc}(l)^2) \cdot poly(1/\epsilon, \log(nd/\delta))$ space, supporting the following operations:*

- $\textsc{Init}(\{x_1, x_2, \dots, x_n\} \subset \mathbb{R}^d, \epsilon \in (0,1), \delta \in (0,1))$: *Given n data points $\{x_1, x_2, \dots, x_n\} \subset \mathbb{R}^d$, an accuracy parameter ϵ and a failure probability δ as input, the data structure preprocesses in time $n(d + \mathrm{mmc}(l)^2) \cdot poly(1/\epsilon, \log(nd/\delta))$. Note that* mmc() *is defined as Definition 13.3.*

- $\textsc{UpdateX}(z \in \mathbb{R}^d, i \in [n])$: *Given an update vector $z \in \mathbb{R}^d$ and index $i \in [n]$, the data data structure receives z and i as inputs, and updates the i-th data point $x_i \leftarrow z$, in $d \cdot poly(1/\epsilon, \log(nd/\delta))$ time.*

- $\textsc{Query}(q \in \mathbb{R}^d, \mathsf{S} \subseteq [n])$: *Given a query point $q \in \mathbb{R}^d$ and a subset of the input points $\mathsf{S} \subseteq [n]$, the* Query *operation outputs a ϵ- multiplicative approximation to each distance from q to points in S, in time*

  $$(d + |\mathsf{S}| \cdot \mathrm{mmc}(l)^2) \cdot poly(1/\epsilon, \log(nd/\delta))$$

  i.e. it provides a set of estimates $\{\mathrm{dst}_i\}_{i \in \mathsf{S}}$ such that:

  $$\forall i \in \mathsf{S}, (1-\epsilon)\|q - x_i\|_l \leq \mathrm{dst}_i \leq (1+\epsilon)\|q - x_i\|_l$$

  with probability at least $1 - \delta$.

- $\textsc{EstPair}(i, j \in [n])$ *Given indices $i, j \in [n]$, the* EstPair *operation takes i and j as input and approximately estimates the symmetric norm distances from i-th to the j-th point $x_i, x_j \in \mathbb{R}^d$ in time $\mathrm{mmc}(l)^2 \cdot poly(1/\epsilon, \log(nd/\delta))$ i.e. it provides a estimated distance* pair *such that:*

  $$(1-\epsilon)\|x_i - x_j\|_l \leq \mathrm{pair} \leq (1+\epsilon)\|x_i - x_j\|_l$$

  with probability at least $1 - \delta$.

Roadmap. In Section 13.2, we give an overview of techniques we mainly use in the work. Section 13.3 gives an introduction of the Sparse Recovery Data we use for sketching. Section 13.4 analyze the running time and space of our data structure with their proofs, respectively. Section 13.5 show the correctness of our data structure and give its proof. Finally in Section 13.6 we conclude our work.

13.2 Technique Overview

Our distance oracle follows the "sketch-and-decode" approach, which was extensively used in many other sublinear-time compressed sensing and sparse recovery problems [Pri11, HIKP12a, LNNT16, NS19, SSWZ22]. The main idea is to compress the data points into smaller dimension by computing, for each data point $x_i \in \mathbb{R}^d$, a (randomized) linear sketch $\Phi \cdot x_i \in \mathbb{R}^{d'}$ with $d' \ll d$ at preprocessing time, where Φx_i is an unbiased estimator of $\|x_i\|$. At query time, given a query point $q \in \mathbb{R}^d$, we analogously compute its sketch Φq. By *linearity* of Φ, the distance between q and x_i (i.e., $\ell(q - x_i)$) can be trivially decoded from the sketch *difference* $\Phi q - \Phi x_i$. As we shall see, this simple virtue of linearity is less obvious to retain when dealing with general symmetric norms.

Layer approximation Our algorithm uses the *layer approximation* method proposed by Indyk and Woodruff in [IW05] and generalized to symmetric norms in [BBC$^+$17, SWY$^+$19]. Since symmetric norms are invariant under reordering of the coordinates, the main idea in [IW05] is to construct a "layer vector" as follows: for a vector $v \in \mathbb{R}^d$, round (the absolute value of) each coordinate to the nearest power α^i for some fixed $\alpha \in \mathbb{R}$ and $i \in \mathbb{N}$, and then sort the coordinates in an increasing order. This ensures that the i-th layer contains all the coordinates $j \in [d]$ satisfying: $\alpha^{i-1} < |v_j| \leq \alpha^i$. In particular, the layer vector of v has the form

$$\mathcal{L}(v) := (\underbrace{\alpha^1, \ldots, \alpha^1}_{b_1 \text{ times}}, \underbrace{\alpha^2, \ldots, \alpha^2}_{b_2 \text{ times}}, \cdots, \underbrace{\alpha^P, \ldots, \alpha^P}_{b_P \text{ times}}, 0, \ldots, 0) \in \mathbb{R}^d,$$

where b_i is the number of coordinates in layer-i. More importantly, since the norm is symmetric, the layer vector $\mathcal{L}(v)$ has a succinct representation: $(b_i)_{i \in [P]}$.

Then, it suffices to estimate b_i for each $i \in [P]$, where the Indyk-Woodruff sketching technique can be used to approximate the vector. At the i-th layer, each coordinate of v is sampled with probability P/b_i, and then the algorithm identifies the ℓ_2-*heavy-hitters* of the sampled vector. [BBC$^+$17] gave a criterion for identifying the important layers, whose heavy-hitter coordinates in the corresponding *sampled* vectors, is enough to recover the entire symmetric norm $\|v\|_{\mathrm{sym}}$.

Unfortunately, this technique for norm estimation does not readily translate to estimating *distances* efficiently:

- *Too many layers:* In previous works, each data point x_i is sub-sampled *independently* in R layers, i.e, generates R subsets of coordinates $S_i^1, \ldots, S_i^R \subset [d]$. The sketch of the query point $S(q)$ then needs to be compared to each $S(x_i)$ in every layer. Since there are $R = \Omega(n)$ layers in [BBC+17, IW05] of size $\Omega(d)$ across all all data points, the total time complexity will be at least $\Omega(nd)$, which is the trivial query time.

- *Non-linearity:* The aforementioned sketching algorithms [BBC+17, IW05] involve nonlinear operations, and thus cannot be directly used for distance estimation.

- *Slow decoding:* The aforementioned sketches take linear time to decode the distance from the sketch, which is too slow for our application.

To overcome these challenges, we use the following ideas:

Technique I: shared randomness To reduce the number of layers, we let all the data points use the same set of layers. That is, in the initialization of our algorithm, we independently sample R subsets $S^1, \ldots, S^R$ with different probabilities. Then, for each data point x_i, we consider $(x_i)_{S^j}$ as the sub-sample for the j-th layer, and perform sketch on it. Hence, the number of different layer sets is reduced from nt to t. For a query point q, we just need to compute the sketches for $(q)_{S^1}, \ldots, (q)_{S^R}$. And the distance between q and x_i can be decoded from $\{\Phi(x_i)_{S^j} - \Phi(q)_{S^j}\}_{j \in [R]}$. We also prove that the shared randomness in all data points will not affect the correctness of layer approximation.

Technique II: linearization We choose a different sketching method called BATCHHEAVYHITTER (see Theorem 13.20 for details) to generate and maintain the *linear sketches*, which allows us to decode the distance from sketch difference.

Technical III: locate-and-verify decoding We design a *locate-and-verify* style decoding method to recover distance from sketch. In our data structure, we not only store the sketch of each sub-sample vector, but also the vector itself. Then, in decoding a sketch, we can first apply the efficient sparse-recovery algorithm to identify the position of heavy-hitters. Next, we directly check the entries at those positions to verify that they are indeed "heavy" (comparing the values with some threshold), and drop the non-heavy indices. This verification step is a significant difference from the typical sparse recovery approaches, which employ complex (and time-consuming) subroutines to *reconstruct the values* of the heavy-hitter coordinates. Instead, our simple verification procedure eliminates all the *false-positive* heavy-hitters, therefore dramatically reducing the running time of the second step, which can now be performed directly by reading-off the values from the memory.

With these three techniques, we obtain our sublinear-time distance estimation algorithm. Our data structure first generate a bunch of randomly selected subsets of coordinates as the layer sets. Then, for each data point, we run the BATCHHEAVYHITTER procedure to sketch the sub-sample vector in each layer[1]. In the query phase, we call the DECODE procedure of BATCHHEAVYHITTER for the sketch differences between the query point q and each data point x_i, and obtain the heavy hitters of each layer. We then select some "important layers" and use them to approximately recover the layer vector $\mathcal{L}(q - x_i)$, which gives the estimated distance $\|q - x_i\|_{\mathrm{sym}}$.

Finally, we summarize the time and space costs of our data structure. Let ϵ be the precision parameter and δ be the failure probability parameter. Our data

[1]The total sketch size of each data point is $\mathrm{mmc}(l)^2 \cdot \mathit{poly} \log d$. In the ℓ_p-norm case with $p > 2$, $\mathrm{mmc}(l) = d^{1/2-1/p}$ (see Table 13.1) and our sketch size is $\widetilde{O}(d^{1-2/p})$, matching the lower bound of ℓ_p-sketching. When $p \in (0, 2]$, $\mathrm{mmc}(l) = \Theta(1)$ and our sketch size is $\widetilde{O}(1)$, which is also optimal.

structure achieves $\widetilde{O}(n(d + \mathrm{mmc}(l)^2))$-time for initialization , $\widetilde{O}(d)$-time per data point update, and $\widetilde{O}(d + n \cdot \mathrm{mmc}(l)^2))$-time per query. As for space cost, our data structure uses the space of $\widetilde{O}(n(d + \mathrm{mmc}(l)^2))$ in total. Note that mmc is defined as Defnition 13.3.

13.3 Sparse Recovery Data Structure

We design a data structure named BATCHHEAVYHITTER to generate sketches and manage them. In our design, it is a "linear sketch" data structure, and providing the following functions:

- INIT($\epsilon \in (0, 0.1), n, d$). Create a set of Random Hash functions and all the n copies of sketches share the same hash functions. This step takes $\mathcal{T}_{\mathrm{init}}(\epsilon, n, d)$ time.

- ENCODE($i \in [n], z \in \mathbb{R}^d, d$). This step takes $\mathcal{T}_{\mathrm{encode}}(d)$ encodes z into i-th sketched location and store a size $\mathcal{S}_{\mathrm{space}}$ linear sketch.

- ENCODESINGLE($i \in [n], j \in [d], z \in \mathbb{R}, d$). This step takes $\mathcal{T}_{\mathrm{encodesingle}}(d)$ updates one sparse vector $e_j z \in \mathbb{R}^d$ into i-th sketched location.

- SUBTRACT($i, j, l \in [n]$). Update the sketch at i-th location by j-th sketch minus l-th sketch.

- DECODE($i \in [n], \epsilon \in (0, 0.1), d$). This step takes $\mathcal{T}_{\mathrm{decode}}(\epsilon, d)$ such that it returns a set $L \subseteq [d]$ of size $|L| = O(\epsilon^{-2})$ containing all ϵ-heavy hitters $i \in [n]$ under ℓ_p. Here we say i is an ϵ-heavy hitter under ℓ_2 if $|x_i| \geq \epsilon \cdot \|x_{\overline{[\epsilon^{-2}]}}\|_2$ where $x_{\overline{[k]}}$ denotes the vector x with the largest k entries (in absolute value) set to zero. Note that the number of heavy hitters never exceeds $2/\epsilon^2$.

With this data structure, we are able to generate the sketches for each point, and subtract each other with its function. And one can get the output of heavy hitters of each sketch stored in it with DECODE function. More details are deferred to Section 13.10.2.

13.4 Running Time and Space of Our Algorithm

Algorithm 72 Data structure for symmetric norm estimation: members, init, informal version of Algorithm 75 and Algorithm 76

```
1: data structure DISTANCEONSYMMETRICNORM                          ▷ Theorem 13.19
2: members
3:     $\{x_i\}_{i=1}^n \in \mathbb{R}^d$
4:     $\{S_{r,l,u}\}_{r\in[R],l\in[L],u\in[U]}$                    ▷ A list of the BATCHHEAVYHITTER
5:     $\{H_{r,l,u}\}_{r\in[R],l\in[L],u\in[U]} \subset [d] \times \mathbb{R}$
6: end members
7:
8: public:
9: procedure INIT($\{x_1, \cdots, x_n\} \subset \mathbb{R}^d, \delta, \epsilon$)        ▷ Lemma 13.2
10:     Initialize the sparse-recovery data structure $\{S_{r,l,u}\}$
11:     Create $\{\mathrm{bmap}_{r,l,u}\}$ shared by all $i \in [n]$ ▷ $\{\mathrm{bmap}_{r,l,u}\}$ is list of a layer set map
12:     for $i \in [n], j \in [d]$ do
13:         if $\mathrm{bmap}_{r,u,l}[j] = 1$ then
14:             Sample $x_{i,j}$ into each subvector $\overline{x}_{r,u,l,i}$
15:         end if
16:     end for
17:     Encode $\{x_{r,u,l,i}\}_{i\in[n]}$ into $\{S_{r,l,u}\}$
18: end procedure
19: end data structure
```

We first analyze the running time of different procedures of our data structure DISTANCEONSYMMETRICNORM. See Algorithm 76, with the linear sketch technique, we spend the time of $\widetilde{O}(n(d + \mathrm{mmc}(l)^2))$ for preprocessing and generate the sketches stored in the data structure. When it comes a data update (Algorithm 77), we spend $\widetilde{O}(d)$ to update the sketch. And when a query comes (Algorithm 78), we spend $\widetilde{O}(d + n \cdot \mathrm{mmc}(l)^2))$ to get the output distance estimation. The lemmas of running time and their proof are shown below in this section.

Lemma 13.2 (INIT time, informal). *Given data points $\{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$, an ac-*

Algorithm 73 Data structure for symmetric norm estimation: query, informal version of Algorithm 78

```
1: data structure DistanceOnSymmetricNorm
2: public:
3: procedure Query($q \in \mathbb{R}^d$)    ▷ Lemma 13.4, 13.6
4:     Encode $q$ into $\{S_{r,l,u}\}$
5:     for $i \in [n]$ do
6:         Subtract the sketch of $x_i$ and $q$, get the estimated heavy-hitters of $q - x_i$
7:         Decode the sketch and store returned estimation of heavy-hitters into $H_{r,l,u}$
8:         for $u \in [U]$ do
9:             if $H_{r,l,u}$ provide correct indices of heavy hitters then
10:                Select it as good set and store it in $H_{r,l}$
11:            end if
12:        end for
13:        Generate estimated layer sizes $\{c_k^i\}_{k \in [P]}$
14:        $\mathrm{dst}_i \leftarrow$ LayerVetcorApprox$(\alpha, c_1^i, c_2^i, \dots, c_P^i, d)$
15:        Reset $\{H_{r,l,u}\}$ for next distance
16:    end for
17:    return $\{\mathrm{dst}_i\}_{i \in [n]}$
18: end procedure
19: end data structure
```

curacy parameter $\epsilon \in (0,1)$, and a failure probability $\delta \in (0,1)$ as input, the procedure INIT *(Algorithm 76) runs in time*

$$O(n(d + \mathrm{mmc}(l)^2) \cdot poly(1/\epsilon, \log(nd/\delta))).$$

Proof. The Init time includes these parts:

- Line 12 takes $O(RLU \cdot \mathcal{T}_{\mathrm{init}}(\sqrt{\beta}, n, d))$ to initialize sketches
- Line 17 to Line 19 takes $O(RUdL)$ to generate the bmap;
- Line 24 takes $O(ndRUL \cdot \mathcal{T}_{\mathrm{encodesingle}}(d))$ to generate sketches

By Theorems 13.20, we have

- $\mathcal{T}_{\mathrm{init}}(\sqrt{\beta}, n, d)) = n \cdot O(\beta^{-1} \log^2 d) = O(n \cdot \mathrm{mmc}(l)^2 \log^7(d) \epsilon^{-5})$,

- $\mathcal{T}_{\text{encodesingle}}(d) = O(\log^2(d))$.

Adding them together we got the time of

$$\begin{aligned}
& O(RLU\mathcal{T}_{\text{init}}(\sqrt{\beta}, n, d)) + O(RUdL) + O(ndRUL \cdot \mathcal{T}_{\text{encodesingle}}(d)) \\
= & O(RLU(\mathcal{T}_{\text{init}}(\sqrt{\beta}, n, d) + nd \cdot \mathcal{T}_{\text{encodesingle}}(d))) \\
= & O(\epsilon^{-4} \log(1/\delta) \log^4(d) \cdot \log(d) \cdot \log(d^2/\delta \cdot \log(nd))(n \cdot \text{mmc}(l)^2 \log^7(d)\epsilon^{-5} + nd \log^2(d))) \\
= & O(n(d + \text{mmc}(l)^2) \cdot \mathit{poly}(1/\epsilon, \log(nd/\delta))),
\end{aligned}$$

where the first step follows from merging the terms, the second step follows from the definition of $R, L, U, \mathcal{T}_{\text{encodesingle}}(d), \mathcal{T}_{\text{init}}$, the third step follows from merging the terms.

Thus, we complete the proof. $\square$

Lemma 13.3 (UPDATE time, informal)**.** *Given a new data point $z \in \mathbb{R}^d$, and an index i where it should replace the original data point $x_i \in \mathbb{R}^d$. The procedure* UPDATE *(Algorithm 77) runs in time*

$$O(d \cdot \mathit{poly}(1/\epsilon, \log(nd/\delta))$$

Proof. The UPDATE operation calls BATCHHEAVYHITTER.ENCODE for RLU times, so it has the time of

$$\begin{aligned}
O(RLU \cdot \mathcal{T}_{\text{encode}}(d)) &= O(\epsilon^{-4} \log(1/\delta) \log^4(d) \cdot \log(d) \cdot \log(d^2/\delta) \cdot d \log^2(d) \cdot \log(nd)) \\
&= O(\epsilon^{-4} d \log^9(nd/\delta))
\end{aligned}$$

where the first step follows from the definition of $R, L, U, \mathcal{T}_{\text{encode}}(d)$, the second step follows from

$$\begin{aligned}
& \log(1/\delta) \log^4(d) \log(d) \log(d^2/\delta) \log^2(d) \log(nd) \\
= & (\log(1/\delta))(\log^7 d)(2 \log d + \log(1/\delta)) \log(nd) \\
= & O(\log^9(nd/\delta)).
\end{aligned}$$

Thus, we complete the proof. $\square$

Here, we present a Query for outputting all the n distances. In Section 13.11, we provide a more general version, which can take any input set $\mathsf{S} \subseteq [n]$, and output distance for only them in a shorter time that proportional to $|\mathsf{S}|$.

Lemma 13.4 (Query time, informal)**.** *Given a query point $q \in \mathbb{R}^d$, the procedure* Query *(Algorithm 78) runs in time*

$$O((d + n \cdot \mathrm{mmc}(l)^2) \cdot poly(1/\epsilon, \log(nd/\delta))).$$

Proof. The Query operation has the following two parts:

- **Part 1:** Line 5 takes $O(RLU \cdot \mathcal{T}_{\mathrm{encode}})$ time to call Encode to generate sketches for q.
- **Part 2:** For every $i \in [n]$:
  - Line 10 takes $O(RLU \cdot \mathcal{T}_{\mathrm{subtract}})$ time to compute sketch of the difference between q and x_i, and store the sketch at index of $n + 1$.
  - Line 11 takes $O(RLU \cdot \mathcal{T}_{\mathrm{decode}})$ time to decode the BatchHeavyHitter and get estimated heavy hitters of $q - x_i$.
  - Line 13 to Line 19 takes $O(RLU \cdot 2/\beta)$ time to analyze the BatchHeavyHitter and get the set of indices, where $2/\beta$ is the size of the set.
  - Line 25 takes $O(LP \cdot 2/\beta)$ time to compute size of the layer sets cut by α.
  - Line 27 to Line 31 takes $O(PL)$ time to compute the estimation of each layer.

  The total running time of this part is:

$$n \cdot (O(RLU \cdot \mathcal{T}_{\mathrm{subtract}}) + O(RLU \cdot \mathcal{T}_{\mathrm{decode}}) + O(RLU \cdot 2/\beta) + O(LP \cdot 2/\beta) + O(LP))$$
$$= O(nL(RU(\mathcal{T}_{\mathrm{subtract}} + \mathcal{T}_{\mathrm{decode}} + \beta^{-1}) + P\beta^{-1}))$$

  time in total.

Taking these two parts together we have the total running time of the QUERY procedure:

$$\begin{aligned}
& O(RLU \cdot \mathcal{T}_{\mathrm{encode}}) + O(nL(RU(\mathcal{T}_{\mathrm{subtract}} + \mathcal{T}_{\mathrm{decode}} + \beta^{-1}) + P\beta^{-1})) \\
= & O(RLU(\mathcal{T}_{\mathrm{encode}} + n \cdot \mathcal{T}_{\mathrm{subtract}} + n \cdot \mathcal{T}_{\mathrm{decode}} + n\beta^{-1}) + nLP\beta^{-1}) \\
= & O(\epsilon^{-4} \log^6(d/\delta) \log(nd)(d \log^2(d) + n\beta^{-1} \log^2(d)) + n\epsilon^{-1} \log^2(d)\beta^{-1}) \\
= & O((d + n \cdot \mathrm{mmc}(l)^2) \cdot poly(1/\epsilon, \log(nd/\delta)))
\end{aligned}$$

where the first step follows from the property of big O notation, the second step follows from the definition of $R, L, U, \mathcal{T}_{\mathrm{encode}}, \mathcal{T}_{\mathrm{encode}}, \mathcal{T}_{\mathrm{subtract}}, \mathcal{T}_{\mathrm{decode}}$ (Theorem 13.20) , P, the third step follows from merging the terms.

Thus, we complete the proof.

□

Next, we analyze the space usage in our algorithm. We delay the proofs into Section 13.13.

Lemma 13.5 (Space complexity of our data structure, informal version of Lemma 13.31)**.** *Our data structure (Algorithm 72 and 73) uses space*

$$O(n(d + \mathrm{mmc}(l)^2) \cdot poly(1/\epsilon, \log(d/\delta))).$$

13.5 Correctness of Our Algorithm

The correctness of our distance oracle is proved in the following lemma:

Lemma 13.6 (QUERY correctness)**.** *Given a query point $q \in \mathbb{R}^d$, the procedure* QUERY *(Algorithm 78) takes q as input and approximately estimates $\{\mathrm{dst}_i\}_{i \in [n]}$ the distance between q and every x_i with the norm l, such that for every dst_i, with probability at least $1 - \delta$, we have*

$$(1 - \epsilon) \cdot \|q - x_i\|_{\mathrm{sym}} \leq \mathrm{dst}_i \leq (1 + \epsilon) \cdot \|q - x_i\|_{\mathrm{sym}}$$

Proof. Without loss of generality, we can consider a fixed $i \in [n]$. For simplicity, we denote x_i by x.

Let $v := q - x$. By Lemma 13.14, it is approximated by its layer vector $\mathcal{L}(v)$, namely,

$$\|v\|_{\mathrm{sym}} \leq \|\mathcal{L}(v)\|_{\mathrm{sym}} \leq (1 + O(\epsilon))\|v\|_{\mathrm{sym}}, \tag{13.1}$$

where $\|\cdot\|_{\mathrm{sym}}$ is a symmetric norm, denoted also by $l(\cdot)$.

We assume without loss of generality that $\epsilon \geq 1/poly(d)$. Our algorithm maintains a data structure that eventually produces a vector $\mathcal{J}(v)$, which is created with the layer sizes $c_1, c_2, \dots, c_P$, where c's denotes the estimated layer sizes output by out data structure, and the b's are the ground truth layer sizes. We will show that with high probability, $\|\mathcal{J}(v)\|_{\mathrm{sym}}$ approximates $\|\mathcal{L}(v)\|_{\mathrm{sym}}$. Specifically, to achieve $(1 \pm \epsilon)$-approximation to $\|v\|_{\mathrm{sym}}$, we set the approximation guarantee of the layer sets (Definition 13.6) to be $\epsilon_1 := O(\frac{\epsilon^2}{\log(d)})$ and the importance guarantee to be $\beta_0 := O(\frac{\epsilon^5}{\mathrm{mmc}(l)^2 \log^5(d)})$, where $\mathrm{mmc}(l)$ is defined as Definition 13.3.

Observe that the number of non-empty layer sets $P = O(\log_\alpha(d)) = O(\log(d)/\epsilon)$. Let E_{succeed} denote the event $(1 - \epsilon_1)b_k \leq c_k \leq b_k$. By Lemma 13.25, it happens with high probability $1 - \delta$. Conditioned on this event.

Denote by $\mathcal{J}(v)$ the vector generated with the layer sizes given by Line 30, and by $\mathcal{L}^*(v)$ the vector $\mathcal{L}(v)$ after removing all buckets (Definition 13.7) that are not β-contributing (Definition 13.8), and define $\mathcal{J}^*(v)$ similar to $\mathcal{L}^*(v)$, where we set $\beta := \epsilon/P = O(\epsilon^2/\log(d))$. Every β-contributing layer is necessarily β_0-important (Definition 13.6) by Lemma 13.17 and Lemma 13.18 and therefore satisfies $c_k \geq (1 - \epsilon_1)b_k$ (Lemma 13.25). We bound the error of $\|\mathcal{L}^*(v)\|_{\mathrm{sym}}$ by Lemma 13.16, namely,

$$(1 - O(\epsilon))\|\mathcal{L}(v)\|_{\mathrm{sym}} \leq (1 - O(\log_\alpha d) \cdot \beta)\|\mathcal{L}(v)\|_{\mathrm{sym}} \leq \|\mathcal{L}^*(v)\|_{\mathrm{sym}} \leq \|\mathcal{L}(v)\|_{\mathrm{sym}}.$$

where the first step follows from the definition of β, the second step and the third step follow from Lemma 13.16.

Then, we have

$$\begin{aligned}\|\mathcal{J}(v)\|_{\mathrm{sym}} &\geq \|\mathcal{J}^*(v)\|_{\mathrm{sym}} \\ &= \|\mathcal{L}^*(v)\backslash\mathcal{L}_{k_1}(v)\cup\mathcal{J}_{k_1}(v)\ldots\backslash\mathcal{L}_{k_\kappa}(v)\cup\mathcal{J}_{k_\kappa}(v)\|_{\mathrm{sym}} \\ &\geq (1-\epsilon_1)^P\|\mathcal{L}^*(v)\|_{\mathrm{sym}} \\ &\geq (1-O(\epsilon))\|\mathcal{L}^*(v)\|_{\mathrm{sym}}. \qquad (13.2)\end{aligned}$$

where the first step follows from monotonicity (Lemma 13.8), the second step follows from the definition of $\mathcal{J}^*(v)$, the third step follows from Lemma 13.15, and the fourth step follows from the definition of ϵ_1 and P.

Combining Eq. (13.1) and (13.2), we have

$$(1-O(\epsilon))\cdot\|v\|_{\mathrm{sym}} \leq \|\mathcal{J}^*(v)\|_{\mathrm{sym}} \leq \|v\|_{\mathrm{sym}}, \qquad (13.3)$$

which bounds the error of $\|\mathcal{J}^*(v)\|_{\mathrm{sym}}$ as required. Note that, with Lemma 13.16 we have

$$(1-O(\epsilon))\cdot\|\mathcal{J}(v)\|_{\mathrm{sym}} \leq \|\mathcal{J}^*(v)\|_{\mathrm{sym}} \leq \|\mathcal{J}(v)\|_{\mathrm{sym}} \qquad (13.4)$$

Combining the Eq.(13.3) and (13.4) we have

$$(1-O(\epsilon))\cdot\|v\|_{\mathrm{sym}} \leq \|\mathcal{J}(v)\|_{\mathrm{sym}} \leq (1+O(\epsilon))\cdot\|v\|_{\mathrm{sym}}$$

Note that E_{succeed} has a failure probability of δ. Thus, we complete the proof. □

13.6 Conclusion

Similarity search is the backbone of many large-scale applications in machine-learning, optimization, databases and computational geometry. Our work strengthens and unifies a long line of work on metric embeddings and sketching, by presenting the first Distance Oracle *for any symmetric norm*, with nearly-optimal query and update times. The generality of our data structure allows to apply it as a *black-box* for data-driven *learned* symmetric distance metrics [DKJ+07] and in various optimization problems involving symmetric distances.

Our work raises several open questions for future study:

- The efficiency of our data structure depends on $\mathrm{mmc}(l)$, the concentration property of the symmetric norm. Is this dependence necessary?
- Can we generalize our data structure to certain *asymmetric norms* ?

We believe our work is also likely to influence other fundamental problems in high-dimensional optimization and search, e.g, kernel linear regression, geometric sampling and near-neighbor search.

Roadmap. We divide the appendix into the following sections. Section 13.7 gives the preliminaries for our work. Section 13.8 introduces some useful properties of symmetric norms. Section 13.9 gives the proofs for the layer approximation technique. Section 13.10 states the formal version of our main theorem and algorithms. Section 13.11 gives more details about the time complexity proofs. Section 13.12 gives some details about the correctness proofs. Section 13.13 shows the space complexity of our algorithm. Section 13.14 states a streaming lower bound for the norm estimation problem and shows that our result is tight in this case. Section 13.15 gives an instantiation of the sparse recovery data structure (BatchHeavyHitter) that simplifies the analysis in prior work.

13.7 Preliminaries

In Section 13.7.1, we define the notations we use in this chapter. In Section 13.7.2, we introduce some probability tools. In Section 13.7.3, we define p-stable distributions.

13.7.1 Notations

For any positive integer n, we use $[n]$ to denote a set $\{1, 2, \cdots, n\}$. We use $\mathbb{E}[]$ to denote the expectation. We use $\Pr[]$ to denote the probability. We use $\mathrm{Var}[]$ to denote the variance. We define the unit vector $\xi^{(d_1)} := \frac{1}{\sqrt{d_1}}(1, 1, 1, , \dots, 1, 0, \dots, 0) \in \mathbb{R}^d$, for any $d_1 \in [d]$, which has d_1 nonzero coordinates. We abuse the notation to write $\xi^{(d_1)} \in \mathbb{R}^{d_1}$ by removing zero coordinates, and vice-versa by appending zeros. We use $\|x\|_2$ to denote entry-wise ℓ_2 norm of a vector. We use $\|x\|_{\mathrm{sym}}$ to denote the symmetric norm of a vector x.

We define tail as follows

Definition 13.4 (Tail of a vector)**.** For a given $x \in \mathbb{R}^d$ and an integer k, we use $x_{\overline{[k]}}$ or $x_{\mathrm{tail}(k)}$ to denote the vector that without largest top-k values (in absolute).

13.7.2 Probability Tools

We state some useful inequalities in probability theory in below.

Theorem 13.7 (Levy's isoperimetric inequality, [GMS86])**.** *For a continuous function $f : S^{d-1} \to \mathbb{R}$, Let M_f be the median of f, i.e., $\mu(\{x : f(x) \le M_f\}) \ge 1/2$ and $\mu(\{x : f(x) \ge M_f\}) \ge 1/2$, where $\mu(\cdot)$ is the Haar probability measure on the unit sphere S^{d-1}. Then*

$$\mu(\{x : f(x) = M_f\}_\epsilon) \ge 1 - \sqrt{\pi/2}e^{-\epsilon^2 d/2},$$

where for a set $A \subset S^{d-1}$ we denote $A_\epsilon := \{x : l_2(x, A) \le \epsilon\}$ and $l_2(x, A) := \inf_{y \in A} \|x - y\|_2$.

13.7.3 Stable Distributions

We define p-stable distributions.

Definition 13.5 ([Ind06])**.** A distribution $\mathcal{D}_p$ is called p-stable, if there exists $p \ge 0$ such that for any d real numbers $a_1, a_2, \dots, a_d$ and i.i.d. variables $x_1, x_2, \dots, x_d$ from distribution $\mathcal{D}_p$. the random variable $\sum_{i=1}^d a_i x_i$ has the same distribution as the variable $\|a\|_p y$, where y is a random variable from distribution $\mathcal{D}_p$.

13.8 Symmetric Norms

In this section, we give several technical tools for symmetric norms. In Section 13.8.1, we show the monotonicity of symmetric norms. In Section 13.8.2, we show the concentration properties of symmetric norms. In Section 13.8.3, we show some properties of the median of symmetric norms.

13.8.1 Monotonicity property of symmetric norm

Lemma 13.8 (Monotonicity of Symmetric Norms, see e.g. Proposition IV.1.1 in [Bha97]). *If $\| \cdot \|_{\mathrm{sym}}$ is a symmetric norm and $x, y \in \mathbb{R}^d$ satisfy that for all $i \in [d]$, $|x_i| \leq |y_i|$, then $\|x\|_{\mathrm{sym}} \leq \|y\|_{\mathrm{sym}}$.*

13.8.2 Concentration property of symmetric norms

Let us give the results of the concentration of measure as followed. The following tools and proofs can be found in [BBC+17]. However, for completeness, we state them below.

Lemma 13.9 (Concentration of M_l). *For every norm l on $\mathbb{R}^d$, if $x \in S^{d-1}$ is drawn uniformly at random according to Haar measure on the sphere, then*

$$\Pr[|\|x\|_l - M_l| > \frac{2\mathfrak{b}_l}{\sqrt{d}}] < \frac{1}{3}$$

Proof. With Theorem 13.7, we know that, for a random x distributed according to the Haar measure on the l_2-sphere, with probability at least $1 - \sqrt{\pi/2}e^{-2} > \frac{2}{3}$, we can always find some $y \in S^{d-1}$, such that

$$\|x - y\|_2 \leq \frac{2}{\sqrt{d}}, \text{ and}$$
$$\|y\|_l = M_l.$$

If we view the norm l as a function, then it is obvious that it is $\mathfrak{b}_l$-Lipschitz with respect to $\|\cdot\|_2$, so that we have

$$\begin{aligned}|\|x\|_l - M_l| &= |\|x\|_l - \|y\|_l| \\ &\leq \|x-y\|_l \\ &\leq \mathfrak{b}_l \|x-y\| \\ &\leq \frac{2\mathfrak{b}_l}{\sqrt{d}}\end{aligned}$$

where the first step follows from the definition of y, the second step follows from triangle inequality, the third step follows from that norm l is $\mathfrak{b}_l$-Lipschitz with respect to $\|\cdot\|_2$, and the last step follows from the definition of y.

Thus, we complete the proof. $\square$

Lemma 13.10 (Concentration inequalities for norms). *For every $d > 0$, and norm l on $\mathbb{R}^n$, there is a vector $x \in S^{d-1}$ satisfying*

- $|\|x\|_\infty - M_{l_\infty^{(d)}}| \leq 2/\sqrt{d}$
- $|\|x\|_l - M_{l^{(d)}}| \leq 2\mathfrak{b}_{l^{(d)}}/\sqrt{d}$, *and*
- $|\{i : |x_i| > \frac{1}{K\sqrt{d}}\}| > \frac{d}{2}$ *for some universal constant K.*

Proof. Let x be drawn uniformly randomly from a unit sphere. From Lemma 13.9, we have

$$\begin{aligned} &\Pr[|\|x\|_l - M_{l_\infty^{(d)}}| > \frac{2}{\sqrt{d}}] &< \frac{1}{3} \\ \textit{and} \quad &\Pr[|\|x\|_l - M_{l^{(d)}}| > \frac{2\mathfrak{b}_l}{\sqrt{d}}] &< \frac{1}{3}. \end{aligned}$$

Define $\tau(x,t) := |\{i \in [d] : |x_i| < t\}|$. Now we are giving the proof to show that, for a constant K, over a choice of x, we have $\tau(x, \frac{1}{K\sqrt{d}}) < \frac{d}{2}$ with probability lager than $2/3$.

Let's consider an random vector $z \in \mathbb{R}^d$, such that each entry z_i is independent standard normal random variable. It is well known that $\frac{z}{\|z\|_2}$ is distributed uniformly over a sphere, so it has the same distribution as x. There is a universal constant K_1 such that

$$\Pr[\|z\|_2 > K_1\sqrt{d}] < \frac{1}{6},$$

and similarly, there is a constant K_2, such that

$$\Pr[|z_i| < \frac{1}{K_2}] < \frac{1}{12}.$$

Therefore, by Markov bound we have

$$\Pr[\tau(z, \frac{1}{K_2}) > \frac{d}{2}] < \frac{1}{6}.$$

By union bound, with probability larger than $2/3$, it holds simultaneously that

$$\|z\|_2 \ \leq K_1\sqrt{d} \quad \text{and} \quad \tau(z, \frac{1}{K_2}) \ < \frac{d}{2},$$

which imply:

$$\tau(\frac{z}{\|z\|_2}, \frac{1}{K_1 K_2 \sqrt{d}}) < \frac{d}{2}.$$

Now, by union bound, a random vector x satisfies all of the conditions in the statement of the lemma with positive probability.

Thus, we complete the proof. $\square$

13.8.3 Median of symmetric norm

We state a well-known fact before introducing Lemma 13.12.

Fact 13.11 (Concentration for median on infinity norm, [GMS86])**.** *There are absolute constants $0 < \gamma_1 < \gamma_2$ such that for every integer $d \geq 1$,*

$$\gamma_1\sqrt{\log(d)/d} \leq M_{l_\infty^{(d)}} \leq \gamma_2\sqrt{\log(d)/d}$$

We now give the following Lemma, which says that the l-norm of the (normalized) unit vector $\xi^{(d)}$ is closely related to the median of the norm. We are considering this vector because that it is related to a single layer of $\mathcal{L}$.

Lemma 13.12 (Flat Median Lemma). *Let $l : \mathbb{R}^d \to \mathbb{R}$ be a symmetric norm. Then*

$$\lambda_1 M_l / \sqrt{\log(d)} \leq l(\xi^{(d)}) \leq \lambda_2 M_l$$

where $\lambda_1, \lambda_2 > 0$ are absolute constants.

We notice that the first inequality is tight for l_∞.

Proof. Using Lemma 13.10, we can find a vector $x \in S^{d-1}$ and a constant $\lambda > 0$ satisfying

- $|\|x\|_\infty - M_{l_\infty}| \leq \lambda \sqrt{1/d}$
- $|\|x\|_l - M_l| \leq \lambda \mathfrak{b}_l / \sqrt{d}$, and
- $|\{i : |x_i| > \frac{1}{K\sqrt{d}}\}| > \frac{d}{2}$ for some universal constant K.

With Fact 13.11, $M_{l_\infty} = \Theta(\sqrt{\log(d)/d})$. On the other hand, let $\mathrm{mmc}(l)$ be defined as Definition 13.3, we have

$$\mathrm{mmc}(l) \leq \gamma \sqrt{d}$$

for sufficiently small γ, thus

$$\frac{\lambda \mathfrak{b}_l}{\sqrt{d}} < M_l.$$

So with constants $\gamma_1, \gamma_2 > 0$, we have

$$\begin{aligned} \gamma_1 M_l &\leq \|x\|_l \leq \gamma_2 M_l \\ \textit{and} \quad \gamma_1 \sqrt{\log(d)/d} &\leq \|x\|_\infty \leq \gamma_2 \sqrt{\log(d)/d}. \end{aligned}$$

So that we have

$$|x| \leq \gamma_2 \sqrt{\log(d)} \cdot \xi^{(d)},$$

where the inequality is entry-wise, and with monotonicity of symmetric norms (Lemma 13.8), we have

$$\gamma_1 \le \|x\|_l \le \gamma_2 \sqrt{\log(d)} \cdot \|\xi^{(d)}\|_l.$$

Now we move to the second condition of the lemma, we first let $M = \{i : |x_i| > \frac{1}{K\sqrt{d}}\}$. As $|M| > \frac{d}{2}$, we can find a permutation π satisfying

$$[d] - M \in \pi(M).$$

We define a vector $\pi(x)$ to be the vector by applying the permutation π to each entry of x. Denote by $|x|$ the vector of taking absolute value of x entry-wise. Notice $|x| + \pi(|x|) > \frac{\xi^{(d)}}{K}$ entry-wise, so that we have

$$\begin{aligned} \frac{1}{K} \cdot \|\xi^{(d)}\|_l &\le \||x| + \pi(|x|)\|_l \\ &\le \||x|\|_l + \|\pi(|x|)\|_l \\ &= 2\|x\|_l \\ &\le 2\gamma_2 M_l, \end{aligned}$$

where the first step follows from the monotonicity of symmetric norm (Lemma 13.8), the second step follows from the triangle inequality, the third step follows from the definition of π, and the last step follows from the definition of γ_2.

Thus, we complete the proof. □

The following lemma shows the monotonicity of the median (in d), a very useful property in the norm approximation.

Lemma 13.13 (Monotonicity of Median)**.** *Let $l : \mathbb{R}^d \to \mathbb{R}$ be a symmetric norm. For all $d_1 \le d_2$, where $d_1, d_2 \in [n]$, let $\mathrm{mmc}(l)$ be defined as Definition 13.3, we have*

$$M_{l^{(d_1)}} \le \lambda \cdot \mathrm{mmc}(l) \cdot \sqrt{\log(d_1)} M_{l^{(d_2)}},$$

where $\lambda > 0$ is an absolute constant.

Proof. By Lemma 13.12 and the fact that $\xi^{(d_1)}$ is also a vector in S^{d_2-1},

$$\lambda M_{l^{(d_1)}}/\sqrt{\log(d_1)} \leq \|\xi^{(d_1)}\|_l \leq \mathfrak{b}_{l^{(d_2)}} \leq \mathrm{mmc}(l) M_{l^{(d_2)}}.$$

Thus, we complete the proof. □

13.9 Analysis of Layer Approximation

In this section, we show how to estimate the symmetric norm $l(\cdot)$ of a vector using the layer vectors. Section 13.9.1 gives the definitions of layer vectors and important layers. Section 13.9.2 shows that we can approximate the exact value of the norm and the layer vector. Section 13.9.3 defines the contributing layer and shows its concentration property. Section 13.9.4 proves that contributing layers are also important.

Throughout this section, let $\epsilon \in (0,1)$ be the precision, $\alpha > 0$ and $\beta \in (0,1]$ be some parameters depending on d, ϵ and $\mathrm{mmc}(l)$, where $\mathrm{mmc}(l)$ is defined as Definition 13.3. Furthermore, we assume $\mathrm{mmc}(l) \leq \gamma\sqrt{d}$, for constant parameter $0 \leq \gamma \ll 1/2$ small enough.[2]

13.9.1 Layer vectors and important layers

Definition 13.6 (Important Layers)**.** For $v \in \mathbb{R}^d$, define layer $i \in \mathbb{N}_+$ as

$$B_i := \{j \in [d] : \alpha^{i-1} < |v_j| \leq \alpha^i\},$$

and denote its size by $b_i := |B_i|$. We denote the number of non-zero b_i's by t, the number of non-empty layers. And we say that layer-i is β-important if

- $b_i > \beta \cdot \sum_{j=i+1}^t b_j$
- $b_i\alpha^{2i} \geq \beta \cdot \sum_{j\in[i]} b_j\alpha^{2j}$

[2]We note that beyond this regime, the streaming lower bound in Theorem 13.32 implies that a linear-sized memory (time) is required to approximate the norm.

With out loss of generality, We restrict the entries of the vector v to be in $[-m, m]$, and that $m = \mathit{poly}(d)$. Then we know that the number of non-zero b_i's is at most $P = O(\log_\alpha(d))$. In the view of ℓ_2-norm, for an arbitrary vector $v \in \mathbb{R}^d$, if we normalize it to a unit vector, then the absolute value of each non-zero entry is at least $1/\mathit{poly}(d)$. In order to simplify our analysis and algorithm for approximating $\|v\|_{\mathrm{sym}}$, we introduce the notations we use as follows.

Definition 13.7 (Layer Vectors and Buckets)**.** For each $i \in [P]$, let $\alpha^i \cdot \mathbf{1}_{b_i} \in \mathbb{R}^{b_i}$ denote a vector that has length b_i and every entry is α^i. Define the layer vector for $v \in \mathbb{R}^d$ with integer coordinates to be

$$\mathcal{L}(v) := (\alpha^1 \cdot \mathbf{1}_{b_1}, \alpha^2 \cdot \mathbf{1}_{b_2}, \cdots, \alpha^P \cdot \mathbf{1}_{b_P}, 0 \cdot \mathbf{1}_{d-\sum_{j\in[P]} b_j}) \in \mathbb{R}^d;$$

and define the i-th bucket of $\mathcal{L}(v)$ to be

$$\mathcal{L}_i(v) := (0 \cdot \mathbf{1}_{b_1+b_2+\cdots+b_{i-1}}, \alpha^i \cdot \mathbf{1}_{b_i}, 0 \cdot \mathbf{1}_{d-\sum_{j\in[i]} b_j}) \in \mathbb{R}^d;$$

We also define $\mathcal{J}(v)$ and $\mathcal{J}_i(v)$ as above by replacing $\{b_i\}$ with the approximated values $\{c_i\}$. Denote $\mathcal{L}(v)\backslash\mathcal{L}_i(v)$ as the vector with the i-th bucket of $\mathcal{L}(v)$ replaced by 0. We also denote $(\mathcal{L}(v)\backslash\mathcal{L}_i(v)) \cup \mathcal{J}_i(v)$ as the vector by replacing the i-th bucket of $\mathcal{L}(v)$ with $\mathcal{J}_i(v)$, i.e.,

$$(\mathcal{L}(v)\backslash\mathcal{L}_i(v)) \cup \mathcal{J}_i(v) := (\alpha^1 \cdot \mathbf{1}_{b_1}, \alpha^2 \cdot \mathbf{1}_{b_2}, \cdots, \alpha^i \cdot \mathbf{1}_{c_i}, \cdots, \alpha^P \cdot \mathbf{1}_{b_P}, 0 \cdot \mathbf{1}_{d-\sum_{j\in[P]} b_j+b_i-c_i}) \in \mathbb{R}^d;$$

13.9.2 Approximated layers provides a good norm approximation

We now prove that, $\|v\|_{\mathrm{sym}}$ can be approximated by using layer vector V. We first choose a base to be $\alpha := 1 + O(\epsilon)$.

Lemma 13.14 (Approximattion with Layer Vector)**.** *For all $v \in \mathbb{R}^d$, we have*

$$\|\mathcal{L}(v)\|_{\mathrm{sym}}/\alpha \leq \|v\|_{\mathrm{sym}} \leq \|\mathcal{L}(v)\|_{\mathrm{sym}}.$$

Proof. The lemma follows from the monotonicity of symmetric norms(Lemma 13.8) directly. □

The next key lemma shows that $\|\mathcal{J}(v)\|_{\text{sym}}$ is a good approximation to $\|\mathcal{L}(v)\|_{\text{sym}}$.

Lemma 13.15 (Bucket Approximation)**.** *For every layer $i \in [P]$,*

- *if $c_i \leq b_i$, then $\|(\mathcal{L}(v)\backslash\mathcal{L}_i(v)) \cup \mathcal{J}_i(v)\|_{\text{sym}} \leq \|\mathcal{L}(v)\|_{\text{sym}}$;*
- *if $c_i \geq (1-\epsilon)b_i$, then $\|(\mathcal{L}(v)\backslash\mathcal{L}_i(v)) \cup \mathcal{J}_i(v)\|_{\text{sym}} \geq (1-\epsilon)\|\mathcal{L}(v)\|_{\text{sym}}$.*

Proof. With the monotonicity of norm (Lemma 13.8), the upper bound is quite obvious. So we just focus on the lower bound. Let us take the vector

$$\mathcal{J}_i(v) := (0 \cdot \mathbf{1}_{c_1+c_2+\cdots+c_{i-1}}, \alpha^i \cdot \mathbf{1}_{c_i}, 0, \cdots, 0) \in \mathbb{R}^d;$$

Here we define $\mathcal{K}(v) := \mathcal{L}(v) - \mathcal{L}_i(v)$. Then notice that $\mathcal{K}(v) + \mathcal{J}_i(v)$ is a permutation of the vector $(\mathcal{L}(v)\backslash\mathcal{L}_i(v)) \cup \mathcal{J}_i(v)$. We will then show that, under assumptions of the lemma, we have

$$\|\mathcal{K}(v) + \mathcal{J}_i(v)\|_{\text{sym}} \geq (c_i/b_i)\|\mathcal{L}(v)\|_{\text{sym}}.$$

Assume a vector $v \in \mathbb{R}^d$ and a permutation $\pi \in \Sigma_d$, we define a vector $\pi(x)$ to be the vector by applying the permutation π to each entry of x. Using the property of the symmetric norm, we have that $\|v\|_{\text{sym}} = \|\pi(v)\|_{\text{sym}}$. Consider a set of permutations that are cyclic shifts over the non-zero coordinates of $\mathcal{L}_i$, and do not move any other coordinates. That is, there is exactly b_i permutations in S, and for every $\pi \in S$, we have $\pi(\mathcal{K}(v)) = \mathcal{K}(v)$. By the construction of S, we have,

$$\sum_{\pi \in S} \pi(\mathcal{J}_i(v)) = c_i \mathcal{L}_i(v)$$

and therefore $\sum_{\pi \in S} \pi(\mathcal{K}(v) + \mathcal{J}_i(v)) = c_i\mathcal{L}_i(v) + b_i\mathcal{K}(v)$. As the vectors $\mathcal{L}_i(v)$ and $\mathcal{K}(v)$ have disjoint support, by monotonicity of symmetric norm (Lemma 13.8) with respect to each coordinates we can deduce $\|c_i\mathcal{L}_i(v) + b_i\mathcal{K}(v)\|_{\text{sym}} \geq \|c_i(\mathcal{L}_i(v) + \mathcal{K}(v))\|_{\text{sym}}$.

By plugging those together,

$$\begin{aligned}
c_i\|\mathcal{L}_i(v)+\mathcal{K}(v)\|_{\mathrm{sym}} &\leq \|c_i\mathcal{L}_i(v)+b_i\mathcal{K}(v)\|_{\mathrm{sym}}\\
&= \|\sum_{\pi\in S}\pi(\mathcal{J}_i(v)+\mathcal{K}(v))\|_{\mathrm{sym}}\\
&\leq \sum_{\pi\in S}\|\pi(\mathcal{J}_i(v)+\mathcal{K}(v))\|_{\mathrm{sym}}\\
&= b_i\|\pi(\mathcal{J}_i(v)+\mathcal{K}(v))\|_{\mathrm{sym}}
\end{aligned}$$

where the first step follows from $c_i\|\mathcal{L}_i(v)+\mathcal{K}(v)\|_{\mathrm{sym}} = \|c_i(\mathcal{L}_i(v)+\mathcal{K}(v))\|_{\mathrm{sym}}$ and the monotonicity of the norm l, the second step follows from $\sum_{\pi\in S}\pi(\mathcal{J}_i(v)+\mathcal{K}(v)) = c_i\mathcal{L}_i(v)+b_i\mathcal{K}(v)$, the third step follows from triangle inequality, and the last step follows from the property of symmetric norm and $|S|=b_i$.

Hence,

$$\|\mathcal{J}_i(v)+\mathcal{K}(v)\|_{\mathrm{sym}} \geq \frac{c_i}{b_i}\|\mathcal{L}(v)\|_{\mathrm{sym}} \geq (1-\epsilon)\|\mathcal{L}(v)\|_{\mathrm{sym}},$$

Thus, we complete the proof. $\square$

13.9.3 Contributing layers

Definition 13.8 (Contributing Layers)**.** For $i\in[P]$, layer i is called β-contributing if

$$\|\mathcal{L}_i(v)\|_{\mathrm{sym}} \geq \beta\|\mathcal{L}(v)\|_{\mathrm{sym}}.$$

Lemma 13.16 (Concentration with contributing layers)**.** *Let $\mathcal{L}^*(v)$ be the vector obtained from V by removing all layers that are not β-contributing. Then*

$$(1-O(\log_\alpha(d))\cdot\beta)\cdot\|\mathcal{L}(v)\|_{\mathrm{sym}} \leq \|\mathcal{L}^*(v)\|_{\mathrm{sym}} \leq \|\mathcal{L}(v)\|_{\mathrm{sym}}.$$

Proof. Let $i_1,\ldots,i_k\in[P]$ be the layers that are not β- contributing.

Then we apply the triangle inequality and have,

$$\begin{aligned}
\|\mathcal{L}(v)\|_{\mathrm{sym}} &\geq \|\mathcal{L}(v)\|_{\mathrm{sym}} - \|\mathcal{L}_{i_1}(v)\|_{\mathrm{sym}} - \cdots - \|\mathcal{L}_{i_k}(v)\|_{\mathrm{sym}}\\
&\geq (1-k_\beta)\|\mathcal{L}(v)\|_{\mathrm{sym}}
\end{aligned}$$

The proof follows by bounding k by $P = O(\log_\alpha(n))$, which is the total number of non-zero b_i's. $\square$

13.9.4 Contributing Layers Are Important

In this section, we give two lemmas to show that every β-contributing layer (Definition 13.8) is β'-important (Definition 13.6), where β' is depending on $\mathrm{mmc}(l)$ (Definition 13.3). The first property of the important layer is proved in Lemma 13.17, and the second property is proved in Lemma 13.18.

Lemma 13.17 (Importance of contributing layers (Part 1))**.** *For $i \in [P]$, if layer i is β-contributing, then for some absolute constant $\lambda > 0$, we have*

$$b_i \geq \frac{\lambda \beta^2}{\mathrm{mmc}(l)^2 \log^2(d)} \cdot \sum_{j=i+1}^{P} b_j,$$

where $\mathrm{mmc}(l)$ *is defined as Definition 13.3.*

Proof. We first fix a layer i which is β-contributing. Let $\mathcal{U}(v)$ be the vector $\mathcal{L}(v)$ after removing buckets $j = 0, \ldots, i$. By Lemma 13.12, there is an absolute constant $\lambda_1 > 0$ such that

$$\begin{aligned} \|\mathcal{L}_i(v)\|_{\mathrm{sym}} &= \alpha^i \cdot \sqrt{b_i} \cdot l(\xi^{(b_i)}) \\ &\leq \lambda_1 \cdot \alpha^i \cdot \sqrt{b_i} \cdot M_{l^{(b_i)}}, \end{aligned}$$

and similarly

$$\|\mathcal{U}(v)\|_{\mathrm{sym}} \geq \frac{\lambda_2 \alpha^i}{\sqrt{\log(d)}} \cdot (\sum_{j=i+1}^{P} b_j)^{1/2} M_{l^{(\Sigma_{j=i+1}^{P} b_j)}}.$$

With these two inequalities, we can have the following deduction.

First, we have

$$\|\mathcal{L}_i(v)\|_{\mathrm{sym}} \geq \beta \cdot \|\mathcal{L}(v)\|_{\mathrm{sym}} \geq \beta \cdot \|\mathcal{U}(v)\|_{\mathrm{sym}}.$$

Second, we assume that $b_i < \sum_{j=i+1}^{P} b_j$, as otherwise we are done:

$$b_i \geq \sum_{j=i+1}^{P} b_j \geq \frac{\lambda \beta^2}{\mathrm{mmc}(l)^2 \log^2(d)} \cdot \sum_{j=i+1}^{P} b_j.$$

Then, by the monotonicity of the median (Lemma 13.13), we have

$$M_{l^{(b_i)}} \leq \lambda_3 \cdot \mathrm{mmc}(l) \cdot \sqrt{\log(d)} \cdot M_{l^{(\sum_{j=i+1}^{P} b_j)}}$$

for some absolute constant $\lambda_3 > 0$.

Putting it all together, we get

$$\beta \cdot \frac{\lambda_2 \alpha^i}{\sqrt{\log(d)}} \cdot (\sum_{j=i+1}^{P} b_j)^{1/2} \leq \lambda_1 \cdot \alpha^i \sqrt{b_i} \cdot \lambda_3 \cdot \mathrm{mmc}(l) \cdot \sqrt{\log(d)}.$$

Therefore, we finish the proof. □

Lemma 13.18 (Importance of contributing layers (Part 2))**.** *For a symmetric l, let $\mathrm{mmc}(l)$ be defined as Definition 13.3. If layer $i \in [P]$ is β-contributing, then there is an absolute constant $\lambda > 0$ such that*

$$b_i \alpha^{2i} \geq \frac{\lambda \beta^2}{\mathrm{mmc}(l)^2 \cdot \log_\alpha(n) \cdot \log^2(n)} \cdot \sum_{j \in [i]} b_j \alpha^{2j}.$$

Proof. We first fix a layer i which is β-contributing, and let $h := \arg\max_{j \leq i} \sqrt{b_j} \alpha^j$. We consider the two different cases as follows.

First, if $b_i \geq b_h$ then the lemma follows obviously by

$$\begin{aligned} & \sum_{j \in [i]} b_j \alpha^{2j} \\ & \leq t \cdot b_h \cdot \alpha^{2h} \\ & \leq O(\log_\alpha(d)) \cdot b_i \cdot \alpha^{2i}. \end{aligned}$$

The second case is when $b_i < b_h$. With Definition 13.8 and Lemma 13.12, we have

$$\begin{aligned}
&\lambda_1 \cdot \alpha^i \cdot \sqrt{b_i} \cdot M_{l^{(b_i)}} \\
\geq\ & \|\mathcal{L}_i\|_{\mathrm{sym}} \\
\geq\ & \beta \cdot \|\mathcal{L}\|_{\mathrm{sym}} \\
\geq\ & \lambda_2 \cdot \beta \cdot \alpha^h \cdot \sqrt{\frac{b_h}{\log(d)}} \cdot M_{l^{(b_h)}},
\end{aligned}$$

for some absolute constants $\lambda_1, \lambda_2 > 0$, where the first step follows from Lemma 13.12, the second step follows from Definition 13.8, and the last step follows from Lemma 13.12.

following from monotonicity of the median (Lemma 13.13), we can plugging in $M_{l^{(b_i)}} \leq \lambda_3 \cdot \mathrm{mmc}(l) \cdot \sqrt{\log(d)} \cdot M_{l^{(b_h)}}$, for some absolute constant $\lambda_3 > 0$, so that we have

$$\begin{aligned}
\lambda_1 \cdot \alpha^i \cdot \sqrt{b_i} \cdot M_{l^{(b_i)}} &\geq \frac{\lambda_2 \cdot \beta \cdot \sqrt{b_h} \cdot \alpha^h}{\sqrt{\log(d)}} \cdot \frac{M_{l^{(b_i)}}}{\lambda_3 \cdot \mathrm{mmc}(l) \cdot \sqrt{\log(d)}}, \\
\sqrt{b_i} \cdot \alpha^i &\geq \frac{\lambda_2 \cdot \beta \cdot \sqrt{b_h} \cdot \alpha^h}{\lambda_1 \lambda_3 \cdot \mathrm{mmc}(l) \cdot \log(d)}.
\end{aligned}$$

Square the above inequality and we can see that $b_h \cdot \alpha^{2h} \geq \frac{1}{O(\log_\alpha(d))} \cdot \sum_{j \in [i]} b_j \alpha^{2j}$.

Thus we complete the proof. $\square$

13.10 Formal Main Result and Algorithms

In this section, we state the formal version of our main theorem and algorithms. Section 13.10.1 presents our main result: a data structure for distance estimation with symmetric norm. Section 13.10.2 introduces the sparse recovery tools for sketching.

13.10.1 Formal version of our main result

Here we divide function QUERY presented in Theorem 13.1 into two versions. One takes only the query point $q \in \mathbb{R}^d$ as input to ask all the distances. Another takes a query point $q \in \mathbb{R}^d$ and a set $\mathsf{S} \subset [n]$ to ask for the distance with the specific

Algorithm 74 Data structure for symmetric norm estimation

```
1: data structure DistanceOnSymmetricNorm                    ▷ Theorem 13.19
2:
3: private:
4: procedure LayerVectorApprox($\alpha, b_1, b_2, \dots, b_P, d$)       ▷ Lemma 13.14
5:     For each $i \in [P]$, let $\alpha^i \cdot \mathbf{1}_{b_i} \in \mathbb{R}^{b_i}$ denote a vector that has length $b_i$ and every
   entry is $\alpha^i$
6:     $\mathcal{L} \leftarrow (\alpha^1 \cdot \mathbf{1}_{b_1}, \cdots, \alpha^P \cdot \mathbf{1}_{b_P}, 0, \dots, 0) \in \mathbb{R}^d$      ▷ Generate the layer vector
7:     return $\|\mathcal{L}\|_{\mathrm{sym}}$              ▷ Return the norm of the estimated layer vector
8: end procedure
9: end data structure
```

Algorithm 75 Data structure for symmetric norm estimation: members, formal version of Algorithm 72

```
1: data structure DistanceOnSymmetricNorm                    ▷ Theorem 13.19
2: members
3:     $d, n \in \mathbb{N}_+$                    ▷ $n$ is the number of points, $d$ is dimension
4:     $X = \{x_i \in \mathbb{R}^d\}_{i=1}^n$                          ▷ Set of points being queried
5:     $L \in \mathbb{N}_+$                              ▷ number of layers we subsample
6:     $R \in \mathbb{N}_+$                          ▷ number of substreams in one layer
7:     $\epsilon$, $\delta$
8:     $\beta$                                          ▷ used to cut important layer
9:     $U \in \mathbb{N}_+$                              ▷ number of parallel processing
10:    BatchHeavyHitter $\{S_{r,l,u}\}_{r \in [R], l \in [L], u \in [U]}$
11:    $\{H_{r,l,u} \subset [d] \times \mathbb{R}\}_{r \in [R], l \in [L], u \in [U]}$ ▷ each set $H$ has a size of $2/\beta$, and is used to
   store the output of BatchHeavyHitter
12:    $\gamma$                               ▷ parameter used when cutting layer vector
13:    bmap $\in \{0,1\}^{R \times L \times U \times d}$
14:    $\overline{x}_{r,l,u} \in \mathbb{R}^{n \times d}$, for each $r \in [R], l \in [L], u \in [U]$              ▷ Substreams
15: end members
16: end data structure
```

set of points. The latter can be viewed as a more general version of the former. In the former parts, we have proved the correctness of the query (Lemma 13.6), and the running time of version for all points (Lemma 13.4). Now we state the both in the following theorem, and the running time analysis for the latter will be stated in Section 13.11.

Algorithm 76 Data structure for symmetric norm estimation: init, formal version of Algorithm 72

```
1: data structure DistanceOnSymmetricNorm    ▷ Theorem 13.19
2: public:
3: procedure Init($\{x_1, \cdots, x_i\} \subset \mathbb{R}^d, n \in \mathbb{N}_+, d \in \mathbb{N}_+, \delta \in (0, 0.1), \epsilon \in (0, 0.1)$)    ▷ Lemma 13.2
4:     $n \leftarrow n$, $d \leftarrow d$, $\delta \leftarrow \delta$, $\epsilon \leftarrow \epsilon$
5:     for $i = 1 \to n$ do
6:         $x_i \leftarrow x_i$
7:     end for
8:     $\epsilon_1 \leftarrow O(\frac{\epsilon^2}{\log d})$    ▷ We define this notation for purpose of analysis
9:     $L \leftarrow \log(d)$, $R \leftarrow \Theta(\epsilon_1^{-2} \log(n/\delta) \log^2 d)$, $U \leftarrow \lceil \log(nd^2/\delta) \rceil$
10:    $\beta \leftarrow O(\frac{\epsilon^5}{\mathrm{mmc}(l)^2 \log^5 d})$
11:    for $r \in [R], l \in [L], u \in [U]$ do
12:        $S_{r,l,u}$.Init($\sqrt{\beta}, n + 2, d$)    ▷ Theorem 13.20
13:    end for
14:    for $r \in [R], u \in [U], j \in [d], l \in [L]$ do
15:        Draw $\xi \in [0, 1]$
16:        if $\xi \in [0, 2^{-l}]$ then
17:            $\mathrm{bmap}[r, l, u, j] \leftarrow 1$
18:        else
19:            $\mathrm{bmap}[r, l, u, j] \leftarrow 0$
20:        end if
21:    end for
22:    for $r \in [R], u \in [U], i \in [n], j \in [d], l \in [L]$ do
23:        if $\mathrm{bmap}[r, l, u, j] = 1$ then
24:            $S_{r,l,u}$.EncodeSingle($i, j, x_{i,j}, d$)    ▷ Theorem 13.20
25:            $[\overline{x}_{r,l,u}]_{i,j} \leftarrow x_{i,j}$    ▷ Create a copy of subvectors
26:        else
27:            $[\overline{x}_{r,l,u}]_{i,j} \leftarrow 0$
28:        end if
29:    end for
30: end procedure
31: end data structure
```

Theorem 13.19 (Main result, formal version of Theorem 13.1)**.** *There is a data structure (Algorithm 75, 76, 78, 77) uses $O(\epsilon^{-9} n(d + \mathrm{mmc}(l)^2) \log^{14}(nd/\delta))$ spaces for the Online Approximate Adaptive Symmetric Norm Distance Estimation Problem*

Algorithm 77 Data structure for symmetric norm estimation: update

```
1:  data structure DistanceOnSymmetricNorm                    ▷ Theorem 13.19
2:  public:
3:  procedure Update($i \in [n], z \in \mathbb{R}^d$)          ▷ Lemma 13.3
4:                                                   ▷ You want to replace $x_i$ by $z$
5:      for $r \in [R], u \in [U], j \in [d], l \in [L]$ do
6:          if $\mathrm{bmap}[r, l, u] = 1$ then
7:              $S_{r,l,u}$.EncodeSingle($i, j, z_j, d$)             ▷ Theorem 13.20
8:              $[\overline{x}_{r,l,u}]_{i,j} \leftarrow z_j$        ▷ Create a copy of subvectors
9:          else
10:             $[\overline{x}_{r,l,u}]_{i,j} \leftarrow 0$
11:         end if
12:     end for
13: end procedure
14: end data structure
```

(Definition 13.1) with the following procedures:

- Init($\{x_1, x_2, \dots, x_n\} \subset \mathbb{R}^d, \epsilon \in (0,1), \delta \in (0,1)$)*: Given n data points* $\{x_1, x_2, \dots, x_n\} \subset \mathbb{R}^d$*, an accuracy parameter* ϵ *and a failure probability* δ *as input, the data structure preprocesses in time* $O(\epsilon^{-9} n (d + \mathrm{mmc}(l)^2) \log^{14}(nd/\delta))$.
- UpdateX($z \in \mathbb{R}^d, i \in [n]$)*: Given an update vector* $z \in \mathbb{R}^d$ *and index* $i \in [n]$*, the* UpdateX *takes* z *and* i *as input and updates the data structure with the new* i*-th data point in* $O(\epsilon^{-4} d \log^9(nd/\delta))$ *time.*
- Query($q \in \mathbb{R}^d$) *(Querying all points): Given a query point* $q \in \mathbb{R}^d$*, the* Query *operation takes* q *as input and approximately estimates the symmetric norm distances from* q *to all the data points* $\{x_1, x_2, \dots, x_n\} \subset \mathbb{R}^d$ *in time*

$$O(\epsilon^{-9}(d + n \cdot \mathrm{mmc}(l)^2) \log^{14}(nd/\delta))$$

i.e. it provides a set of estimates $\{\mathrm{dst}_i\}_{i=1}^n$ *such that:*

$$\forall i \in [n], (1 - \epsilon)\|q - x_i\|_{\mathrm{sym}} \le \mathrm{dst}_i \le (1 + \epsilon)\|q - x_i\|_{\mathrm{sym}}$$

with probability at least $1 - \delta$.

Algorithm 78 Data structure for symmetric norm estimation: query, formal version of Algorithm 73

```
data structure DistanceOnSymmetricNorm                          ▷ Theorem 13.19
procedure Query($q \in \mathbb{R}^d$)                                ▷ Lemma 13.4, 13.6
    $P \leftarrow O(\log_\alpha(d))$          ▷ $P$ denotes the number of non-empty layer sets
    for $r \in [R], l \in [L], u \in [U]$ do
        $S_{r,l,u}$.Encode($n+1, q, d$)                           ▷ Generate Sketch for $q$
    end for
    $\xi \leftarrow$ chosen uniformly at random from $[1/2, 1]$, $\gamma \leftarrow \Theta(\epsilon)$, $\alpha \leftarrow 1 + \gamma \cdot \xi$
    for $i \in [n]$ do
        for $r \in [R], l \in [L], u \in [U]$ do
            $S_{r,l,u}$.Subtract($n+2, n+1, i$)
            $H_{r,l,u} \leftarrow S_{r,l,u}$.Decode($n+2, \sqrt{\beta}, d$)   ▷ This can be done in $\frac{2}{\beta} poly(\log d)$
              ▷ At this point $H_{r,l,u}$ is a list of index, the value of each index is reset to 0
            for $k \in H_{r,l,u}$ do
                value $\leftarrow [\overline{x}_{r,l,u}]_{i,k}$
                $H_{r,l,u}[k] \leftarrow$ value, $w \leftarrow \lceil \log(\text{value})/\log(\alpha) \rceil$
                if $\alpha^{w-1} \geq \text{value}/(1+\epsilon)$ then
                    $H_{r,l,u} \leftarrow$ null, break
                end if
            end for
            if $H_{r,l,u} \neq$ null then
                $H_{r,l} \leftarrow H_{r,l,u}$
            end if
        end for
        for $l \in [L], k \in [P]$ do
            $A^i_{l,k} \leftarrow |\{k \mid \exists k \in H_{r,l}, \alpha^{k-1} < |H_{r,l}[k]| \leq \alpha^k\}|$
        end for
        for $k \in [P]$ do
            $q^i_k \leftarrow \max_{l \in [L]} \{l \mid A^i_{l,k} \geq \frac{R \log(1/\delta)}{100 \log(d)}\}$     ▷ Definition 13.12
            If $q^i_k$ does not exist, then $\widehat{\eta}^i_k \leftarrow 0$; Else $\widehat{\eta}^i_k \leftarrow \frac{A_{q^i_k,k}}{R(1+\epsilon_1)}$
            If $\widehat{\eta}^i_k = 0$ then $c^i_k \leftarrow 0$; Else $c^i_k \leftarrow \frac{\log(1-\widehat{\eta}^i_k)}{1-w^{-q_k}}$
        end for
        $\text{dst}_i \leftarrow$ LayerVetcorApprox($\alpha, c^i_1, c^i_2, \ldots, c^i_P, d$)
        for $r \in [R], l \in [L], u \in [U]$ do
            $\{H_{r,l,u}\} \leftarrow \{0\}$                       ▷ Reset the sets to use for next point
        end for
    end for
    return $\{\text{dst}_i\}_{i \in [n]}$
end procedure
end data structure
```

- QUERY($q \in \mathbb{R}^d, \mathsf{S} \subseteq [n]$) *(Querying a specific set* S *of points). Given a query point* $q \in \mathbb{R}^d$ *and an index* $i \in [n]$*, the* QUERYONE *operation takes* q *and* i *as input and approximately estimates the symmetric norm distances from* q *to the* i*-th point* $x_i \in \mathbb{R}^d$ *in time*

$$O(\epsilon^{-9}(d + |\mathsf{S}| \cdot \mathrm{mmc}(l)^2) \log^{14}(nd/\delta))$$

i.e. it provides a estimated distance $\mathrm{dst} \in \mathbb{R}^{\mathsf{S}}$ *such that:*

$$(1-\epsilon)\|q - x_i\|_{\mathrm{sym}} \leq \mathrm{dst}_i \leq (1+\epsilon)\|q - x_i\|_{\mathrm{sym}}, \forall i \in \mathsf{S}$$

with probability at least $1 - \delta$*.*

- ESTPAIR($i, j \in [n]$) *Given indices* $i, j \in [n]$*, the* ESTPAIR *operation takes* i *and* j *as input and approximately estimates the symmetric norm distances from* i*-th to the* j*-th point* $x_i, x_j \in \mathbb{R}^d$ *in time*

$$O(\epsilon^{-9} \cdot \mathrm{mmc}(l)^2 \log^{14}(nd/\delta))$$

i.e. it provides a estimated distance pair *such that:*

$$(1-\epsilon)\|x_i - x_j\|_{\mathrm{sym}} \leq \mathrm{pair} \leq (1+\epsilon)\|x_i - x_j\|_{\mathrm{sym}}$$

with probability at least $1 - \delta$*.*

Proof. In Lemma 13.23, Lemma 13.24, Lemma 13.21 and Lemma 13.22 we analyze the running time for INIT, UPDATE and QUERY and ESTPAIR respectively.

Lemma 13.31 shows the space complexity.

In Lemma 13.6 we give the correctness of QUERY, and correctness for ESTPAIR follows directly.

Thus, by putting them all together, we prove the Theorem.

□

13.10.2 Sparse recovery tools

We start by describing a data structure problem

Definition 13.9 (Batch Heavy Hitter)**.** Given an $n \times d$ matrix, the goal is to design a data structure that supports the following operations:

- $\textsc{Init}(\epsilon \in (0, 0.1), n, d)$. Create a set of Random Hash functions and all the n copies of sketches share the same hash functions.
- $\textsc{Encode}(i \in [n], z \in \mathbb{R}^d, d)$. This step encodes z into i-th sketched location and store a size $\mathcal{S}_{\mathrm{space}}$ linear sketch.
- $\textsc{EncodeSingle}(i \in [n], j \in [d], z \in \mathbb{R}, d)$. This step updates one sparse vector $e_j z \in \mathbb{R}^d$ into i-th sketched location.
- $\textsc{Subtract}(i, j, l \in [n])$. This function updates the sketch at i-th location by j-th sketch minus l-th sketch.
- $\textsc{Decode}(i \in [n], \epsilon \in (0, 0.1), d)$. This function returns a set $L \subseteq [d]$ of size $|L| = O(\epsilon^{-2})$ containing all ϵ-heavy hitters $i \in [n]$ under ℓ_p. Here we say i is an ϵ-heavy hitter under ℓ_2 if $|x_i| \geq \epsilon \cdot \|x_{\overline{[\epsilon^{-2}]}}\|_2$ where $x_{\overline{[k]}}$ denotes the vector x with the largest k entries (in absolute value) set to zero. Note that the number of heavy hitters never exceeds $2/\epsilon^2$.

The existing work [KNPW11, Pag13] implies the following result. However their proofs are very decent and complicated. We provide another data structure in Section 13.15 that significantly simplifies the analysis (by only paying some extra log factors). We believe it is of independent interest.

Theorem 13.20. *There is (linear sketch) data structure* $\textsc{BatchHeavyHitter}(\epsilon, n, d)$ *that uses* $\mathcal{S}_{\mathrm{space}}$ *space that support the following operations:*

- $\textsc{Init}(\epsilon \in (0, 0.1), n, d)$. *This step takes* $\mathcal{T}_{\mathrm{init}}(\epsilon, n, d)$ *time.*

- ENCODE($i \in [n], z \in \mathbb{R}^d, d$). *This step takes* $\mathcal{T}_{\mathrm{encode}}(d)$ *time.*
- ENCODESINGLE($i \in [n], j \in [d], z \in \mathbb{R}, d$). *This step takes* $\mathcal{T}_{\mathrm{encodesingle}}(d)$ *time.*
- SUBTRACT($i, j, l \in [n]$). *This step takes* $\mathcal{T}_{\mathrm{encodesingle}}(d)$ *time.*
- DECODE($i \in [n], \epsilon \in (0, 0.1), d$). *This step takes* $\mathcal{T}_{\mathrm{decode}}(\epsilon, d)$ *time.*

The running time of function can be summarize as

- $\mathcal{S}_{\mathrm{space}}(\epsilon, d) = n \cdot O(\epsilon^{-2} \log^2 d)$
- $\mathcal{T}_{\mathrm{init}}(\epsilon, n, d) = n \cdot O(\epsilon^{-2} \log^2 d)$
- $\mathcal{T}_{\mathrm{encode}}(d) = O(d \log^2(d))$
- $\mathcal{T}_{\mathrm{encodesingle}}(d) = O(\log^2(d))$
- $\mathcal{T}_{\mathrm{subtract}}(\epsilon, d) = O(\epsilon^{-2} \log^2 d)$
- $\mathcal{T}_{\mathrm{decode}}(\epsilon, d) = O(\epsilon^{-2} \log^2 d)$

Note that the succeed probability is at least 0.99.

We remark that, to boost the probability from constant to $1 - 1/poly(nd)$ we just need to pay an extra $\log(nd)$ factor.

13.11 More Details of the Time Complexity

In this section, we analyze the running time of the general version of QUERY. Lemma 13.4 is a special case of the following Lemma when $\mathsf{S} = [n]$.

Lemma 13.21 (QUERY time for general version)**.** *Given a query point* $q \in \mathbb{R}^d$ *and a set* $\mathsf{S} \subseteq [n]$*, the procedure* QUERY *(Algorithm 79) runs in time*

$$O(\epsilon^{-9}(d + |\mathsf{S}| \cdot \mathrm{mmc}(l)^2) \log^{14}(nd/\delta)).$$

Proof. The QUERY operation for a stored vector (Algorithm 79) has the following two parts:

- **Part 1:** Line 5 takes $O(RLU \cdot \mathcal{T}_{\mathrm{encode}})$ time to call ENCODE to generate sketches for q.
- **Part 2:** For each $i \in \mathsf{S}$
  - Line 11 takes $O(RLU \cdot \mathcal{T}_{\mathrm{subtract}})$ time to compute sketch of the difference between q and x_i, and store the sketch at index of $n+2$.
  - Line 12 takes $O(RLU \cdot \mathcal{T}_{\mathrm{decode}})$ time to decode the BATCHHEAVYHITTER and get estimated heavy hitters of $q - x_i$.
  - Line 14 to Line 20 takes $O(RLU \cdot 2/\beta)$ time to analyze the BATCHHEAVY-HITTER and get the set of indices, where $2/\beta$ is the size of the set.
  - Line 26 takes $O(LP \cdot 2/\beta)$ time to compute size of the layer sets cut by α.
  - Line 28 to Line 32 takes $O(PL)$ time to compute the estimation of each layer.

  The total running time of this part is:

  $$\begin{aligned}&|\mathsf{S}| \cdot (O(RLU \cdot \mathcal{T}_{\mathrm{subtract}}) + O(RLU \cdot \mathcal{T}_{\mathrm{decode}}) + O(RLU \cdot 2/\beta) + O(LP \cdot 2/\beta) + O(LP)))\\ =\ & O(|\mathsf{S}| \cdot L(RU(\mathcal{T}_{\mathrm{subtract}} + \mathcal{T}_{\mathrm{decode}} + \beta^{-1}) + P\beta^{-1}))\end{aligned}$$

  time in total.

Taking these two parts together we have the total running time of the QUERY procedure:

$$\begin{aligned}&O(RLU \cdot \mathcal{T}_{\mathrm{encode}}) + O(|\mathsf{S}| \cdot L(RU(\mathcal{T}_{\mathrm{subtract}} + \mathcal{T}_{\mathrm{decode}} + \beta^{-1}) + P\beta^{-1}))\\ =\ & O(RLU(\mathcal{T}_{\mathrm{encode}} + |\mathsf{S}| \cdot (\mathcal{T}_{\mathrm{subtract}} + \mathcal{T}_{\mathrm{decode}} + \beta^{-1})) + |\mathsf{S}| \cdot LP\beta^{-1})\\ =\ & O(\epsilon^{-4} \log^6(nd/\delta)(d \log^2(d) \log(nd) + |\mathsf{S}| \cdot \beta^{-1} \log^2(d) \log(nd)) + |\mathsf{S}| \cdot \epsilon^{-1} \log^2(d) \beta^{-1})\\ =\ & O(\epsilon^{-9}(d + |\mathsf{S}| \cdot \mathrm{mmc}(l)^2) \log^{14}(nd/\delta))\end{aligned}$$

where the first step follows from the property of big O notation, the second step follows from the definition of $R, L, U, \mathcal{T}_{\mathrm{encode}}, \mathcal{T}_{\mathrm{encode}}, \mathcal{T}_{\mathrm{subtract}}, \mathcal{T}_{\mathrm{decode}}$ (Theorem 13.20) , P, the third step follows from merging the terms.

Thus, we complete the proof.

$\square$

Lemma 13.22 (EstPair time)**.** *Given a query point $q \in \mathbb{R}^d$, the procedure* EstPair *(Algorithm 80) runs in time*

$$O(\epsilon^{-9} \cdot \mathrm{mmc}(l)^2 \log^{14}(nd/\delta)).$$

Proof. The EstPair operation (Algorithm 80) has the following two parts:

- Line 9 takes $O(RLU \cdot \mathcal{T}_{\mathrm{subtract}})$ time to compute sketch of the difference between x_i and x_j, and store the sketch at index of $n+2$.
- Line 10 takes $O(RLU \cdot \mathcal{T}_{\mathrm{decode}})$ time to decode the BatchHeavyHitter and get estimated heavy hitters of $x_i - x_j$.
- Line 12 to Line 20 takes $O(RLU \cdot 2/\beta)$ time to analyze the BatchHeavyHitter and get the set of indices, where $2/\beta$ is the size of the set.
- Line 26 takes $O(LP \cdot 2/\beta)$ time to compute size of the layer sets cut by α.
- Line 28 to Line 32 takes $O(PL)$ time to compute the estimation of each layer.

The total running time is:

$$\begin{aligned}
& O(RLU \cdot \mathcal{T}_{\mathrm{subtract}}) + O(RLU \cdot \mathcal{T}_{\mathrm{decode}}) + O(RLU \cdot 2/\beta) + O(LP \cdot 2/\beta) + O(LP) \\
= {} & O(RLU(\mathcal{T}_{\mathrm{subtract}} + \mathcal{T}_{\mathrm{decode}} + \beta^{-1}) + LP\beta^{-1}) \\
= {} & O(\epsilon^{-4} \log^6(nd/\delta)(\beta^{-1} \log^2(d) \log(nd) + \epsilon^{-1} \log^2(d) \beta^{-1} \log(nd)) \\
= {} & O(\epsilon^{-9} \cdot \mathrm{mmc}(l)^2 \log^{14}(nd/\delta))
\end{aligned}$$

time in total, where the first step follows from the property of big O notation, the second step follows from the definition of $R, L, U, \mathcal{T}_{\text{encode}}, \mathcal{T}_{\text{encode}}, \mathcal{T}_{\text{subtract}}, \mathcal{T}_{\text{decode}}$ (Theorem 13.20) , P, the third step follows from merging the terms.

Thus, we complete the proof.

$\square$

Lemma 13.23 (Init time, formal version of Lemma 13.2)**.** *Given data points $\{x_1, x_2, \dots, x_n\} \subset \mathbb{R}^d$, an accuracy parameter $\epsilon > 0$, and a failure probability $\delta > 0$ as input, the procedure* INIT *(Algorithm 76) runs in time*

$$O(\epsilon^{-9} n(d + \mathrm{mmc}(l)^2) \log^{14}(nd/\delta)).$$

Proof. The Init time includes these parts:

- Line 12 takes $O(RLU \cdot \mathcal{T}_{\text{init}}(\sqrt{\beta}, n, d))$ to initialize sketches
- Line 17 to Line 19 takes $O(RUdL)$ to generate the bmap;
- Line 24 takes $O(ndRUL \cdot \mathcal{T}_{\text{encodesingle}}(d))$ to generate sketches

By Theorems 13.20, we have

- $\mathcal{T}_{\text{init}}(\sqrt{\beta}, n, d)) = n \cdot O(\beta^{-1} \log^2(d) \log(nd)) = O(n \cdot \mathrm{mmc}(l)^2 \log^7(d) \log(nd) \epsilon^{-5})$,
- $\mathcal{T}_{\text{encodesingle}}(d) = O(\log^2(d) \log(nd))$.

Adding them together we got the time of

$$\begin{aligned}
& O(RLU\mathcal{T}_{\text{init}}(\sqrt{\beta}, n, d)) + O(RUdL) + O(ndRUL \cdot \mathcal{T}_{\text{encodesingle}}(d)) \\
= & O(RLU(\mathcal{T}_{\text{init}}(\sqrt{\beta}, n, d) + nd \cdot \mathcal{T}_{\text{encodesingle}}(d))) \\
= & O(\epsilon^{-4} \log(d/\delta) \log^4(d) \cdot \log(d) \cdot \log(nd^2/\delta) \log(nd)(n \cdot \mathrm{mmc}(l)^2 \log^7(d)\epsilon^{-5} + nd \log^2(d))) \\
= & O(\epsilon^{-9} n(d + \mathrm{mmc}(l)^2) \log^{14}(nd/\delta)),
\end{aligned}$$

where the first step follows from merging the terms, the second step follows from the definition of $R, L, U, \mathcal{T}_{\mathrm{encodesingle}}(d), \mathcal{T}_{\mathrm{init}}$, the third step follows from merging the terms.

Thus, we complete the proof. $\square$

Lemma 13.24 (UPDATE time, formal version of Lemma 13.3)**.** *Given a new data point $z \in \mathbb{R}^d$, and an index i where it should replace the original data point $x_i \in \mathbb{R}^d$. The procedure* UPDATE *(Algorithm 77) runs in time*

$$O(\epsilon^{-4} d \log^9(nd/\delta)).$$

Proof. The UPDATE operation calls BATCHHEAVYHITTER.ENCODE for RLU times, so it has the time of

$$\begin{aligned} O(RLU \cdot \mathcal{T}_{\mathrm{encode}}(d)) &= O(\epsilon^{-4} \log(n/\delta) \log^4(d) \cdot \log(d) \cdot \log(nd^2/\delta) \cdot d \log^2(d) \log(nd)) \\ &= O(\epsilon^{-4} d \log^9(nd/\delta)) \end{aligned}$$

where the first step follows from the definition of $R, L, U, \mathcal{T}_{\mathrm{encode}}(d)$, the second step follows from

$$\begin{aligned} & \log(n/\delta) \log^4(d) \log(d) \log(nd^2/\delta) \log^2(d) \log(nd) \\ = & (\log(n/\delta))(\log^7 d)(2 \log d + \log(n/\delta)) \log(nd) \\ = & O(\log^9(nd/\delta)) \end{aligned}$$

Thus, we complete the proof. $\square$

13.12 More Details of the Correctness Proofs

In this section, we give the complete proofs of the correctness of our algorithms. In Section 13.12.1 we state and prove the main result of this section, using the technical lemmas in the following subsections. In Section 13.12.2, we define the trackable layers and show the connection with important layers (Definition 13.6). In

Section 13.12.3, we analyze the sample probability and track the probability of a layer. In Section 13.12.4, we show that a good estimation of track probability implies a good approximation of the layer size, which completes the proof of correctness.

13.12.1 Correctness of layer size estimation

We first show that the estimation of layer sizes output by our data structure is good to approximate the exact values.

Lemma 13.25 (Correctness of layer size approximation)**.** *We first show that, the layer sizes $c^i_1, c^i_2, \dots, c^i_P$ our data structure return satisfy*

- *for all $k \in P$, $c^i_k \le b^i_k$;*
- *if k is a β-important layer (Definition 13.6) of $q - x_i$, then $c^i_k \ge (1 - \epsilon_1) b^i_k$,*

with probability at least $1 - \delta$, where the b^i_k is the ground truth k-th layer size of $q - x_i$.

Proof. We first define two events as

- E_1: for all important layers $k \in [P]$, q_k is well defined;
- E_2: for all $k \in [P]$, if $\widehat{\eta}_k > 0$ then $(1 - O(\epsilon))\eta'_{k,q_k} \le \widehat{\eta}_k \le \eta'_{k,q_k}$.

With Lemma 13.28 and Lemma 13.29, we have that

$$\Pr[E_1 \cap E_2] \ge 1 - \delta^{O(\log(d))}.$$

When the output of every BatchHeavyHitter is correct, if follows from Lemma 13.27, Lemma 13.28 and Lemma 13.30 the algorithm outputs an approximation to the layer vector meeting the two criteria.

The BatchHeavyHitter is used a total of LR times, each with error probability at most δ/d. By a union bound over the layers, the failure probability is at most $(poly(\log d)) \cdot \delta/d = o(\delta)$. Therefore, the total failure probability of the algorithm is at most $1 - o(\delta)$.

Thus, we complete the proof. $\square$

13.12.2 Trackability of Layers

Definition 13.10 (Trackability of layers)**.** A layer $k \in [P]$ of a vector x is β-trackable, if

$$\alpha^{2k} \geq \beta \cdot \|x_{\overline{[\beta^{-1}]}}\|_2^2$$

where $x_{\overline{[\kappa]}}$ is defined as Definition 13.4.

Lemma 13.26 (Importance and Trackability)**.** *Let α be some parameter such that $\alpha \in [0, 2]$. Suppose subvector $\widetilde{x}$ is obtained by subsampling the original vector with probability p.*

If $k \in [P]$ is a β-important layer, then for any $\lambda > P$, with probability at least $1 - P\exp(-\frac{\lambda p b_k}{P\beta})$, layer k is $\frac{\beta}{\lambda p b_k}$-trackable.

In particular, if $pb_k = O(1)$, then with probability at least $1 - P\exp(-\Omega(\frac{\lambda}{P\beta}))$, layer k is $\frac{\beta}{\lambda}$-trackable.

Proof. Let $(\zeta_0, \zeta_1, \dots)$ be the new level sizes of the subvector $\widetilde{x}$ we sampled. Thus, for $k \in [P]$, we have

$$\mathbb{E}[\zeta_k] = p \cdot b_k.$$

By the definition of important layer (Definition 13.6) we have

$$\mathbb{E}[\zeta_k] \geq \beta \cdot \mathbb{E}\Big[\sum_{j=k+1}^{P} \zeta_j\Big]$$

and

$$\mathbb{E}[\zeta_k \cdot \alpha^{2k}] \geq \beta \cdot \mathbb{E}\Big[\sum_{j\in[k]} \zeta_j \cdot \alpha^{2j}\Big].$$

To have layer k trackable, it has to be in the top $\lambda p b_k/\beta$ elements, so that we have $\sum_{j=k+1}^{P} \zeta_j \leq \lambda p b_k/\beta$. By Chernoff bound (Lemma A.2), we can know the probability that this event does not happen is

$$\Pr\Big[\sum_{j=k+1}^{P} b_j > \frac{\lambda p b_k}{\beta}\Big] \leq \exp(-\Omega(\lambda p b_k/\beta)).$$

On the other hand, for level k to be trackable $\alpha^{2k} \geq \frac{\beta}{\lambda p b_k} \sum_{j \in [k]} \zeta_j \alpha^{2j}$.

Thus, the complement occurs with probability

$$\begin{aligned}
\Pr\Big[\frac{\beta}{\lambda p b_k} \sum_{j \in [k]} \zeta_j \alpha^{2j} > \alpha^{2k}\Big] \leq & \ \Pr\Big[\exists j, \zeta_j \alpha^{2j} \geq \frac{\lambda p b_k \alpha^{2k}}{P\beta}\Big] \\
\leq & \ \sum_{j \in [P]} \Pr\Big[\zeta_j \alpha^{2j} \geq \frac{\lambda p b_k \alpha^{2k}}{P\beta}\Big].
\end{aligned}$$

By Chernoff bound (Lemma A.2) and the fact that $\mathbb{E}[\zeta_j \alpha^{2j}] \leq p b_k \alpha^{2k}$, we have

$$\begin{aligned}
\Pr\Big[\frac{\beta}{\lambda p b_k} \sum_{j \in [k]} \zeta_j \alpha^{2j} > \alpha^{2k}\Big] \leq & \ P \cdot \exp\Big(- \Omega(\frac{\lambda p b_k \alpha^{2k}}{\alpha^{2j} P \beta})\Big) \\
\leq & \ P \cdot \exp\Big(- \Omega(\frac{\lambda p b_k}{\beta})\Big).
\end{aligned}$$

Thus we complete the proof. $\square$

13.12.3 Probability analysis

We first define some parameters:

Definition 13.11. For each $k \in [p]$, $l \in \mathbb{R}^+$, we define $\eta_{k,l} := 1 - (1 - p_l)^{b_k}$ to be the probability that at least one element from B_k is sampled with the sampling probability of $p_l = 2^{-l}$.

Set $\lambda = \Theta(P \log(1/\delta))$. Let $\eta^*_{k,l}$ be the probability that an element from B_k is contained in $H_{1,l}$, such that, for $H_{r,l}$ with any other r, the probability is the same as $\eta^*_{k,l}$.

Lemma 13.27 (Sample Probability and Track Probability)**.** *For any layer $k \in [P]$ we have $\eta^*_{k,l} < \eta_{k,l}$. Let δ and ϵ denote the two parameters such that $0 < \delta < \epsilon < 1$. If layer k is a β-important layer and $p_l b_k = O(1)$, then*

$$\eta^*_{k,l} \geq (1 - \Theta(\epsilon)) \cdot \eta_{k,l}.$$

Proof. If one is in $H_{r,l}$, it has to be sampled, so we have $\eta^*_{k,l} \leq \eta_{k,l}$. On the other hand, using Lemma 13.26, with probability at least $1 - t\exp(-\Omega(\frac{\lambda p b_k}{t\beta}))$, layer k is $\frac{\beta}{\lambda p b_k}$-trackable.

We have

$$\frac{\beta}{\lambda p b_k} = \Theta(\frac{\beta}{P\log(1/\delta)})$$

where the last step follows from definition of λ.

Thus with probability at least

$$\eta_{k,l}(1 - \Theta(\delta)) \geq \eta_{k,l}(1 - O(\epsilon))$$

an element from B_k is sampled and the element is reported by BatchHeavy-Hitter.

Thus we complete the proof. $\square$

Lemma 13.28 (Probability Approximation)**.** *For $k \in [P]$, let $\widehat{\eta}_k$ be defined as in Line 29 in Algorithm 78. With probability at least $1 - \delta^{\Omega(\log d)}$, for all $k \in [P]$, if $\widehat{\eta}_k \neq 0$ then*

$$(1 - O(\epsilon_1))\eta^*_{k,q_k} \leq \widehat{\eta}_k \leq \eta^*_{k,q_k}.$$

Proof. We define

$$\gamma := \frac{R\log(1/\delta)}{100\log d}.$$

Recall the condition of Line 29, if $\widehat{\eta}_k \neq 0$, then we have

$$A_{q_k,k} \geq \gamma.$$

For a fixed $k \in [P]$, since the sampling process is independent for each $r \in [R]$, we can assume that

$$\mathbb{E}[A_{q_k,k}] = R\eta_{k,q_k} \geq \gamma.$$

Otherwise, by Chernoff bound (Lemma A.2), we get that

$$\Pr[A_{q_k,k} \geq \gamma] = o(\delta^{\Omega(\log d)}),$$

which implies that with probability at least $1 - \delta^{\Omega(\log d)}$, $\widehat{\eta}_k = 0$ for all $k \in [P]$, and we are done.

Thus, under this assumption, by Chernoff bound (Lemma A.2), we have

$$\Pr[|A_{q_k,k} - R \cdot \eta^*_{k,q_k}| \geq \epsilon R \eta_{k,q_k}] \leq \exp(-\Omega(\epsilon^2 \gamma)) = \delta^{\Omega(\log d)}.$$

Since $P = poly \log(d)$, by union bound over the B_k , the event

$$(1 - \epsilon_1)\eta_{k,q_k} \leq \frac{A_{q_k,k}}{R_k} \leq (1 + \epsilon_1)\eta_{k,q_k}$$

holds for all $k \in [P]$ with probability at least $1 - \delta^{\Omega(\log(d))}$. Since $\widehat{\eta}_k = \frac{A_{k,q_k}}{R(1+\epsilon_1)}$ by definition, we get the desired bounds.

Thus we complete the proof. $\square$

Definition 13.12. We define q_k as

$$q_k := \max_{l \in [L]} \Big\{ l | A_{l,k} \geq \frac{R \log(1/\delta)}{100 \log(d)} \Big\}.$$

We say that q_k is well-defined if the set in the RHS of the above definition is non-empty.

At Line 28 in Algorithm 78, we define the q_k^i for i-th vector as the definition above.

Lemma 13.29 (Maximizer Probability)**.** *If layer $k \in [P]$ is important (Line 28 in Algorithm 78), then with probability at least $1 - \delta^{\Omega(\log d)}$, the maximizer q_k is well defined.*

Proof. Lemma 13.27 tells us that, when $p_k b_k = O(1)$, layer k is at least $\Omega(\frac{\beta}{P\log(1/\delta)})$-trackable. On the other hand, if $P_k = 2^{-l_0} = 1/b_k$, we have $\eta_{k,l_0} = 1-(1-p_k)^{b_k} \geq 1/e$. Thus we have

$$\mathbb{E}[A_{l_0,k}] \geq R/e,$$

so by Chernoff bound (Lemma A.2),

$$\begin{aligned}\Pr[A_{l_0,k} \leq \frac{R\log(1/\delta)}{\log d}] &\leq \exp(-\Omega(\frac{R\log(1/\delta)}{\log d})) \\ &\leq \delta^{\Omega(\log d)}.\end{aligned}$$

Thus we show that, there exists one $q_k \geq l_0$ with probability at least $1 - \delta^{\Omega(\log n)}$.

Since there are at most $P = \mathit{poly}\log(d)$ important layers, with probability at least $1-\delta^{\Omega(\log n)}$, the corresponding value of q_k is well defined for all important layers.

Thus we complete the proof. □

13.12.4 From probability estimation to layer size approximation

The following lemma shows that, if we have a sharp estimate for the track probability of a layer, then we can obtain a good approximation for its size.

Lemma 13.30 (Track probability implies layer size approximation)**.** *Suppose $q_k \geq 1$, $\epsilon \in (0, 1/2)$, and d is sufficiently large. Define $\epsilon_1 := O(\epsilon^2/\log(d))$. We have for $k \in [P]$,*

- *Part 1. If $\widehat{\eta}_k \leq \eta_{k,q_k}$ then $c_k \leq b_k$.*
- *Part 2. If $\widehat{\eta}_i \geq (1-\epsilon_1)\eta_{k,q_k}$ then $c_k \geq (1-O(\epsilon_1))b_k$.*

Proof. We know that

$$b_k = \frac{\log(1-\eta_{k,q_k})}{\log(1-2^{-q_k})},$$

which is a increasing function of η_{k,q_k}.

Part 1. If $\widehat{\eta}_k \leq \eta_{k,q_k}$, then $c_k \leq b_k$.

Part 2. If $\widehat{\eta}_k \geq (1-O(\epsilon))\eta_{k,q_k}$ we have

$$c_k \geq b_k + \frac{\epsilon_1 \eta_{k,q_k}}{(1-\eta_{k,q_k})\log(1-2^{-q_k})} \geq b_k - O(\epsilon) b_k.$$

Thus we complete the proof. □

13.13 Space Complexity

In this section, we prove the space complexity of our data structure.

Lemma 13.31 (Space complexity of our data structure, formal version of Lemma 13.5)**.** *Our data structure (Algorithm 75 and 76) uses $O(\epsilon^{-9} n(d + \mathrm{mmc}(l)^2)\log^{14}(nd/\delta))$ space.*

Proof. First, we store the original data,

$$\text{space for } x = O(nd).$$

Second, we store the sub stream/sample of orignal data

$$\begin{aligned}
\text{space for } \overline{x} &= O(RLUnd) \\
&= O(\epsilon^{-4}\log(n/\delta)\log^4(d)\cdot\log(d)\cdot\log(nd^2/\delta)\cdot nd) \\
&= O(\epsilon^{-4} nd \log^6(nd/\delta)).
\end{aligned}$$

Our data structure holds a set $\{S_{r,l,u}\}_{r\in[R],l\in[L],u\in[U]}$ (Line 10). Each $S_{r,l,u}$ has a size of $\mathcal{S}_{\mathrm{space}}(\sqrt{\beta}, d) = O(n\cdot\beta^{-1}\log^2(d)\log(nd))$, which uses the space of

$$\begin{aligned}
\text{space for } S &= O(RLUn\cdot\beta^{-1}\log^2(d)\log(nd)) \\
&= O(\epsilon^{-4}\log(n/\delta)\log^4(d)\cdot\log(d)\cdot\log(nd)\cdot\log(nd^2/\delta)\cdot n\cdot(\epsilon^{-5}\cdot\mathrm{mmc}(l)^2\log^5(d))\log^2(d)) \\
&= O(\epsilon^{-9} n\cdot\mathrm{mmc}(l)^2\log^{14}(nd/\delta))
\end{aligned}$$

where the first step follows from the definition of R, L, U, and the second step follows just simplifying the last step.

We hold a bmap (Line 17 and Line 19) of size $O(RLUd)$, which uses the space of

$$\begin{aligned}\text{space for bmap} &= O(RLUd)\\ &= O(\epsilon^{-4}\log(n/\delta)\log^4(d)\cdot\log(d)\cdot\log(nd^2/\delta)\cdot d)\\ &= O(\epsilon^{-4}d\log^6(nd/\delta)).\end{aligned}$$

In Query, we generate a set of sets $\{H_{r,l,u}\}_{r\in[R],l\in[L],u\in[U]}$, each of the sets has size of $O(\beta^{-1})$, so the whole set uses space of

$$\begin{aligned}\text{space for } H &= O(RLU\cdot\beta^{-1})\\ &= O(\epsilon^{-4}\log(n/\delta)\log^4(d)\cdot\log(d)\cdot\log(nd^2/\delta)\cdot\epsilon^{-5}\cdot\mathrm{mmc}(l)^2\log^5(d))\\ &= O(\epsilon^{-9}\cdot\mathrm{mmc}(l)^2\log^{11}(nd/\delta)).\end{aligned}$$

By putting them together, we have the total space is

$$\begin{aligned}&\text{total space}\\ =&\ \text{space for } x + \text{space for } \overline{x} + \text{space for } S + \text{space for bmap} + \text{space for } H\\ =&\ O(nd) + O(\epsilon^{-4}nd\log^6(nd/\delta)) + O(\epsilon^{-9}n\cdot\mathrm{mmc}(l)^2\log^{13}(nd/\delta))\\ &+ O(\epsilon^{-4}d\log^6(nd/\delta)) + O(\epsilon^{-9}\cdot\mathrm{mmc}(l)^2\log^{11}(nd/\delta))\\ =&\ O(\epsilon^{-9}n(d+\mathrm{mmc}(l)^2)\log^{14}(nd/\delta)).\end{aligned}$$

Thus, we complete the proof.

□

13.14 Lower Bound From Previous Work

We first define turnstile streaming model (see page 2 of [LNNT16] as an example) as follows

Definition 13.13 (Turnstile streaming model)**.** We define two different turnstile streaming models here:

- Strict turnstile: Each update Δ may be an arbitrary positive or negative number, but we are promised that $x_i \geq 0$ for all $i \in [n]$ at all points in the stream.
- General turnstile: Each update Δ may be an arbitrary positive or negative number, and there is no promise that $x_i \geq 0$ always. Entries in x may be negative.

Under the turnstile model, the *norm estimation problem* has the following streaming lower bound:

Theorem 13.32 (Theorem 1.2 in [BBC+17])**.** *Let l be a symmetric norm on $\mathbb{R}^n$. Any turnstile streaming algorithm (Definition 13.13) that outputs, with probability at least 0.99, a $(1 \pm 1/6)$-approximation for $l(\cdot)$ must use $\Omega(\mathrm{mmc}(l)^2)$ bits of space in the worst case.*

We note that this problem is a special case of our symmetry norm distance oracle problem (i.e., with $n = 1$ and query vector $q = \mathbf{0}_d$ where $\mathbf{0}_d$ is a all zeros length-d vector). And in this case, the query time of our data structure becomes $\widetilde{O}(\mathrm{mmc}(l)^2)$ for a constant-approximation, matching the streaming lower bound in Theorem 13.32.

13.15 Details About Sparse Recovery Tools

In this section we give an instantiation of the sparse recovery tool we use (Definition 13.9) in Algorithms 81 - 83. Although the running times of our data structure are slightly worse (by some log factors) than the result of [KNPW11], it's enough for our symmetric norm estimation task. In terms of space requirement, the classical sparse recovery/compressed sensing only sublinear space is allowed. In our application, we're allowed to use linear space (e.g. d per point, nd in total). And more importantly, our instantiation has much simpler algorithm and analysis than the prior result.

The following lemma shows that the sparse recovery data structure satisfies our requirements in Theorem 13.20.

13.15.1 Our sparse recovery tool

We first state the correctness, the proof follows from framework of [KNPW11].

Lemma 13.33. *The function* DECODE(i, ϵ, d, δ) *(Algorithm 83) returns a set* $S \subseteq d$ *of size* $|S| = O(\epsilon^{-2})$ *containing all* ϵ*-heavy hitters of the* i*-column of the matrix under* l_2 *with probability of* $1 - \delta$*. Here we say* j *is an* ϵ*-heavy hitter under* l_2 *if* $|x_j| \geq \epsilon \cdot \|x_{\overline{[\epsilon^{-2}]}}\|_2$ *where* $x_{\overline{[k]}}$ *denotes the vector* x *with the largest* k *entries (in absolute value) set to zero. Note that the number of heavy hitters never exceeds* $2/\epsilon^2$*.*

Proof. The correctness follows from the framework of [KNPW11], and combining tail estimation (Lemma 13.36) and norm estimation.

□

We next analyze the running time of our data structure in the following lemma:

Lemma 13.34. *The time complexity of our data structure (Algorithm 81, Algorithm 82 and Algorithm 83) is as follows:*

- INIT *takes time of* $O(\epsilon^{-1}(n + d) \log^2(nd/\delta))$.
- ENCODESINGLE *takes time of* $O(\log^2(nd/\delta))$.

- ENCODE *takes time of* $O(d\log^2(nd/\delta))$.
- SUBTRACT *takes time of* $O(\epsilon^{-1}\log^2(nd/\delta))$.
- DECODE *takes time of* $O(\epsilon^{-2}\log^2(nd/\delta))$.

Proof. We first notice that $L = \log_2 d$ and $\delta' = \epsilon\delta/(12(\log(d)+1))$.

For the procedure INIT (Algorithm 81), Line 22 takes time

$$O(L\epsilon^{-1}n\log(n/\delta')) = O(\epsilon^{-1}n\log^2(\delta^{-1}\epsilon^{-1}nd\log(d)).$$

Line 25 takes time $O((n+d)\log(n/\delta))$. Taking together, we have the total running time of INIT is $O(\epsilon^{-1}(n+d)\log^2(\delta^{-1}\epsilon^{-1}nd\log(d))$

For the procedure ENCODESINGLE (Algorithm 82), Line 4 takes time of

$$O(L\log(n/\delta')) = O(\log^2(\epsilon^{-1}\delta^{-1}nd\log(d))).$$

Line 6 takes time of $O(\log(n/\delta))$. Taking together we have the total running time of ENCODESINGLE to be $O(\log^2(\epsilon^{-1}\delta^{-1}nd\log(d)))$.

For the procedure ENCODE (Algorithm 82), it runs ENCODESINGLE for d times, so its running time is $O(d\log^2(\epsilon^{-1}\delta^{-1}nd\log(d)))$.

For the procedure SUBTRACT (Algorithm 83), Line 4 runs in time

$$O(L\epsilon^{-1}\log(n/\delta')) = O(\epsilon^{-1}\log^2(\epsilon^{-1}\delta^{-1}nd\log(d))).$$

Line 6 runs in time $O(\log(n/\delta))$. So the total running time is $O(\epsilon^{-1}\log^2(\epsilon^{-1}\delta^{-1}nd\log(d)))$.

For the procedure DECODE (Algorithm 83), Line 10 runs in time $O(\log(n/\delta))$. Line 14 runs in time

$$O(L\epsilon^{-2}\log(n/\delta')) = O(\epsilon^{-2}\log^2(\epsilon^{-1}\delta^{-1}nd\log(d))).$$

So the total running time is $O(\epsilon^{-2}\log^2(\epsilon^{-1}\delta^{-1}nd\log(d)))$

Thus we complete the proof. □

The space complexity of our data structure is stated in below.

Lemma 13.35. *Our batch heavy hitter data structure (Algorithm 81, Algorithm 82 and Algorithm 83) takes the space of*

$$O(\epsilon^{-1}(n+d)\log^2(\epsilon^{-1}\delta^{-1}nd\log(d))).$$

Proof. Our data structure has these two parts to be considered:

- The instantiations of FpEst we maintain.
- The instantiation of LpLpTailEstimation we maintain .

The first part takes the space of

$$O(L\epsilon^{-1}n\log(n/\delta')) = O(\epsilon^{-1}n\log^2(\epsilon^{-1}\delta^{-1}nd\log(d))).$$

And the second part takes the space of $O(d\log(n/\delta))$. Adding them together we complete the proof. $\square$

13.15.2 Lp tail estimation

[NS19] provide a linear data structure LpLpTailEstimation(x,k,p,C_0,δ) that can output the estimation of the contribution of non-heavy-hitter entries. We restate their Lemma as followed.

Lemma 13.36 (Lemma C.4 of [NS19])**.** *Let $C_0 \geq 1000$ denote some fixed constant. There is an oblivious construction of matrix $A \in \mathbb{R}^{m\times n}$ with $m = O(\log(1/\delta))$ along with a decoding procedure LpLpTailEstimation(x,k,p,C_0,δ) such that, given Ax, it is possible to output a value V in time $O(m)$ such that*

$$\frac{1}{10k}\|x_{\overline{C_0k}}\|_p^p \leq V \leq \frac{1}{k}\|x_{\overline{k}}\|_p^p,$$

holds with probability $1-\delta$.

Lemma 13.37. *The running time of the above data structure is*

- INIT *runs in time of* $O((n+d)\log(n/\delta))$
- UPDATE *runs in time of* $O(\log(n/\delta))$
- SUBTRACT *runs in time of* $O(\log(n/\delta))$
- QUERY *runs in time of* $O(\log(n/\delta))$

And its space is $O(d\log(n/\delta))$.

13.15.3 Lp norm estimation

Following the work of [KNW10], we provide a linear sketch satisfying the following requirements.

Lemma 13.38 ([KNW10])**.** *There is a linear sketch data structure* FPEST *using space of*

$$O(\epsilon^{-1}\phi^{-2}n\log(n/\delta))$$

and it provide these functions:

- INIT$(n \in \mathbb{Z}^+, d \in \mathbb{Z}^+, l \in \mathbb{Z}^+, \phi, \epsilon, \delta)$*: Initialize the sketches, running in time* $O(\epsilon^{-1}n\log(n/\delta))$
- UPDATE$(i \in [n], j \in [d], z \in \mathbb{R})$*: Update the sketches, running in time* $O(\log(n/\delta))$
- SUBTRACT$(i, j, k \in [n])$*: Subtract the sketches, running in time* $O(\epsilon^{-1}\phi^{-2}\log(n/\delta))$
- QUERY$(i \in [n], \xi \in [2^l])$*: This function will output a* V *satisfying*

$$(1-\phi)\cdot F_i(l,\xi) \le V \le (1+\phi)\cdot(F_i(l,\xi)+5\epsilon\|x_i\|_2^2),$$

where $F(l,\xi)$ is defined as

$$F_i(l,\xi) := \sum_{j=\frac{\xi}{2^l}d}^{\frac{\xi+1}{2^l}d-1} |x_{i,j}|^2,$$

running in time $O(\phi^{-2}\log(n/\delta))$

We choose ϕ to be constant when we use the above Lemma.

Algorithm 79 Data structure for symmetric norm estimation: query set

```
1: data structure DistanceOnSymmetricNorm    ▷ Theorem 13.19
2: procedure Query($q \in \mathbb{R}^d$, $\mathsf{S} \subseteq [n]$)    ▷ Lemma 13.4, 13.6
3:     $P \leftarrow O(\log_\alpha(d)) = O(\log(d)/\epsilon)$    ▷ $P$ denotes the number of non-empty layer sets
4:     for $r \in [R], l \in [L], u \in [U]$ do
5:         $S_{r,l,u}$.Encode($n+1, q, d$)    ▷ Generate Sketch for $q$
6:     end for
7:     $\xi \leftarrow$ chosen uniformly at random from $[1/2, 1]$
8:     $\gamma \leftarrow \Theta(\epsilon)$, $\alpha \leftarrow 1 + \gamma \cdot \xi$
9:     for $i \in \mathsf{S}$ do
10:        for $r \in [R], l \in [L], u \in [U]$ do
11:            $S_{r,l,u}$.Subtract($n+2, n+1, i$)
12:            $H_{r,l,u} \leftarrow S_{r,l,u}$.Decode($n+2, \sqrt{\beta}, d$)    ▷ This can be done in $\frac{2}{\beta}\mathrm{poly}(\log d)$
13:                ▷ At this point $H_{r,l,u}$ is a list of index, reset to 0
14:            for $k \in H_{r,l,u}$ do
15:                value $\leftarrow [\overline{x}_{r,l,u}]_{i,k}$
16:                $H_{r,l,u}[k] \leftarrow$ value, $w \leftarrow \lceil \log(\text{value})/\log(\alpha) \rceil$
17:                if $\alpha^{w-1} \geq \text{value}/(1+\epsilon)$ then
18:                    $H_{r,l,u} \leftarrow \mathsf{null}$, break
19:                end if
20:            end for
21:            if $H_{r,l,u} \neq \mathsf{null}$ then
22:                $H_{r,l} \leftarrow H_{r,l,u}$
23:            end if
24:        end for
25:        for $l \in [L], k \in [P]$ do
26:            $A^i_{l,k} \leftarrow |\{k \mid \exists k \in H_{r,l}, \alpha^{k-1} < |H_{r,l}[k]| \leq \alpha^k\}|$
27:        end for
28:        for $k \in [P]$ do
29:            $q^i_k \leftarrow \max_{l \in [L]} \{l \mid A^i_{l,k} \geq \frac{R \log(1/\delta)}{100 \log(d)}\}$
30:            If $q^i_k$ does not exist, then $\widehat{\eta}^i_k \leftarrow 0$; Else $\widehat{\eta}^i_k \leftarrow \frac{A_{q^i_k,k}}{R(1+\epsilon_2)}$
31:            If $\widehat{\eta}^i_k = 0$ then $c^i_k \leftarrow 0$; Else $c^i_k \leftarrow \frac{\log(1-\widehat{\eta}^i_k)}{1-w^{-q_k}}$
32:        end for
33:        $\mathrm{dst}_i \leftarrow$ LayerVetcorApprox($\alpha, c^i_1, c^i_2, \ldots, c^i_P, d$)
34:        for $r \in [R], l \in [L], u \in [U]$ do
35:            $\{H_{r,l,u}\} \leftarrow \{0\}$    ▷ Reset the sets to use for next point
36:        end for
37:    end for
38:    return $\{\mathrm{dst}_i\}_{i \in \mathsf{S}}$
39: end procedure
40: end data structure
```

Algorithm 80 Data structure for symmetric norm estimation: query pair

```
1: data structure DistanceOnSymmetricNorm                              ▷ Theorem 13.19
2: procedure EstPair($i \in [n]$, $j \in [n]$)                         ▷ Lemma 13.4, 13.6
3:     $P \leftarrow O(\log_\alpha(d))$            ▷ $P$ denotes the number of non-empty layer sets
4:     $\xi \leftarrow$ chosen uniformly at random from $[1/2, 1]$
5:     $\gamma \leftarrow \Theta(\epsilon)$
6:     $\alpha \leftarrow 1 + \gamma \cdot \xi$
7:     $P \leftarrow O(\log_\alpha(d)) = O(\log(d)/\epsilon)$    ▷ $P$ denotes the number of non-empty layer sets
8:     for $r \in [R], l \in [L], u \in [U]$ do
9:         $S_{r,l,u}$.Subtract($n+2, j, i$)
10:        $H_{r,l,u} \leftarrow S_{r,l,u}$.Decode($n+2, \sqrt{\beta}, d$)   ▷ This can be done in $\frac{2}{\beta} poly(\log d)$
11:        ▷ At this point $H_{r,l,u}$ is a list of index, the value of each index is reset to 0
12:        for $k \in H_{r,l,u}$ do
13:            value $\leftarrow [\overline{x}_{r,l,u}]_{i,k}$
14:            $H_{r,l,u}[k] \leftarrow$ value
15:            $w \leftarrow \lceil \log(\text{value}) / \log(\alpha) \rceil$
16:            if $\alpha^{w-1} \geq \text{value}/(1+\epsilon)$ then
17:                $H_{r,l,u} \leftarrow$ null
18:                break
19:            end if
20:        end for
21:        if $H_{r,l,u} \neq$ null then
22:            $H_{r,l} \leftarrow H_{r,l,u}$
23:        end if
24:    end for
25:    for $l \in [L], k \in [P]$ do
26:        $A_{l,k} \leftarrow |\{k \mid \exists k \in H_{r,l}, \alpha^{k-1} < |H_{r,l}[k]| \leq \alpha^k\}|$
27:    end for
28:    for $k \in [P]$ do
29:        $q_k \leftarrow \max_{l \in [L]} \{l \mid A_{l,k} \geq \frac{R \log(1/\delta)}{100 \log(d)}\}$
30:        If $q_k$ does not exist, then $\widehat{\eta}_k \leftarrow 0$; Else $\widehat{\eta}_k \leftarrow \frac{A_{q_k,k}}{R(1+\epsilon_2)}$
31:        If $\widehat{\eta}_k = 0$ then $c_k \leftarrow 0$; Else $c_k \leftarrow \frac{\log(1-\widehat{\eta}_k)}{1-w^{-q_k}}$
32:    end for
33:    dst $\leftarrow$ LayerVetcorApprox($\alpha, c_1, c_2, \dots, c_P, d$)
34:    return dst
35: end procedure
36: end data structure
```

Algorithm 81 Our CountSketch for Batch heavy hitter

```
data structure BasicBatchHeavyHitter    ▷ Definition 13.9
members
    $d, n \in \mathbb{N}^+$    ▷ $n$ is the number of vectors, $d$ is the dimension.
    $\epsilon$    ▷ We are asking for $\epsilon$-heavy hitters
    $\delta$    ▷ $\delta$ is the failure probability
    $B$    ▷ $B$ is the multiple number of each counter to take mean
    $L$    ▷ $L$ is the number of number of the level of the binary tree.
    $\eta$    ▷ $\eta$ is the precision for $l_2$ norm estimation.
    $K$    ▷ $K$ is the number of hash functions.
    $\{h_{l,k}\}_{l \in \{0,\dots,L\}, k \in [K]} \subseteq [2^l] \times [B]$    ▷ hash functions.
    $\{C^i_{l,b}\}_{i \in [n], l \in \{0,\dots,L\}, b \in [B]}$    ▷ The counters we maintain in CountSketch.
    $\{\sigma_{l,k}\}_{l \in \{0,\dots,L\}, k \in [K]} \subseteq [d] \times \{+1,-1\}$    ▷ The hash function we use for norm estimation
    $\mathcal{Q}$    ▷ A instantiation of LpLpTailEstimation (Algorithm 84)
    $\{\mathcal{D}_l\}_{l \in [L]}$    ▷ Instantiations of FpEst (Algorithm 85)
end members

public:
procedure Init($\epsilon, n \in \mathbb{N}^+, d \in \mathbb{N}^+, \delta$)
    $L \leftarrow \log_2 d$
    $\delta' \leftarrow \epsilon\delta/(12(\log d) + 1)$
    for $l \in \{0, \dots, L\}$ do
        $\mathcal{D}_l$.Init($n, d, l, 1/7, \epsilon, \delta'$)
    end for
    $C_0 \leftarrow$ greater than 1000
    $\mathcal{Q}$.Init($n, \epsilon^{-2}, 2, C_0, \delta$)
end procedure
```

Algorithm 82 Our CountSketch for Batch heavy hitter

```
data structure BasicBatchHeavyHitter                         ▷ Definition 13.9
procedure EncodeSingle($i \in [n], j \in [d], z \in \mathbb{R}, d$)
    for $l \in \{0, \ldots, L\}$ do
        $\mathcal{D}_l$.Update($i, j, z$)
    end for
    $\mathcal{Q}$.Update($i, j, z$)
end procedure

procedure Encode($i \in [n], z \in \mathbb{R}^d, d$)
    for $j \in [d]$ do
        EncodeSingle($i, j, z_j, d$)
    end for
end procedure

end data structure
```

Algorithm 83 Our CountSketch for Batch heavy hitter

```
1: data structure BasicBatchHeavyHitter                    ▷ Definition 13.9
2: procedure Subtract($i, j, k \in [n]$)
3:     for $l \in \{0, \dots, L\}$ do
4:         $\mathcal{D}_l$.Subtract($i, j, k$)
5:     end for
6:     $\mathcal{Q}$.Subtract($i, j, k$)
7: end procedure
8:
9: procedure Decode($i \in [n], \epsilon, d$)
10:     EstNorm $\leftarrow \mathcal{Q}$.Query($i$) ▷ Here the EstNorm is the estimated tail $l_2$-norm of
    $i$-vector.
11:     $S \leftarrow \{0\}, S' \leftarrow \emptyset$
12:     for $l \in \{0, \dots, L\}$ do                         ▷ The dyadic trick.
13:         for $\xi \in S$ do
14:             Est $\leftarrow \mathcal{D}$.Query($i, \xi$)
15:             if Est $\geq (3/4)\epsilon^2 \cdot$ EstNorm then
16:                 $S' \leftarrow S' \cup \{2\xi, 2\xi + 1\}$
17:             end if
18:         end for
19:         $S \leftarrow S'$, $S' \leftarrow \emptyset$
20:     end for
21:     return $S$
22: end procedure
23: end data structure
```

Algorithm 84 Batched ℓ_p tail estimation algorithm, based on [NS19]

```
1: data structure LpLpTailEstimation
2: members
3:     $m$                                                    ▷ $m$ is the sketch size
4:     $\{g_{j,t}\}_{j\in[d],t\in[m]}$    ▷ random variable that sampled i.i.d. from distribution $\mathcal{D}_p$
5:     $\{\delta_{j,t}\}_{j\in[d],t\in[m]}$          ▷ Bernoulli random variable with $\mathbb{E}[\delta_{j,t}] = 1/(100k)$
6:     $\{y_{i,t}\}_{i\in[n],t\in[m]}$                                          ▷ Counters
7: end members
8: public:
9: procedure Init($n, k, p, C_0, \delta$)
10:     $m \leftarrow O(\log(n/\delta))$
11:     Choose $\{g_{j,t}\}_{j\in[d],t\in[m]}$ to be random variable that sampled i.i.d. from distribution $\mathcal{D}_p$
12:     Choose $\{\delta_{j,t}\}_{j\in[d],t\in[m]}$ to be Bernoulli random variable with $\mathbb{E}[\delta_{j,t}] = 1/(100k)$ ▷ Matrix $A$ in Lemma 13.36 is implicitly constructed based on $g_{j,t}$ and $\delta_{j,t}$
13:     initialize $\{y_{i,t}\}_{i\in[n],t\in[m]} = \{0\}$
14: end procedure
15:
16: procedure Update($i \in [n], j \in [d], z \in \mathbb{R}$)
17:     for $t \in [m]$ do
18:         $y_{i,t} \leftarrow y_{i,t} + \delta_{j,t} \cdot g_{jt} \cdot z$
19:     end for
20: end procedure
21:
22: procedure Subtract($i, j, k \in [n]$)
23:     for $t \in [m]$ do
24:         $y_{i,t} \leftarrow y_{j,t} - y_{k,t}$
25:     end for
26: end procedure
27:
28: procedure Query($i \in [n]$)
29:     $V \leftarrow median_{t\in[m]} |y_{i,t}|^2$
30:     return $V$
31: end procedure
```

Algorithm 85 Batched ℓ_p norm estimation algorithm, based on [KNW10]

```
1: data structure LpNormEst
2: members
3:     $R, T$                                                    ▷ parallel parameters
4:     $m$                                                       ▷ Sketch size
5:     $\{y^i_{r,t}\}_{i\in[n],r\in[R],t\in[T]} \subset \mathbb{R}^m$   ▷ Sketch vectors
6:     $\{A\}_{r,t} \subset \mathbb{R}^{m\times 2^l}$                  ▷ Sketch matrices
7:     $\{h_t\}_{t\in[T]} : \{0,\dots,2^l\} \to [R]$                 ▷ hash functions
8: end members
9:
10: public:
11: procedure Init($n, d, l, \phi, \epsilon$)
12:     $R \leftarrow \lceil 1/\epsilon \rceil$
13:     $T \leftarrow \Theta(\log(n/\delta))$
14:     $m \leftarrow O(1/\phi^2)$
15:     for $i \in [n], r \in [R], t \in [T]$ do
16:         $y^i_{r,t} \leftarrow \mathbf{0}$                              ▷ Sketch vectors
17:     end for
18:     for $i \in [n], r \in [R], t \in [T]$ do
19:         generate $A_{r,t} \in \mathbb{R}^{m\times 2^l}$             ▷ See [KNW10] for details
20:     end for
21:     initialize $h$
22: end procedure
23:
24: procedure Update($i \in [n], j \in [d], z \in \mathbb{R}$)
25:     $\xi \leftarrow j \cdot 2^l/d$
26:     for $t \in [T]$ do
27:         $y^i_{h_t(\xi),t} \leftarrow y^i_{h_t(\xi),t} + A_{h_t(\xi),t}\mathbf{i}^l_{j,z}$ ▷ $\mathbf{i}^l_{j,z}$ is define to be the $2^l$-dimensional vector
    with $j$-th entry of $z$ and others to be 0
28:     end for
29: end procedure
30:
```

Algorithm 86 Batched ℓ_p norm estimation algorithm, based on [KNW10], continued

```
procedure SUBTRACT($i, j, k \in [n]$)
    for $r \in [R], t \in [T]$ do
        $y^i_{r,t} \leftarrow y^j_{r,t} - y^k_{r,t}$
    end for
end procedure

procedure QUERY($i \in [n], \xi \in [2^l]$)
    for $t \in [T]$ do
        $V_{h_t(\xi),t} \leftarrow median_{\zeta\in[m]} y^i_{r,t,\zeta} / median(|\mathcal{D}_p|)$        ▷ Definition 13.5
    end for
    $V \leftarrow median_{t\in[T]} V_{h_t(\xi),t}$
    return $V$
end procedure
```

Part III

Machine Learning

Chapter 14: Training Two-Layer Over-Parameterized Neural Networks

14.1 Introduction

Over the last decade, deep learning has achieved dominating performance over many areas, e.g., computer vision [LBBH98, KSH12, SLJ$^+$15, HZRS16], natural language processing [CWB$^+$11, DCLT18], game playing [SHM$^+$16, SSS$^+$17] and beyond. The computational resource requirement for deep neural network training grows very quickly. Designing a fast and provable training method for neural networks is, therefore, a fundamental and demanding challenge.

Almost all deep learning models are optimized by gradient descent (or its variants). The total training time can be split into two components, the first one is the number of iterations and the second one is the cost per spent per iteration. Nearly all the iterative algorithms for acceleration can be viewed as two separate lines of research correspondingly, the first line is aiming for an algorithm that has as small as possible number of iterations, the second line is focusing on designing as efficient as possible data structures to improve the cost spent per iteration of the algorithm [Vai89b, CLS19, LSZ19, JLSW20, JKL$^+$20, JSWZ21]. In this chapter, our major focus is on the second line.

There are a number of practical works trying to use a nearest neighbor search data structure to speed up the per-step computation of the deep neural network training [CMJF$^+$20, LXJ$^+$20, CLP$^+$21, DMZS21]. However, none of the previous work is able to give a provable guarantee. In this chapter, our goal is to develop training algorithms that provably reduce per step time complexity. Let us consider the ReLU activation neural network and two-layer neural network[1]. Let n denote the number of training data points. Let d denote the dimension of each data point.

[1]An alternative name of the two-layer neural network is "one-hidden layer neural network".

Let m denote the number of neurons. In each iteration of gradient descent (GD), we need to compute prediction for each point in the neural network. Each point $x_i \in \mathbb{R}^d$, requires to compute m inner product in d dimension. Thus, $\Omega(mnd)$ is a natural barrier for cost per iteration in training neural networks (in both forward computation and backward computation).

A natural question to ask is

Is it possible to improve the cost per iteration of training neural network algorithm? E.g., is $o(mnd)$ possible?

We list our contributions as follows:

- We provide a new theoretical framework for speeding up neural network training by: 1) adopting the shifted neural tangent kernel; 2) showing that only a small fraction ($o(m)$) of neurons are activated for each input data in each training iteration; 3) identifying the sparsely activated neurons via geometric search; 4) proving that the algorithm can minimize the training loss to zero in a linear convergence rate.
- We provide two theoretical results 1) our first result (Theorem 14.8) builds a dynamic half-space report data structure for the weights of a neural network, to train neural networks in sublinear cost per iteration; 2) our second result (Theorem 14.9) builds a static half-space report data-structure for the input data points of the training data set for training a neural network in sublinear time.

Acceleration via high-dimensional search data-structure. High-dimensional search data structures support efficiently finding points in some geometric query regions (e.g., half-spaces, simplices, etc). Currently, there are two main approaches: one is based on Locality Sensitive Hashing (LSH) [IM98], which aims to find the close-by points (i.e., small ℓ_2 distance [DIIM04, AR15, AIL$^+$15, ARN17, Raz17, AIR18, BIW19, DIRW20] or large inner product [SL14, SL15b, SL15a]) of a query $q \in \mathbb{R}^d$

in a given set of points $S \subset \mathbb{R}^d$. This kind of algorithms runs very fast in practice, but most of them only support approximate queries. Another approach is based on space partitioning data structures, for example, partition trees [Mat92a, Mat92b, AEM92, AC09, Cha12], k-d trees / range trees [CT17, TOG17, Cha19], Voronoi diagrams [ADBMS98, Cha00a], which can exactly search the query regions. Recent works have successfully applied high-dimensional geometric data structure to reduce the complexity of training deep learning models. SLIDE [CMJF+20] accelerates the forward pass by retrieving neurons with maximum inner product via an LSH-based data structure; Reforemer [KKL20] similarly adopts LSH to reduce the memory usage for processing long sequence; MONGOOSE [CLP+21] accelerates the forward pass by retrieving neurons with maximum inner products via a learnable LSH-based data structure [Cha02] and lazy update framework [CLS19]. Despite the great empirical success, there is no theoretical understanding of such acceleration.

The goal of our paper is to theoretically characterize the acceleration brought by the high-dimensional geometric data structure. Specifically, our algorithm and analysis are built upon the HSR data structures [AEM92] which can find all the points that have large inner products and support efficient data update. Note that HSR comes with a stronger recovery guarantee than LSH, in the sense that HSR, whereas LSH is guaranteed to find some of those points.

Convergence via over-parameterization. Over the last few years, there has been a tremendous work studying the convergence result of deep neural network explicilty or implicitly based on neural tangent kernel (NTK) [JGH18], e.g. [LL18, DZPS19, AZLS19a, AZLS19b, DLL+19, ADH+19a, ADH+19b, SY19, CGH+19, ZMG19, CG19, ZG19, OS20, LSS+20, JT20, ZPD+20, HLSY21, BPSW21]. It has been shown that (S)GD can train a sufficiently wide NN with random initialization will converge to a small training error in polynomial steps.

14.2 Challenges and Techniques

- Empirical works combine high-dimensional search data structures (e.g., LSH) with neural network training, however, they do not work theoretically due to the following reasons:
  - Without shifting, the number of activated (and therefore updated) neurons is $\Theta(m)$. There is no hope to theoretically prove $o(m)$ complexity (See **Challenge 1**).
  - Approximate high-dimensional search data structures might miss some important neurons, which can potentially prevent the training from converging (see **Challenge 2**).
- Our solutions are:
  - We propose a shifted ReLU activation that is guaranteed to have $o(m)$ number of activated neurons. Along with the shifted ReLU, we also propose a shifted NTK to rigorously provide a convergence guarantee (see **Solution 1**).
  - We adopt an exact high-dimensional search data structure that better couples with the shifted NTK. It takes $o(m)$ time to identify the activated neurons and fits well with the convergence analysis as it avoids missing important neurons (see **Solution 2**).

Challenge 1: How to sparsify an over-parameterized neural network? To speed up the training process, we need the neural network to be "sparse", that is, for each training data $x \in \mathbb{R}^d$, the number of activated neurons is small. Then, in the forward computation, we can just evaluate a small subset of neurons. However, in the previous NTK analysis (e.g., [DZPS19]), the activation function is $\sigma(x) = \max\{\langle w_r, x\rangle, 0\}$, and the weights vectors w_r are initially sampled from a standard d-dimensional Gaussian distribution. Then, by the symmetry of Gaussian distribution, we know that for every input data x, there will be about half of the neurons being activated, which means that we can only obtain a constant-factor speedup.

Solution 1 The problem actually comes from the activation function. In practice, people use a shifted ReLU function $\sigma_b(x) = \max\{\langle w_r, x\rangle, b_r\}$ to train neural networks. The main observation of our work is that *threshold implies sparsity*. We consider the setting where all neurons have a unified threshold parameter b. Then, by the concentration of Gaussian distribution, there will be $O(\exp(-b^2) \cdot m)$ activated neurons after the initialization.

The next step is to show that the number of activated neurons will not blow up too much in the following training iterations. In [DZPS19, SY19], they showed that the weights vectors are changing slowly during the training process. In our work, we open the black box of their proof and show a similar phenomenon for the shifted ReLU function. More specifically, a key component is to prove that for each training data, a large fraction of neurons will not change their status (from non-activated to activated and vice versa) in the next iteration with high probability. To achieve this, they showed that this is equivalent to the event that a standard Gaussian random variable in a small centered interval $[-R, R]$, and applied the anti-concentration inequality to upper-bound the probability. In our setting, we need to upper-bound the probability of $z \sim \mathcal{N}(0, 1)$ in a shifted interval $[b - R, b + R]$. On the one hand, we can still apply the anti-concentration inequality by showing that the probability is at most $\Pr[z \in [-R, R]]$. On the other hand, this probability is also upper-bounded by $\Pr[z > b - R]$, and for small R, we can apply the concentration inequality for a more accurate estimation. In the end, by some finer analysis of the probability, we can show that with high probability, the number of activated neurons in each iteration is also $O(\exp(-b^2) \cdot m)$ for each training data. If we take $b = \Theta(\sqrt{\log m})$, we only need to deal with truly sublinear in m of activated neurons in the forward evaluation.

Challenge 2: How to find the small subset of activated neurons? A linear scan of the neurons will lead to a time complexity linear in m, which we hope to avoid. Randomly sampling or using LSH for searching can potentially miss important neurons which are important for a rigorous convergence analysis.

Solution 2 Given the shifted ReLU function $\sigma_b(\langle w_r, x\rangle) = \max\{\langle w_r, x\rangle - b, 0\}$, the active neurons are those with weights w_r lying in the half space of $\langle w_r, x\rangle - b > 0$. Finding such neurons is equivalent to a computational geometry problem: given m points in $\mathbb{R}^d$, in each query and a half space $\mathcal{H}$, the goal is to output the points contained in $\mathcal{H}$. Here we use the Half-Space Reporting (HSR) data structure proposed by [AEM92]: after proper initialization, the HSR data structure can return all points lying in the queried half space with complexity as low as $O(\log(n) + k)$, where k is the number of such points. Note that the HSR data structure well couples with the shifted ReLU, as the number of activated neurons k is truly sublinear in m as per the setting of $b = \Theta(\sqrt{\log m})$.

14.3 Preliminaries

Notations For an integer n, we use $[n]$ to denote the set $\{1, 2, \cdots, n\}$. For a vector x and $p \in \{0, 1, 2, \infty\}$, we use $\|x\|_p$ to denote the entry-wise ℓ_p norm of a vector. We use I_d to denote d-dimensional identity matrix. We use $\mathcal{N}(\mu, \sigma^2)$ to denote Gaussian distribution with mean mu and variance σ^2. We use $\widetilde{O}$ to hide the polylog factors.

This section is organized as follows. Section 14.3.1 introduces the neural network and present problem formulation. Section 14.3.2 presents the half-space report data-structure, Section 14.3.3 proposes our new sparsity-based Characterizations.

14.3.1 Problem formulation

In this section, we introduce the neural network model we study in this chapter. Let us consider a two-layer ReLU activated neural network f that has width m and

ℓ_2 loss function. [2]

Definition 14.1 (Prediction function and loss function)**.** Given $b \in \mathbb{R}$, $x \in \mathbb{R}^d$, $W \in \mathbb{R}^{d\times m}$ and $a \in \mathbb{R}^m$,

$$f(W,x,a) := \frac{1}{\sqrt{m}} \sum_{r=1}^{m} a_r \sigma_b(\langle w_r, x\rangle),$$
$$L(W) := \frac{1}{2} \sum_{i=1}^{n} (f(W,x_i,a) - y_i)^2.$$

We say function f is $\mathsf{2NN}(m,b)$ for simplicity.

Here W are weights that connect input nodes with hidden nodes, $a_1, \cdots, a_m \in \mathbb{R}$ are the weights that connect hidden nodes with output node. The ReLU function $\sigma_b(x) := \max\{x-b, 0\}$, where b is the threshold parameter. Following the literature, we mainly focus on optimizing $W \in \mathbb{R}^{d\times m}$. For weights $a \in \mathbb{R}^m$, we will never change a during the training after we randomly choose them at the initialization.[3]

Definition 14.2 (Weights at initialization)**.** We use the following initialization,

- For each r, we sample $w_r(0) \sim \mathcal{N}(0, I_d)$
- For each r, we sample a_r from $\{-1,+1\}$ uniformly at random

Next, we can calculate the gradient

Fact 14.1 (Gradient of the prediction function and loss function)**.** *For each $r \in [m]$,*

$$\frac{\partial f(W,x,a)}{\partial w_r} = \frac{1}{\sqrt{m}} a_r x \mathbf{1}_{w_r^\top x \geq b}. \tag{14.1}$$

and

$$\frac{\partial L(W)}{\partial w_r} = \frac{1}{\sqrt{m}} \sum_{i=1}^{n} (f(W,x_i,a) - y_i) a_r x_i \mathbf{1}_{\langle w_r, x_i\rangle \geq b}. \tag{14.2}$$

[2]This is a very standard formulation in the literature, e.g., see [DZPS19, SY19, BPSW21]

[3]We remark, in some previous work, they do choose shift, but their shift is a random shift. In our application, it is important that the same b is fixed for all neurons and never trained.

To update the weights from iteration k to iteration $k+1$, we follow the standard update rule of the GD algorithm,

$$\text{GD:} \quad W(k+1) = W(k) - \eta \cdot \Delta W(k), \quad \text{where} \quad \Delta W(k) = \frac{\partial L(W(k))}{\partial W(k)}. \tag{14.3}$$

The ODE of the gradient flow is defined as

$$\frac{\mathrm{d}w_r(t)}{\mathrm{d}t} = -\frac{\partial L(W)}{\partial w_r}. \tag{14.4}$$

Definition 14.3 (Error of prediction)**.** For each $t \in \{0, 1, \cdots, T\}$, we define $\mathrm{err}(t) \in \mathbb{R}^n$ to be the error of prediction $\mathrm{err}(t) = y - u(t)$, where $u(t) := f(W(t), a, X) \in \mathbb{R}^n$

14.3.2 Data structure for Half-Space Reporting

The half-space range reporting problem is an important problem in computational geometry, which is formally defined as following:

Definition 14.4 (Half-space range reporting)**.** Given a set S of n points in $\mathbb{R}^d$. There are two operations:

- Query(H): given a half-space $H \subset \mathbb{R}^d$, output all of the points in S that contain in H, i.e., $S \cap H$.
- Update: add or delete a point in S.
  - Insert(q): insert q into S
  - Delete(q): delete q from S

Let $\mathcal{T}_{\mathsf{init}}$ denote the pre-processing time to build the data structure, $\mathcal{T}_{\mathsf{query}}$ denote the time per query and $\mathcal{T}_{\mathsf{update}}$ time per update.

We use the data-structure proposed in [AEM92] to solve the half-space range reporting problem, which admits the interface summarized in Algorithm 87. Intuitively, the data-structure recursively partitions the set S and organizes the points in a tree data-structure. Then for a given query (a, b), all k points of S with $sgn(\langle a, x\rangle - b) \geq 0$ are reported quickly. Note that the query (a, b) here defines the half-space H in Definition 14.4.

Algorithm 87 Half Space Report Data Structure

```
1: data structure HalfSpaceReport
2:     procedures:
3:         Init($S, n, d$)  ▷ Initialize the data structure with a set $S$ of $n$ points in $\mathbb{R}^d$
4:         Query($a, b$)    ▷ $a, b \in \mathbb{R}^d$. Output the set $\{x \in S : sgn(\langle a, x\rangle - b) \geq 0\}$
5:         Add($x$)         ▷ Add point $x \in \mathbb{R}^d$ to $S$
6:         Delete($x$)      ▷ Delete point $x \in \mathbb{R}^d$ from $S$
7: end data structure
```

Adapted from [AEM92], the algorithm comes with the following complexity:

Corollary 14.2 ([AEM92])**.** *Given a set of n points in $\mathbb{R}^d$, the half-space reporting problem can be solved with the following performances:*

- *Part 1.* $\mathcal{T}_{\mathsf{query}}(n, d, k) = O_d(n^{1-1/\lfloor d/2\rfloor} + k)$, *amortized* $\mathcal{T}_{\mathsf{update}} = O_d(\log^2(n))$.
- *Part 2.* $\mathcal{T}_{\mathsf{query}}(n, d, k) = O_d(\log(n) + k)$, *amortized* $\mathcal{T}_{\mathsf{update}} = O_d(n^{\lfloor d/2\rfloor - 1})$.

We remark that Part 1 will be used in Theorem 14.8 and Part 2 will be used in Theorem 14.9.

14.3.3 Sparsity-based characterizations

In this section, we consider the ReLU function with a nonzero threshold: $\sigma_b(x) = \max\{0, x - b\}$, which is commonly seen in practise, and also has been considered in theoretical work [ZPD+20].

We first define the set of neurons that are firing at time t.

Definition 14.5 (fire set)**.** For each $i \in [n]$, for each $t \in \{0, 1, \cdots, T\}$, let $\mathcal{S}_{i,\text{fire}}(t) \subset [m]$ denote the set of neurons that are "fire" at time t, i.e.,

$$\mathcal{S}_{i,\text{fire}}(t) := \{r \in [m] : \langle w_r(t), x_i\rangle > b\}.$$

We define $k_{i,t} := |\mathcal{S}_{i,\mathrm{fire}}(t)|$, for all t in $\{0, 1, \cdots, T\}$.

We propose a new "sparsity" lemma in this chapter. It shows that σ_b gives the desired sparsity.

Lemma 14.3 (Sparsity after initialization)**.** *Let $b > 0$ be a tunable parameter. If we use the σ_b as the activation function, then after the initialization, with probability at least $1 - n \cdot \exp(-\Omega(m \cdot \exp(-b^2/2)))$, it holds that for each input data x_i, the number of activated neurons $k_{i,0}$ is at most $O(m \cdot \exp(-b^2/2))$, where m is the total number of neurons.*

Proof. By the concentration of Gaussian distribution, the initial fire probability of a single neuron is

$$\Pr[\sigma_b(\langle w_r(0), x_i \rangle) > 0] = \Pr_{z \sim \mathcal{N}(0,1)}[z > b] \leq \exp(-b^2/2).$$

Hence, for the indicator variable $\mathbf{1}_{r \in \mathcal{S}_{i,fire}(0)}$, we have

$$\mathbb{E}[\mathbf{1}_{r \in \mathcal{S}_{i,fire}(0)}] \leq \exp(-b^2/2).$$

By standard concentration inequality (Lemma A.4),

$$\Pr\left[|\mathcal{S}_{i,fire}(0)| > k_0 + t\right] \leq \exp\left(-\frac{t^2/2}{k_0 + t/3}\right), \forall t > 0 \tag{14.5}$$

where $k_0 := m \cdot \exp(-b^2/2)$. If we choose $t = k_0$, then we have:

$$\Pr\left[|\mathcal{S}_{i,fire}(0)| > 2k_0\right] \leq \exp\left(-3k_0/8\right)$$

Then, by union bound over all $i \in [n]$, we have that with high probability

$$1 - n \cdot \exp(-\Omega(m \cdot \exp(-b^2/2))),$$

the number of initial fire neurons for the sample x_i is bounded by $k_{i,0} \leq 2m \cdot \exp(-b^2/2)$. $\square$

The following remark gives an example of setting the threshold b, and will be useful for showing the sublinear complexity in the next section.

Remark 14.1. If we choose $b = \sqrt{0.4 \log m}$ then $k_0 = m^{4/5}$. For $t = m^{4/5}$, Eq. (14.5) implies that

$$\Pr\left[|\mathcal{S}_{i,fire}(0)| > 2m^{4/5}\right] \leq \exp\left(-\min\{mR, O(m^{4/5})\}\right).$$

14.4 Training Neural Network with Half-Space Reporting Data Structure

In this section, we present two sublinear time algorithms for training over-parameterized neural networks. The first algorithm (Section 14.4.1) relies on building a high-dimensional search data-structure for the weights of the neural network. The second algorithm (Section 14.4.2) is based on building a data structure for the input data points of the training set. Both of the algorithms use the HSR to quickly identify the fired neurons to avoid unnecessary calculations. The time complexity and the sketch of the proof are provided after each of the algorithms.

14.4.1 Weights preprocessing

We first introduce the algorithm that preprocesses the weights w_r for $r \in [m]$, which is commonly used in practice [CLP+21, CMJF+20, KKL20]. Recall $\mathsf{2NN}(m, b)$ is $f(W, x, a) := \frac{1}{\sqrt{m}} \sum_{r=1}^{m} a_r \sigma_b(\langle w_r, x\rangle)$. By constructing a HSR data-structure for w_r's, we can quickly find the set of active neurons $S_{i,fire}$ for each of the training sample x_i. See pseudo-code in Algorithm 88.

In the remaining part of this section, we focus on the time complexity analysis of Algorithm 88. The convergence proof will be given in Section 14.5.

Lemma 14.4 (Running time part of Theorem 14.8)**.** *Given n data points in d-dimensional space. Running gradient descent algorithm (Algorithm 88) on $\mathsf{2NN}(m, b = \sqrt{0.4 \log m})$ (Definition 14.1) the expected cost per-iteration of the gradient descent*

Algorithm 88 Training Neural Network via building a data structure of weights of the neural network

```
 1: procedure TrainingWithPreprocessWeights($\{(x_i, y_i)\}_{i\in[n]}$,$n$,$m$,$d$)    ▷ Theorem 14.8
 2:     Initialize $w_r, a_r$ for $r \in [m]$ and $b$ according to Definition 14.2 and Remark 14.1
 3:     HalfSpaceReport hsr.Init($\{w_r(0)\}_{r\in[m]}, m, d$)    ▷ Algorithm 87
 4:     for $t = 1 \to T$ do
 5:         $S_{i,fire} \leftarrow$ hsr.Query($x_i, b$) for $i \in [n]$
 6:         Forward pass for $x_i$ only on neurons in $S_{i,fire}$ for $i \in [n]$
 7:         Calculate gradient for $x_i$ only on neurons in $S_{i,fire}$ for $i \in [n]$
 8:         Gradient update for the neurons in $\cup_{i\in[n]} S_{i,fire}$
 9:         hsr.Delete($w_r(t)$) for $r \in \cup_{i\in[n]} S_{i,fire}$
10:         hsr.Add($w_r(t+1)$) for $r \in \cup_{i\in[n]} S_{i,fire}$
11:     end for
12:     return Trained weights $w_r(T+1)$ for $r \in [m]$
13: end procedure
```

algorithm is

$$\widetilde{O}(m^{1-\Theta(1/d)}nd).$$

Proof. The per-step time complexity is

$$\sum_{i=1}^{n} \mathcal{T}_{\text{Query}}(m, d, k_{i,t}) + (\mathcal{T}_{\text{Delete}} + \mathcal{T}_{\text{Insert}}) \cdot |\cup_{i\in[n]} S_{i,fire}(t)| + d \sum_{i\in[n]} k_{i,t}$$

The first term $\sum_{i=1}^{n} \mathcal{T}_{\text{Query}}(m, d, k_{i,t})$ corresponds to the running time of querying the active neuron set $S_{i,fire}(t)$ for all training samples $i \in [n]$. With the first result in Corollary 14.2, the complexity is bounded by $\widetilde{O}(m^{1-\Theta(1/d)}nd)$.

The second term $(\mathcal{T}_{\text{Delete}} + \mathcal{T}_{\text{Insert}}) \cdot |\cup_{i\in[n]} S_{i,fire}(t)|$ corresponds to updating w_r in the high-dimensional search data-structure (Lines 9 and 10). Again with the first result in Corollary 14.2, we have $\mathcal{T}_{\text{Delete}} + \mathcal{T}_{\text{Insert}} = O(\log^2 m)$. Combining with the fact that $|\cup_{i\in[n]} S_{i,fire}(t)| \le |\cup_{i\in[n]} S_{i,fire}(0)| \le O(nm^{4/5})$, the second term is bounded by $O(nm^{4/5} \log^2 m)$.

The third term is the time complexity of gradient calculation restricted to the

Algorithm 89 Training Neural Network via building a data-structure of the input data points

```
1: procedure TRAININGWITHPROCESSDATA($\{(x_i, y_i)\}_{i\in[n]}$,$n$,$m$,$d$)    ▷ Theorem 14.9
2:     Initialize $w_r, a_r$ for $r \in [m]$ and $b$ according to Definition 14.2 and Remark 14.1
3:     HALFSPACEREPORT HSR.INIT($\{x_i\}_{i\in[n]}, n, d$)    ▷ Algorithm 87
4:     $\widetilde{S}_{r,fire} \leftarrow$ HSR.QUERY($w_r(0), b$) for $r \in [m]$ ▷ $\widetilde{S}_{r,fire}$ are samples which neuron $r$ fires for
5:     $S_{i,fire} \leftarrow \{r \mid i \in \widetilde{S}_{r,fire}\}$    ▷ $S_{i,fire}$ is the set of neurons, which fire for $x_i$
6:     for $t = 1 \to T$ do
7:         Forward pass for $x_i$ only on neurons in $S_{i,fire}$ for $i \in [n]$
8:         Calculate gradient for $x_i$ only on neurons in $S_{i,fire}$ for $i \in [n]$
9:         Gradient update for the neurons in $\cup_{i\in[n]} S_{i,fire}$
10:        for $r \in \cup_{i\in[n]} \mathcal{S}_{i,fire}$ do
11:            $S_{i,fire}$.DEL($r$) for $i \in \widetilde{S}_{r,fire}$
12:            $\widetilde{S}_{r,fire} \leftarrow$ HSR.QUERY($w_r(t+1), b$)
13:            $S_{i,fire}$.ADD($r$) for $i \in \widetilde{S}_{r,fire}$
14:        end for
15:    end for
16:    return Trained weights $w_r(T+1)$ for $r \in [m]$
17: end procedure
```

set $\mathcal{S}_{i,fire}(t)$. With the bound on $\sum_{i\in[n]} k_{i,t}$ (Lemma 14.18), we have $d\sum_{i\in[n]} k_{i,t} \leq O(m^{4/5}nd)$.

Putting them together completes the proof. □

14.4.2 Data preprocessing

While the weights preprcessing algorithm is inspired by the common practise, the dual relationship between the input x_i and model weights w_r inspires us to preprocess the dataset before training (i.e., building HSR data-structure for x_i). This largely improves the per-iteration complexity and avoids the frequent updates of the data structure since the training data is fixed. More importantly, once the training dataset is preprocessed, it can be reused for different models or tasks, thus one does not need to perform the expensive preprocessing for each training.

The corresponding pseudocode is presented in Algorithm 89. With x_i preprocessed, we can query HSR with weights w_r and the result $\widetilde{S}_{r,fire}$ is the set of training samples x_i for which w_r fires for. Given $\widetilde{S}_{r,fire}$ for $r \in [m]$, we can easily reconstruct the set $S_{i,fire}$, which is the set of neurons fired for sample x_i. The forward and backward pass can then proceed similar to Algorithm 88.

At the end of each iteration, we will update $\widetilde{S}_{r,fire}$ based on the new w_r estimation and update $S_{i,fire}$ accordingly. For Algorithm 89, the HSR data-structure is static for the entire training process. This is the main difference from Algorithm 88, where the HSR needs to be updated every time step to account for the changing weights w_r.

We defer the convergence analysis to Section 14.5 and focus on the time complexity analysis of Algorithm 88 in the rest of this section. We consider d being a constant for the rest of this subsection.

Lemma 14.5 (Running time part of Theorem 14.9). *Given n data points in d-dimensional space. Running gradient descent algorithm (Algorithm 88) on $\mathsf{2NN}(m, b = \sqrt{0.4 \log m})$ (Definition 14.1), the expected per-iteration running time of initializing $\widetilde{S}_{r,fire}, S_{i,fire}$ for $r \in [m], i \in [n]$ is $O(m \log n + m^{4/5} n)$. The cost per iteration of the training algorithm is $O(m^{4/5} n \log n)$.*

Proof. We analyze the initialization and training parts separately.

Initialization In Lines 4 and 5, the sets $\widetilde{S}_{r,fire}, S_{i,fire}$ for $r \in [m], i \in [n]$ are initialized. For each $r \in [m]$, we need to query the data structure the set of data points x's such that $\sigma_b(w_r(0)^\top x) > 0$. Hence, the running time of this step is

$$\begin{aligned}\sum_{r=1}^{m} \mathcal{T}_{\mathsf{query}}(n, d, \widetilde{k}_{r,0}) &= O(m \log n + \sum_{r=1}^{m} \widetilde{k}_{r,0}) \\ &= O(m \log n + \sum_{i=1}^{n} k_{i,0}) \\ &= O(m \log n + m^{4/5} n).\end{aligned}$$

where the second step follows from $\sum_{r=1}^m \widetilde{k}_{r,0} = \sum_{i=1}^n k_{i,0}$.

Training Consider training the neural network for T steps. For each step, first notice that the forward and backward computation parts (Line 7 - 9) are the same as previous algorithm. The time complexity is $O(m^{4/5} n \log n)$.

We next show that maintaining $\widetilde{S}_{r,fire}, r \in [m]$ and $S_{i,fire}, i \in [n]$ (Line 10 - 14) takes $O(m^{4/5} n \log n)$ time. For each fired neuron $r \in [m]$, we first remove the indices of data in the sets $S_{i,\mathsf{fire}}$, which takes time

$$O(1) \cdot \sum_{r \in \cup_{i \in [n]} S_{i,fire}} \widetilde{k}_{r,t} = O(1) \cdot \sum_{r=1}^m \widetilde{k}_{r,t} = O(m^{4/5} n).$$

Then, we find the new set of x's such that $\sigma_b(\langle w_r(t+1), x \rangle) > 0$ by querying the half-space reporting data structure. The total running time for all fired neurons is

$$\sum_{r \in \cup_{i \in [n]} S_{i,fire}} \mathcal{T}_{\mathsf{query}}(n, d, \widetilde{k}_{r,t+1}) \lesssim m^{4/5} n \log n + \sum_{r \in \cup_{i \in [n]} S_{i,fire}} \widetilde{k}_{r,t+1} = O(m^{4/5} n \log n)$$

Then, we update the index sets $S_{i,fire}$ in time $O(m^{4/5} n)$. Therefore, each training step takes $O(m^{4/5} n \log n)$ time, which completes the proof. $\square$

14.5 Convergence of Our Algorithm

We state the result of our training neural network algorithms (Lemma 14.7) can converge in certain steps. An important component in our proof is to find out a lower bound on minimum eigenvalue of the continuous Hessian matrix $\lambda_{\min}(H^{cts})$. It turns out to be an anti-concentration problem of the Gaussian random matrix. In [OS20], they gave a lower bound on $\lambda_{\min}(H^{cts})$ for ReLU function with $b = 0$, assuming the input data are separable. One of our major technical contribution is generalizing it to arbitrary $b \geq 0$.

Proposition 14.6 (Informal version of Theorem 14.31). *Given n (normalized) input data points $\{x_1, x_2, \cdots, x_n\} \subseteq \mathbb{R}^d$ such that $\forall i \in [n], \|x_i\|_2 = 1$. Let parameter $\delta :=$*

$\min_{i \neq j}\{\|x_i - x_j\|_2, \|x_i + x_j\|_2\}$ *denote the data separability. For any shift parameter* $b \geq 0$, *we define shifted NTK* $H^{cts} \in \mathbb{R}^{n \times n}$ *as follows*

$$H^{cts}_{i,j} := \mathbb{E}_{w \sim \mathcal{N}(0, I_d)} \left[\langle x_i, x_j \rangle \cdot \mathbf{1}_{\langle w, x_i \rangle \geq b} \cdot \mathbf{1}_{\langle w, x_j \rangle \geq b} \right], \forall i \in [n], j \in [n].$$

Then

$$\lambda_{\min}(H^{cts}) \geq 0.01 e^{-b^2/2} \delta / n^2.$$

With proposition 14.6, we are ready to show the convergence rate of training an over-parameterized neural network with shifted ReLU function.

Lemma 14.7 (Convergence part of Theorem 14.8 and Theorem 14.9)**.** *Suppose input data-points are* δ*-separable, i.e.,* $\delta := \min_{i \neq j}\{\|x_i - x_j\|_2, \|x_i + x_j\|_2\}$. *Let* $m = \mathrm{poly}(n, 1/\delta, \log(n/\rho))$ *and* $\eta = O(\lambda/n^2)$. *Let* $b = \Theta(\sqrt{\log m})$. *Then*

$$\Pr\left[\|\mathrm{err}(k)\|_2^2 \leq (1 - \eta\lambda/2)^k \cdot \|\mathrm{err}(0)\|_2^2, \ \forall k \in \{0, 1, \cdots, T\} \right] \geq 1 - \rho.$$

Note that the randomness is over initialization. Eventually, we choose $T = \lambda^{-2} n^2 \log(n/\epsilon)$ *where* ϵ *is the final accuracy.*

This result shows that despite the shifted ReLU and sparsely activated neurons, we can still retain the linear convergence. Combined with the results on per-step complexity in the previous section, it gives our main theoretical results of training deep learning models with sublinear time complexity (Theorem 14.8 and Theorem 14.9).

14.6 Main Classical Results

We present two theorems (under classical computation model) of our work, showing the sublinear running time and linear convergence rate of our two algorithms. We leave the quantum application into Appendix 14.14. The first algorithm is relying on building a high-dimensional geometric search data-structure for the weights of a neural network.

Theorem 14.8 (Main result I, informal of Theorem 14.30)**.** *Given n data points in d-dimensional space. We preprocess the initialization weights of the neural network. Running gradient descent algorithm (Algorithm 88) on a two-layer, m-width, over-parameterized ReLU neural network will minimize the training loss to zero, and the expected running time of gradient descent algorithm (per iteration) is*

$$\widetilde{O}(m^{1-\Theta(1/d)}nd).$$

The second algorithm is based on building a data structure for the input data points of the training set. Our second algorithm can further reduce the cost per iteration from $m^{1-1/d}$ to truly sublinear in m, e.g. $m^{4/5}$.

Theorem 14.9 (Main result II, informal of Theorem 14.30)**.** *Given n data points in d-dimensional space. We preprocess all the data points. Running gradient descent algorithm (Algorithm 89) on a two-layer, m-width, over-parameterized ReLU neural network will minimize the training loss to zero, and the expected running time of gradient descent algorithm (per iteration) is*

$$\widetilde{O}(m^{4/5}nd).$$

14.7 Discussion

In this chapter, we propose two sublinear algorithms to train neural networks. By preprocessing the weights of the neuron networks or preprocessing the training data, we rigorously prove that it is possible to train a neuron network with sublinear complexity, which overcomes the $\Omega(mnd)$ barrier in classical training methods. Our results also offer theoretical insights for many previously established fast training methods.

Our algorithm is intuitively related to the lottery tickets hypothesis [FC18]. However, our theoretical results can not be applied to explain lottery tickets immediately for two reasons: 1) the lottery ticket hypothesis focuses on pruning weights;

while our results identify the important neurons. 2) the lottery ticket hypothesis identifies the weights that need to be pruned after training (by examining their magnitude), while our algorithms accelerate the training via preprocessing. It would be interesting to see how our theory can be extended to the lottery ticket hypothesis.

Roadmap. In Section 14.8, we present our main algorithms. In Section 14.9, we provide some preliminaries. In Section 14.10, we provide sparsity analysis. We show convergence analysis in Section 14.11. In Section 14.12, we show how to combine the sparsity, convergence, running time all together. In Section 14.13, we show correlation between sparsity and spectral gap of Hessian in neural tangent kernel. In Section 14.14, we discuss how to generalize our result to quantum setting.

14.8 Complete Algorithms

In this section, we present three algorithms (Alg. 90, Alg. 91 and Alg. 92) which are the complete version of Alg. 87, Alg. 88 and Alg. 89.

Algorithm 90 Half Space Report Data Structure

```
1: data structure HalfSpaceReport
2:     procedures:
3:         Init($S, n, d$)    ▷ Initialize the data structure with a set $S$ of $n$ points in $\mathbb{R}^d$
4:         Query($a, b$)      ▷ $a, b \in \mathbb{R}^d$. Output the set $\{x \in S : sgn(\langle a, x \rangle - b) \geq 0\}$
5:         Add($x$)           ▷ Add a point $x \in \mathbb{R}^d$ to $S$
6:         Delete($x$)        ▷ Delete the point $x \in \mathbb{R}^d$ from $S$
7: end data structure
```

Algorithm 91 Training Neural Network via building a data structure of weights.

```
1: procedure TrainingWithPreprocessWeights($\{x_i\}_{i\in[n]}, \{y_i\}_{i\in[n]}, n, m, d$)    ▷ Theorem 14.8
2:     /*Initialization step*/
3:     Sample $W(0)$ and $a$ according to Definition 14.2
4:     $b \leftarrow \sqrt{0.4\log m}$.
5:     /*A dynamic data-structure*/
6:     HalfSpaceReport hsr    ▷ Algorithm 87, Part 1 of Corollary 14.2
7:     hsr.Init($\{w_r(0)\}_{r\in[m]}, m, d$)    ▷ It takes $\mathcal{T}_{\mathsf{init}}(m,d)$ time
8:     /*Iterative step*/
9:     for $t = 0 \to T$ do
10:        /*Forward computation step*/
11:        for $i = 1 \to n$ do
12:            $S_{i,\mathrm{fire}} \leftarrow$ hsr.Query($x_i, b$)    ▷ It takes $\mathcal{T}_{\mathsf{query}}(m,d,k_{i,t})$ time
13:            $u(t)_i \leftarrow \frac{1}{\sqrt{m}} \sum_{r\in\mathcal{S}_{i,\mathrm{fire}}} a_r \cdot \sigma_b(w_r(t)^\top x_i)$    ▷ It takes $O(d\cdot k_{i,t})$ time
14:        end for
15:        /*Backward computation step*/
16:        $P \leftarrow 0^{n\times m}$    ▷ $P \in \mathbb{R}^{n\times m}$
17:        for $i = 1 \to n$ do
18:            for $r \in \mathcal{S}_{i,\mathrm{fire}}$ do
19:                $P_{i,r} \leftarrow \frac{1}{\sqrt{m}} a_r \cdot \sigma_b'(w_r(t)^\top x_i)$
20:            end for
21:        end for
22:        $M \leftarrow X\mathrm{diag}(y - u(t))$    ▷ $M \in \mathbb{R}^{d\times n}$, it takes $O(n\cdot d)$ time
23:        $\Delta W \leftarrow \underbrace{M}_{d\times n}\underbrace{P}_{n\times m}$    ▷ $\Delta W \in \mathbb{R}^{d\times m}$, it takes $O(d\cdot nnz(P))$ time, $nnz(P) = O(nm^{4/5})$
24:        $W(t+1) \leftarrow W(t) - \eta\cdot\Delta W$.
25:        /*Update data structure*/
26:        Let $Q \subset [m]$ where for each $r \in Q$, the $\Delta W_{*,r}$ is not all zeros    ▷ $|Q| \le O(nm^{4/5})$
27:        for $r \in Q$ do
28:            hsr.Delete($w_r(t)$)
29:            hsr.Insert($w_r(t+1)$)
30:        end for
31:    end for
32:    return $W$    ▷ $W \in \mathbb{R}^{d\times m}$
33: end procedure
```

Algorithm 92 Training Neural Network via building a data-structure of the input points.

```
1: procedure TrainingWithProcessData($\{x_i\}_{i\in[n]}, \{y_i\}_{i\in[n]}$,$n$,$m$,$d$)   ▷ Theorem 14.9
2:     /*Initialization step*/
3:     Sample $W(0)$ and $a$ according to Definition 14.2
4:     $b \leftarrow \sqrt{0.4\log m}$.
5:     /*A static data-structure*/
6:     HalfSpaceReport hsr   ▷ Algorithm 87, Part 2 of Corollary 14.2
7:     hsr.Init($\{x_i\}_{i\in[n]}, n, d$)   ▷ It takes $\mathcal{T}_{\mathsf{init}}(n,d)$ time
8:     /*Initialize $\widetilde{S}_{r,\mathrm{fire}}$ and $S_{i,\mathrm{fire}}$ */
9:         ▷ It takes $\sum_{r=1}^m \mathcal{T}_{\mathsf{query}}(n,d,\widetilde{k}_{r,t}) = O(m\log n + m^{1/2}n)$ time
10:    $\widetilde{S}_{r,\mathrm{fire}} \leftarrow \emptyset$ for $r \in [m]$.   ▷ $\widetilde{S}_{r,\mathrm{fire}}$ is the set of samples, for which neuron $r$ fires
11:    $S_{i,\mathrm{fire}} \leftarrow \emptyset$ for $i \in [n]$.   ▷ $S_{i,\mathrm{fire}}$ is the set of neurons, which fire for $x_i$
12:    for $r = 1 \to m$ do
13:        $\widetilde{S}_{r,\mathrm{fire}} \leftarrow$ hsr.Query($w_r(0), b$)
14:        for $i \in \widetilde{S}_{r,\mathrm{fire}}$ do
15:            $S_{i,\mathrm{fire}}$.Add($r$)
16:        end for
17:    end for
18:    /*Iterative step*/
19:    for $t = 1 \to T$ do
20:        /*Forward computation step*/
21:        for $i = 1 \to n$ do
22:            $u(t)_i \leftarrow \frac{1}{\sqrt{m}} \sum_{r\in\mathcal{S}_{i,\mathrm{fire}}} a_r \cdot \sigma_b(w_r(t)^\top x_i)$   ▷ It takes $O(d \cdot k_{i,t})$ time
23:        end for
24:        /*Backward computation step*/
25:        $P \leftarrow 0^{n\times m}$   ▷ $P \in \mathbb{R}^{n\times m}$
26:        for $i = 1 \to n$ do
27:            for $r \in \mathcal{S}_{i,\mathrm{fire}}$ do
28:                $P_{i,r} \leftarrow \frac{1}{\sqrt{m}} a_r \cdot \sigma_b'(w_r(t)^\top x_i)$
29:            end for
30:        end for
31:        $M \leftarrow X\mathrm{diag}(y - u(t))$   ▷ $M \in \mathbb{R}^{d\times n}$, it takes $O(n \cdot d)$ time
32:        $\Delta W \leftarrow \underbrace{M}_{d\times n}\underbrace{P}_{n\times m}$   ▷ $\Delta W \in \mathbb{R}^{d\times m}$, it takes $O(d \cdot nnz(P))$ time,
   $nnz(P) = O(nm^{4/5})$
33:        $W(t+1) \leftarrow W(t) - \eta \cdot \Delta W$.
34:        /*Update $\widetilde{S}_{r,\mathrm{fire}}$ and $S_{i,\mathrm{fire}}$ step*/
35:            ▷ It takes $O(\sum_{i=1}^n k_{i,t} + \sum_{r\in S_{[n],\mathrm{fire}}} \mathcal{T}_{\mathsf{query}}(n,d,\widetilde{k}_{r,t+1})) = O(n \cdot \log n \cdot m^{4/5})$
36:        $S_{[n],\mathrm{fire}} \leftarrow \cup_{i\in[n]} \mathcal{S}_{i,\mathrm{fire}}$
37:        for $r \in S_{[n],\mathrm{fire}}$ do
38:            for $i \in \widetilde{S}_{r,\mathrm{fire}}$ do   ▷ Removing old fired neuron indices. It takes $O(\widetilde{k}_{r,t})$ time
39:                $S_{i,\mathrm{fire}}$.Del($r$)
40:            end for
41:            $\widetilde{S}_{r,\mathrm{fire}} \leftarrow$ hsr.Query($w_r(t+1), b$)   ▷ It takes $\mathcal{T}_{\mathsf{query}}(n,d,\widetilde{k}_{r,t+1})$ time
42:            for $i \in \widetilde{S}_{r,\mathrm{fire}}$ do   ▷ Adding new fired neuron indices. It takes $O(\widetilde{k}_{r,t+1})$ time
43:                $S_{i,\mathrm{fire}}$.Add($r$)
44:            end for
45:        end for
46:    end for
```

14.9 Preliminaries

Notations For an integer n, we use $[n]$ to denote the set $\{1, 2, \cdots, n\}$. For a vector x, we use $\|x\|_2$ to denote the entry-wise ℓ_2 norm of a vector. We use $\mathbb{E}[]$ to denote the expectation and $\Pr[]$ to denote the probability. We use $M^\top$ to denote the transpose of M. We define matrix Frobenius norm as $\|M\|_F = (\sum_{i,j} M_{i,j}^2)^{1/2}$. We use $\|M\|$ to denote the operator norm of M. For $d \times m$ weight matrix W, we define $\|W\|_{\infty,2} := \max_{r \in [m]} \|w_r\|_2$. We use $x^\top y$ to denote the inner product between vectors x and y. We use I_d to denote d-dimensional identity matrix. We use $\mathcal{N}(\mu, \sigma^2)$ to denote Gaussian distribution with mean μ and variance σ^2. We use $\lambda_{\min}(M)$ and $\lambda_{\max}(M)$ to denote the minimum and the maximum eigenvalue of the matrix M, respectively.

14.9.1 Half-space reporting data structures

The time complexity of HSR data structure is:

Theorem 14.10 (Agarwal, Eppstein and Matousek [AEM92])**.** *Let d be a fixed constant. Let t be a parameter between n and $n^{\lfloor d/2 \rfloor}$. There is a dynamic data structure for half-space reporting that uses $O_{d,\epsilon}(t^{1+\epsilon})$ space and pre-processing time, $O_{d,\epsilon}(\frac{n}{t^{1/\lfloor d/2 \rfloor}} \log n + k)$ time per query where k is the output size and $\epsilon > 0$ is any fixed constant, and $O_{d,\epsilon}(t^{1+\epsilon}/n)$ amortized update time.*

As a direct corollary, we have

Corollary 14.11 (HSR data-structure time complexity [AEM92])**.** *Given a set of n points in $\mathbb{R}^d$, the half-space reporting problem can be solved with the following performances:*

- *Part 1.*$\mathcal{T}_{\mathsf{init}}(n, d) = O_d(n \log n)$, $\mathcal{T}_{\mathsf{query}}(n, d, k) = O_{d,\epsilon}(n^{1-1/\lfloor d/2 \rfloor + \epsilon} + k)$, *amortized* $\mathcal{T}_{\mathsf{update}} = O_{d,\epsilon}(\log^2(n))$.
- *Part 2.*$\mathcal{T}_{\mathsf{init}}(n, d) = O_{d,\epsilon}(n^{\lfloor d/2 \rfloor + \epsilon})$, $\mathcal{T}_{\mathsf{query}}(n, d, k) = O_{d,\epsilon}(\log(n) + k)$, *amortized* $\mathcal{T}_{\mathsf{update}} = O_{d,\epsilon}(n^{\lfloor d/2 \rfloor - 1 + \epsilon})$.

14.9.2 Basic algebras

Claim 14.12 ([Sch11]). *Let $M_1, M_2 \in \mathbb{R}^{n\times n}$ be two PSD matrices. Let $M_1 \circ M_2$ denote the Hadamard product of M_1 and M_2. Then,*

$$\lambda_{\min}(M_1 \circ M_2) \geq (\min_{i\in[n]} M_{2i,i}) \cdot \lambda_{\min}(M_1),$$
$$\lambda_{\max}(M_1 \circ M_2) \leq (\max_{i\in[n]} M_{2i,i}) \cdot \lambda_{\max}(M_1).$$

14.10 Sparsity Analysis

14.10.1 Bounding difference between continuous kernel and discrete kernel

In [DZPS19, SY19], they proved the following lemma for $b = 0$. Here, we provide a more general statement for any $b \geq 0$.

Lemma 14.13. *For any shift parameter $b \geq 0$, we define continuous version of shifted NTK H^{cts} and discrete version of shifted NTK H^{dis} as:*

$$H^{cts}_{i,j} := \mathbb{E}_{w\sim\mathcal{N}(0,I)} \left[x_i^\top x_j \mathbf{1}_{w^\top x_i \geq b, w^\top x_j \geq b}\right],$$
$$H^{dis}_{i,j} := \frac{1}{m}\sum_{r=1}^{m} \left[x_i^\top x_j \mathbf{1}_{w_r^\top x_i \geq b, w_r^\top x_j \geq b}\right].$$

We define $\lambda := \lambda_{\min}(H^{cts})$.

Let $m = \Omega(\lambda^{-1} n \log(n/\rho))$ be number of samples of H^{dis}, then

$$\Pr\left[\lambda_{\min}(H^{dis}) \geq \frac{3}{4}\lambda\right] \geq 1 - \rho.$$

Proof. We will use the matrix Chernoff bound (Theorem A.6) to provide a lower bound on the least eigenvalue of discrete version of shifted NTK H^{dis}.

Let $H_r := \frac{1}{m}\widetilde{X}(w_r)\widetilde{X}(w_r)^\top$, where $\widetilde{X}(w_r) \in \mathbb{R}^{d\times n}$ is defined as follows:

$$\widetilde{X}(w_r) = \begin{bmatrix}\mathbf{1}_{w_r^\top x_i \geq b} \cdot x_1 & \cdots & \mathbf{1}_{w_r^\top x_n \geq b} \cdot x_n\end{bmatrix}.$$

Hence, $H_r \succeq 0$. We need to upper-bound $\|H_r\|$. Naively, we have

$$\|H_r\| \leq \|H_r\|_F \leq \frac{n}{m},$$

since for each entry at $(i,j) \in [n] \times [n]$,

$$(H_r)_{i,j} = \frac{1}{m} x_i^\top x_j \mathbf{1}_{w_r^\top x_i \geq b, w_r^\top x_j \geq b} \leq \frac{1}{m} x_i^\top x_j \leq \frac{1}{m}.$$

Then, $H^{dis} = \sum_{r=1}^m H_r$, and $\mathbb{E}[H^{dis}] = H^{cts}$. And we assume that $\lambda_{\min}(H^{cts}) = \lambda$.

Hence, by matrix Chernoff bound (Theorem A.6) and choosing choose $m = \Omega(\lambda^{-1} n \cdot \log(n/\rho))$, we can show

$$\begin{aligned} \Pr\left[\lambda_{\min}(H^{dis}) \leq \frac{3}{4}\lambda\right] &\leq n \cdot \exp(-\frac{1}{16}\lambda/(2n/m)) \\ &= n \cdot \exp(-\frac{\lambda m}{32n}) \\ &\leq \rho, \end{aligned}$$

Thus, we finish the proof. □

14.10.2 Handling Hessian if perturbing weight

We present a tool which is inspired by a list of previous work [DZPS19, SY19].

Lemma 14.14 (perturbed w for shifted NTK)**.** *Let $b > 0$ and $R \leq 1/b$. Let $c > 0$ and $c' > 0$ denote two fxied constants. We define function H that is mapping $\mathbb{R}^{m \times d}$ to $\mathbb{R}^{n \times n}$ as follows:*

$$\textit{the } (i,j)\textit{-th entry of } H(W) \textit{ is } \frac{1}{m} x_i^\top x_j \sum_{r=1}^m \mathbf{1}_{w_r^\top x_i \geq b, w_r^\top x_j \geq b}.$$

Let $\widetilde{W} \in \mathbb{R}^{d \times m}$ be m vectors that are sampled from $\mathcal{N}(0, I_d)$. Consider $W \in \mathbb{R}^{d \times m}$ that satisfy, $\|\widetilde{W} - W\|_{\infty,2} \leq R$, it has

- *Part 1, $\|H(\widetilde{W}) - H(W)\|_F \leq n \cdot \min\{c \cdot \exp(-b^2/2), 3R\}$ holds with probability at least $1 - n^2 \cdot \exp(-m \cdot \min\{c' \cdot \exp(-b^2/2), R/10\})$.*

- *Part 2,* $\lambda_{\min}(H(W)) \geq \frac{3}{4}\lambda - n \cdot \min\{c \cdot \exp(-b^2/2), 3R\}$ *holds with probability at least* $1 - n^2 \cdot \exp(-m \cdot \min\{c' \cdot \exp(-b^2/2), R/10\}) - \rho$.

Proof. Consider

$$\begin{aligned}\|H(W) - H(\widetilde{W})\|_F^2 &= \sum_{i\in[n]}\sum_{j\in[n]}(H(\widetilde{W})_{i,j} - H(W)_{i,j})^2 \\ &\leq \frac{1}{m^2}\sum_{i\in[n]}\sum_{j\in[n]}\left(\sum_{r\in[m]} \mathbf{1}_{\widetilde{w}_r^\top x_i\geq b, \widetilde{w}_r^\top x_j \geq b} - \mathbf{1}_{w_r^\top x_i \geq b, w_r^\top x_j\geq b}\right)^2 \\ &= \frac{1}{m^2}\sum_{i\in[n]}\sum_{j\in[n]}\Big(\sum_{r\in[m]} s_{r,i,j,b}\Big)^2,\end{aligned}$$

where the first step follows from definition of Frobenius norm, the last third step follows from by defining

$$s_{r,i,j,b} := \mathbf{1}_{\widetilde{w}_r^\top x_i\geq b, \widetilde{w}_r^\top x_j \geq b} - \mathbf{1}_{w_r^\top x_i \geq b, w_r^\top x_j\geq b}.$$

For simplicity, we use s_r to $s_{r,i,j,b}$ (note that we fixed (i,j) and b).

Define $A_{i,r}$ to be the event that

$$A_{i,r} = \{\exists w \in \mathbb{R}^d : \|w - w_r\|_2 \leq R, \mathbf{1}_{w^\top x_i \geq b} \neq \mathbf{1}_{w_r^\top x_i\geq b}\}.$$

Note that event $A_{i,r}$ happens iff $|w_r^\top x_i - b| \leq R$ happens.

Prior work [DZPS19, SY19] only one way to bound $\Pr[A_{i,r}]$. We present two ways of arguing the upper bound on $\Pr[A_{i,r}]$. One is anti-concentration, and the other is concentration.

By anticoncentration, (Lemma A.7),

$$\Pr[A_{i,r}] \leq \frac{2R}{\sqrt{2\pi}} \leq R.$$

By concentration,

$$\Pr[A_{i,r}] \leq \exp(-(b-R)^2/2) \leq c_1 \cdot \exp(-b^2/2).$$

where the last step follows from $R < 1/b$ and $c_1 \geq \exp(1 - R^2/2)$ is a constant.

Hence,

$$\Pr[A_{i,r}] \leq \min\{R, c_1 \exp(-b^2/2)\}.$$

If the event $\neg A_{i,r}$ happens and the event $\neg A_{j,r}$ happens, then we have

$$\left|\mathbf{1}_{\widetilde{w}_r^\top x_i \geq b, \widetilde{w}_r^\top x_j \geq b} - \mathbf{1}_{w_r^\top x_i \geq b, w_r^\top x_j \geq b}\right| = 0.$$

If the event $A_{i,r}$ happens or the event $A_{j,r}$ happens, then we obtain

$$\left|\mathbf{1}_{\widetilde{w}_r^\top x_i \geq b, \widetilde{w}_r^\top x_j \geq b} - \mathbf{1}_{w_r^\top x_i \geq b, w_r^\top x_j \geq b}\right| \leq 1.$$

Case 1: $c_1 \exp(-b^2/2) < R$. So we have

$$\begin{aligned} \mathbb{E}_{\widetilde{w}_r}[s_r] \leq &~ \Pr[A_{i,r}] + \Pr[A_{j,r}] \\ \leq &~ c_1 \cdot \exp(-b^2/2) \end{aligned}$$

Now, we calculate the variance

$$\begin{aligned} \mathbb{E}_{\widetilde{w}_r}\left[(s_r - \mathbb{E}_{\widetilde{w}_r}[s_r])^2\right] = &~ \mathbb{E}_{\widetilde{w}_r}[s_r^2] - \mathbb{E}_{\widetilde{w}_r}[s_r]^2 \\ \leq &~ \mathbb{E}_{\widetilde{w}_r}[s_r^2] \\ \leq &~ \mathbb{E}_{\widetilde{w}_r}\left[\left(\mathbf{1}_{A_{i,r} \vee A_{j,r}}\right)^2\right] \\ \leq &~ c_1 \cdot \exp(-b^2/2). \end{aligned}$$

Note that $|s_r| \leq 1$ for all r.

Define $\overline{s} = \frac{1}{m}\sum_{r=1}^m s_r$. Thus, we are able to use Lemma A.4,

$$\begin{aligned} \Pr\left[m \cdot \overline{s} \geq m \cdot c_1 \exp(-b^2/2) + mt\right] \leq &~ \Pr\left[\sum_{r=1}^m (s_r - \mathbb{E}[s_r]) \geq mt\right] \\ \leq &~ \exp\left(-\frac{m^2t^2/2}{m \cdot c_1 \exp(-b^2/2) + mt/3}\right), \quad \forall t \geq 0. \end{aligned}$$

Define $\overline{s} = \frac{1}{m}\sum_{r=1}^m s_r$. Thus, it gives

$$\Pr\left[\overline{s} \geq c_2 \cdot \exp(-b^2/2)\right] \leq \exp(-c_3 \cdot m \exp(-b^2/2)),$$

where $c_2 := 2c_1, c_3 := \frac{3}{8}c_1$ are some constants.

Case 2: $\exp(-b^2/2) > R$**.** Then, we have

$$\mathbb{E}_{\widetilde{w}_r}[s_r] \leq 2R, \quad \mathbb{E}_{\widetilde{w}_r}\left[(s_r - \mathbb{E}_{\widetilde{w}_r}[s_r])^2\right] \leq 2R.$$

Define $\overline{s} = \frac{1}{m}\sum_{r=1}^m s_r$. By Lemma A.4,

$$\Pr\left[\overline{s} \geq 3R\right] \leq \exp\left(-mR/10\right).$$

Combining two cases:

Thus, we obtain

$$\begin{aligned} &\Pr\left[\|H(\widetilde{W}) - H(W)\|_F \leq n \cdot \min\{c_2 \exp(-b^2/2), 3R\}\right] \\ \geq\ & 1 - n^2 \cdot \exp(-m \cdot \min\{c_3 \exp(-b^2/2), R/10\}). \end{aligned}$$

For the second part, by Lemma 14.14, $\Pr[\lambda_{\min}(H(\widetilde{W})) \geq 0.75 \cdot \lambda] \geq 1 - \rho$. Hence,

$$\begin{aligned} \lambda_{\min}(H(W)) &\geq \lambda_{\min}(H(\widetilde{W})) - \|H(\widetilde{W}) - H(W)\| \\ &\geq \lambda_{\min}(H(\widetilde{W})) - \|H(\widetilde{W}) - H(W)\|_F \\ &\geq 0.75 \cdot \lambda - n \cdot \min\{c_2 \cdot \exp(-b^2/2), 3R\}, \end{aligned}$$

which happens with probability $1 - n^2 \cdot \exp(-m \cdot \min\{c_3 \cdot \exp(-b^2/2), R/10\}) - \rho$ by the union bound. $\square$

14.10.3 Total movement of weights

Definition 14.6 (Hessian matrix at time t)**.** For $t \geq 0$, let $H(t)$ be an $n \times n$ matrix with (i,j)-th entry:

$$H(t)_{i,j} := \frac{1}{m} x_i^\top x_j \sum_{r=1}^m \mathbf{1}_{\langle w_r(t), x_i\rangle \geq b} \mathbf{1}_{\langle w_r(t), x_j\rangle \geq b}$$

We follow the standard notation D_{cts} in Lemma 3.5 in [SY19].

Definition 14.7 (D_{cts})**.** Let $y \in \mathbb{R}^n$ be the vector of the training data labels. Let $\mathrm{err}(0) \in \mathbb{R}^n$ denote the error of prediction of the neural network function (Definition 14.3). Define the actual moving distance of weight D_{cts} to be

$$D_{cts} := \lambda^{-1} \cdot m^{-1/2} \cdot \sqrt{n} \cdot \|\mathrm{err}(0)\|_2.$$

We state a tool from previous work [DZPS19, SY19] (more specifically, Lemma 3.4 in [DZPS19], Lemma 3.6 in [SY19]). Since adding the shift parameter b to NTK doesn't affect the proof of the following lemma, thus we don't provide a proof and refer the readers to prior work.

Lemma 14.15 ([DZPS19, SY19])**.** *The condition $D_{cts} < R$ implies $\lambda_{\min}(H(t)) \geq \lambda/2$, $\forall t \geq 0$. Let $\mathrm{err}(t)$ be defined as Definition 14.3. Further,*

1. $\|W(t) - W(0)\|_{\infty,2} \leq D_{cts}$,
2. $\|\mathrm{err}(t)\|_2^2 \leq \exp(-\lambda t) \cdot \|\mathrm{err}(0)\|_2^2$.

14.10.4 Bounded gradient

The proof of Lemma 3.6 in [SY19] implicitly implies the following basic property of gradient.

Claim 14.16 (Bounded gradient)**.** *Let $\mathrm{err}(s)$ be defined as Definition 14.3. For any $0 \leq s \leq t$, We have*

$$\left\| \frac{\partial L(W(s))}{\partial w_r(s)} \right\|_2 \leq \frac{\sqrt{n}}{\sqrt{m}} \|\mathrm{err}(s)\|_2$$
$$\left\| \frac{\mathrm{d}}{\mathrm{d}s} w_r(s) \right\|_2 \leq \frac{\sqrt{n}}{\sqrt{m}} \|\mathrm{err}(s)\|_2$$

Proof. For the first part,

$$\begin{aligned}\Big\|\frac{\partial L(W(s))}{\partial w_r(s)}\Big\|_2 &= \Big\|\sum_{i=1}^n \mathrm{err}_i(s)\frac{1}{\sqrt{m}}a_r x_i \cdot \mathbf{1}_{w_r(s)^\top x_i \geq b}\Big\|_2 && \text{by Eq. (14.4)}\\ &\leq \frac{1}{\sqrt{m}}\sum_{i=1}^n |\mathrm{err}_i(s)| && \text{by Eq. (14.2)}\\ &\leq \frac{\sqrt{n}}{\sqrt{m}}\|\mathrm{err}(s)\|_2.\end{aligned}$$

For second part, we use ODE to prove it.

□

14.10.5 Upper bound on the movement of weights per iteration

The following Claim is quite standard in the literature, we omitt the details.

Claim 14.17 (Corollary 4.1 in [DZPS19], Lemma 3.8 in [SY19])**.** *Let* $\mathrm{err}(i)$ *be defined as Definition 14.3. If* $\forall i \in [t], \|\mathrm{err}(i)\|_2^2 \leq (1-\eta\lambda/2)^i \cdot \|\mathrm{err}(0)\|_2^2$*, then*

$$\|W(t+1) - W_r(0)\|_{\infty,2} \leq 4\lambda^{-1}m^{-1/2} \cdot \sqrt{n} \cdot \|\mathrm{err}(0)\|_2 := D.$$

14.10.6 Bounding the number of fired neuron per iteration

In this section, we will show that for $t = 0, 1, \ldots, T$, the number of fire neurons $k_{i,t} = |\mathcal{S}_{i,\mathrm{fire}}(t)|$ is small with high probability.

We define the set of neurons that are flipping at time t:

Definition 14.8 (flip set)**.** For each $i \in [n]$, for each time $t \in [T]$ let $\mathcal{S}_{i,\mathrm{flip}}(t) \subset [m]$ denote the set of neurons that are never flipped during the entire training process,

$$\mathcal{S}_{i,\mathrm{flip}}(t) := \{r \in [m] : \ sgn(\langle w_r(t), x_i\rangle - b) \neq sgn(\langle w_r(t-1), x_i\rangle - b)\}.$$

Over all the iterations of training algorithm, there are some neurons that never flip states. We provide a mathematical formulation of that set,

Definition 14.9 (noflip set)**.** For each $i \in [n]$, let $S_i \subset [m]$ denote the set of neurons that are never flipped during the entire training process,

$$S_i := \{r \in [m] : \forall t \in [T] \; sgn(\langle w_r(t), x_i \rangle - b) = sgn(\langle w_r(0), x_i \rangle - b)\}. \tag{14.6}$$

In Lemma 14.3, we already show that $k_{i,0} = O(m \cdot \exp(-b^2/2))$ for all $i \in [n]$ with high probability. We can show that it also holds for $t > 0$.

Lemma 14.18 (Bounding the number of fired neuron per iteration)**.** *Let $b \geq 0$ be a parameter, and let $\sigma_b(x) = \max\{x, b\}$ be the activation function. For each $i \in [n], t \in [T]$, $k_{i,t}$ is the number of activated neurons at the t-th iteration. For $0 < t \leq T$, with probability at least $1 - n \cdot \exp\left(-\Omega(m) \cdot \min\{R, \exp(-b^2/2)\}\right)$, $k_{i,t}$ is at most $O(m \exp(-b^2/2))$ for all $i \in [n]$.*

Proof. We prove this lemma by induction.

The base case of $t = 0$ is shown by Lemma 14.3 that $k_{i,0} = O(m \cdot \exp(-b^2/2))$ for all $i \in [n]$ with probability at least $1 - n \exp(-\Omega(m \cdot \exp(-b^2/2)))$.

Assume that the statement holds for $0, \ldots, t-1$. By Claim 14.17, we know $\forall k < t$,

$$\|W(k+1) - W(0)\|_{\infty,2} < R.$$

Consider the t-th iteration. For each $i \in [n]$, consider the set of activated neurons $\mathcal{S}_{i,\mathrm{fire}}$. We note that for the neurons in S_i, with high probability these neurons will not be activated in the t-th iteration if they are not activated in the $(t-1)$-th iteration. By Claim 14.19, for $r \in [m]$,

$$\Pr[r \notin S_i] \leq \min\left\{R, O(\exp(-b^2/2))\right\}.$$

On the one hand, if $R < O(\exp(-b^2/2))$, then $\mathbb{E}[|\overline{S}_i|] \leq mR$. By Lemma A.4,

$$\Pr\left[|\overline{S}_i| > t\right] \leq \exp\left(-\frac{t^2/2}{mR + t/3}\right).$$

If we take $t := mR$, then we have

$$\Pr\left[|\overline{S}_i| > mR\right] \leq \exp\left(-3mR/8\right).$$

On the other hand, if $O(\exp(-b^2/2)) < R$, then $\mathbb{E}[|\overline{S_i}|] \leq O(m\exp(-b^2/2))$. By Lemma A.4, we have that

$$\Pr\left[|\overline{S}_i| > t\right] \leq \exp\left(-\frac{t^2/2}{O(m\exp(-b^2/2)) + t/3}\right).$$

If we take $t := m\exp(-b^2/2)$, we have that

$$\Pr\left[|\overline{S}_i| > m\exp(-b^2/2)\right] \leq \exp(-\Omega(m\exp(-b^2/2))).$$

Then, we know that in addition to the fire neurons in $S_{i,\text{noflip}}$, there are at most $m \cdot \min\{R, \exp(-b^2/2)\}$ neurons are activated in t-th iteration with high probability.

By a union bound for $i \in [n]$, we obtain with probability

$$\geq 1 - n \cdot \exp(-\Omega(m) \cdot \min\{R, \exp(-b^2/2)\}),$$

the number of activated neurons for x_i at the t-th iteration of the algorithm is

$$k_{i,t} = |S_{i,\text{fire}}(t)| \leq k_{i,0} + m\min\{R, \exp(-b^2/2)\} \leq O(m\exp(-b^2/2)),$$

where the last step follows from $k_{i,0} = O(m\exp(-b^2/2))$ by Lemma 14.3.

The Lemma is then proved for all $t = 0, \ldots, T$. $\square$

Claim 14.19 (Bound on noflip probability). *Let $R \leq 1/b$. For $i \in [n]$, let S_i be the set defined by Eq. (14.6).*

Part 1. *For $r \in [m]$, $r \notin S_i$ if and only if $|\langle w_r(0), x_i\rangle - b| < R$.*

Part 2. *If $w_r(0) \sim \mathcal{N}(0, I_d)$, then*

$$\Pr[r \notin S_i] \leq \min\{R, O(\exp(-b^2/2))\} \quad \forall r \in [m].$$

Proof. **Part 1.** We first note that $r \notin S_i \subset [m]$ is equivalent to the event that

$$\exists w \in \mathbb{R}^d, \text{s.t.} \mathbf{1}_{\langle w_r(0), x_i \rangle \geq b} \neq \mathbf{1}_{\langle w, x_i \rangle \geq b} \wedge \|w - w_r(0)\|_2 < R.$$

Assume that $\|w - w_r(0)\|_2 = R$. Then, we can write $w = w_r(0) + R \cdot v$ with $\|v\|_2 = 1$ and $\langle w, x_i \rangle = \langle w_r(0), x_i \rangle + R \cdot \langle v, x_i \rangle$.

Now, suppose there exists a w such that $\mathbf{1}_{\langle w_r(0), x_i \rangle \geq b} \neq \mathbf{1}_{\langle w, x_i \rangle \geq b}$.

- If $\langle w_r(0), x_i \rangle > b$, then there exists a vector $v \in \mathbb{R}^d$ such that $R \cdot \langle v, x_i \rangle < b - \langle w_r(0), x_i \rangle$,
- If $\langle w_r(0), x_i \rangle < b$, then there exists a vector $v \in \mathbb{R}^d$ such that $R \cdot \langle v, x_i \rangle > b - \langle w_r(0), x_i \rangle$.

Since $\|x_i\|_2 = 1$ and $\langle v, x_i \rangle \in [-1, 1]$, we can see that the above conditions hold if and only if

$$\begin{aligned} b - \langle w_r(0), x_i \rangle &> -R, \quad \text{and} \\ b - \langle w_r(0), x_i \rangle &< +R. \end{aligned}$$

In other words, $r \notin S_i$ if and only if $|\langle w_r(0), x_i \rangle - b| < R$.

Part 2.

We have

$$\begin{aligned} \Pr[r \notin S_i] = & \Pr_{z \sim \mathcal{N}(0,1)}[|z - b| < R] && \text{by } \langle w_r, x_i \rangle \sim \mathcal{N}(0,1) \\ \leq & \Pr_{z \sim \mathcal{N}(0,1)}[|z| < R] && \text{by symmetric property of Gaussian distribution} \\ \leq & \frac{2R}{\sqrt{2\pi}} && \text{by anti-concentration inequality of Gaussian (Lemma A.7)} \\ \leq & R. \end{aligned}$$

On the other hand, we also know

$$\Pr[r \notin S_i] \leq \Pr_{z \sim \mathcal{N}(0,1)}[z \geq b - R] \leq \exp(-(b - R)^2/2) \leq O(\exp(-b^2/2)),$$

where the last step follows from $R < 1/b$. $\square$

14.11 Convergence Analysis

14.11.1 Upper bound the initialization

The following Claim provides an upper bound for initialization. Prior work only shows it for $b = 0$, we generalize it to $b \geq 0$. The modification to the proof of previous Claim 3.10 in [SY19] is quite straightforward, thus we omit the details here.

Claim 14.20 (Upper bound the initialization, shited NTK version of Claim 3.10 in [SY19])**.** *Let $b \geq 0$ denote the NTK shifted parameter. Let parameter $\rho \in (0, 1)$ denote the failure probability. Then*

$$\Pr[\|\mathrm{err}(0)\|_2^2 = O(n(1 + b^2) \log^2(n/\rho))] \geq 1 - \rho.$$

14.11.2 Bounding progress per iteration

In previous work, [SY19] define H and $H^\perp$ only for $b = 0$. In this section, we generalize it to $b \geq 0$. Let us define two shifted matrices H and $H^\perp$

$$H(k)_{i,j} := \frac{1}{m} \sum_{r=1}^{m} \langle x_i, x_j \rangle \mathbf{1}_{\langle w_r(k), x_i \rangle \geq b, \langle w_r(k), x_j \rangle \geq b}, \tag{14.7}$$

$$H(k)_{i,j}^{\perp} := \frac{1}{m} \sum_{r \in \overline{S}_i} \langle x_i, x_j \rangle \mathbf{1}_{\langle w_r(k), x_i \rangle \geq b, \langle w_r(k), x_j \rangle \geq b}. \tag{14.8}$$

We define

$$v_{1,i} := \frac{1}{\sqrt{m}} \sum_{r \in S_i} a_r(\sigma_b(w_r(k+1)^\top x_i) - \sigma_b(w_r(k)^\top x_i))$$

$$v_{2,i} := \frac{1}{\sqrt{m}} \sum_{r \in \overline{S}_i} a_r(\sigma_b(w_r(k+1)^\top x_i) - \sigma_b(w_r(k)^\top x_i)) \tag{14.9}$$

Following the same proof as Claim 3.9 [SY19], we can show that the following Claim. The major difference between our claim and Claim 3.9 in [SY19] is, they only proved it for the case $b = 0$. We generalize it to $b \geq 0$. The proof is several basic algebra computations, we omit the details here.

Claim 14.21 (Shifted NTK version of Claim 3.9 in [SY19])**.** *Let* $\mathrm{err}(k) = y - u(k)$ *be defined as Definition 14.3.*

$$\|\mathrm{err}(k+1)\|_2^2 = \|\mathrm{err}(k)\|_2^2 + B_1 + B_2 + B_3 + B_4,$$

where

$$\begin{aligned}
B_1 :=& \ -2\eta \cdot \mathrm{err}(k)^\top \cdot H(k) \cdot \mathrm{err}(k), \\
B_2 :=& \ +2\eta \cdot \mathrm{err}(k)^\top \cdot H(k)^\perp \cdot \mathrm{err}(k), \\
B_3 :=& \ -2\mathrm{err}(k)^\top v_2, \\
B_4 :=& \ +\|u(k+1) - u(k)\|_2^2.
\end{aligned}$$

The nontrivial parts in our analysis is how to bound B_1, B_2, B_3 and B_4 for the shifted cases (We will provide a proof later). Once we can bound all these terms, we can show the following result for one iteration of the algorithm:

Lemma 14.22 (Shifted NTK version of Page 13 in [SY19])**.** *We have*

$$\|\mathrm{err}(k+1)\|_2^2 \leq \|\mathrm{err}(k)\|_2^2 \cdot (1 - \eta\lambda + 4\eta n \cdot \min\{R, \exp(-b^2/2)\} + \eta^2 n^2))$$

holds with probability at least

$$1 - 2n^2 \cdot \exp(-\Omega(m) \cdot \min\{R, \exp(-b^2/2)\}) - \rho.$$

Proof. We are able to provide the following upper bound for $\|\mathrm{err}(k+1)\|_2^2$:

$$\begin{aligned}
& \|\mathrm{err}(k+1)\|_2^2 \\
= & \ \|\mathrm{err}(k)\|_2^2 + B_1 + B_2 + B_3 + B_4 && \text{by Claim 14.21} \\
\leq & \ \|\mathrm{err}(k)\|_2^2 (1 - \eta\lambda + 4\eta n \cdot \min\{R, \exp(-b^2/2)\} + \eta^2 n^2) && \text{by Claim 14.24, 14.25, 14.26 and 14.27}
\end{aligned}$$

□

14.11.3 Upper bound on the norm of dual Hessian

The proof of the following fact is similar to Fact C.1 in [SY19]. We generalize the $b = 0$ to $b \geq 0$. The same bound will hold as Fact C.1 in [SY19] if we replace $\mathbf{1}_{w_r(k)^\top x_i \geq 0}$ by $\mathbf{1}_{w_r(k)^\top x_i \geq b}$. Thus, we omit the details here.

Fact 14.23 (Shifted NTK version of Fact C.1 in [SY19])**.** *Let $b \geq 0$. Let shifted matrix $H(k)^\perp$ be defined as Eq. (14.8). For all $k \geq 0$, we have*

$$\|H(k)^\perp\|_F \leq \frac{n}{m^2} \sum_{i=1}^n |\overline{S}_i|^2.$$

14.11.4 Bounding the gradient improvement term

Claim 14.24 (Bounding the gradient improvement term)**.** *Let $H(k)$ be shifted matrix (see Eq. (14.7)). Assume $b \geq 0$. Denote $\rho_0 = n^2 \cdot \exp(-m \cdot \min\{c' \cdot \exp(-b^2/2), R/10\}) + \rho$. We define $B_1 := -2\eta \mathrm{err}(k)^\top H(k) \mathrm{err}(k)$. Assuming either of the following condition,*

- $R \leq \frac{\lambda}{12n}$,
- $b \geq \sqrt{2 \cdot \log(4cn/\lambda)}$.

Then, we have

$$\Pr[B_1 \leq -\eta\lambda \cdot \|\mathrm{err}(k)\|_2^2] \geq 1 - \rho_0.$$

Proof. By Lemma 14.14, there exists constants $c, c' > 0$ such that

$$\lambda_{\min}(H(W)) \geq \frac{3}{4}\lambda - n \cdot \min\{c \cdot \exp(-b^2/2), 3R\}$$

with probability at least $1 - \rho_0$.

If we have $R \leq \frac{\lambda}{12n}$ or $b \geq \sqrt{2 \cdot \log(4cn/\lambda)}$, then

$$\lambda_{\min}(H(W)) \geq \frac{1}{2}\lambda.$$

Finally, we have

$$\mathrm{err}(k)^\top \cdot H(k) \cdot \mathrm{err}(k) \geq \|\mathrm{err}(k)\|_2^2 \cdot \lambda/2.$$

□

14.11.5 Bounding the blowup by the dual Hessian term

Claim 14.25 (Bounding the blowup by the dual Hessian term)**.** *Let shifted matrix $H(k)^\perp$ be defined as Eq. (14.8). Let $\rho_0 = n\exp(-\Omega(m) \cdot \min\{R, \exp(-b^2/2)\})$. Let $b \geq 0$ be shifted NTK parameter. We define $B_2 := 2\eta \cdot \mathrm{err}(k)^\top \cdot H(k)^\perp \cdot \mathrm{err}(k)$. Then*

$$\Pr[B_2 \leq 2\eta n \cdot \min\{R, \exp(-b^2/2)\} \cdot \|\mathrm{err}(k)\|_2^2] \geq 1 - \rho_0.$$

Proof. By property of spectral norm,

$$B_2 \leq 2\eta \|\mathrm{err}(k)\|_2^2 \|H(k)^\perp\|.$$

Using Fact 14.23, we have $\|H(k)^\perp\|_F \leq \frac{n}{m^2}\sum_{i=1}^n |\overline{S}_i|^2$.

By Lemma 14.18, $\forall i \in \{1, 2, \cdots, n\}$, it has

$$\Pr\left[|\overline{S_i}| \leq m \cdot \min\{R, \exp(-b^2/2)\}\right] \geq 1 - \rho_0. \tag{14.10}$$

Hence, with probability at least $1 - \rho_0$

$$\|H(k)^\perp\|_F^2 \leq \frac{n}{m^2} \cdot n \cdot m^2 \cdot \min\{R^2, \exp(-b^2)\} = n^2 \cdot \min\{R^2, \exp(-b^2)\}.$$

Putting all together, we have

$$\|H(k)^\perp\| \leq \|H(k)^\perp\|_F \leq n \cdot \min\{R, \exp(-b^2/2)\}$$

with probability at least $1 - \rho_0$.

□

14.11.6 Bounding the blowup by the flip-neurons term

Claim 14.26 (Bounding the blowup by flipping neurons term)**.** *Let $\rho_0 = n\exp(-\Omega(m)\cdot \min\{R, \exp(-b^2/2)\})$. We define $B_3 := -2\mathrm{err}(k)^\top v_2$. Let $b \geq 0$ be shifted NTK parameter. Then we have*

$$\Pr[B_3 \leq 2\eta n \cdot \min\{R, \exp(-b^2/2)\} \cdot \|\mathrm{err}(k)\|_2^2] \geq 1 - \rho_0.$$

Proof. Using Cauchy-Schwarz inequality, we have $B_3 \leq 2\|\mathrm{err}(k)\|_2 \cdot \|v_2\|_2$.

Then we focus on $\|v_2\|_2$,

$$\begin{aligned}
\|v_2\|_2^2 \leq & \sum_{i=1}^n \left(\frac{\eta}{\sqrt{m}} \sum_{r\in \overline{S}_i} \left| (\frac{\partial L(W(k))}{\partial w_r(k)})^\top x_i \right| \right)^2 && \text{by Eq. (14.9)} \\
= & \frac{\eta^2}{m} \sum_{i=1}^n \left(\sum_{r=1}^m \mathbf{1}_{r\in\overline{S}_i} \left| (\frac{\partial L(W(k))}{\partial w_r(k)})^\top x_i \right| \right)^2 \\
\leq & \frac{\eta^2}{m} \cdot \max_{r\in[m]} \left| \frac{\partial L(W(k))}{\partial w_r(k)} \right|^2 \cdot \sum_{i=1}^n \left(\sum_{r=1}^m \mathbf{1}_{r\in\overline{S}_i} \right)^2 \\
\leq & \frac{\eta^2}{m} \cdot (\frac{\sqrt{n}}{\sqrt{m}} \|\mathrm{err}(k)\|_2)^2 \cdot \sum_{i=1}^n \left(\sum_{r=1}^m \mathbf{1}_{r\in\overline{S}_i} \right)^2 && \text{by Claim 14.16} \\
\leq & \frac{\eta^2}{m} \cdot (\frac{\sqrt{n}}{\sqrt{m}} \|\mathrm{err}(k)\|_2)^2 \cdot \sum_{i=1}^n m^2 \cdot \min\{R^2, \exp(-b^2)\} && \text{by Eq. (14.10)} \\
= & \eta^2 n^2 \cdot \min\{R^2, \exp(-b^2)\} \cdot \|\mathrm{err}(k)\|_2^2,
\end{aligned}$$

□

14.11.7 Bounding the blowup by the prediction movement term

The proof of the following Claim is quite standard and simple in literature, see Claim 3.14 in [SY19]. We omit the details here.

Claim 14.27 (Bounding the blowup by the prediction movement term)**.**

$$B_4 \leq \eta^2 n^2 \cdot \|\mathrm{err}(k)\|_2^2.$$

14.11.8 Putting it all together

The goal of this section to combine all the convergence analysis together.

Lemma 14.28 (Convergence). *Let $\eta = \lambda/(4n^2)$, $R = \lambda/(12n)$, let $b \in [0, n]$, and*

$$m \geq \Omega(\lambda^{-4} n^4 b^2 \log^2(n/\rho)),$$

we have

$$\Pr\left[\|\mathrm{err}(t)\|_2^2 \leq (1 - \eta\lambda/2)^t \cdot \|\mathrm{err}(0)\|_2^2\right] \geq 1 - 2\rho.$$

Proof. We know with probability $\geq 1 - 2n^2 \cdot \exp(-\Omega(m) \cdot \min\{R, \exp(-b^2/2)\}) - \rho$,

$$\|\mathrm{err}(t+1)\|_2^2 \leq \|\mathrm{err}(t)\|_2^2 \cdot (1 - \eta\lambda + 4\eta n \cdot \min\{R, \exp(-b^2/2)\} + \eta^2 n^2)),$$

and we want to show that

$$1 - \eta\lambda + 4\eta n \cdot \min\{R, \exp(-b^2/2)\} + \eta^2 n^2 \leq 1 - \eta\lambda/2, \quad \text{and} \tag{14.11}$$

$$2n^2 \cdot \exp(-\Omega(m) \cdot \min\{R, \exp(-b^2/2)\}) \leq \rho. \tag{14.12}$$

Claim 14.17 requires the following relationship between D and R,

$$D = \frac{4\sqrt{n}\|\mathrm{err}(0)\|_2}{\sqrt{m}\lambda} < R$$

By Claim 14.20, we can upper bound the prediction error at the initialization,

$$\|\mathrm{err}(0)\|_2^2 = O(nb^2 \log^2(n/\rho)),$$

Combining the above two equations gives

$$R > \Omega(\lambda^{-1} n m^{-1/2} b \log(n/\rho)). \tag{14.13}$$

Claim 14.24 (where $0 < c < e$ is a constant) requires an upper bound on R,[4]

$$R \leq \frac{\lambda}{12n}. \tag{14.14}$$

[4]Due to the relationship between b and λ, we are not allowed to choose b in an arbitrary function of λ. Thus, we should only expect to use R to fix the problem.

Combing the lower bound and upper bound of R, it implies the lower bound on m in our Lemma statement.

And Lemma 14.13 also requires that

$$m = \Omega(\lambda^{-1} n \log(n/\rho)). \tag{14.15}$$

which is dominated by the lower bound on m in our lemma statement, thus we can ignore it.

Lemma 14.13 and Claim 14.19 require that

$$R < 1/b. \tag{14.16}$$

which is equivalent to

$$b < 12n/\lambda$$

However, by Theorem 14.31, it will always hold for any $b > 0$.

Note that Eq. (14.11) can be rewritten as

$$4\eta n \cdot \min\{R, \exp(-b^2/2)\} + \eta^2 n^2 \leq \eta\lambda/2.$$

where it follows from taking $\eta := \lambda/(4n^2)$ and $R = \lambda/(12n)$.

Therefore, we can take the choice of the parameters m, b, R and Eqs. (14.11), (14.12) imply

$$\Pr[\|\mathrm{err}(t+1)\|_2^2 \leq (1 - \eta\lambda/2) \cdot \|\mathrm{err}(t)\|_2^2] \geq 1 - 2\rho.$$

□

14.12 Combine

Corollary 14.29 (Sublinear cost per iteration)**.** *Let n denote the number of points. Let d denote the dimension of points. Let $\rho \in (0, 1/10)$ denote the failure probability.*

Let δ be the separability of data points. For any parameter $\alpha \in (0,1]$, we choose $b = \sqrt{0.5(1-\alpha)\log m}$, if

$$m = \Omega((\delta^{-4} n^{10} \log^4(n/\rho))^{1/\alpha})$$

then the sparsity is

$$O(m^{\frac{3+\alpha}{4}}).$$

Furthermore,

- *If we preprocess the initial weights of the neural network, then we choose $\alpha = 1 - 1/\Theta(d)$ to get the desired running time.*
- *If we preprocess the training data points, then we choose α to be an arbitrarily small constant to get the desired running time.*

Proof. From Theorem 14.31, we know

$$\lambda \geq \exp(-b^2/2) \cdot \frac{\delta}{100n^2}$$

which is equivalent to

$$\lambda^{-1} \leq \exp(b^2/2) \cdot \frac{100n^2}{\delta}.$$

For convergence, we need

$$m = \Omega(\lambda^{-4} n^4 b^2 \log^2(n/\rho))$$

Since we know the upper bound of λ^{-1}, thus we need to choose

$$m = \Omega(\exp(4 \cdot b^2/2) \cdot \delta^{-4} \cdot n^{10} b^2 \log^2(n/\rho))$$

From sparsity, we have

$$O(m \cdot \exp(-b^2/2))$$

Let us choose $b = \sqrt{0.5(1-\alpha)\log m}$, for any $\alpha \in (0, 1]$.

For the lower bound on m, we obtain

$$m \geq (\delta^{-4} n^{10} \log^4(n/\rho))^{1/\alpha}$$

For the sparsity, we obtain

$$m \cdot m^{-(1-\alpha)/4} = m^{\frac{3+\alpha}{4}}$$

$\square$

Theorem 14.30 (Main result, formal of Theorem 14.8 and 14.9)**.** *Given n data points in d-dimensional space. Running gradient descent algorithm on a two-layer ReLU (over-parameterized) neural network with m neurons in the hidden layers is able to minimize the training loss to zero, let $\mathcal{T}_{\mathsf{init}}$ denote the preprocessing time and $\mathcal{C}_{\mathsf{iter}}$ denote the cost per iteration of gradient descent algorithm.*

- *If we preprocess the initial weights of the neural network (Algorithm 88), then*

$$\mathcal{T}_{\mathsf{init}} = O_d(m \log m), \mathcal{C}_{\mathsf{iter}} = \widetilde{O}(m^{1-\Theta(1/d)} nd).$$

- *If we preprocess the training data points (Algorithm 89), then*

$$\mathcal{T}_{\mathsf{init}} = O(n^d), \mathcal{C}_{\mathsf{iter}} = \widetilde{O}(m^{3/4+o(1)} nd).$$

14.13 Bounds for the Spectral Gap with Data Separation

Theorem 14.31 (Formal version of Proposition 14.6)**.** *Let $x_1, \ldots, x_n$ be points in $\mathbb{R}^d$ with unit Euclidean norm and $w \sim \mathcal{N}(0, I_d)$. Form the matrix $X \in \mathbb{R}^{n \times d} = [x_1 \ \ldots \ x_n]^\top$. Suppose there exists $\delta \in (0, \sqrt{2})$ such that*

$$\min_{i \neq j \in [n]} \{\|x_i - x_j\|_2, \|x_i + x_j\|_2\} \geq \delta.$$

Let $b \geq 0$. Recall the continuous Hessian matrix H^{cts} is defined by

$$H^{cts}_{i,j} := \mathbb{E}_{w \sim \mathcal{N}(0,I)} \left[x_i^\top x_j \mathbf{1}_{w^\top x_i \geq b, w^\top x_j \geq b} \right] \quad \forall (i,j) \in [n] \times [n].$$

Let $\lambda := \lambda_{\min}(H^{cts})$. Then, we have

$$\exp(-b^2/2) \geq \ \lambda \geq \ \exp(-b^2/2) \cdot \frac{\delta}{100n^2}. \tag{14.17}$$

Proof. **Part 1: Lower bound.**

Define the covariance of the vector $\mathbf{1}_{Xw>b} \in \mathbb{R}^n$ as

$$\mathbb{E}_{w \sim \mathcal{N}(0,I_d)} \left[(\mathbf{1}_{Xw>b})(\mathbf{1}_{Xw>b})^\top \right].$$

Then, H^{cts} can be written as

$$H^{cts} = \mathbb{E}_{w \sim \mathcal{N}(0,I_d)} \left[(\mathbf{1}_{Xw>b})(\mathbf{1}_{Xw>b})^\top \right] \circ XX^\top,$$

where $A \circ B$ denotes the Hadamard product between A and B.

By Claim 14.12, and since $\|x_i\|_2 = 1$ for all $i \in [n]$, we only need to show:

$$\mathbb{E}_{w \sim \mathcal{N}(0,I_d)} \left[(\mathbf{1}_{Xw>b})(\mathbf{1}_{Xw>b})^\top \right] \succeq \exp(-b^2/2) \cdot \frac{\delta}{100n^2} \cdot I_n.$$

Fix a unit length vector $a \in \mathbb{R}^n$. Suppose there exist constants c_1, c_2 such that

$$\Pr \left[\left| a^\top \mathbf{1}_{Xw>b} \right| \geq c_1 \|a\|_\infty \right] \geq \frac{c_2 \delta}{n}. \tag{14.18}$$

This would imply that

$$\begin{aligned} \mathbb{E} \left[\left(a^\top \mathbf{1}_{Xw>b} \right)^2 \right] &\geq \mathbb{E} \left[\left| a^\top \mathbf{1}_{Xw>b} \right| \right]^2 \\ &\geq c_1^2 \|a\|_\infty^2 (\frac{c_2 \delta}{n})^2 \\ &\geq c_1^2 c_2 \frac{\delta}{n^2}, \end{aligned}$$

where the first step follows from Jensen's inequality, the second step follows from Markov's inequality, the last step follows from $\|a\|_2 = 1$.

Since this is true for all a, we find Eq. (14.17) with $c_1^2 c_2 = \frac{1}{100}$ by choosing $c_1 = 1/2, c_2 = 1/25$ as described later.

Hence, our goal is proving Eq. (14.18). Without loss of generality, assume $|a_1| = \|a\|_\infty$ and construct an orthonormal basis $Q \in \mathbb{R}^{d\times d}$ in $\mathbb{R}^d$ where the first column is equal to $x_1 \in \mathbb{R}^d$ and $Q = [x_1 \ \overline{Q}] \in \mathbb{R}^{d\times d}$. Note that $g = Q^\top w \sim \mathcal{N}(0, I_d)$ and we have

$$w = Qg = g_1 x_1 + \overline{Q}\overline{g},$$

where $g = \begin{bmatrix} g_1 \\ \overline{g} \end{bmatrix} \in \mathbb{R}^d$ and the first step follows from $QQ^\top = I_d$.

For $0 \le \gamma \le 1/2$, Gaussian small ball guarantees

$$\Pr[|g_1| \le \gamma] \ge \frac{7\gamma}{10}.$$

Then, by Theorem 3.1 in [LS01] (Lemma A.8), we have

$$\Pr[|g_1 - b| \le \gamma] \ge \exp(-b^2/2) \cdot \frac{7\gamma}{10}.$$

Next, we argue that $z_i := \langle \overline{Q}\overline{g}, x_i\rangle$ is small for all $i \neq 1$. For a fixed $i \ge 2$, observe that

$$z_i \sim \mathcal{N}(0, 1 - \langle x_1, x_i\rangle^2).$$

Let $\tau_{i,1} := \langle x_i, x_1\rangle$.

Note that δ-separation implies

$$1 - |\langle x_1, x_i\rangle| = \frac{1}{2}\min\{\|x_1 - x_i\|_2^2, \|x_1 + x_i\|_2^2\} \ge \frac{\delta^2}{2}$$

Hence $|\tau_{i,1}| \le 1 - \delta^2/2$.

Then, from Gaussian anti-concentration bound (Lemma A.7) and variance bound on z_i, we have

$$\begin{aligned}\Pr[|z_i| \leq |\tau_{i,1}|\gamma] &\leq \sqrt{\frac{2}{\pi}}\frac{|\tau_{i,1}|\gamma}{\sqrt{1-\tau_{i,1}^2}} \\ &\leq \frac{2\gamma}{\delta\sqrt{\pi}} \\ &\leq \frac{2\gamma}{\delta},\end{aligned}$$

which implies that

$$\Pr[|z_i - (1-\tau_{i,1})b| \leq |\tau_{i,1}|\cdot\gamma] \leq \Pr[|z_i| \leq \gamma] \leq \frac{2\gamma}{\delta}.$$

Hence, by union bound,

$$\Pr[\forall i \in \{2,\cdots,n\} : |z_i - (1-\tau_{i,1})b| \leq |\tau_{i,1}|\cdot\gamma] \geq 1 - n\frac{2\gamma}{\delta}$$

Define $\mathcal{E}$ to be the following event:

$$\mathcal{E} := \Big\{|g_1 - b| \leq \gamma \text{ and } |z_i - (1-\tau_{i,1})b| \leq |\tau_{i,1}|\cdot\gamma, \quad \forall i \in \{2,\cdots,n\}\Big\}.$$

Since $g_1 \in \mathbb{R}$ is independent of $\overline{g}$, we have

$$\begin{aligned}\Pr[\mathcal{E}] =&\ \Pr[|g_1 - b| \leq \gamma]\cdot\Pr[\forall i \in \{2,\cdots,n\} : |z_i - (1-\tau_{i,1})b| \leq |\tau_{i,1}|\cdot\gamma] \\ \geq&\ \exp(-b^2/2)\cdot\frac{7\gamma}{10}\cdot(1-2n\gamma/\delta) \\ \geq&\ \exp(-b^2/2)\cdot\frac{7\delta}{80n}.\end{aligned}$$

where the last step follows from choosing $\gamma := \frac{\delta}{4n} \in [0,1/2]$.

To proceed, define

$$\begin{aligned}f(g) &:= \langle a, \mathbf{1}_{Xw>b}\rangle \\ &= a_1\cdot\mathbf{1}_{g_1>b} + \sum_{i=2}^n (a_i\cdot\mathbf{1}_{x_i^\top x_1\cdot g_1 + x_i^\top\overline{Q}\overline{g}>b}) \\ &= a_1\cdot\mathbf{1}_{g_1>b} + \sum_{i=2}^n (a_i\cdot\mathbf{1}_{\tau_{i,1}\cdot g_1 + x_i^\top\overline{Q}\overline{g}>b}).\end{aligned}$$

where the third step follows from $\tau_{i,1} = x_i^\top x_1$.

On the event $\mathcal{E}$, by Claim 14.32, we have that $\mathbf{1}_{\tau_{i,1} \cdot g_1 + z_i > b} = \mathbf{1}_{z_i > (1-\tau_{i,1})b}$.

Hence, conditioned on $\mathcal{E}$,

$$f(g) = a_1 \mathbf{1}_{g_1 > b} + \mathrm{rest}(\overline{g}),$$

where

$$\mathrm{rest}(\overline{g}) := \sum_{i=2}^{n} a_i \cdot \mathbf{1}_{x_i^\top \overline{Q} \overline{g} > (1-\tau_{i,1})b}.$$

Furthermore, conditioned on $\mathcal{E}$, $g_1, \overline{g}$ are independent as z_i's are function of $\overline{g}$ alone. Hence, $\mathcal{E}$ can be split into two equally likely events that are symmetric with respect to g_1 i.e. $g_1 \geq b$ and $g_1 < b$.

Consequently,

$$\Pr\Big[|f(g)| \geq \max\{|a_1 \mathbf{1}_{g_1 > b} + \mathrm{rest}(\overline{g})|, |a_1 \mathbf{1}_{g_1 < b} + \mathrm{rest}(\overline{g})|\} \; \Big| \; \mathcal{E}\Big] \geq 1/2 \tag{14.19}$$

Now, using $\max\{|a|, |b|\} \geq |a - b|/2$, we find

$$\begin{aligned}
& \Pr[|f(g)| \geq 0.5|a_1| \cdot |\mathbf{1}_{g_1 > b} - \mathbf{1}_{g_1 < b}| \mid \mathcal{E}] \\
= & \Pr[|f(g)| \geq 0.5|a_1| \mid \mathcal{E}] \\
= & \Pr[|f(g)| \geq 0.5\|a\|_\infty \mid \mathcal{E}] \\
\geq & 1/2,
\end{aligned}$$

where $|a_1| = \|a\|_\infty$.

This yields

$$\Pr[|f(g)| \geq \|a\|_\infty/2] \geq \Pr[\mathcal{E}]/2 \geq \exp(-b^2/2) \cdot \frac{7\delta}{160n}.$$

Part 2: Upper bound.

$$\begin{aligned}\lambda &= \lambda_{\min}(H^{cts}) \\ &= \min_{x \in \mathbb{R}^d : \|x\|_2 = 1} x^\top H^{cts} x \\ &\leq e_1^\top H^{cts} e_1 \\ &= (H^{cts})_{1,1} \\ &= \mathbb{E}_w \left[x_1^\top x_1 \mathbf{1}_{w^\top x_1 \geq b,} \right] \\ &= \Pr_w [w^\top x_1 \geq b] \\ &\leq \exp(-b^2/2),\end{aligned}$$

where $e_1 := \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^\top$, and the sixth step follows from $\|x_1\|_2 = 1$, the last step follows from the concentration of Gaussian distribution. In Line 5 and 6 of the above proof, w is sampled from $\mathcal{N}(0, I_d)$. $\square$

Claim 14.32. *Suppose $|g_1 - b| \leq \gamma$.*

- *If $\tau_{i,1} > 0$, then $|z_i - (1 - \tau_{i,1})b| > +\tau_{i,1}\gamma$ implies that $\mathbf{1}_{\tau_{i,1} \cdot g_1 + z_i > b} = \mathbf{1}_{z_i > (1-\tau_{i,1})b}$.*
- *If $\tau_{i,1} < 0$, then $|z_i - (1 - \tau_{i,1})b| > -\tau_{i,1}\gamma$ implies that $\mathbf{1}_{\tau_{i,1} \cdot g_1 + z_i > b} = \mathbf{1}_{z_i > (1-\tau_{i,1})b}$.*

That is, if $|z_i - (1 - \tau_{i,1})b| > |\tau_{i,1}|\gamma$, then we have $\mathbf{1}_{\tau_{i,1} \cdot g_1 + z_i > b} = \mathbf{1}_{z_i > (1-\tau_{i,1})b}$.

Proof. **Case 1.** We can assume $\tau_{i,1} > 0$. By assumption, we know that $g_1 \in [b - \gamma, b + \gamma]$.

Consider the forward direction first.

If $\tau_{i,1} g_1 + z_i > b$, then

$$z_i > b - \tau_{i,1}(b + \gamma) = (1 - \tau_{i,1})b - \tau_{i,1}\gamma.$$

According to the range of z_i, it implies $z_i > (1 - \tau_{i,1})b$.

Then, consider the backward direction.

If $z_i > (1-\tau_{i,1})b$, then by the range of z_i, we have $z_i > (1-\tau_{i,1})b + \tau_{i,1}\gamma$.

Hence,

$$\tau_{i,1}g_1 + z_i > \tau_{i,1}(b-\gamma) + (1-\tau_{i,1})b + \tau_{i,1}\gamma = b.$$

Case 2. The $\tau_{i,1} < 0$ case can be proved in a similar way. $\square$

14.14 Quantum Algorithm for Training Neural Network

In this section, we provide a quantum-classical hybrid approach to train neural networks with truly sub-quadratic time per iteration. The main observation is that the classical HSR data structure can be replaced with Grover's search algorithm in quantum.

We first state our main result in below, showing the running time of our quantum training algorithm:

Corollary 14.33 (Main theorem)**.** *Given n data points in d-dimensional space. Running gradient descent algorithm on a two-layer, m-with, over-parameterized, and ReLU neural network will minimize the training loss to zero, let $\mathcal{C}_{\mathsf{iter}}$ denote the cost per iteration of gradient descent algorithm. Then, we have*

$$\mathcal{C}_{\mathsf{iter}} = \widetilde{O}(m^{9/10}nd).$$

by applying Grover's search algorithm for the neurons (Algorithm 93) or the input data points (Algorithm 94).

Remark 14.2. We remark that previous works ([KLP19, AHKZ20]) on training classical neural networks use the quantum linear algebra approach, which achieves quantum speedup in the linear algebra operations in the training process. For example, [KLP19] used the block encoding technique to speed up the matrix multiplication in training a convolutional neural network (CNN). [AHKZ20] used the quantum inner-product estimation to reduce each neuron's computational cost. One drawback of this approach is that the quantum linear algebra computation incurs some non-negligible errors. Hence, extra efforts of error analysis are needed to guarantee that the intermediate errors will not affect the convergence of their algorithms.

Compared with the previous works, the only quantum component of our algorithm is Grover's search. So, we do not need to worry about the quantum algorithm's error in the training process. And we are able to use our fast training framework to exploit a sparse structure, which makes Grover's search algorithm run very fast, and further leads to a truly sub-quadratic training algorithm.

Remark 14.3. We also remark the difference between the two algorithms in this quantum section the first algorithm runs Grover's search for each data point to find the activated neurons, while the second one runs Grover's search for each neuron to find the data points that make it activated. The advantage of Algorithm 94 is it uses fewer quantum resources since its search space is of size $O(n)$ and the first algorithm's search space is of size $O(m)$.

Algorithm 93 Quantum-Classical Hybrid Training Neural Network, Version 1

```
1: procedure OraclePrep($i \in [n], t \in [T]$)
2:     Prepare the quantum query oracle $\mathcal{O}_{i,t}$ such that    ▷ Each oracle call takes $O(d)$ time
$$\mathcal{O}_{i,t} : |r\rangle\,|0\rangle \mapsto \begin{cases} |r\rangle\,|1\rangle & \text{if } w_r(t)^\top x_i > b, \\ |r\rangle\,|0\rangle & \text{otherwise.} \end{cases}$$
3: end procedure
4: procedure QTrainingAlgorithmI($\{x_i\}_{i\in[n]}, \{y_i\}_{i\in[n]}, n, m, d$)    ▷ Corollary 14.33
5:     Sample $w(0)$ and $a$ according to def. 14.2
6:     $b \leftarrow \sqrt{0.4 \log m}$.
7:     for $t = 0 \to T$ do
8:         /*Quantum part*/
9:         for $i = 1 \to n$ do
10:            $\mathcal{O}_{i,t} \leftarrow$ OraclePrep($i, t$)
11:            Use Grover's search with oracle $\mathcal{O}_{i,t}$ to find the set $\mathcal{S}_{i,\mathrm{fire}} \subset [m]$
12:                                        ▷ It takes $\widetilde{O}(\sqrt{m \cdot k_{i,t}} \cdot d)$ time
13:        end for
14:        /*Classical part*/
15:        for $i = 1 \to n$ do
16:            $u(t)_i \leftarrow \frac{1}{\sqrt{m}} \sum_{r\in\mathcal{S}_{i,\mathrm{fire}}} a_r \sigma_b(w_r(t)^\top x_i)$    ▷ It takes $O(d \cdot k_{i,t})$ time
17:        end for
18:        for $i = 1 \to n$ do
19:            for $r \in \mathcal{S}_{i,\mathrm{fire}}$ do
20:                $P_{i,r} \leftarrow \frac{1}{\sqrt{m}} a_r \sigma_b'(w_r(t)^\top x_i)$
21:            end for
22:        end for
23:        $M \leftarrow X \mathrm{diag}(y - u(t))$    ▷ $M \in \mathbb{R}^{d\times n}$, it takes $O(n \cdot d)$ time
24:        $\Delta W \leftarrow MP$    ▷ $\Delta W \in \mathbb{R}^{d\times m}$, it takes $O(d \cdot nnz(P))$ time
25:        $W(t+1) \leftarrow W(t) - \eta \cdot \Delta W$.
26:    end for
27:    return $W$
28: end procedure
```

Algorithm 94 Quantum-Classical Hybrid Training Neural Network, Version 2.

```
1: procedure QTrainingAlgorithmII($\{x_i\}_{i\in[n]}, \{y_i\}_{i\in[n]}, n, m, d$)    ▷ Corollary 14.33
2:     Sample $w(0)$ and $a$ according to def. 14.2
3:     $b \leftarrow \sqrt{0.4\log m}$.
4:     /*Initialize $\widetilde{S}_{r,\mathrm{fire}}$ and $S_{i,\mathrm{fire}}$ */    ▷ It takes $\sum_{r=1}^m \widetilde{O}((n\widetilde{k}_{r,t})^{1/2}d) \leq O(m^{9/10}nd)$ time in total.
5:     $\widetilde{S}_{r,\mathrm{fire}} \leftarrow \emptyset$ for $r \in [m]$.
6:     $S_{i,\mathrm{fire}} \leftarrow \emptyset$ for $i \in [n]$.
7:     for $r = 1 \to m$ do
8:         $\widetilde{S}_{r,\mathrm{fire}} \leftarrow$ use Grover's serach to find all $i \in [n]$ s.t. $\sigma_b(w_r(1)^\top x_i) \neq 0$.
9:         for $i \in \widetilde{S}_{r,\mathrm{fire}}$ do
10:            $S_{i,\mathrm{fire}}$.Add($r$)
11:        end for
12:    end for
13:    /*Iterative step*/
14:    for $t = 0 \to T$ do
15:        for $i = 1 \to n$ do
16:            $u(t)_i \leftarrow \frac{1}{\sqrt{m}} \sum_{r\in\mathcal{S}_{i,\mathrm{fire}}} a_r \cdot \sigma_b(w_r(t)^\top x_i)$    ▷ It takes $O(d \cdot k_{i,t})$ time
17:        end for
18:        $P \leftarrow 0^{n\times m}$    ▷ $P \in \mathbb{R}^{n\times m}$
19:        for $i = 1 \to n$ do
20:            for $r \in \mathcal{S}_{i,\mathrm{fire}}$ do
21:                $P_{i,r} \leftarrow \frac{1}{\sqrt{m}} a_r \cdot \sigma_b'(w_r(t)^\top x_i)$
22:            end for
23:        end for
24:        $M \leftarrow X\mathrm{diag}(y - u(t))$    ▷ $M \in \mathbb{R}^{d\times n}$, it takes $O(n \cdot d)$ time
25:        $\Delta W \leftarrow MP$    ▷ $\Delta W \in \mathbb{R}^{d\times m}$, it takes $O(m^{4/5}nd)$ time
26:        $W(t+1) \leftarrow W(t) - \eta \cdot \Delta W$.
27:        /*Update $\widetilde{S}_{r,\mathrm{fire}}$ and $S_{i,\mathrm{fire}}$ step*/    ▷ It takes $\widetilde{O}(m^{9/10}nd)$ time in total
28:        $S_{[n],\mathrm{fire}} \leftarrow \cup_{i\in[n]} \mathcal{S}_{i,\mathrm{fire}}$
29:        for $r \in S_{[n],\mathrm{fire}}$ do
30:            for $i \in \widetilde{S}_{r,\mathrm{fire}}$ do    ▷ It takes $O(\widetilde{k}_{r,t})$ time
31:                $S_{i,\mathrm{fire}}$.Del($r$)
32:            end for
33:            $\widetilde{S}_{r,\mathrm{fire}} \leftarrow$ use Grover's search to find all $i \in [n]$ s.t. $\sigma_b(w_r(t+1)^\top x_i) \neq 0$.
34:            for $i \in \widetilde{S}_{r,\mathrm{fire}}$ do    ▷ It takes $O(\widetilde{k}_{r,t+1})$ time
35:                $S_{i,\mathrm{fire}}$.Add($r$)
36:            end for
37:        end for
38:    end for
39:    return $W$    ▷ $W \in \mathbb{R}^{d\times m}$
40: end procedure
```

We first state a famous result about the quadratic quantum speedup for the unstructured search problem using Grover's search algorithm.

Theorem 14.34 (Grover's search algorithm [Gro96, BHMT02])**.** *Given access to the evaluation oracle for an unknown function $f : [n] \to \{0, 1\}$ such that $|f^{-1}(1)| = k$ for some unknown number $k \leq n$, we can find all i's in $f^{-1}(1)$ in $\widetilde{O}(\sqrt{nk})$-time quantumly.*

Lemma 14.35 (Running time)**.** *For $t = 0, 1, \ldots, T$, the time complexity of the t-th iteration in Algorithm 93 is*

$$\widetilde{O}\Big(nd\sqrt{m} \cdot \max_{i\in[n]} \sqrt{k_{i,t}}\Big),$$

where $k_{i,t} = |\mathcal{S}_{i,\mathrm{fire}}(t)|$.

Proof. We first consider the quantum part of the algorithm, which is dominated by the for-loop at Line 9. For each $i \in [n]$, we need to find the set $\mathcal{S}_{i,\mathrm{fire}}(t)$ by Grover's search, which takes $\widetilde{O}(\sqrt{mk_{i,t}} \cdot \mathcal{T}_{\mathrm{oracle}})$ time. In our case, each oracle call takes $O(d)$ time. Hence, the quantum part's running time is

$$\widetilde{O}\Big(\sum_{i=1}^{n} \sqrt{mk_{i,t}} \cdot d\Big) = \widetilde{O}\Big(nd\sqrt{m} \cdot \max_{i\in[n]} \sqrt{k_{i,t}}\Big).$$

Then, consider the classical part of the algorithm. Since we already get the sets $\mathcal{S}_{i,\mathrm{fire}}$, the for-loop at Line 15 takes $O(k_t d)$ time, and the for-loop at Line 18 takes $O(k_t)$ time, where $k_t = \sum_{i=1}^{n} k_{i,t} \leq n \cdot \max_{i\in[n]} k_{i,t}$. Then, at Line 24, we compute the matrix product $X\mathrm{diag}(y - u(t))P$. It's easy to see that $M = X\mathrm{diag}(y - u(t))$ can be computed in time $O(nd)$. Since P is a sparse matrix, MP can be computed in $O(d \cdot nnz(P)) = O(dk_t)$ time. Namely, we maintain a data structure for all the non-zero entries of P. Then calculate each row of MP in time $O(nnz(P))$. Hence, the total running time of the classical part is $O(nd \cdot \max_{i\in[n]} k_{i,t})$.

Since $k_{i,t} \leq m$ for all $i \in [n]$, the running time per iteration of Algorithm 93 is $\widetilde{O}(nd\sqrt{m} \cdot \max_{i\in[n]} \sqrt{k_{i,t}})$, which completes the proof of the lemma. □

The following lemma proves the running time of Algorithm 94.

Lemma 14.36. *For $t = 0, 1, \dots, T$, the time complexity of the t-th iteration in Algorithm 94 is*

$$\widetilde{O}\Big(\sqrt{n}d \cdot \sum_{r=1}^{m} \sqrt{\widetilde{k}_{r,t}}\Big),$$

where $\widetilde{k}_{r,t} = |\widetilde{\mathcal{S}}_{r,\mathrm{fire}}|$ at time t.

Proof. For the quantum part, the difference is at Line 8, where we use Grover's search to find the data points such that the r-th neuron is activated. By Theorem 7.5, it takes $\widetilde{O}((n\widetilde{k}_{r,0})^{1/2})$-time quantumly. And at Line 33, we re-compute $\widetilde{\mathcal{S}}_{r,\mathrm{fire}}$, which takes $\widetilde{O}((n\widetilde{k}_{r,t+1})^{1/2})$-time quantumly. Thus, the quantum running time of Algorithm 94 is $\widetilde{O}(\sum_{r\in[m]}(n\widetilde{k}_{r,t})^{1/2})$ per iteration.

The classical part is quite similar to Algorithm 92, which takes $O(nd \cdot \max_{i\in[n]} k_{i,t})$-time per iteration.

Therefore, the cost per iteration is $\widetilde{O}(\sum_{r\in[m]}(n\widetilde{k}_{r,t})^{1/2})$, and the lemma is then proved. $\square$

Combining Lemma 14.35 and Lemma 14.36 proves the main result of this section:

Proof of Corollary 14.33. In Section 14.12, we prove that $k_{i,t} = m^{4/5}$ with high probability for all $i \in [n]$ if we take $b = \sqrt{0.4 \log m}$. Hence, by Lemma 14.35, each iteration in Algorithm 93 takes

$$\widetilde{O}\Big(nd\sqrt{m} \cdot \max_{i\in[n]} \sqrt{k_{i,t}}\Big) = \widetilde{O}\Big(ndm^{9/10}\Big)$$

time in quantum. On the other hand, by Lemma 14.36, each iteration in Algorithm 94 takes quantum time

$$\begin{aligned} \widetilde{O}\Big(\sqrt{n}d \sum_{r\in[m]} \sqrt{\widetilde{k}_{r,t}}\Big) &\leq \widetilde{O}\Big(\sqrt{n}d\sqrt{m} \sum_{r\in[m]} \widetilde{k}_{r,t}\Big) && \text{(Cauchy-Schwartz inequality.)} \\ &= \widetilde{O}\Big(\sqrt{n}d\sqrt{m}\big(\sum_{i\in[n]} k_{i,t}\big)^{1/2}\Big) \\ &= \widetilde{O}\Big(ndm^{9/10}\Big), \end{aligned}$$

where the second step is by $\sum_{r\in[m]} \widetilde{k}_{r,t} = \sum_{i\in[n]} k_{i,t}$, which completes the proof of the corollary. □

14.15 More Efficient Data Structures

The classical training algorithms introduced in this chapter employ the half-space reporting data structures [AEM92], which have very slow pre-processing and are very complicated to implement in practice. In this section, we provide two alternatives that are much simpler with less pre-processing time. In addition, we also prove a fine-grained hardness result showing that our algorithms are nearly optimal.

We first identify the following dynamic algorithms problem, which we prove appears as a key subroutine of the training process.

Definition 14.10 (Dynamic Detection of Firing Neurons (DDFN))**.** Given two set of points $X = \{x_1, \ldots, x_n\} \subset \mathbb{Z}^d$, $Y = \{y_1, \ldots, y_m\} \subset \mathbb{Z}^d$ and a threshold $b \in \mathbb{R}$, design a data structure to support the following operations:

- UPDATE($j \in [m], z \in \mathbb{Z}^d$), set y_j to z
- QUERY(), either output the set

$$Q = \{(i, j) \in [n] \times [m] \mid \langle x_i, y_j \rangle \geq b\},$$

  or report that $|Q| > m^{4/5}n$.

The main technical result of this section is a data structure for solving DDFN with $O(mnd)$-time for preprocessing, $\widetilde{O}(nd)$-time per update, and $O(\min\{|Q|, m^{4/5}n\})$-time per query.

In Section 14.15.1, we introduce our data structures. In Section 14.15.2, we show how these data structures help neural network training. Finally, in Section 14.15.3, we prove our fine-grained lower bound for DDFN.

14.15.1 Correlation tree data structure

In this section, we consider a neural network $2\mathrm{NN}(m, b)$ (Definition 14.1) with n data points. We let $\{w_1, \cdots, w_m\} \subset \mathbb{R}^d$ be the weights, $\{x_1, \cdots, x_n\} \subset \mathbb{R}^d$ be the data points, and $\{(w_r, x_i)\}_{r\in[m],i\in[n]} \subset \mathbb{R}^{m+n}$ be the weight-data pairs.

We propose two data structures: Correlation DTree and Correlation WTree. For the DTree data structure, it contains n binary trees indexed by n data points and supports the following operations:

- **Initialize** It takes data points $\{x_1, \cdots, x_n\} \subset \mathbb{R}^d$ and weights $\{w_1, \cdots, w_m\} \subset \mathbb{R}^d$ as input and compute inner products of all weight-data pairs (w_r, x_i). It uses these inner products to create n different trees. For the i-th tree based on data point x_i, it is constructed from m leaf nodes $\langle w_r, x_i\rangle, r \in [m]$ and satisfies the property that the value of parent node is the maximum value of its child nodes.

- **Update** It takes a new weight $z \in \mathbb{R}^d$ and an index $r \in [m]$ as input. For the i-th tree, it calculate the new inner product $\langle z, x_i\rangle$ and stores the value into the r-th leaf node. Then it compares the new value with its parent node. It replaces parent node with new value if it is larger and continue this comparing process. Otherwise it stops. Repeat the same operation for all n trees.

- **Query** It takes a threshold $b \in \mathbb{R}_{\geq 0}$ and an index $i \in [n]$ as input. Starting from the root of the i-th tree, it checks if its value is greater than threshold b.

If no, search ends. If yes, it treats the child nodes as the root of a new subtree and repeat this searching process until stop. Then it finds all indices $r \in [m]$ that satisfy $\{w_r : sgn(\langle w_r, x_i \rangle - b) \geq 0\}$.

Algorithm 95 Correlation DTree Data Structure

```
1: data structure CORRELATIONDTREE
2:     procedures:
3:         INIT($S \subset \mathbb{R}^d, W \subset \mathbb{R}^d, n, m, d$) ▷ Initialize the data structure via building $n$ trees
4:         QUERY($i, b$)    ▷ $i \in [n], b \in \mathbb{R}$. Output the set $\{r \in [m] : sgn(\langle w_r, x_i \rangle - b) \geq 0\}$
5:         UPDATE($x, i$)                     ▷ Update the $i$'th point in $\mathbb{R}^d$ with $x$
6: end data structure
```

For the WTree data structure, it contains m binary trees indexed by m weights and supports the following operations:

- **Initialize** Similar to DTree, it takes data points $\{x_1, \cdots, x_n\} \subset \mathbb{R}^d$ and weights $\{w_1, \cdots, w_m\} \subset \mathbb{R}^d$ as input and compute inner products of all weight-data pairs (w_r, x_i). It uses these inner products to create nm different trees. For the r-th tree based on weight w_i, it is constructed from n leaf nodes $\langle w_r, x_i \rangle, i \in [n]$ and satisfies the property that the value of parent node is the maximum value of its child nodes.

- **Update** It takes a new weight $z \in \mathbb{R}^d$ and an index $r \in [m]$ as input. Then it re-constructs the r-th tree with weight z.

- **Query** It takes a threshold $b \in \mathbb{R}_{\geq 0}$ and an index $r \in [m]$ as input. Starting from the root of the r-th tree, it checks if its value is greater than threshold b. If no, search ends. If yes, it treats the child nodes as the root of a new subtree and repeat this searching process until stop. Then it finds all indices $i \in [n]$ that satisfy $\{w_r : sgn(\langle w_r, x_i \rangle - b) \geq 0\}$.

Below, we provide the details of the DTree and WTree data structures.

Algorithm 96 Correlation WTree Data Structure

```
1: data structure CorrelationWTree
2:     procedures:
3:         Init($S \subset \mathbb{R}^d, W \subset \mathbb{R}^d, n, m, d$)    ▷ Initialize the data structure via building $m$ trees
4:         Query($r, b$)    ▷ $r \in [m], b \in \mathbb{R}$. Output the set $\{i \in [n] : sgn(\langle w_r, x_i \rangle - b) \geq 0\}$
5:         Update($w, r$)    ▷ Update the $r$'th point in $\mathbb{R}^d$ with $w$
6: end data structure
```

14.15.1.1 Correlation DTree data structure

We start by stating the main theorem of correlation DTree data structure. The pseudocodes are presented in Algorithms 97 and 98.

Theorem 14.37 (Correlation DTree data structure). *There exists a data structure with the following procedures:*

- Init($\{w_1, w_2, \cdots, w_m\} \subset \mathbb{R}^d, \{x_1, x_2, \cdots, x_n\} \subset \mathbb{R}^d, n \in \mathbb{N}, m \in \mathbb{N}, d \in \mathbb{N}$). *Given a series of weights $w_1, w_2, \cdots, w_m$ and datas $x_1, x_2, \cdots, x_n$ in d-dimensional space, it preprocesses in time $O(nmd)$*
- Update($z \in \mathbb{R}^d, r \in [m]$). *Given a weight z and index r, it updates weight w_r with z in time $O(n \cdot (d + \log m))$*
- Query($i \in [n], \tau \in \mathbb{R}$). *Given an index i indicating data point x_i and a threshold τ, it finds all index $r \in [m]$ such that $\langle w_r, x_i \rangle > \tau$ in time $O(|\widetilde{S}(\tau)| \cdot \log m)$, where $\widetilde{S}(\tau) := \{r : \langle w_r, x_i \rangle > \tau\}$*

Algorithm 97 Correlation DTree data structure

```
data structure CorrelationDTree                          ▷ Theorem 14.37
members
    $W \in \mathbb{R}^{m \times d}$ ($m$ weight vectors )
    $X \in \mathbb{R}^{n \times d}$ ($n$ data points)
    Binary tree $T_1, T_2, \cdots, T_n$                  ▷ $n$ binary search trees
end members
public:
procedure Init($w_1, w_2, \cdots, w_m \in \mathbb{R}^d, m, x_1, x_2, \cdots, x_n \in \mathbb{R}^d$, $n$, $m$, $d$)    ▷ Lemma 14.38
    for $i = 1 \to n$ do
        $x_i \leftarrow x_i$
    end for
    for $j = 1 \to m$ do
        $w_j \leftarrow w_j$
    end for
    for $i = 1 \to n$ do                                 ▷ for data point, we create a tree
        for $j = 1 \to m$ do
            $u_j \leftarrow \langle x_i, w_j \rangle$
        end for
        $T_i \leftarrow$ MakeMaxTree($u_1, \cdots, u_m$)   ▷ Each node stores the maximum value for his two children, Algorithm 99
    end for
end procedure
procedure Update($z \in \mathbb{R}^d, r \in [m]$)         ▷ Lemma 14.39
    $w_r \leftarrow z$
    for $i = 1 \to n$ do
        $l \leftarrow$ the $l$-th leaf of tree $T_i$
        $l$.value $= \langle z, x_i \rangle$
        while $l$ is not root do
            $p \leftarrow$ parent of $l$
            $p$.value $\leftarrow \max\{p$.value$, l$.value$\}$
            $l \leftarrow p$
        end while
    end for
end procedure
end data structure
```

Algorithm 98 Correlation DTrees

```
data structure CorrelationDTree                                ▷ Theorem 14.37
public:
procedure Query($i \in [n], \tau \in \mathbb{R}_{\geq 0}$)      ▷ Lemma 14.40
    return Find($\tau$, root($T_i$))
end procedure

private:
procedure Find($\tau \in \mathbb{R}_{\geq 0}, r \in T$)
    if $r$ is leaf then
        return $r$
    else
        $r_1 \leftarrow$ left child of $r$, $r_2 \leftarrow$ right child of $r$
        if $r_1$.value $\geq \tau$ then
            $S_1 \leftarrow$Find($\tau, r_1$)
        end if
        if $r_2$.value $\geq \tau$ then
            $S_2 \leftarrow$Find($\tau, r_2$)
        end if
    end if
    return $S_1 \cup S_2$
end procedure
end data structure
```

Running time for CorrelationDTree Then, we prove the running time of Init, Update, and Query. We start by showing the running time of Init.

Lemma 14.38 (Running time of Init)**.** *Given a series of weights* $\{w_1, w_2, \cdots, w_m\} \subset \mathbb{R}^d$ *and datas* $\{x_1, x_2, \cdots, x_n\} \subset \mathbb{R}^d$*, it preprocesses in time* $O(nmd)$*.*

Proof. The Init consists of two independent forloop and two recursive forloops. The first forloop (start from line 9) has n interations, which takes $O(n)$ time. The second forloop (start from line 12) has m iterations, which takes $O(m)$ time. Now we consider the recursive forloop. The outer loop (line 15) has n iterations. In inner loop has m iterations. In each iteration of the inner loop, line 17 takes $O(d)$ time. Line 19 takes

Algorithm 99 Constructing a tree satisfying the property that the value of the parent node is the max value of its child node.

```
procedure MakeMaxTreeInner($r_1, \cdots, r_n$)
    if $n = 1$ then
        return $r_1$
    else
        for $i \in [n/2]$ do
            Create node $r'_i$
            if $r_{2i-1}$.value $>$ $r_{2i}$.value then
                $r'_i \leftarrow r_{2i-1}$
            else
                $r'_i \leftarrow r_{2i}$
            end if
            Insert $r_{2i-1}$ as left child
            Insert $r_{2i}$ as right child
        end for
                                        ▷ If $n$ is odd, create a parent node for the last node.
        return MakeMaxTreeInner($\{r'_1, \cdots, r'_i\}$)
    end if
end procedure
procedure MakeMaxTree($u_1, \cdots, u_n$)
    for $i \in [n]$ do
        Create nodes $r_i$
        $r_i$.value $\leftarrow u_i$
    end for
    return MakeMaxTreeInner($r_1, \cdots, r_n$)
end procedure
```

$O(m)$ time. Putting it all together, the running time of Init is

$$O(n + m + n(md + m)) = O(nmd).$$

Thus, we complete the proof. □

Next, we analyze the running time of Update.

Lemma 14.39 (Running time of Update)**.** *Given a weight $z \in \mathbb{R}^d$ and index $j \in [m]$, it updates weight w_j with z in time $O(n \cdot (d + \log m))$.*

Proof. The running time of UPDATE mainly comes from the forloop (line 24), which consists of n iterations. In each iteration, line 25 takes $O(\log m)$ time, line 26 takes $O(d)$ time and the while loop takes $O(\log m)$ time since it go through a path bottom up. Putting it together, the running time of UPDATE is $O(n(d + \log m))$. $\square$

Finally, we state the running time for QUERY procedure.

Lemma 14.40 (Running time of QUERY)**.** *Given a query $q \in \mathbb{R}^d$ and a threshold $\tau > 0$, it finds all index $i \in [n]$ such that $\langle w_i, q \rangle > \tau$ in time $O(|S(\tau)| \cdot \log m)$, where $S(\tau) := \{i : \langle w_i, q \rangle > \tau\}$.*

Proof. The running time comes from FIND with input τ and root(T_i). In FIND, we start from the root node r and find indices in a recursive way. The INIT guarantees that for a node r satisfying r.value $> \tau$, the sub-tree with root r must contains a leaf whose value is greater than τ If not satisfied, all the values of the nodes in the sub-tree with root r is less thanτ. This guarantees that all the paths it search does not have any branches that leads to the leaf we don't want and it will report all the indices i satisfying $\langle w_i, q \rangle > 0$. Note that the depth of T is $O(\log n)$, the running time of QUERY is $O(|S(\tau)| \cdot \log n)$ $\square$

14.15.1.2 Correlation WTree data structure

In this section, we state the main theorem of correlation WTree data structure. The pseudocodes are presented in Algorithms 100 and 101.

Theorem 14.41 (Correlation WTree data structure)**.** *There exists a data structure with the following procedures:*

- INIT$(\{w_1, w_2, \cdots, w_m\} \subset \mathbb{R}^d, \{x_1, x_2, \cdots, x_n\} \subset \mathbb{R}^d, n \in \mathbb{N}, m \in \mathbb{N}, d \in \mathbb{N})$. *Given a series of weights $w_1, w_2, \cdots, w_m$ and datas $x_1, x_2, \cdots, x_n$ in d-dimensional space, it preprocesses in time $O(nmd)$*

- UPDATE($z \in \mathbb{R}^d, r \in [m]$). *Given a weight z and index r, it updates weight w_r with z in time $O(nd)$*
- QUERY($r \in [m], \tau \in \mathbb{R}$). *Given an index r indicating weight w_r and a threshold τ, it finds all index $i \in [n]$ such that $\langle w_r, x_i \rangle > \tau$ in time $O(|S(\tau)| \cdot \log m)$, where $S(\tau) := \{i : \langle w_r, x_i \rangle > \tau\}$*

Running time for Correlation WTree Then, we prove the running time of INIT, UPDATE and QUERY. As in DTree, we first show the running time for INIT.

Lemma 14.42 (Running time of INIT). *Given a series of weights $\{w_1, w_2, \cdots, w_m\} \subset \mathbb{R}^d$ and datas $\{x_1, x_2, \cdots, x_n\} \subset \mathbb{R}^d$, it preprocesses in time $O(nmd)$*

Proof. The INIT consists of two independent forloop and two recursive forloops. The first forloop (start from line 10) has n interations, which takes $O(n)$ time. The second forloop (start from line 13) has m iterations, which takes $O(m)$ time. Now we consider the recursive forloop. The outer loop (line 16) has m iterations. In inner loop has n iterations. In each iteration of the inner loop, line 18 takes $O(d)$ time. Line 20 takes $O(n)$ time. Putting it all together, the running time of INIT is

$$O(n + m + m(nd + n)) = O(nmd).$$

Thus, we complete the proof. □

Next, we turn to the running time for UPDATE.

Lemma 14.43 (Running time of UPDATE). *Given a weight $z \in \mathbb{R}^d$ and index $r \in [m]$, it updates weight w_j with z in time $O(nd)$.*

Proof. In this procedure, it generates a new tree for weight w_r with n leaves, which takes $O(nd)$ time. Thus, we complete the proof. □

Finally, we present the running time of QUERY.

Algorithm 100 Correlation WTree data structure

```
1: data structure CorrelationWTree                                  ▷ Theorem 14.41
2: members
3:     $W \in \mathbb{R}^{m \times d}$ ($m$ weight vectors )
4:     $X \in \mathbb{R}^{n \times d}$ ($n$ data points)
5:     Binary tree $T_1, T_2, \cdots, T_M$                           ▷ $m$ binary search trees
6: end members
7:
8: public:
9: procedure Init($w_1, w_2, \cdots, w_m \in \mathbb{R}^d, m, x_1, x_2, \cdots, x_n \in \mathbb{R}^d$, $n$, $m$, $d$)   ▷ Lemma 14.42
10:     for $i = 1 \to n$ do
11:         $x_i \leftarrow x_i$
12:     end for
13:     for $j = 1 \to m$ do
14:         $w_j \leftarrow w_j$
15:     end for
16:     for $i = 1 \to m$ do                                    ▷ for weight, we create a tree
17:         for $j = 1 \to n$ do
18:             $u_j \leftarrow \langle x_i, w_j \rangle$
19:         end for
20:         $T_i \leftarrow$ MakeTree($u_1, \cdots, u_n$) ▷ Each node stores the maximum value for his two children
21:     end for
22: end procedure
23: procedure Update($z \in \mathbb{R}^d, r \in [m]$)                          ▷ Lemma 14.43
24:     $w_r \leftarrow z$
25:     for $j = 1 \to n$ do
26:         $u_j \leftarrow \langle x_j, w_r \rangle$
27:         $T_i \leftarrow$ MakeTree($u_1, \cdots, u_n$) ▷ Each node stores the maximum value for his two children
28:     end for
29: end procedure
30: end data structure
```

Lemma 14.44 (Running time of Query). *Given a query $q \in \mathbb{R}^d$ and a threshold $\tau > 0$, it finds all index $i \in [n]$ such that $\langle w_i, q \rangle > \tau$ in time $O(|S(\tau)| \cdot \log m)$, where $S(\tau) := \{i : \langle w_i, q \rangle > \tau\}$.*

Algorithm 101 Correlation WTree

```
data structure CorrelationWTree
public:
procedure Query($r \in [m], \tau \in \mathbb{R}_{\geq 0}$)                ▷ Lemma 14.44
    return Find($\tau$, root($T_r$))
end procedure

private:
procedure Find($\tau \in \mathbb{R}_{\geq 0}, r \in T$)
    if $r$ is leaf then
        return $r$
    else
        $r_1 \leftarrow$ left child of $r$, $r_2 \leftarrow$ right child of $r$
        if $r_1$.value $\geq \tau$ then
            $S_1 \leftarrow$Find($\tau, r_1$)
        end if
        if $r_2$.value $\geq \tau$ then
            $S_2 \leftarrow$Find($\tau, r_2$)
        end if
    end if
    return $S_1 \cup S_2$
end procedure
end data structure
```

Proof. The running time comes from Find with input τ and root(T_i). In Find, we start from the root node r and find indices in a recursive way. The Init guarantees that for a node r satisfying r.value $> \tau$, the sub-tree with root r must contain a leaf whose value is greater than τ If not satisfied, all the values of the nodes in the sub-tree with root r is less thanτ. This guarantees that all the paths it search does not have any branches that lead to the leaf we don't want and it will report all the index i satisfying $\langle w_i, q \rangle > 0$. Note that the depth of T is $O(\log n)$, the running time of Query is $O(|S(\tau)| \cdot \log n)$ □

14.15.2 Training algorithms with correlation tree data structures

In this section, we show two neural network training algorithms using DTree and WTree data structures.

14.15.2.1 Weights Preprocessing

We first show our training algorithm using DTree, which preprocesses weights for each data point. The pseudocode is in Algorithm 102.

Theorem 14.45. *Given n data points in $\mathbb{R}^d$. Running gradient descent algorithm (Algorithm 102) on $2\mathrm{NN}(m, b = \sqrt{0.4 \log m})$ (Definition 14.1) the expected cost per iteration of the gradient descent algorithm is*

$$O(m^{4/5} n^2 d)$$

Proof. The per-step time complexity is

$$\begin{aligned} \mathcal{T} &= \mathcal{T}_1 + \mathcal{T}_2 + \mathcal{T}_3 \\ &= \sum_{i=1}^{n} \mathcal{T}_{\textsc{Query}}(m, d, k_{i,t}) + \mathcal{T}_{\textsc{Update}} \cdot |\cup_{i \in [n]} S_{i,\mathrm{fire}}(t)| + d \sum_{i \in [n]} k_{i,t} \end{aligned}$$

The first term $\mathcal{T}_1 = \sum_{i=1}^{n} \mathcal{T}_{\textsc{Query}}(m, d, k_{i,t})$ corresponds to the running time of querying the active neuron set $S_{i,\mathrm{fire}}(t)$ for all training samples $i \in [n]$. With the first result in Theorem 14.37, the complexity is bounded by $O(m^{4/5} n \log m)$.

The second term $\mathcal{T}_2 = \mathcal{T}_{\textsc{Update}} \cdot |\cup_{i \in [n]} S_{i,\mathrm{fire}}(t)|$ corresponds to updating w_r in the high-dimensional search data-structure (Line 28). Again with the first result in Theorem 14.37, we have $\mathcal{T}_{\textsc{Update}} = O(n(d + \log m))$. Combining with the fact that $|\cup_{i \in [n]} S_{i,\mathrm{fire}}(t)| \le |\cup_{i \in [n]} S_{i,\mathrm{fire}}(0)| \le O(m^{4/5} n)$, the second term is bounded by $O(m^{4/5} n^2 d)$.

The third term is the time complexity of gradient calculation restricted to the set $S_{i,\mathrm{fire}}(t)$. With the bound on $\sum_{i \in [n]} k_{i,t}$ (Lemma 14.18), we have $d \sum_{i \in [n]} k_{i,t} \le O(m^{4/5} nd)$

Algorithm 102 Training Neural Network via building n trees, where each tree is the correlation between one data point and all the weights

```
1: procedure TrainingWithPreprocessWeights($\{x_i\}_{i\in[n]}, \{y_i\}_{i\in[n]}, n, m, d$)    ▷ Theorem 14.45
2:     /*Initialization step*/
3:     Sample $W(0)$ and $a$ according to Definition
4:     $b \leftarrow \sqrt{0.4\log m}$.
5:     /*A dynamic data-structure*/
6:     CorrelationDTree cdt    ▷ Theorem 14.37
7:     cdt.Init($\{x_i\}_{i\in[n]}, \{w_r(0)\}_{r\in[m]}, n, m, d$)    ▷ It takes $\mathcal{T}_{\mathsf{init}}(n,m,d)$ time. Alg. 97
8:     /*Iterative step*/
9:     for $t = 0 \to T$ do
10:        /*Forward computation step*/
11:        for $i = 1 \to n$ do
12:            $S_{i,\mathrm{fire}} \leftarrow$ cdt.Query($i, b$)    ▷ It takes $\mathcal{T}_{\mathsf{query}}(m, k_{i,t})$ time. Alg. 98
13:            $u(t)_i \leftarrow \frac{1}{\sqrt{m}} \sum_{r\in\mathcal{S}_{i,\mathrm{fire}}} a_r \cdot \sigma_b(w_r(t)^\top x_i)$    ▷ It takes $O(d \cdot k_{i,t})$ time
14:        end for
15:        /*Backward computation step*/
16:        $P \leftarrow 0^{n\times m}$    ▷ $P \in \mathbb{R}^{n\times m}$
17:        for $i = 1 \to n$ do
18:            for $r \in \mathcal{S}_{i,\mathrm{fire}}$ do
19:                $P_{i,r} \leftarrow \frac{1}{\sqrt{m}} a_r \cdot \sigma_b'(w_r(t)^\top x_i)$
20:            end for
21:        end for
22:        $M \leftarrow X\mathrm{diag}(y - u(t))$    ▷ $M \in \mathbb{R}^{d\times n}$, it takes $O(n \cdot d)$ time
23:        $\Delta W \leftarrow \underbrace{M}_{d\times n} \underbrace{P}_{n\times m}$    ▷ $\Delta W \in \mathbb{R}^{d\times m}$, it takes $O(d \cdot nnz(P))$ time, $nnz(P) = O(nm^{4/5})$
24:        $W(t+1) \leftarrow W(t) - \eta \cdot \Delta W$.
25:        /*Update data structure*/
26:        Let $Q \subset [m]$ where for each $r \in Q$, the $\Delta W_{*,r}$ is not all zeros    ▷ $|Q| \leq O(nm^{4/5})$
27:        for $r \in Q$ do
28:            cdt.Update($w_r(t+1), r$)    ▷ Alg. 97
29:        end for
30:    end for
31:    return $W$    ▷ $W \in \mathbb{R}^{d\times m}$
32: end procedure
```

Putting them together, we have

$$\begin{aligned}\mathcal{T} &\leq O(m^{4/5} n \log m) + O(m^{4/5} n^2 d) + O(m^{4/5} nd) \\ &= O(m^{4/5} n^2 d)\end{aligned}$$

Thus, we complete the proof. □

14.15.2.2 Data Preprocessing

Now, we describe a similar version of the aforementioned training algorithm, but it uses WTree to preprocess data points based on weights. The pseudocode is in Algorithms 103 and 104.

Algorithm 103 Training Neural Network via building m trees, where each tree is the correlation between one weight and all the data points

```
1: procedure TrainingWithProcessData($\{x_i\}_{i\in[n]}, \{y_i\}_{i\in[n]}, n, m, d$)   ▷ Theorem 14.46
2:     /*Initialization step*/
3:     Sample $W(0)$ and $a$ according to Definition
4:     $b \leftarrow \sqrt{0.4\log m}$.
5:     /*A static data-structure*/
6:     CorrelationWTree cwt                        ▷ Algorithm 97, Part 2 of Theorem 14.37
7:     cwt.Init($\{x_i\}_{i\in[n]}, \{w_r(0)\}_{r\in[m]}, n, m, d$)        ▷ It takes $\mathcal{T}_{\mathsf{init}}(n,m,d)$ time
8:     /*Initialize $\widetilde{S}_{r,\mathrm{fire}}$ and $S_{i,\mathrm{fire}}$ */
9:                             ▷ It takes $\sum_{r=1}^m \mathcal{T}_{\mathsf{query}}(n, \widetilde{k}_{r,t}) = O(m^{4/5}n\cdot\log n)$ time
10:    $\widetilde{S}_{r,\mathrm{fire}} \leftarrow \emptyset$ for $r \in [m]$.     ▷ $\widetilde{S}_{r,\mathrm{fire}}$ is the set of samples, for which neuron $r$ fires
11:    $S_{i,\mathrm{fire}} \leftarrow \emptyset$ for $i \in [n]$.            ▷ $S_{i,\mathrm{fire}}$ is the set of neurons, which fire for $x_i$
12:    for $r = 1 \to m$ do
13:        $\widetilde{S}_{r,\mathrm{fire}} \leftarrow$ cwt.Query($r, b$)
14:        for $i \in \widetilde{S}_{r,\mathrm{fire}}$ do
15:            $S_{i,\mathrm{fire}}$.Add($r$)
16:        end for
17:    end for
18: end procedure
```

Theorem 14.46. *Given n data points in $\mathbb{R}^d$. Running gradient descent algorithm (Algorithm 103) on $2\mathrm{NN}(m, b = \sqrt{0.4\log m})$, the expected per-iteration running time of initializing $\widetilde{S}_{r,\mathrm{fire}}, S_{i,\mathrm{fire}}$ for $r \in [m], i \in [n]$ is $O(m^{4/5}n\cdot\log n)$. The cost per-iteration of the training algorithm is $O(m^{4/5}n^2 d)$.*

Proof. We analyze the initialization and training parts separately.

Initialization From Line 10 to Line 17, the sets $\widetilde{S}_{r,\mathrm{fire}}, S_{i,\mathrm{fire}}$ for $r \in [m], i \in [n]$ are initialized. For each $r \in [m]$, we need to query the data structure the set of data points x's such that $\sigma_b(w_r(0)^\top x) > 0$. Hence the running time of this step is

$$\begin{aligned}\sum_{r=1}^{m} \mathcal{T}_{\textsc{Query}}(n, \widetilde{k}_{r,0}) &= O(\sum_{r=1}^{m} \widetilde{k}_{r,0} \cdot \log n) \\ &= O(\sum_{i=1}^{n} k_{i,0} \cdot \log n) \\ &= O(m^{4/5} n \cdot \log n)\end{aligned}$$

where the second step follows from $\sum_{r=1}^{m} \widetilde{k}_{r,0} = \sum_{i=1}^{n} k_{i,0}$.

Training Consider training the neural networkfor T steps. For each step, first notice that the forward and backward computation parts (Line 6 - Line 18) are the same as previous algorithm. The time complexity is $O(m^{4/5}n)$.

We next show that maintaining $\widetilde{S}_{r,\mathrm{fire}}, r \in [m]$ and $S_{i,\mathrm{fire}}, i \in [n]$ (Line 21 - Line 30) takes $O(m^{4/5}nd)$ time. For each fired neuron $r \in [m]$, we first remove the indices of data in the sets $S_{i,\mathrm{fire}}$, which takes time

$$O(1) \cdot \sum_{r \in \cup_{i \in [n]} S_{i,\mathrm{fire}}} \widetilde{k}_{r,t} = O(1) \cdot \sum_{r=1}^{m} \widetilde{k}_{r,t} = O(m^{4/5}n)$$

Then, we find the new set of x's such that $\sigma_b(\langle w_r(t+1), x \rangle) > 0$ by querying the correlation tree data structure. The total ruunning time for all fired neurons is

$$\begin{aligned}\sum_{r \in \cup_{i \in [n]} S_{i,\mathrm{fire}}} \mathcal{T}_{\textsc{Update}}(n, d) + \mathcal{T}_{\textsc{Query}}(n, \widetilde{k}_{r,t+1}) &\lesssim m^{4/5} n^2 (d + \log m) + \sum_{r \in \cup_{i \in [n]} S_{i,\mathrm{fire}}} \widetilde{k}_{r,t+1} \cdot \log n \\ &= O(m^{4/5} n^2 d)\end{aligned}$$

Then, we update the index sets $S_{i,\mathrm{fire}}$ in time $O(m^{4/5}n)$. Therefore, each training step takes $O(m^{4/5}n^2 d)$ time, which completes the proof. $\square$

14.15.3 Lower bound for Dynamic Detection of Firing Neurons

The goal of this section is to prove the lower bound for Dynamic Detection of Firing Neurons.

We refer to a theorem about the maximum bichromatic inner product lower bound in [Che18].

Theorem 14.47 (Maximum bichromatic inner product lower bound, [Che18])**.** *Assuming SETH, there is a constant c such that any exact algorithm for $\mathbb{Z}$-$\mathsf{Max\text{-}IP}_{n,d}$ in dimension $d = c^{\log^* n}$ requires $n^{2-o(1)}$ time, with vectors of $O(\log n)$-bit entries.*

Putting things together, we state the main result for the lower bound for Dynamic Detection of Firing Neurons.

Theorem 14.48 (Lower Bound for Dynamic Detection of Firing Neurons)**.** *Let $d = 2^{O(\log^* n)}$. Unless SETH fails, for any constants $c \in (0,1)$, no data structure can solve DDFN with less than $m^{1-c}n^{c-o(1)}$-time per update and $m^{1-c}n^{1+c-o(1)}$-time per query.*

Proof. Without loss of generality, we assume that $m > n$. Let $d = c^{\log^* m}$, where c is defined in Theorem 14.47.

Suppose there exists a data structure that for $(m, n, d+1)$-sized instance, can perform updates in $m^{1-c}n^{c-\epsilon}$-time and answer queries in $m^{1-c}n^{1+c-\epsilon}$-time, for some $c \in (0,1)$ and $\epsilon \in (0,c)$.

Let $X = \{x_1, \dots, x_m\} \subset \mathbb{Z}^d$, $Y = \{y_1, \dots, y_m\} \subset \mathbb{Z}^d$ be a hard instance of $\mathbb{Z}$-$\mathsf{Max\text{-}IP}_{m,d}$ problem constructed in Theorem 14.47. For each vector x_i (or y_j), we construct a new vector $\widetilde{x}_i$ (or $\widetilde{y}_j$) in $(d+1)$-dimension such that $(\widetilde{x}_i)_{d+1} = -1$ and $(\widetilde{y}_j)_{d+1} = w$, where w is a parameter to be chosen later.

Then, we construct $k = \lceil m/n \rceil$ instances of the DDFN problem in Definition 14.10 as follows:

$$\widetilde{X}^{(i)} := \{\widetilde{x}_1, \dots, \widetilde{x}_n\}, \quad \widetilde{Y}^{(i)} := \{\widetilde{y}_1, \dots, \widetilde{y}_m\},$$

and $b = 0$.

We show that the data structures for these instances $\{(\widetilde{X}^{(i)}, \widetilde{Y}^{(i)}, b)\}_{i \in [k]}$ can be used to solve $\mathbb{Z}$-Max-$\mathsf{IP}_{n,d}(X, Y)$.

We perform a binary search for the value of $\mathbb{Z}$-Max-$\mathsf{IP}_{n,d}(X, Y)$. Note that at most $O(\log n)$ iterations suffice to find the exact answer.

Suppose the current value in the binary search is $t \in \mathbb{Z}$. Consider the i-th instance $(\widetilde{X}^{(i)}, \widetilde{Y}^{(i)}, b)$ for any $i \in [k]$. We first call $\textsc{Update}()$ to set $(\widetilde{y}_j)_{d+1} = t$ for each $j \in [m]$. By the data structure's guarantee, this step takes $O(m \cdot m^{1-c} n^{c-\epsilon}) = O(m^{2-c} n^{c-\epsilon})$ time. Then, we call $\textsc{Query}()$. Notice that

$$\langle \widetilde{x}_i, \widetilde{y}_j \rangle = \langle x_i, y_j \rangle - t \geq 0 \iff \langle x_i, y_j \rangle \geq t.$$

Hence, $\textsc{Query}()$ will return all pairs of (i, j) such that $\langle x_i, y_j \rangle \geq t$. This step runs in $O(m^{1-c} n^{1+c-\epsilon})$-time. We repeat this process for all k instances. And based on whether the outputs of all the $\textsc{Query}()$ are empty or not, we know the direction of the binary search for the next iteration.

Hence, the total running time of each iteration is

$$\begin{aligned} & O(k \cdot (m^{2-c} n^{c-\epsilon} + m^{1-c} n^{1+c-\epsilon})) \\ = {} & O(m^{3-c} n^{c-\epsilon-1} + m^{2-c} n^{c-\epsilon}) \\ \leq {} & O(m^{2-\epsilon}), \end{aligned}$$

which follows from the assumption of $m \geq n$. Thus, we can solve $\mathbb{Z}$-Max-$\mathsf{IP}_{n,d}(X, Y)$ in $\widetilde{O}(m^{2-\epsilon}) < m^{2-o(1)}$-time, which contradicts to the lower bound for $\mathbb{Z}$-Max-$\mathsf{IP}_{n,d}$ in Theorem 14.47.

Therefore, no such data structure can exist. $\square$

Algorithm 104 Training with process data (continue)

```
1: procedure TRAININGWITHPROCESSDATA($\{x_i\}_{i\in[n]}$, $\{y_i\}_{i\in[n]}$,$n$,$m$,$d$)
2:     $\cdots$                                                        ▷ Algorithm 103
3:     /*Iterative step*/
4:     for $t = 1 \to T$ do
5:         /*Forward computation step*/
6:         for $i = 1 \to n$ do
7:             $u(t)_i \leftarrow \frac{1}{\sqrt{m}} \sum_{r\in\mathcal{S}_{i,\mathrm{fire}}} a_r \cdot \sigma_b(w_r(t)^\top x_i)$     ▷ It takes $O(d \cdot k_{i,t})$ time
8:         end for
9:         /*Backward computation step*/
10:        $P \leftarrow 0^{n\times m}$                                    ▷ $P \in \mathbb{R}^{n\times m}$
11:        for $i = 1 \to n$ do
12:            for $r \in \mathcal{S}_{i,\mathrm{fire}}$ do
13:                $P_{i,r} \leftarrow \frac{1}{\sqrt{m}} a_r \cdot \sigma_b'(w_r(t)^\top x_i)$
14:            end for
15:        end for
16:        $M \leftarrow X \mathrm{diag}(y - u(t))$                ▷ $M \in \mathbb{R}^{d\times n}$, it takes $O(n \cdot d)$ time
17:        $\Delta W \leftarrow \underbrace{M}_{d\times n} \underbrace{P}_{n\times m}$        ▷ $\Delta W \in \mathbb{R}^{d\times m}$, it takes $O(d \cdot nnz(P))$ time,
   $nnz(P) = O(nm^{4/5})$
18:        $W(t+1) \leftarrow W(t) - \eta \cdot \Delta W$.
19:        /*Update $\widetilde{S}_{r,\mathrm{fire}}$ and $S_{i,\mathrm{fire}}$ step*/
20:        ▷ It takes $O(\sum_{i=1}^n k_{i,t} + \sum_{r\in S_{[n],\mathrm{fire}}} \mathcal{T}_{\mathsf{query}}(n, d, \widetilde{k}_{r,t+1})) = O(n \cdot \log n \cdot m^{4/5})$
21:        $S_{[n],\mathrm{fire}} \leftarrow \cup_{i\in[n]} \mathcal{S}_{i,\mathrm{fire}}$
22:        for $r \in S_{[n],\mathrm{fire}}$ do
23:            for $i \in \widetilde{S}_{r,\mathrm{fire}}$ do  ▷ Removing old fired neuron indices. It takes $O(\widetilde{k}_{r,t})$
   time
24:                $S_{i,\mathrm{fire}}$.DEL($r$)
25:            end for
26:            CWT.UPDATE($w_r(t+1), r$)                         ▷ It takes $\mathcal{T}_{\mathsf{update}}(n, d)$ time
27:            $\widetilde{S}_{r,\mathrm{fire}} \leftarrow$ CWT.QUERY($r, b$)              ▷ It takes $\mathcal{T}_{\mathsf{query}}(n, d, \widetilde{k}_{r,t+1})$ time
28:            for $i \in \widetilde{S}_{r,\mathrm{fire}}$ do  ▷ Adding new fired neuron indices. It takes $O(\widetilde{k}_{r,t+1})$
   time
29:                $S_{i,\mathrm{fire}}$.ADD($r$)
30:            end for
31:        end for
32:    end for
33:    return $W$                                                        ▷ $W \in \mathbb{R}^{d\times m}$
34: end procedure
```

Chapter 15: Training Multi-Layer Over-Parameterized Neural Networks

15.1 Introduction

Convex and non-convex optimizations are popular topics across various communities, such as theoretical computer science, operational research, numerical methods and machine learning. Typically, optimization algorithms comprise of two parts: the number of iterations (or iteration counts) and the cost per iteration. While reducing the iteration count is critical [LS14, LSW15], recently, there are more and more focus on improving the cost per iteration [CLS19, LSZ19, Ye20, BLSS20, JLSW20, JKL$^+$20, BLL$^+$21, DLY21, JSWZ21]. As the popularity of deep neural networks grows in the recent year, developing efficient optimization algorithms to train deep neural networks that can both have a good convergence rate and a fast training time has been a central topic. From a theoretical point of view, it is highly non-trivial to prove the most basic algorithm such as gradient descent will converge in training deep neural networks due to the non-convexity of the architecture. A rich body of works has been devoted to use the power of *over-parametrization* for convergence analysis [LL18, JGH18, DZPS19, AZLS19a, AZLS19b, DLL$^+$19]. The key ingredient of over-parametrization involves using a much wider neural network, where the network width $m = \mathrm{poly}(n, d)$. However, this induces a weight matrix $W \in \mathbb{R}^{m \times m}$ (except for the input layer, which is of size $m \times d$). Training the neural network involves performing matrix-vector products with the $m \times m$ matrix per layer, which would take $\Theta(m^2)$ time assuming no structures on the weight matrix W and the vector h being multiplied, and in standard network training, W and h are both dense. Hence, it is critical to reduce the cost of training per iteration.

We hence ask the following question:

Is it possible to reduce the cost per iteration during the training process to truly *subquadratic in m?*

In the case of training two-layer over-parametrized neural networks, two interesting results have been obtained by Brand, Peng, Song and Weinstein [BPSW21] and Song, Yang and Zhang [SYZ21]. However, we note that the setting both of these papers studied is different from ours, specifically,

- In the two-layer case, they only need to train a weight matrix of size $m \times d$. Since evaluating one data point in d dimensional space for neural network functions takes $O(md)$ time, the cost per iteration bound they want to match [BPSW21] or beat [SYZ21] is $O(md)$.
- In the multiple layer case, instead of only having a weight matrix of size $m \times d$, we will have at least one weight matrix of size $m \times m$. Since evaluating one data point in d dimensional space for neural network functions takes $O(m^2)$ time, the cost per iteration bound we're trying to beat in this chapter is $O(m^2)$.

In [BPSW21], they provide an algorithm that can adaptively choose the step size for different gradient directions associated to different data points, which is one of the goal we want to achieve. However, their method involves forming a Jacobian matrix of the data, which is of size $n \times md$ in the two-layer case, but of size $n \times m^2$ in our case. Hence, their algorithm can only imply an $O(nm^2)$ cost per iteration, which cannot match our subquadratic goal.

In [SYZ21], they beat the nmd barrier for cost per iteration by leveraging two key ideas: use a shifted ReLU activation and via a sparsity analysis, they show that only $o(m)$ number of neurons being fired up[1]. They further use data structure to preprocess both data points and training weights. However, their algorithm only works for small d due to the high (exponential) dependence on d in the proprocessing

[1]Such phenomenon has been observed in practice as in [CMJF$^+$20, CLP$^+$21]

phase. In multiple layer neural network, instead of only having weight matrix with size $m \times d$, we will have at least one weight matrix with size $m \times m$, this directly translates to a high dependence on m in the proprocessing phase. Therefore, the techniques presented in [SYZ21] cannot give $n \cdot o(m^2)$ cost per iteration.

In this chapter, we take the first step to break the m^2 barrier on cost per iteration for training multiple layer over-parametrized neural networks.

15.1.1 Our result

Our main result can be summarized in the following three theorems, with one analyzing the convergence behavior of a general Gram-based optimization framework, one designing an efficient algorithm to realize the subqudratic cost per iteration, and the third is a novel algorithm to solve tensor-based regression in high precision and fast, which is a key step in our meta algorithm.

Throughout this chapter, we will use n to denote the number of training data points, d to denote the dimension of input data points, m to denote the width of the network and L to denote the number of layers of the network. We use λ_L to denote the smallest eigenvalue of the neural tangent kernel induced by our neural network. We use $f_t \in \mathbb{R}^n$ to denote the prediction of neural network at time t.

Our first theorem demonstrates the fast convergence rate of our algorithm.

Theorem 15.1 (Convergence, informal version of Theorem 15.32)**.** *Suppose the width of the neural network satisfies $m \geq \mathrm{poly}(n, L, \lambda_L)$, then there exists an algorithm (Algorithm 105) such that, over the randomness of initialization of the network and the algorithm, with probability at least $1 - e^{-\Omega(\log^2 n)}$, we have*

$$\|f_{t+1} - y\|_2 \leq \frac{1}{3}\|f_t - y\|_2.$$

The above theorem establishes the linear convergence behavior of our method. However, compared to one-hidden layer case, our analysis is much more sophisticated since we have to carefully control the probability so that it does not blow up exponentially with respect to the number of layers.

The next theorem concerns the *cost per iteration* of our algorithm.

Theorem 15.2 (Runtime, informal version of Theorem 15.5)**.** *There exists a randomized algorithm (Algorithm 105) that trains a multi-layer neural network of width m with the cost per training iteration being*

$$\widetilde{O}(nm^{2-\Omega(1)}).$$

We improve the overall training time of multi-layer over-parametrized networks from $\mathcal{T}_{\mathrm{init}} + T \cdot \widetilde{O}(nm^2)$ to $\mathcal{T}_{\mathrm{init}} + T \cdot \widetilde{o}(nm^2)$, where $\mathcal{T}_{\mathrm{init}}$ is the initialization time of training, typically takes $O(nm^2)$. As we have argued before, multi-layer over-parametrized networks require m to be in the order of n^4, hence improving the cost per iteration from quadratic to subquadratic is an important gain in speeding up training. Its advantage is even more evident when one seeks a *high precision solution*, and hence the number of iterations T is large.

We highlight that it is non-trivial to obtain a subquadratic running time per iteration: If not handled properly, computing the matrix-vector product with weight matrices will take $O(m^2)$ time! This means that even for method such as gradient descent, it is not clear how to achieve a subquadratic running time, since one has to multiply the weight matrix with a vector in both forward evaluation and backward computation. In our case, we also have a Jacobian matrix of size $n \times m^2$, so forming it naively will cost $O(nm^2)$ time, which is prohibitively large. Finally, note that the update matrix is also an $m \times m$ matrix. In order to circumvent these problems, we exploit the fact that the gradient is of low rank (rank n), hence one can compute a rank-n factorization and use it to support fast matrix-vector product. We also observe that each row of the Jacobian matrix can be formulated as a tensor product of two vectors, therefore we can make use of fast randomized linear algebra to approximate the tensor product efficiently. As a byproduct, we have the following technical theorem:

Theorem 15.3 (Fast Tensor Regression, informal version of Theorem 15.17)**.** *Given two $n \times m$ matrices U and V with $m \gg n$ and a target vector $c \in \mathbb{R}^n$. Let $J = [\mathrm{vec}(u_1 v_1^\top)^\top, \dots, \mathrm{vec}(u_n v_n^\top)^\top] \in \mathbb{R}^{n \times m^2}$ where u_i is the i-th row of matrix U $\forall i \in [n]$. There is a randomized algorithm that takes $\widetilde{O}(nm + n^2(\log(\kappa/\epsilon) + \log(m/\delta)) + n^\omega)$ time and outputs a vector $\widehat{x} \in \mathbb{R}^n$ such that*

$$\|JJ^\top \widehat{x} - c\|_2 \leq \epsilon \|c\|_2$$

holds with probability at least $1 - \delta$, and κ is the condition number of J.

From a high level, the algorithm proceeds as follows: given matrices U and V, it forms an approximation $\widetilde{J} \in \mathbb{R}^{n \times n \log(m/\delta)}$, where each row is generated by applying fast tensor sketching technique to u_i and v_i ([AKK+20]). Then, it uses another sketching matrix for $\widetilde{J}$ to obtain a good preconditioner R for $\widetilde{J}$. Subsequently, it runs a gradient descent to solve the regression.

To understand this runtime better, we note that nm term is the size of matrices U and V, hence reading the entries from these matrices will take at least $O(nm)$ time. The algorithm then uses tensor-based sketching techniques ([AKK+20]) to squash length m^2 tensors to length $O(n \log(m/\epsilon\delta))$. All subsequent operations are performed on these much smaller vectors. Finally, computing the preconditioner takes $\widetilde{O}(n^\omega)$ time[2], and running the gradient descent takes $\widetilde{O}(n^2 \log(\kappa/\epsilon))$ time.

15.1.2 Related Work

Convex and Non-Convex Optimization. In convex optimization problems, such as linear programming ([Vai89b, DS08, LS14, CLS19, BLSS20, Ye20, SY21, DLY21, JSWZ21]), empirical risk minimization ([LSZ19]), cutting plane method ([JLSW20]), maximum bipartite matching and max-flow [BLL+21, GLP21, vdBGJ+22, AMV21] and semi-definite programming ([LSW15, JKL+20, HJS+22]), one typically uses an

[2]Here ω is the exponent of matrix multiplication [Wil12, LG14, AW21]

algorithm that can dynamic adjust the search direction and step size to reduce the iteration count. Due to the prohibitively high cost of implementing one step of these methods, most of these works focus on improving the *cost per iteration*.

In non-convex setting, there's a vast body of ongoing works ([MG15, BRB17, PW17, ABH17, BLH18, CGH$^+$19, ZMG19, BPSW21, YGS$^+$21]) that try to improve the iteration count and cost per iteration, especially in the setting of training deep neural network. As shown in [CGH$^+$19], it is possible to exploit the equivalence between over-parametrized networks and neural tangent kernel to optimize an $n \times n$ matrix instead of an $m^2 \times m^2$ matrix, which is an important breakthrough in gaining speedup for such method. Sketching and sampling-based methods can also be used to accelerate the computation of inverses of the Hessian matrix ([PW17]). In spirit, our work resembles most with [CGH$^+$19] and [BPSW21], in the sense that our optimization also works on an $n \times n$ Gram matrix. Our algorithm also makes use of sketching and sampling, as in [PW17].

Over-parameterized Neural Networks. In the deep learning community, understanding the geometry and convergence behavior of various optimization algorithms on over-parameterized neural networks has received a lot of attention ([LL18, JGH18, DZPS19, AZLS19a, AZLS19b, DLL$^+$19, SY19, BPSW21, ZCZG18, CG19, LXS$^+$19, LZB20, LSS$^+$20, OS20, LZB22, CCZG21, HLSY21, SYZ21]). The seminal work of [JGH18] initiates the study of *neural tangent kernel* (NTK), which is a very useful analytical model in the deep learning theory area. By over-parametrizing the neural network so that the network width is relatively large ($m \geq \Omega(n^4)$), one can show that the training dynamic on a neural network is almost the same as that on a NTK.

Sketching. Using randomized linear algebra to reduce the dimension of the problem and speedup the algorithms for various problems has been a growing trend in machine learning community ([Sar06, CW13, Woo14]) due to its wide range of applications to various tasks, especially the efficient approximation of kernel matrices ([ANW14, AKK$^+$20, WZ20, SWYZ21]). The standard “Sketch-and-Solve” ([CW13]) paradigm involves reducing the dimension of the problem via sketching and then using a black-box for the original problem to gain an edge on computational efficiency. Another line of work is to use sketching as a preconditioner ([Woo14, BPSW21]) to obtain a high precision solution.

Roadmap. In Section 15.2, we give a preliminary view of the training setup we consider in this chapter. In Section 15.2.1, we introduce the notations that will be used throughout this chapter. In Section 15.2.2, we consider the training setting. In Section 15.3, we overview the techniques employed in this chapter. In Section 15.3.1, we examine the algorithmic tools utilized in this chapter to achieve subquadratic cost per iteration. In Section 15.3.2, we demonstrate various techniques to prove the convergence of our second-order method. In Section 15.4, we summarize the results in this chapter and point out some future directions.

15.2 Preliminaries

15.2.1 Notations

For any integer $n > 0$, let $[n]$ denote the set $\{1, 2, \cdots, n\}$. Let $\Pr[\cdot]$ denote probability and $\mathbb{E}[\cdot]$ denote expectation. We use $\|x\|_2$ to denote the ℓ_2 norm of a vector x. We use $\|A\|$ and $\|A\|_F$ to denote the spectral norm and the Frobenius norm of matrix A, respectively. We use $A^\top$ to denote the transpose of matrix A. We use I_m to denote the identity matrix of size $m \times m$. For α being a vector or matrix, we use $\|\alpha\|_0$ to denote the number of nonzero entries of α. Given a real square matrix A, we use $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$ to denote its largest and smallest eigenvalues respectively. Given a real matrix A, we use $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$ to denote its largest and smallest singular values respectively. We use $\mathcal{N}(\mu, \sigma^2)$ to denote the Gaussian distribution with mean μ and variance σ^2. We use $\widetilde{O}(f(n))$ to denote $O(f(n) \cdot \operatorname{poly}\log(f(n)))$. We use $\langle \cdot, \cdot \rangle$ to denote the inner product, when applying to two vectors, this denotes the standard dot product between two vectors, and when applying to two matrices, this means $\langle A, B \rangle = tr[A^\top B]$, i.e., the trace of $A^\top B$.

15.2.2 Problem setup

Let $X \in \mathbb{R}^{m_0 \times n}$ denote the data matrix with n data points and m_0 features. By proper re-scaling, we have $\|x_i\|_2 = 1$ for all $i \in [n]$. Consider an L layer neural network with one vector $a \in \mathbb{R}^{m_L}$ and L matrices $W_L \in \mathbb{R}^{m_L \times m_{L-1}}, \cdots, W_2 \in \mathbb{R}^{m_2 \times m_1}$ and $W_1 \in \mathbb{R}^{m_1 \times m_0}$. We will use $W_\ell(t)$ to denote the weight matrix at layer ℓ at time t, and $\nabla W_\ell(t)$ to denote its gradient. We also use $W(t) = \{W_1(t), \ldots, W_L(t)\}$ to denote the collection of weight matrices at time t.

Architecture. We first describe our network architecture. The network consists of L hidden layers, each represented by a weight matrix $W_\ell \in \mathbb{R}^{m_\ell \times m_{\ell-1}}$ for any $\ell \in [L]$. The output layer consists of a vector $a \in \mathbb{R}^{m_L}$. We define the neural network prediction function $f : \mathbb{R}^{m_0} \to \mathbb{R}$ as follows:

$$f(W, x) = a^\top \phi(W_L(\phi(\cdots \phi(W_1 x)))),$$

where $\phi : \mathbb{R} \to \mathbb{R}$ is the (shifted) ReLU activation function ($\sigma_b(x) = \max\{x - b, 0\}$) applied coordinate-wise to a vector.

We measure the loss via the squared-loss function:

$$\mathcal{L}(W) = \frac{1}{2} \sum_{i=1}^{n} (y_i - f(W, x_i))^2.$$

This is also the objective function for our training.

The prediction function $f_t : \mathbb{R}^{m_0 \times n} \to \mathbb{R}^n$ is defined as

$$f_t(X) = \begin{bmatrix} f(W(t), x_1) & f(W(t), x_2) & \cdots & f(W(t), x_n) \end{bmatrix}^\top.$$

Initialization. Our neural networks are initialized as follows:

- For each $\ell \in [L]$, the layer-ℓ's weight parameter $W_\ell(0) \in \mathbb{R}^{m_\ell \times m_{\ell-1}}$ is initialized such that each entry is sampled from $\mathcal{N}(0, \frac{2}{m_\ell})$.
- Each entry of a is an i.i.d. sample from $\{-1, +1\}$ uniformly at random.

Gradient. In order to write gradient in an elegant way, we define some artificial variables:

$$\begin{aligned}
g_{i,1} &= W_1 x_i, & h_{i,1} &= \phi(W_1 x_i), & &\forall i \in [n] \\
g_{i,\ell} &= W_\ell h_{i,\ell-1}, & h_{i,\ell} &= \phi(W_\ell h_{i,\ell-1}), & &\forall i \in [n], \forall \ell \in [L]\backslash\{1\} \\
D_{i,1} &= \mathrm{diag}\big(\phi'(W_1 x_i)\big), & & & &\forall i \in [n] \\
D_{i,\ell} &= \mathrm{diag}\big(\phi'(W_\ell h_{i,\ell-1})\big), & & & &\forall i \in [n], \forall \ell \in [L]\backslash\{1\}
\end{aligned} \tag{15.1}$$

Using the definitions of f and h, we have

$$f(W, x_i) = a^\top h_{i,L}, \quad \in \mathbb{R}, \quad \forall i \in [n]$$

We can compute the gradient of $\mathcal{L}$ in terms of $W_\ell \in \mathbb{R}^{m_\ell \times m_{\ell-1}}$, for all $\ell \geq 2$

$$\frac{\partial \mathcal{L}(W)}{\partial W_\ell} = \sum_{i=1}^{n} (f(W, x_i) - y_i) \underbrace{D_{i,\ell}}_{m_\ell \times m_\ell} \left(\prod_{k=\ell+1}^{L} \underbrace{W_k^\top}_{m_{k-1} \times m_k} \underbrace{D_{i,k}}_{m_k \times m_k} \right) \underbrace{a}_{m_L \times 1} \underbrace{h_{i,\ell-1}^\top}_{1 \times m_{\ell-1}} \tag{15.2}$$

Note that the gradient for $W_1 \in \mathbb{R}^{m_1 \times m_0}$ (recall that $m_0 = d$) is slightly different and can not be written by general form. By the chain rule, the gradient of the variables in W_1 can be expressed as:

$$\frac{\partial \mathcal{L}(W)}{\partial W_1} = \sum_{i=1}^{n} (f(W, x_i) - y_i) \underbrace{D_{i,1}}_{m_1 \times m_1} \left(\prod_{k=2}^{L} \underbrace{W_k^\top}_{m_{k-1} \times m_k} \underbrace{D_{i,k}}_{m_k \times m_k} \right) \underbrace{a}_{m_L \times 1} \underbrace{x_i^\top}_{1 \times m_0} \tag{15.3}$$

It is worth noting that the gradient matrix is of rank n, since it's a sum of n rank-1 matrices.

Jacobian. For each layer $\ell \in [L]$ and time $t \in [T]$, we define the Jacobian matrix $J_{\ell,t} \in \mathbb{R}^{n \times m_\ell m_{\ell-1}}$ via the following formulation:

$$J_{\ell,t} := \left[\mathrm{vec}(\tfrac{\partial f(W(t),x_1)}{\partial W_\ell(t)}) \quad \mathrm{vec}(\tfrac{\partial f(W(t),x_2)}{\partial W_\ell(t)}) \quad \cdots \quad \mathrm{vec}(\tfrac{\partial f(W(t),x_n)}{\partial W_\ell(t)}) \right]^\top .$$

The Gram matrix at layer ℓ and time t is then defined as $G_{\ell,t} = J_{\ell,t} J_{\ell,t}^\top \in \mathbb{R}^{n \times n}$ whose (i,j)-th entry is

$$\Big\langle \frac{\partial f(W(t),x_i)}{\partial W_\ell}, \frac{\partial f(W(t),x_j)}{\partial W_\ell} \Big\rangle.$$

15.3 Technique Overview

In this section, we give an overview of the techniques employed in this chapter. In Section 15.3.1, we showcase our algorithm and explain various techniques being used to obtain a subquadratic cost per iteration. In Section 15.3.2, we give an overview of the proof to show the convergence of our algorithm. To give a simpler and cleaner presentation, we assume $m_\ell = m$ for all $\ell \in [L]$.

15.3.1 Subquadratic time

In this section, we study the different techniques being used to achieve the subquadratic cost per iteration.

We start by demonstrating our algorithm:

Algorithm 105 Informal version of our algorithm.

```
1: procedure OurAlgorithm($f, \{x_i, y_i\}_{i \in [n]}$)    ▷ Theorem 15.1,15.2
2:     /*Initialization*/
3:     Initialize $W_\ell(0)$, $\forall \ell \in [L]$
4:     Store $h_{i,L-1}$ in memory, $\forall i \in [n]$    ▷ Takes $O(nm^2)$ time
5:     for $t = 0 \to T$ do
6:         /*Forward computation*/
7:         $v_{i,L} \leftarrow h_{i,L-1}, \forall i \in [n]$
8:         $h_{i,L} \leftarrow \phi((W_L(0) + \Delta W_L) h_{i,L-1}), \forall i \in [n]$    ▷ Takes $o(nm^2)$ time
9:         $D_{i,L} \leftarrow \mathrm{diag}(h_{i,L}), \forall i \in [n]$
10:        $f_t \leftarrow [a^\top h_{1,L}, \ldots, a^\top h_{n,L}]^\top$    ▷ Takes $O(nm)$ time
11:        /*Backward computation*/
12:        $u_{i,L} \leftarrow a^\top D_{i,L}$
13:        Form $\widetilde{J}_{L,t}$ that approximates $J_{L,t}$ using $\{u_{i,L}\}_{i=1}^n, \{v_{i,L}\}_{i=1}^n$
14:            ▷ Takes $\widetilde{O}(mn)$ time, $\widetilde{J}_{L,t} \in \mathbb{R}^{n \times s}$ where $s = \widetilde{O}(n)$
15:        Compute $g_L$ that approximates $(\widetilde{J}_{L,t} \widetilde{J}_{L,t}^\top)^{-1}(f_t - y)$
16:        Form $J_{L,t}^\top g_\ell$ via low rank factorization $\sum_{i=1}^n g_{L,i} u_{i,L} v_{i,L}^\top$
17:        Implicitly update $\Delta W_L \leftarrow \Delta W_L + \sum_{i=1}^n g_{L,i} u_{i,L} v_{i,L}^\top$
18:    end for
19: end procedure
```

Step 1: Invert Gram by solving regression. The update rule of our algorithm is given by

$$W_L(t+1) \leftarrow W_L(t) - J_{L,t}^\top (J_{L,t} J_{L,t}^\top)^{-1} (f_t - y),$$

where c is $f_t - y$ after proper scaling. Naively forming the Gram matrix $J_{L,t} J_{L,t}^\top$ will take $O(n^2 m^2)$ time and inverting it will take $O(n^\omega)$ time. To avoid the quadratic cost at this step, we instead solve a regression, or a linear system since the Gram matrix has full rank: find the vector $g_{L,t} \in \mathbb{R}^n$ such that

$$\|J_{L,t} J_{L,t}^\top g_{L,t} - (f_t - y)\|_2^2$$

is minimized. This enables us to utilize the power of sketching to solve the regression efficiently.

Step 2: Solve Gram regression via preconditioning. In order to solve the above regression, we adapt the idea of obtaining a good preconditioner via sketching then apply iterative method to solve it ([BPSW21]). Roughly speaking, we first use a random matrix $S \in \mathbb{R}^{s \times m^2}$ that has the *subspace embedding property* ([Sar06]) to reduce the number of rows of $J^\top$, then we run a QR decomposition on matrix $SJ^\top$. This gives us a matrix R such that $SJ^\top R$ has orthonormal columns. We then use gradient descent to optimize the objective $\|JJ^\top R z_t - y\|_2^2$. Since S is a subspace embedding for $J^\top$, we can make sure that the condition number of the matrix $J^\top R$ is small ($O(1)$), hence the gradient descent converges after $\log(\kappa/\epsilon)$ iterations, where κ is the condition number of J. However, in order to implement the gradient descent, we still need to multiply an $m^2 \times n$ matrix with a length n vector, in the worst case this will incur a time of $O(nm^2)$. In order to bypass this barrier, we need to exploit extra structural properties of the Jocobian, which will be demonstrated in the following steps.

Step 3: Low rank structure of the gradient. Instead of studying Jacobian directly, we first try to understand the *low rank structure* of the gradient. Consider $\frac{\partial f(W,x_i)}{\partial W_L} \in \mathbb{R}^{m \times m}$, it can be written as (for simplicity, we use $h_{i,0}$ to denote x_i):

$$\frac{\partial f(W,x_i)}{\partial W_L} = \underbrace{h_{i,L-1}}_{v_i \in \mathbb{R}^{m \times 1}} \underbrace{a^\top D_{i,L}}_{u_i^\top \in \mathbb{R}^{1 \times m}} .$$

This means the gradient is essentially an outer product of two vectors, and hence has rank one. This has several interesting consequences: for over-parametrized networks, the gradient is merely of rank n instead of m. When using first-order method such as gradient descent or stochastic gradient descent, the weight is updated via a low rank matrix. To some extent, this explains why the weight does not move too far from initialization in over-parametrized networks when using first-order method to train. Also, as we will illustrate below, this enables the efficient approximation of Jacobian matrices and maintenance of the change.

Step 4: Fast approximation of the Jacobian matrix. We now turn our attention to design a fast approximation algorithm to the Jacobian matrix. Recall that Jacobian matrix $J_{L,t} \in \mathbb{R}^{n \times m^2}$ is an $n \times m^2$ matrix, therefore writing down the matrix will take $O(nm^2)$ time. However, it is worth noticing that each row of $J_{L,t}$ is $\mathrm{vec}(u_i v_i^\top)^\top$, $\forall i \in [n]$, or equivalently, $u_i \circ v_i$ where $\circ$ denotes the tensor product between two vectors. Suppose we are given the collection of $\{u_1, \dots, u_n\} \in (\mathbb{R}^m)^n$ and $\{v_1, \dots, v_n\} \in (\mathbb{R}^m)^n$, then we can compute the tensor product $u_i \circ v_i$ via tensor-based sketching techniques, such as TensorSketch ([ANW14, DSSW17, DJS+19]) or TensorSRHT ([AKK+20, WZ20]) in time nearly linear in m and the targeted sketching dimension s, in contrast to the naive $O(m^2)$ time. Since it suffices to preserve the length of all vectors in the column space of $J_{L,t}^\top$, the target dimension s can be chosen as $O(\epsilon^{-2} n \cdot \mathrm{poly}(\log(m/\epsilon\delta)))$. Use $\widetilde{J}_{L,t} \in \mathbb{R}^{n \times s}$ to denote this approximation of $J_{L,t}$, we perform the preconditioned gradient descent we described above on this smaller matrix. This enables to lower the overall cost of the regression step to be subquadratic in m.

Step 5: Efficient update via low rank factorization. The low rank structure of the gradient can further be utilized to represent the change on weight matrices ΔW in a way such that any matrix-vector product involving ΔW can be performed fast. Let $g_L \in \mathbb{R}^n$ denote the solution to the regression problem posed in Step 1. Note that by the update rule of our method, we shall use $J_{L,t}^\top g_L \in \mathbb{R}^{m \times m}$ to update the weight matrix, but writing down the matrix will already take $O(m^2)$ time. Therefore, it is instructive to find a succinct representation for the update. The key observation is that each column of $J_{L,t}^\top$ is a tensor product of two vectors: $u_i \circ v_i$ or equivalently, $u_i v_i^\top$. The update matrix can be rewritten as $\sum_{i=1}^n g_{L,i} u_i v_i^\top$, and we can use this representation for the update on the weight, instead of adding it directly. Let

$$U_L := \begin{bmatrix} | & | & \cdots & | \\ g_{L,1}u_1 & g_{L,2}u_2 & \dots & g_{L,n}u_n \\ | & | & \cdots & | \end{bmatrix} \in \mathbb{R}^{m \times n},\ V_L := \begin{bmatrix} | & | & \cdots & | \\ v_1 & v_2 & \dots & v_n \\ | & | & \cdots & | \end{bmatrix} \in \mathbb{R}^{m \times n},$$

then the update can be represented as $U_L V_L^\top$. Consider multiplying a vector $y \in \mathbb{R}^m$ with this representation, we first multiply y with $V_L^\top \in \mathbb{R}^{n\times m}$, which takes $O(mn)$ time. Then we multiply $V_L^\top y \in \mathbb{R}^n$ with $U_L \in \mathbb{R}^{m\times n}$ which takes $O(mn)$ time. This drastically reduces the cost of multiplying the weight matrix with a vector from $O(m^2)$ to $O(mn)$.

It is tempting to store all intermediate low rank representations across all iterations and use them to facilitate matrix-vector product, which incurs a runtime of $O(Tmn)$. This is fine when T is relatively small, however, if one looks for a high precision solution which requires a large number of iterations, then T might be too large and $O(Tmn)$ might be in the order of $O(m^2)$. To circumvent this problem, we design a data structure so that it will exactly compute the $m \times m$ change matrix and update the weight and clean up the cumulative changes. This can be viewed as a "restart" of the data structure. To choose the correct number of updates before restarting, we utilize the *dual exponent of matrix multiplication* α ([GU18]), which means it takes $O(m^{2+o(1)})$ time to multiply an $m \times m$ by an $m \times m^\alpha$ matrix. Hence, we restart the data structure after around m^α/n updates. Therefore, we achieve an amortized $o(m^2)$ time, which is invariant even though the number of iterations T grows larger and larger.

15.3.2 Convergence analysis

In this section, we demonstrate the strategy to prove that our second-order method achieves a linear convergence rate on the training loss.

Step 1: Initialization. Let $W(0)$ be the random initialization. We first show that for any data point x_i, the initial neural network output $f(W(0), x_i) = \widetilde{O}(1)$. The analysis draws inspiration from [AZLS19a]. The general idea is, given a fixed unit length vector x, multiplying it with a random Gaussian matrix W will make sure that $\|Wx\|_2^2 \approx 2$. Since W is a random Gaussian matrix, applying shifted ReLU activation gives a random vector with a truncated Gaussian distribution conditioned on a binomial random variable indicating which neurons are activated. We will end up with $\|\phi(Wx)\|_2 \approx 1$ as well as $\phi(Wx)$ being sparse. Inductively applying this idea to each layer and carefully controlling the error occurring at each layer, we can show that with good probability, $\|h_{i,L}\|_2$ is a constant. We conclude the argument by exploiting the fact that a is a Rademacher random vector so the inner product $\langle a, h_{i,L}\rangle$ concentrates around $\|h_{i,L}\|_2$, and hence with good probability, the value of $f(W(0), x_i) = \widetilde{O}(1)$.

Furthermore, we show that the Gram matrix for the multiple-layer over-parametrized neural network, which is defined as $J_{\ell,0}J_{\ell,0}^\top$, has a nontrivial minimum eigenvalue after the initialization. In particular, we adapt the neural tangent kernel (NTK) for multiple-layer neural networks defined by [DLL+19] into our setting by analyzing the corresponding Gaussian process with shifted ReLU activation function. Then, we can prove that with high probability, the least eigenvalue of the initial Gram matrix is lower bounded by the least eigenvalue of the neural tangent kernel matrix.

Step 2: Small perturbation. The next step is to show that if all weight matrices undergo a small perturbation from initialization (in terms of spectral norm), then the corresponding Jacobian matrix has not changed too much. As long as the perturbation is small enough, it is possible to show that the change of the h vector (in terms of ℓ_2 norm) and the consecutive product (in terms of spectral norm) is also small. Finally, we use the concentration property of Rademacher random variables and Gaussian random variables to conclude that the change of Jacobian has a relatively small spectral norm and Frobenious norm.

Step 3: Connect everything via a double induction. Put things together, we use a double induction argument, where we assume the perturbation of weight matrix is small and the gap between f_t and y is at most $1/3$ of the gap between f_{t-1} and y. By carefully bounding various terms and exploiting the fact the Jacobian matrix *always* has a relative small spectral norm ($\widetilde{O}(\sqrt{n})$), we first show that the weights are not moving too far from the initialization, then use this fact to derive a final convergence bound for $\|f_t - y\|_2$.

15.4 Discussion and Future Directions

In this chapter, we propose and analyze a second-order method to train multi-layer over-parametrized neural networks. Our algorithm achieves a linear convergence rate in terms of training loss and achieves a subquadratic ($o(m^2)$) cost per training iteration. From an analytical perspective, we greatly extend the analysis of ([AZLS19a]) to our method, coupled with the use of the equivalence between multi-layer over-parametrized networks and neural tangent kernels ([DLL$^+$19]). From an algorithmic perspective, we achieve a subquadratic cost per iteration, which is a significant improvement from $O(m^2)$ time per iteration due to the prohibitively large network width m. Our algorithm combines various techniques, such as training with the Gram matrix, solving the Gram regression via sketching-based preconditioning, fast tensor computation and dimensionality reduction, and low-rank decomposition of weight updates. Our algorithm is especially valuable when one requires a high precision solution on training loss, and hence the number of iterations is large.

One of the interesting questions from our work is: is it possible to obtain an algorithm that has a *nearly linear* cost per iteration on m as in the case of training one-hidden layer over-parametrized networks ([BPSW21])? In particular, can this runtime be achieved under the current best width of multi-layer over-parametrized networks ($m \geq n^8$)? We note that the major limitation in our method is the *sparsity* of the change of the diagonal matrices (ΔD) is directly related to the magnitude of the change of weights ($\|\Delta W\|$). In our analysis of convergence, we go through a careful double induction argument, which in fact imposes on a lower bound on $\|\Delta W\|$. It seems to us that, in order to achieve a nearly linear runtime, one has to adapt a different analytical framework or approach the problem from a different perspective.

A related question is, how can we maintain the changes of weight more efficiently? In our work, we achieve speedup in the neural network training process by observing that the change of the weights are small in each iteration. Similar phenomenon also appears in some classical optimization problem (e.g., solving linear program [CLS19, JSWZ21] and solving semi-definite program [JKL$^+$20]) and they achieve further speedup by using lazy update and amortization techniques to compute the weight changes, or using more complicated data structure to maintain the *changes* of the weight changes. Can we adapt their techniques to neural network training? An orthogonal direction to maintain the change is to design an initialization setup such that while we still have enough randomness to obtain provable guarantees, the matrix-vector product with the initial weight matrix can be performed faster than $O(m^2)$ by sparsifying the Gaussian matrix as in [DLPM21] or imposing extra structural assumption such as using circulant Gaussian [RRT12, NN13, KMR14].

Another question concerns activation functions. In this chapter, we consider the shifted ReLU activation and design our algorithm and analysis around its properties. Is it possible to generalize our algorithm and analysis to various other activation functions, such as sigmoid, tanh or leaky ReLU? If one chooses a smooth activation, can we get a better result in terms of convergence rate? Can we leverage this structure to design faster algorithms?

Finally, the network architecture considered in this chapter is the standard feedforward network. Is it possible to extend our analysis and algorithm to other architectures, such as recurrent neural networks (RNN)? For RNN, the weight matrices for each layer are the same. Hence it is trickier to analyze the training dynamics on such networks. Though the convergence of the first-order method on over-parametrized multi-layer RNN has been established, it is unclear whether such analysis can be extended to our method.

Roadmap. In Section 15.5, we remind readers with the notations and some probability tools. In Section 15.6, we illustrate the complete version of our algorithm and give a runtime analysis of it. In Section 15.7, we design a simple low rank maintenance data structure and use it to efficiently implement matrix-vector product. In Section 15.8, we introduce an efficient regression solver handling our Jacobian and Gram regression. In Section 15.9, we study the spectral property of the Gram matrix at each layer and connects it with multi-layer neural tangent kernels. In Section 15.10, we analyze the convergence of our algorithm by using some heavy machinery such as structural analysis of the gradient and a careful double induction. In Section 15.11, we give a detailed proof of one technical lemma.

15.5 Preliminaries

In this section, we introduce notations that will be used throughout the rest of the chapter and a technical tool that will be heavily exploited in the later proofs.

Notations. For any $n \in \mathbb{N}_+$, let $[n]$ denote the set $\{1, 2, \cdots, n\}$. We use $\mathbb{E}[\cdot]$ for expectation and $\Pr[\cdot]$ for probability. For any vector $x \in \mathbb{R}^d$, we use $\|x\|_2$ for the ℓ_2 norm of a vector x. For any matrix $A \in \mathbb{R}^{m \times m}$, We use $\|A\|$ for the spectral norm of matrix A, i.e., $\|A\| = \max_{i \in [m]}\{|\lambda_i(A)\}$ where $\lambda_i(A)$ is the i-th eigenvalue. We use $\|A\|_F$ for the Frobenius norm of A. For any $A \in \mathbb{R}^{n \times m}$, we define $A^\top \in \mathbb{R}^{m \times n}$ to be the transpose of A. We use I_m for the identity matrix of size $m \times m$. For matrix A or vector x, $\|A\|_0, \|x\|_0$ denote the number of nonzero entries of A and x respectively. Note that $\|\cdot\|_0$ is a semi-norm since it satisfies triangle inequality. Given a real square matrix A, we use $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$ for its largest and smallest eigenvalues respectively. Given a real matrix A, we use $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$ for its largest and smallest singular values respectively. We use $\mathcal{N}(\mu, \sigma^2)$ for the Gaussian distribution with mean μ and variance σ^2. We use $\widetilde{O}(f(n))$ for $O(f(n) \cdot \operatorname{poly}\log(f(n))$. Let $\langle \cdot, \cdot \rangle$ be the inner product between two vectors or two matrices. When applying it to two vectors, this means the standard dot product between two vectors, and when applying to two matrices, this means $\langle A, B \rangle = tr[A^\top B]$ where $tr[A]$ denote the trace of matrix

A. We use $x \circ y = \mathrm{vec}(xy^\top)$ for the tensor product between two conforming vectors x and y.

Fact 15.4 (Minimum eigenvalue of Hadamard product matrices, [Sch11])**.** *Let $A, B \in \mathbb{R}^{n \times n}$ be two PSD matrices. Then, we have*

$$\lambda_{\min}(A \odot B) \geq \min_{i \in [n]} (B)_{i,i} \cdot \lambda_{\min}(A).$$

15.6 Complete Algorithm and Its Runtime Analysis

In this section, we first present our complete algorithm, then we analyze its running time. We show that as long as we use the shifted ReLU activation so that the number of activated neurons is sparse, then all our operations can be realized in subquadratic time.

Algorithm 106 Training last layer.

```
1: procedure CompleteAlgorithm($X \in \mathbb{R}^{d \times n}, y \in \mathbb{R}^n$)          ▷ Theorem 15.5
2:     /*Initialization*/
3:     Initialize $W_\ell(0), \forall \ell \in [L]$
4:     Compute $h_{i,\ell}$ for $\ell \in [L-1]$                         ▷ Takes $O(nm^2L)$ time
5:     Store $h_{i,L-1}$ in memory, $\forall i \in [n]$
6:     LowRankMaintenance LMR                                  ▷ Algorithm 107
7:     LMR.Init($\{W_1(0) \dots, W_L(0)\}$)
8:     for $t = 0 \to T$ do
9:         /*Forward computation*/
10:        $v_{i,L} \leftarrow h_{i,L-1}, \forall i \in [n]$
11:        $g_{i,L} \leftarrow$ LMR.Query($L, h_{i,L-1}), \forall i \in [n]$         ▷ Takes $o(nm^2)$ time
12:        $h_{i,L} \leftarrow \phi(g_{i,L}), \forall i \in [n]$                          ▷ $h_{i,L}$ is sparse
13:        $D_{i,L} \leftarrow \mathrm{diag}(\phi'(g_{i,L})), \forall i \in [n]$                 ▷ $D_{i,L}$ is sparse
14:        $f_t \leftarrow [a^\top h_{1,L}, \dots, a^\top h_{n,L}]^\top$                  ▷ Takes $O(nm)$ time
15:        /*Backward computation*/
16:        $u_{i,L} \leftarrow a^\top D_{i,L}, \forall i \in [n]$                        ▷ Takes $o(nm)$ time
17:        $g_L \leftarrow$ FastTensorRegression($\{u_{i,L}\}_{i=1}^n, \{v_{i,L}\}_{i=1}^n, f_t - y$) with precision
   $\sqrt{\lambda_L / n}$
18:        LMR.Update($\{g_{L,i} u_{i,L}\}_{i=1}^n, \{v_{i,L}\}_{i=1}^n$)
19:    end for
20: end procedure
```

Theorem 15.5 (Formal version of Theorem 15.2)**.** *Let $X \in \mathbb{R}^{d \times n}$ and $y \in \mathbb{R}^n$, and let k denote the sparsity of $D_{i,\ell}$ and s denote the sparsity of $\Delta D_{i,\ell}$, $\forall \ell \in [L], i \in [n]$. Let m denote the width of neural network, L denote the number of layers and α denote the dual matrix multiplication exponent (Def. 15.1),then the running time of Algorithm 106 is*

$$O(\mathcal{T}_{\mathrm{init}} + T \cdot \mathcal{T}_{\mathrm{iter}}),$$

where

$$\mathcal{T}_{\mathrm{init}} = O(m^2 nL),$$
$$\mathcal{T}_{\mathrm{iter}} = \widetilde{O}(n \cdot (m^{2-\alpha} + m \cdot (s+k))).$$

Therefore, the cost per iteration *of Algorithm 106 is*

$$\widetilde{O}(n \cdot (m^{2-\alpha} + m \cdot (s+k))).$$

Proof. We analyze $\mathcal{T}_{\mathrm{init}}$ and $\mathcal{T}_{\mathrm{iter}}$ separately.

Initialization time. We will first initialize $(L-1)$ $m \times m$ matrices and one $m \times d$ matrix, which takes $O(m^2 L)$ time. Compute $h_{i,L-1}$ for all $i \in [n]$ takes $O(m^2 nL)$ time. Finally, initialize the data structure takes $O(m^2 L)$ time. Hence, $\mathcal{T}_{\mathrm{init}} = O(m^2 nL)$.

Cost per iteration. For each iteration, we perform one forward computation from layer 1 to L, then we train the last layer via solving a regression based on its Jacobian matrix.

- **Forward computation:** In forward computation, we first compute $g_{i,L} \in \mathbb{R}^m$, which involves using the QUERY procedure of LMR data structure, hence by Lemma 15.7, it takes $O(m \cdot (s+k+m^\alpha))$ time. Compute $h_{i,L}$ and $D_{i,L}$ takes $O(m)$ time. Hence the overall runtime of forward computation is $O(nm \cdot (s+k+m^\alpha))$.

- **Backward computation:** In backward computation, we first compute $u_{i,L} \in \mathbb{R}^m$, which takes $O(m(s+k))$ time owing to the sparsity of $D_{i,L}$. Then, we call Algorithm 110 to solve the Gram regression problem, which due to Theorem 15.17 takes $\widetilde{O}(mn + n^\omega)$ time. Note that even we want a high probability version of the solver with $e^{-\log^2 nL}$ failure probability, we only pay extra $\log^2 nL$ term in running time, which is absorbed by the $\widetilde{O}(\cdot)$ notation. Finally, the update takes $O(m^{2-\alpha}n)$ amortized time owing to Lemma 15.7. Put things together, we get an overall running time of $\widetilde{O}(n(m(s+k) + m^{2-\alpha}))$ time.

This concludes the proof of our Theorem. $\square$

Corollary 15.6. *Suppose the network width m is chosen as in 15.11 and the shift parameter b is chosen as in 15.10, then the cost per iteration of Algorithm 106 is*

$$\widetilde{O}(m^{2-\alpha}n).$$

Remark 15.1. As long as the neural network is wide enough, as in 15.11 and we choose the shift threshold properly, as in 15.10, then we can make sure that both sparsity parameters k and s to be $o(m)$, and we achieve subquadratic cost per iteration.

We also compare our result with our approaches. Note that a naive implementation of variants of gradient descent will take $O(nm^2)$ time, namely, one evaluates the gradient with respect to each data point and sum them up. By batching the n data points and use fast rectangular matrix multiplication, the running time can be improved to $\mathcal{T}_{\mathrm{mat}}(m, n, m)$, in the setting where $n \leq m^\alpha$, this will only take $O(m^{2+o(1)})$ time.

In the specific parameter set we choose, we need that $m^{2-\alpha}n < m^2$ to truly beat the quadratic barrier, which implies that $n < m^\alpha$. As we will later see the choice of m (Def. 15.11), we will have $n \leq m^{1/4}$, which means that we get a *truly subquadratic time* in m.

15.7 Low Rank Maintenance and Efficient Computation of the Change

The goal of this section is to present a data structure that maintains the low rank representation of the change of weights in a lazy fashion, so that it can support matrix-vector product query efficiently.

15.7.1 Low rank maintenance

In this section, we design a lazy data structure that maintains a low rank representation of the change of weights. We also show that using this data structure, we can implement matrix-vector product query fast.

Before moving, we define some notions related to rectangular matrix multiplication.

Definition 15.1 ([Wil12, GU18])**.** Let ω be the matrix multiplication exponent such that it takes $n^{\omega+o(1)}$ time to multiply two $n \times n$ matrices.

Let α be the dual exponent of the matrix multiplication which is the supremum among all $a \geq 0$ such that it takes $n^{2+o(1)}$ time to multiply an $n \times n$ by $n \times n^a$ matrix.

Additionally, we define the function $\omega(\cdot)$ where $\omega(b)$ denotes the exponent of multiplying an $n \times n$ matrix by an $n \times n^b$ matrix. Hence, we have $\omega(1) = \omega$ and $\omega(\alpha) = 2$.

The overall idea of our low rank maintenance data structure is as follows: we keep accumulating the low rank change, when the rank of the change reaches a certain threshold (m^α), then we restart the data structure and update the weight matrix.

Algorithm 107 Low rank maintenance data structure

```
1: data structure LowRankMaintenance                                        ▷ Lemma 15.7
2:     members
3:         $r_\ell, \forall \ell \in [L]$                 ▷ $r_\ell$ denotes the accumulated rank of the change
4:         $W_\ell, \forall \ell \in [L]$                                          ▷ $\{W_\ell\}_{\ell=1}^L \in (\mathbb{R}^{m \times m})^L$
5:         $\Delta W_\ell, \forall \ell \in [L]$                                   ▷ $\{\Delta W_\ell\}_{\ell=1}^L \in (\mathbb{R}^{m \times m})^L$
6:     end members
7:
8:     procedures
9:         Init($\{W_1(0), \dots W_L(0)\}$)                          ▷ Initialize the data structure
10:        Update($U_\ell, V_\ell$)                             ▷ Update the low rank representation
11:        Query($\ell, y$)       ▷ Compute the matrix-vector product between $\Delta W_\ell$ and $y$
12:    end procedures
13: end data structure
```

Algorithm 108 Procedures of LRM data structure

```
1: procedure Init($\{W_1(0), \dots, W_L(0)\}$)                              ▷ Lemma 15.7
2:     $W_\ell \leftarrow W_\ell(0)$
3:     $\Delta W_\ell \leftarrow 0, \forall \ell \in [L]$
4:     $r_\ell \leftarrow 0, \forall \ell \in [L]$
5: end procedure
6:
7: procedure Update($U_\ell \in \mathbb{R}^{m \times n}, V_\ell \in \mathbb{R}^{m \times n}$)                ▷ Lemma 15.7
8:     $\Delta W_\ell \leftarrow \Delta W_\ell + U_\ell V_\ell^\top$ without forming the product and sum the two matrices
9:     $r_\ell \leftarrow r_\ell + n$
10:    if $r_\ell = m^a$ where $a = \omega(2)$ then
11:        $W_\ell \leftarrow W_\ell + \Delta W_\ell$                                    ▷ Takes $O(m^2)$ time
12:        $r_\ell \leftarrow 0$
13:        $\Delta W_\ell \leftarrow 0$
14:    end if
15: end procedure
16:
17: procedure Query($\ell \in [L], y \in \mathbb{R}^m$)                             ▷ Lemma 15.7
18:    $z \leftarrow W_\ell \cdot y + \Delta W_\ell \cdot y$                  ▷ Takes $O(\mathrm{nnz}(y) \cdot m + m r_\ell)$ time
19:    return $z$
20: end procedure
```

Lemma 15.7. *There exists a deterministic data structure (Algorithm 107) such that* *maintains*

$$\Delta W_1, \dots, \Delta W_L$$

such that

- *The procedure* Init *(Algorithm 108) takes $O(m^2 L)$ time.*
- *The procedure* Update *(Algorithm 108) takes $O(nm^{2-\alpha+o(1)})$ amortized time, where $\alpha = \omega(2)$.*
- *The procedure* Query *(Algorithm 108) takes $O(m \cdot (\mathrm{nnz}(y) + r_\ell))$ time, where r_ℓ is the rank of ΔW_ℓ when* Query *is called.*

Proof. The runtime for Init is straightforward, for Query, notice that we are multiplying vector y with a (possibly) dense matrix $W_\ell \in \mathbb{R}^{m \times m}$, which takes $O(\mathrm{nnz}(y) \cdot m)$ time, and an accumulated low rank matrix ΔW_ℓ with rank r_ℓ. By using the low rank decomposition $\Delta W_\ell = UV^\top$ with $U, V \in \mathbb{R}^{m \times r_\ell}$, the time to multiply y with ΔW is $O(mr_\ell)$. Combine them together, we get a running time of $O(m \cdot (\mathrm{nnz}(y) + r_\ell))$.

It remains to analyze the amortized cost of Update. Note that if $r_\ell < m^a$, then we just pay $O(1)$ time to update corresponding variables in the data structure. If $r_\ell = m^a$, then we will explicitly form the $m \times m$ matrix ΔW_ℓ. To form it, notice we have accumulated r_ℓ / n different sums of rank-n decompositions, which can be represented as

$$U = [U_\ell(1), U_\ell(2), \dots, U_\ell(r_\ell/n)] \in \mathbb{R}^{m \times r_\ell},$$
$$V = [V_\ell(1), V_\ell(2), \dots, V_\ell(r_\ell/n)] \in \mathbb{R}^{m \times r_\ell},$$

and $\Delta W_\ell = UV^\top$, which takes $O(m^{2+o(1)})$ time to compute since $r_\ell = m^a$ and $a = \omega(2)$. Finally, note that this update of W_ℓ only happens once per r_ℓ/n number of calls to Update, therefore we can charge each step by $O(\frac{m^2}{r_\ell/n}) = O(m^{2-a}n) = O(m^{2-\alpha}n)$, arrives at our final amortized running time. $\square$

Remark 15.2. Currently, the dual matrix multiplication exponent $\alpha \approx 0.31$ [GU18], hence the amortized time for UPDATE is $O(nm^{1.69})$. If $m \geq n^4$, then we achieve an update time of $o(m^2)$. Similarly, the time for QUERY is $O(m \cdot (\mathrm{nnz}(y) + r_\ell)) = O(m \cdot \mathrm{nnz}(y) + m^{1+\alpha}) = O(m \cdot \mathrm{nnz}(y) + m^{1.31})$, as long as $\mathrm{nnz}(y) = o(m)$, then its running time is also $o(m^2)$. In our application of training neural networks, we will make sure that the inputted vector y is sparse.

15.7.2 Efficient computation of rank-1 decompositions

We illustrate the method to compute the vectors $u_{i,\ell}, v_{i,\ell} \in \mathbb{R}^m$ using the low rank maintenance data structure. Recall the definition of these vectors:

$$\begin{aligned} u_{i,\ell}(t)^\top &= a^\top D_{i,L}(t) W_L(t) \dots D_{i,\ell+1}(t) W_{\ell+1}(t) D_{i,\ell}(t) \in \mathbb{R}^{1\times m}, \\ v_{i,\ell}(t) &= h_{i,\ell-1}(t) \in \mathbb{R}^m. \end{aligned}$$

Before proceeding, we list the assumptions we will be using:

- For any $\ell \in [L]$, $D_{i,\ell}(t)$ is s_D-sparse, where $s_D := k + s$, k is the sparsity of $D_{i,\ell}(0)$ and s is the sparsity of $D_{i,\ell}(t) - D_{i,\ell}(0)$.
- For any $\ell \in [L]$, the change of the weight matrix W_ℓ, $\Delta W_\ell(t) := W_\ell(t) - W_\ell(0)$, is of low-rank. That is, $\Delta W_\ell(t) = \sum_{j=1}^{r_t} y_{\ell,j} z_{\ell,j}^\top$.
- For any $i \in [n]$, $W_1(0)x_i$ is pre-computed.

We first note that as a direct consequence of $D_{i,\ell}(0)$ is k-sparse, $h_{i,\ell}(0)$ is k-sparse as well. Similarly, $h_{i,\ell}(t) - h_{i,\ell}(0)$ has sparsity s. Hence $h_{i,\ell}(t)$ has sparsity bounded by s_D.

Compute $u_{i,\ell}(t)$. Compute $u_{i,\ell}(t)$ is equivalent to compute the following vector:

$$D_{i,\ell}(t)(W_{\ell+1}(0) + \Delta W_{\ell+1}(t))^\top D_{i,\ell+1}(t) \cdots (W_L(0) + \Delta W_L(t))^\top D_{i,L}(t) a.$$

First, we know that $D_{i,L}(t)a \in \mathbb{R}^m$ is an s_D-sparse vector, and it takes $O(s_D)$ time. The next matrix is $(W_L(0)+\Delta W_L(t))^\top$, which gives two terms: $W_L(0)^\top(D_{i,L}(t)a)$ and $\Delta W_L(t)^\top(D_{i,L}(t)a)$. For the first term, since $D_{i,L}(t)a$ is s_D-sparse, it takes $O(ms_D)$-time. For the second term, we have

$$\begin{aligned}\Delta W_L(t)^\top(D_{i,L}(t)a) = & \sum_{j=1}^{r_t} z_{L,j} y_{L,j}^\top (D_{i,L}(t)a) \\ = & \sum_{j=1}^{r_t} z_{L,j} \cdot \langle y_{L,j}, D_{i,L}(t)a\rangle.\end{aligned}$$

Each inner-product takes $O(s_D)$-time and it takes $O(mr_t + s_D r_t) = O(mr_t)$-time in total. Hence, in $O(m(s_D + r_t))$-time, we compute the vector $W_L(t)^\top D_{i,L}(t)a$. Note that we do not assume the sparsity of a.

Thus, by repeating this process for the $L-\ell$ intermediate matrices $W_j^\top(t)D_{i,j}(t)$, we can obtain the vector

$$\left(\prod_{j=\ell+1}^{L} W_j^\top(t)D_{i,j}(t)\right)a$$

in time $O((L-\ell)m(s_D+r_t))$. Finally, by multiplying a sparse diagonal matrix $D_{i,\ell}(t)$, we get the desired vector $u_{i,\ell}(t)$.

Compute $v_{i,\ell}(t)$. Note that $v_{i,\ell}(t)$ is essentially $h_{i,\ell-1}(t)$, so we consider how to compute $h_{i,\ell}(t)$ for general $\ell \in [L]$. Recall that

$$h_{i,\ell}(t) = \phi((W_\ell(0) + \Delta W_\ell(t))h_{i,\ell-1}(t)),$$

since $h_{i,\ell-1}(t)$ is s_D-sparse, the product $W_\ell(0)h_{i,\ell-1}(t)$ can be computed in $O(ms_D)$ time. For the product $\Delta W_\ell(t)h_{i,\ell-1}(t)$ can be computed use the low rank decomposition, which takes $O(mr_t)$ time. Apply the shifted ReLU takes $O(m)$ time. Hence, the total time is $O(m(r_t + s_D))$ time.

The running time results are summarized in the following lemma:

Lemma 15.8. *For $\ell \in [L]$ and $i \in [n]$, suppose $\|D_{i,\ell}(0)\|_0 \leq k$. Let $t > 0$. Suppose the change of $D_{i,\ell}$ is sparse, i.e., $\|D_{i,\ell}(t) - D_{i,\ell}(0)\|_0 \leq s$. For $\ell \in [L]$, $i \in [n]$, for any $t > 0$, suppose the change of W_ℓ is of low-rank, i.e., $\Delta W_\ell(t) = \sum_{j=1}^{r_t} y_{\ell,j} z_{\ell,j}^\top$. We further assume that $\{y_{\ell,j}, z_{\ell,j}\}_{\ell \in [L], j \in [r_t]}$ and $\{W_1(0)x_i\}_{i \in [n]}$ are pre-computed.*

Then, for any $\ell \in [L]$ and $i \in [n]$, the vectors $u_{\ell,i}(t), v_{\ell,i}(t) \in \mathbb{R}^m$ can be computed in

$$O(mL(s + k + r_t))$$

time.

As a direct consequence, if we combine Lemma 15.7 and Lemma 15.8, then we get the following corollary:

Corollary 15.9. *For $\ell \in [L]$ and $i \in [n]$, we can compute $v_{i,\ell}(t), u_{i,\ell}(t) \in \mathbb{R}^m$ as in Algorithm 106 with the following time bound:*

- *Compute $u_{i,\ell}(t)$ in time $O(mL(s + k + r_\ell))$.*
- *Compute $v_{i,\ell}(t)$ in time $O(m(s + k + r_\ell))$.*

Remark 15.3. We note that the result we present is more general than needed for our algorithm, since it can handle the updates across all layers. This means we can use it to implement a subquadratic cost per iteration algorithm for gradient descent algorithm over all layers. In this chapter, we focus our attention to the training of last layer, since the step size of that algorithm is chosen adaptively.

15.8 Fast Tensor Product Regression

In this section, we show how to solve a specific type of regression fast using both tensor-based sketching matrices for approximation, and sketching-based preconditioner for high precision solution.

Consider the following problem: Given two matrices $U = [u_1^\top, \dots, u_n^\top]^\top, V = [v_1^\top, \dots, v_n^\top] \in \mathbb{R}^{m \times n}$ with $m \gg n$, consider the matrix $J \in \mathbb{R}^{n \times m^2}$ formed by

$$J = \begin{bmatrix} \mathrm{vec}(u_1 v_1^\top)^\top \\ \mathrm{vec}(u_2 v_2^\top)^\top \\ \vdots \\ \mathrm{vec}(u_n v_n^\top)^\top \end{bmatrix}.$$

We are also given a vector c in n dimension, our task is to find a solution to the following regression problem:

$$\min_{x \in \mathbb{R}^n} \|JJ^\top x - c\|_2^2.$$

Our main theorem for this section is as follows:

Theorem 15.10 (Restatement of Theorem 15.17)**.** *Given two $n \times m$ matrices U and V, and a target vector $c \in \mathbb{R}^n$. Let $J = [\mathrm{vec}(u_1 v_1^\top)^\top, \dots, \mathrm{vec}(u_n v_n^\top)^\top] \in \mathbb{R}^{n \times m^2}$. There is an algorithm (Algorithm 110) takes $\widetilde{O}(nm + n^2(\log(\kappa/\epsilon) + \log(m/\epsilon\delta)\epsilon^{-2}) + n^\omega)$ time and outputs a vector $\widehat{x} \in \mathbb{R}^n$ such that*

$$\|JJ^\top \widehat{x} - c\|_2 \le \epsilon \|c\|_2$$

holds with probability at least $1 - \delta$, and κ is the condition number of J.

15.8.1 Approximate J via TensorSketch

We introduce the notion of TensorSketch for two vectors:

Definition 15.2. Let $h_1, h_2 : [m] \to [s]$ be 3-wise independent hash functions, also let $\sigma : [m] \to \{\pm 1\}$ be a 4-wise independent random sign function. The degree two TensorSketch transform, $S : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^s$ is defined as follows: for any $i, j \in [m]$ and $r \in [s]$,

$$S_{r,(i,j)} = \sigma_1(i) \cdot \sigma_2(j) \cdot \mathbf{1}[h_1(i) + h_2(j) = r \bmod s].$$

Remark 15.4. Apply S to two vectors $x, y \in \mathbb{R}^m$ can be implemented in time $O(s \log s + \mathrm{nnz}(x) + \mathrm{nnz}(y))$.

We introduce one key technical lemma from [ANW14]:

Lemma 15.11 (Theorem 1 of [ANW14])**.** *Let $S \in \mathbb{R}^{s \times m^2}$ be the* TensorSketch *matrix, consider a fixed n-dimensional subspace V. If $s = \Omega(n^2/(\epsilon^2\delta))$, then with probability at least $1-\delta$, $\|Sx\|_2 = (1 \pm \epsilon)\|x\|_2$ simultaneously for all $x \in V$.*

Now we are ready to prove the main lemma of this section:

Lemma 15.12. *Let $\epsilon, \delta \in (0,1)$ denote two parameters. Let $J \in \mathbb{R}^{n \times m^2}$ represent a matrix such that the i-th row of J is equal to $\mathrm{vec}(u_i v_i^\top)$ for some $u_i, v_i \in \mathbb{R}^m$. Then, we can compute a matrix $\widetilde{J} \in \mathbb{R}^{n \times s}$ such that for any vector $x \in \mathbb{R}^n$, with probability at least $1-\delta$, we have*

$$\|\widetilde{J}^\top x\|_2 = (1 \pm \epsilon)\|J^\top x\|_2,$$

where $s = \Omega(n^2/(\epsilon^2\delta))$. The time to compute $\widetilde{J}$ is $O(ns \log s + \mathrm{nnz}(U) + \mathrm{nnz}(V))$.

Proof. Notice that the row space of matrix J can be viewed as an n-dimensional subspace, hence, by Lemma 15.11, the TensorSketch matrix S with $s = \Omega(n^2/(\epsilon^2\delta))$ can preserve the length of all vectors in the subspace generated by $J^\top$ with probability $1-\delta$, to a multiplicative factor of $1 \pm \epsilon$.

The running time part is to apply the FFT algorithm to each row of J with a total of n rows. For each row, it takes $O(s \log s + m)$ time, hence the overall running time is $O(n(s \log s + m))$. □

15.8.2 Approximate J via TensorSRHT

We note that the dependence on the target dimension of sketching is $O(1/\delta)$ for TensorSketch. We introduce another kind of sketching technique for tensor, called TensorSRHT. The tradeoff is we lose input sparsity runtime of matrices U and V.

Definition 15.3. We define the TensorSRHT $S : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^s$ as $S = \frac{1}{\sqrt{s}} P \cdot (HD_1 \times HD_2)$, where each row of $P \in \{0,1\}^{s \times m^2}$ contains only one 1 at a random coordinate, one can view P as a sampling matrix. H is a $m \times m$ Hadamard matrix, and D_1, D_2 are two $m \times m$ independent diagonal matrices with diagonals that are each independently set to be a Rademacher random variable (uniform in $\{-1, 1\}$).

Remark 15.5. By using FFT algorithm, apply S to two vectors $x, y \in \mathbb{R}^m$ takes time $O(m \log m + s)$.

We again introduce a technical lemma for TensorSRHT.

Lemma 15.13 (Theorem 3 of [AKK+20]). *Let $S \in \mathbb{R}^{s \times m^2}$ be the* TensorSRHT *matrix, consider a fixed n-dimensional subspace V. If $s = \Omega(n \log^3(nm/\epsilon\delta)\epsilon^{-2})$, then with probability at least $1 - \delta$, $\|Sx\|_2 = (1 \pm \epsilon)\|x\|_2$ simultaneously for all $x \in V$.*

Lemma 15.14. *Let $\epsilon, \delta \in (0,1)$ denote two parameters. Given a list of vectors $u_1, \cdots, u_m, v_1, \cdots, v_m \in \mathbb{R}^m$. Let $J \in \mathbb{R}^{n \times m^2}$ represent a matrix where the i-th row of J is equal to $\mathrm{vec}(u_i v_i^\top)$. Then, we can compute a matrix $\widetilde{J} \in \mathbb{R}^{n \times s}$ such that for any vector $x \in \mathbb{R}^n$, with probability at least $1 - \delta$, we have*

$$\|\widetilde{J}^\top x\|_2 = (1 \pm \epsilon)\|J^\top x\|_2,$$

where $s = \Omega(n \log^3(nm/(\epsilon\delta))\epsilon^{-2})$. The time to compute $\widetilde{J}$ is $O(n(m \log m + s))$.

Proof. The correctness follows directly from Lemma 15.13. The running time follows from the FFT algorithm to each row of J, each application takes $O(m \log m + s)$ time, and we need to apply it to n rows. $\square$

15.8.3 Sketching-based Preconditioner

In this section, we first use TensorSketch and TensorSRHT to approximate J, then use a general sketching matrix as a preconditioner to solve a regression task involving $JJ^\top$.

Before proceeding, we introduce the notion of *subspace embedding*:

Definition 15.4 (Subspace Embedding, [Sar06]). Let $A \in \mathbb{R}^{N \times k}$, we say a matrix $S \in \mathbb{R}^{s \times N}$ is a $(1 \pm \epsilon) - \ell_2$ subspace embedding for A if for any $x \in \mathbb{R}^k$, we have $\|SAx\|_2^2 = (1 \pm \epsilon)\|Ax\|_2^2$. Equivalently, $\|I - U^\top S^\top SU\| \leq \epsilon$ where U is an orthonormal basis for the column space of A.

We will mainly utilize efficient subspace embedding.

Definition 15.5 ([LDFU13, Woo14]). Given a matrix $A \in \mathbb{R}^{N \times k}$ with $N = \mathrm{poly}(k)$, then we can compute an $S \in \mathbb{R}^{k\mathrm{poly}(\log(k/\delta))/\epsilon^2 \times N}$ such that with probability at least $1 - \delta$, we have

$$\|SAx\|_2 = (1 \pm \epsilon)\|Ax\|_2$$

hols for all $x \in \mathbb{R}^k$. Moreover, SA can be computed in $O(Nk \log((k \log N)/\epsilon))$ time.

Algorithm 109 Fast Regression algorithm of [BPSW21]

```
1:  procedure FastRegression($A, y, \epsilon$)                         ▷ Lemma 15.15
2:                                          ▷ $A \in \mathbb{R}^{N \times k}$ is full rank, $\epsilon \in (0, 1/2)$
3:      Compute a subspace embedding $SA$                  ▷ $S \in \mathbb{R}^{k\mathrm{poly}(\log k)} \times N$
4:      Compute $R$ such that $SAR$ has orthonormal columns via QR decomposition
        ▷ $R \in \mathbb{R}^{k \times k}$
5:      $z_0 = \mathbf{0}_k \in \mathbb{R}^k$
6:      $t \leftarrow 0$
7:      while $\|A^\top ARz_t - y\|_2 \geq \epsilon$ do
8:          $z_{t+1} \leftarrow z_t - (R^\top A^\top AR)^\top (R^\top A^\top ARz_t - R^\top y)$
9:          $t \leftarrow t + 1$
10:     end while
11:     return $Rz_t$
12: end procedure
```

Lemma 15.15 (Lemma 4.2 of [BPSW21]). *Let $N = \Omega(k\mathrm{poly}(\log k))$. Given a matrix $A \in \mathbb{R}^{N \times k}$, let κ denote its condition number. Consider the following regression task:*

$$\min_{x \in \mathbb{R}^k} \|A^\top Ax - y\|_2.$$

Using the procedure FASTREGRESSION *(Algorithm 109), with probability at least* $1-\delta$*, we can compute an* ϵ*-approximate solution* $\widehat{x}$ *satisfying*

$$\|A^\top A\widehat{x} - y\|_2 \leq \epsilon\|y\|_2$$

in time $\widetilde{O}(Nk\log(\kappa/\epsilon) + k^\omega)$*.*

Our algorithm is similar to the ridge regression procedure in [SWYZ21], where they first apply their sketching algorithm as a bootstrapping to reduce the dimension of the original matrix, then use another subspace embedding to proceed and get stronger guarantee.

We shall first prove a useful lemma.

Lemma 15.16. *Let* $A \in \mathbb{R}^{N\times k}$*, suppose* SA *is a subspace embedding for* A *(Def. 15.5), then we have for any* $x \in \mathbb{R}^k$*, with probability at least* $1-\delta$*,*

$$\|(SA)^\top SAx - b\|_2 = (1\pm\epsilon)\|A^\top Ax - b\|_2.$$

Proof. Throughout the proof, we condition on the event that S preserves the length of all vectors in the column space of A.

Note that

$$\|(SA)^\top SAx - b\|_2^2 = \|(SA)^\top SAx\|_2^2 + \|b\|_2^2 - 2\langle (SA)^\top SAx, b\rangle.$$

We will first bound the norm of $(SA)^\top SAx$, then the inner product term.

Bounding $\|(SA)^\top SAx\|_2^2$**.**

Let $U \in \mathbb{R}^{N\times k}$ represent an orthonormal basis of A, then use an alternative definition of subspace embedding, we have $\|U^\top S^\top SU - I\| \leq \epsilon$, this means all the eigenvalues of $U^\top S^\top SU$ lie in the range of of $[(1-\epsilon)^2, (1+\epsilon)^2]$. Let V denote the matrix $U^\top S^\top SU$, then we know that all eigenvalues of $V^\top V$ lie in range $[(1-\epsilon)^4, (1+\epsilon)^4]$. Setting ϵ as $\epsilon/4$, we arrive at $\|V^\top V - I\| \leq \epsilon$. This shows that for any $x \in \mathbb{R}^k$, we have $\|(SA)^\top SAx\|_2 = (1\pm\epsilon)\|A^\top Ax\|_2$.

Bounding $\langle (SA)^\top SAx, b\rangle$.

Note that

$$\begin{aligned}\langle (SA)^\top SAx, b\rangle &= \langle SAx, SAb\rangle \\ &= 1/2 \cdot (\|SAx\|_2^2 + \|SAb\|_2^2 - \|SA(x-b)\|_2^2) \\ &= 1/2 \cdot (1\pm\epsilon)(\|Ax\|_2^2 + \|Ab\|_2^2 - \|A(x-b)\|_2^2) \\ &= (1\pm\epsilon)\langle A^\top Ax, b\rangle.\end{aligned}$$

Combining these two terms, we conclude that, with probability at least $1-\delta$,

$$\|(SA)^\top SAx - b\|_2 = (1\pm\epsilon)\|A^\top Ax - b\|_2.$$

□

Algorithm 110 Fast Regression via tensor trick

```
1:  procedure FastTensorRegression($\{u_i\}_{i=1}^n \in \mathbb{R}^{m\times n}, \{v_i\}_{i=1}^n \in \mathbb{R}^{m\times n}, c \in \mathbb{R}^n$)    ▷ Theorem 15.17
2:                                   ▷ $J = [\mathrm{vec}(u_1 v_1^\top)^\top,\ \mathrm{vec}(u_2 v_2^\top)^\top, \ldots,\ \mathrm{vec}(u_n v_n^\top)^\top]^\top \in \mathbb{R}^{n\times m^2}$
3:      $s_1 \leftarrow \Theta(n\log^3(nm/\delta))$
4:      $s_2 \leftarrow \Theta((n+\log m)\log n)$
5:      Let $S_1 \in \mathbb{R}^{s_1\times m^2}$ be a sketching matrix    ▷ $S_1$ can be TensorSketch or TensorSRHT
6:      Compute $\widetilde{J} = JS_1^\top$ via FFT algorithm    ▷ $\widetilde{J} \in \mathbb{R}^{n\times s_1}$
7:      Choose $S_2 \in \mathbb{R}^{s_2\times s_1}$ to be a sketching matrix (see Def. 15.5)
8:      Compute a subspace embedding $S_2\widetilde{J}^\top$
9:      Compute $R$ such that $S_2\widetilde{J}^\top R$ has orthonormal columns via QR decomposition    ▷ $R \in \mathbb{R}^{n\times n}$
10:     $z_0 \leftarrow \mathbf{0}_k \in \mathbb{R}^k$
11:     $t \leftarrow 0$
12:     while $\|\widetilde{J}\widetilde{J}^\top Rz_t - c\|_2 \geq \epsilon$ do
13:         $z_{t+1} \leftarrow z_t - (R^\top \widetilde{J}\widetilde{J}^\top R)^\top (R^\top \widetilde{J}\widetilde{J}^\top Rz_t - R^\top c)$
14:         $t \leftarrow t+1$
15:     end while
16:     return $Rz_t$
17: end procedure
```

Theorem 15.17. *Given two $n \times m$ matrices U and V, and a target vector $c \in \mathbb{R}^n$. Let $J = [\mathrm{vec}(u_1 v_1^\top)^\top, \ldots, \mathrm{vec}(u_n v_n^\top)^\top] \in \mathbb{R}^{n \times m^2}$. There is an algorithm (Algorithm 110) takes $\widetilde{O}(nm + n^2(\log(\kappa/\epsilon) + \log(m/\delta)) + n^\omega)$ time and outputs a vector $\widehat{x} \in \mathbb{R}^n$ such that*

$$\|JJ^\top \widehat{x} - c\|_2 \leq \epsilon \|c\|_2$$

holds with probability at least $1 - \delta$, and κ is the condition number of J.

Proof. We can decompose Algorithm 110 into two parts:

- Applying S_1 to efficiently form matrix $\widetilde{J}$ to approximate J and reduce its dimension, notice here we only need ϵ for this part to be a small constant, pick $\epsilon = 0.1$ suffices.
- Using S_2 as a preconditioner and solve the regression problem iteratively.

Let $\widehat{x}$ denote the solution found by the iterative regime. We will prove this statement in two-folds:

- First, we will show that $\|\widetilde{J}\widetilde{J}^\top \widehat{x} - c\|_2 \leq \epsilon \|c\|_2$ with probability at least $1 - \delta$;
- Then, we will show that $\|JJ^\top \widehat{x} - c\|_2 = (1 \pm 0.1)\|\widetilde{J}\widetilde{J}^\top \widehat{x} - c\|_2$ with probability at least $1 - \delta$.

Combining these two statements, we can show that

$$\begin{aligned} \|JJ^\top \widehat{x} - c\|_2 &= (1 \pm 0.1)\|\widetilde{J}\widetilde{J}^\top \widehat{x} - c\|_2 \\ &\leq 1.1\epsilon \|c\|_2 \end{aligned}$$

Setting ϵ to $\epsilon/1.1$ and δ to $\delta/2$, we conclude our proof. It remains to prove these two parts.

Part 1 $\|\widetilde{J}\widetilde{J}^\top\widehat{x}-c\|_2 \leq \epsilon\|c\|_2$. We observe the iterative procedure is essentially the same as running FastRegression on input $\widetilde{J}^\top, y, \epsilon$, hence by Lemma 15.15, we know

$$\Pr\left[\|\widetilde{J}\widetilde{J}^\top\widehat{x}-c\|_2 \leq \epsilon\|c\|_2\right] \geq 1-\delta.$$

Part 2 $\|JJ^\top\widehat{x}-c\|_2 = (1\pm 0.1)\|\widetilde{J}\widetilde{J}^\top\widehat{x}-c\|_2$. To prove this part, note that by Lemma 15.13, we know that $\widetilde{J}^\top$ is a subspace embedding for $J^\top$. Hence, we can utilize Lemma 15.16 and get that,

$$\Pr\left[\|JJ^\top\widehat{x}-c\|_2 = (1\pm 0.1)\|\widetilde{J}\widetilde{J}^\top\widehat{x}-c\|_2\right] \geq 1-\delta.$$

Combining these two parts, we have proven the correctness of the theorem. It remains to justify the running time. Note that running time can be decomposed into two parts: 1). The time to generate $\widetilde{J}$, 2). The time to compute $\widehat{x}$ via iterative scheme.

Part 1 Generate $\widetilde{J}$. To generate $\widetilde{J}$, we apply $S_1 \in \mathbb{R}^{s_1\times m^2}$ which is a TensorSRHT. By Lemma 15.14, it takes $O(n(m\log m+s_1))$ time to compute $\widetilde{J}$, plug in $s_1=\Theta(n\log^3(nm/\delta))$, the time is $\widetilde{O}(nm)$.

Part 2 Compute $\widehat{x}$. To compute $\widehat{x}$, essentially we run FastRegression on $\widetilde{J}^\top, c, \epsilon$, hence by Lemma 15.15, it takes $\widetilde{O}(s_2 n\log(\kappa/\epsilon)+n^\omega)$ time, with $s_2 = \Theta((n+\log m)\log n)$ and κ is the condition number of $\widetilde{J}$, which has the guarantee $\kappa=(1\pm\epsilon)\kappa(J)$. Hence, the overall running time of this part is $\widetilde{O}(n^2\log(\kappa/\epsilon)+n^\omega)$.

Put things together, the overall running time is $\widetilde{O}(nm+n^2\log(\kappa/\epsilon)+n^\omega)$. $\square$

Remark 15.6. Due to the probability requirement (union bounding over all data points), here we only prove by using TensorSRHT. One can use similar strategy to obtain an input sparsity time version using TensorSketch. We remark that this framework is similar to the approach [SWYZ21] takes to solve kernel ridge regression, where one first uses a shallow but fast sketch to bootstrap, then use another sketching to proceed with the main task.

We also point out that in a more general neural network architecture, the network width between different layers might differ, i.e., we are in fact dealing with the tensor product $u \otimes v$ where $u \in \mathbb{R}^{m_\ell}$ and $v \in \mathbb{R}^{m_{\ell-1}}$. Our sketching can be modified to handle this kind of inputs. Specifically, for TensorSketch, it is defined by a pair of hash functions and sign functions, we can change their domain to handle different input dimensions. For TensorSRHT, it's more tricky, however, we note that the Hadamard matrix is merely for speeding up computation via FFT algorithm, hence we can differ the size of D_1 and D_2, and change the size of sampling matrix P accordingly.

15.9 Spectral Properties of Over-parametrized Deep Neural Network

In this section, we study the spectral structure of our Gram matrix and its connection to the multi-layer NTK matrix. We show two different results regarding the minimum eigenvalue of the last layer and the intermediate layers. We also show that as long as the weight matrix does not move too far, the eigenvalue of the Gram matrix is relatively stable around its initialization.

15.9.1 Bounds on the Least Eigenvalue of Kernel at Initialization

The following fact gives the exact form of the Gram matrix of the ℓ-th layer of the neural network.

Fact 15.18 (Multi-layer Gram matrix)**.** *For any $\ell \in [L]$, let $G_\ell = J_\ell J_\ell^\top \in \mathbb{R}^{n \times n}$ be the layer-ℓ's Gram matrix. Then, for any $i, j \in [n]$,*

$$(G_\ell)_{i,j} = h_{i,\ell-1}^\top h_{j,\ell-1} \cdot a^\top \left(D_{i,\ell} \prod_{k=\ell+1}^{L} W_k^\top D_{i,k} \right)^\top \left(D_{j,\ell} \prod_{k=\ell+1}^{L} W_k^\top D_{j,k} \right) a,$$

where $h_{i,\ell-1} = \prod_{k=1}^{\ell-1} D_{i,k} W_k x_i$.

Proof. For for any $i, j \in [n]$, by definition,

$$\begin{aligned}
(G_\ell)_{i,j} &= \mathrm{vec}(\frac{\partial f(W, x_i)}{\partial W_\ell})^\top \mathrm{vec}(\frac{\partial f(W, x_j)}{\partial W_\ell}) \\
&= \mathrm{vec}\left(D_{i,\ell} \prod_{k=\ell+1}^{L} W_k^\top D_{i,k} a h_{i,\ell-1}^\top\right)^\top \mathrm{vec}\left(D_{j,\ell} \prod_{k=\ell+1}^{L} W_k^\top D_{j,k} a h_{j,\ell-1}^\top\right) \\
&= \left((h_{i,\ell-1} \otimes I_{m_\ell})\left(D_{i,\ell} \prod_{k=\ell+1}^{L} W_k^\top D_{i,k} a\right)\right)^\top (h_{j,\ell-1} \otimes I_{m_\ell})\left(D_{j,\ell} \prod_{k=\ell+1}^{L} W_k^\top D_{j,k} a\right) \\
&= \left(D_{i,\ell} \prod_{k=\ell+1}^{L} W_k^\top D_{i,k} a\right)^\top (h_{i,\ell-1}^\top \otimes I_{m_\ell})(h_{j,\ell-1} \otimes I_{m_\ell})\left(D_{j,\ell} \prod_{k=\ell+1}^{L} W_k^\top D_{j,k} a\right) \\
&= a^\top \left(D_{i,\ell} \prod_{k=\ell+1}^{L} W_k^\top D_{i,k}\right)^\top \left(D_{j,\ell} \prod_{k=\ell+1}^{L} W_k^\top D_{j,k}\right) a \cdot h_{i,\ell-1}^\top h_{j,\ell-1}.
\end{aligned}$$

□

The following lemma handles the least eigenvalue for all intermediate layers $\ell \in [L-1]$.

Lemma 15.19 (Bounds on the least eigenvalue at initialization for layer 1 to $L-1$)**.** *Let $\lambda := \min_{\ell\in[L-1]} \lambda_\ell$. Then, for all $\ell \in [L-1]$, with probability at least $1-\delta$, we have*

$$\lambda_{\min}(G_\ell) \geq \Omega(\lambda \delta^2 n^{-2} L^{-1}).$$

Proof. Let $\ell \in [L]$ and $i, j \in [n]$. By Fact 15.18, the (i, j)-th entry of the layer-ℓ Gram matrix can be expressed as

$$\begin{aligned}
(G_\ell)_{i,j} &= h_{i,\ell-1}^\top h_{j,\ell-1} \cdot a^\top \left(D_{i,\ell} \prod_{k=\ell+1}^{L} W_k^\top D_{i,k}\right)^\top \left(D_{j,\ell} \prod_{k=\ell+1}^{L} W_k^\top D_{j,k}\right) a \\
&= (H_{\ell-1})_{i,j} \cdot (A_\ell)_{i,j},
\end{aligned}$$

where $(H_{\ell-1})_{i,j} = h_{i,\ell-1}^\top h_{j,\ell-1}$ and $(A_\ell)_{i,j} = a^\top (D_{i,\ell} \prod_{k=\ell+1}^{L} W_k^\top D_{i,k})^\top (D_{j,\ell} \prod_{k=\ell+1}^{L} W_k^\top D_{j,k}) a$. Hence, we can write G_ℓ as

$$G_\ell = H_{\ell-1} \odot A_\ell,$$

where $\odot$ represents the Hadamard product.

By Fact 15.4, we have

$$\lambda_{\min}(G_\ell) \geq \min_{i \in [n]} \ (A_\ell)_{i,i} \cdot \lambda_{\min}(H_\ell)$$

Note that for $i \in [n]$,

$$\begin{aligned}
(A_\ell)_{i,i} &= a^\top (D_{i,\ell} \prod_{k=\ell+1}^{L} W_k^\top D_{i,k})^\top (D_{i,\ell} \prod_{k=\ell+1}^{L} W_k^\top D_{i,k}) a \\
&= \Big\| D_{i,\ell} \prod_{k=\ell+1}^{L} W_k^\top D_{i,k} a \Big\|_2^2 \\
&\geq \frac{\Big\langle W_\ell \prod_{k=1}^{\ell-1} D_{i,k} W_k x_i, D_{i,\ell} \prod_{k=\ell+1}^{L} W_k^\top D_{i,k} a \Big\rangle^2}{\| W_\ell \prod_{k=1}^{\ell-1} D_{i,k} W_k x_i \|_2^2} \\
&= \frac{\langle a, h_{i,L} \rangle^2}{\| W_\ell \prod_{k=1}^{\ell-1} D_{i,k} W_k x_i \|_2^2}.
\end{aligned}$$

By Lemma 15.28 part (a), we have

$$\Big\| W_\ell \prod_{k=1}^{\ell-1} D_{i,k} W_k x_i \Big\|_2^2 \leq O(\sqrt{L})$$

with probability $1 - e^{-\Omega(k/L^2)}$ for all $i \in [n]$, where $k = m \exp(-b^2 m/4) = m^{0.8}$ by our choice of parameters.

By Lemma 15.27,

$$\Pr[\forall i \in [n], \quad \|h_{i,L}\|_2 \in 1 \pm \epsilon] \geq 1 - O(nL) \cdot \exp(-\Omega(k\epsilon^2/L^2)).$$

Conditioning on this event, let $h_{i,L}$ be a fixed vector h with length $1 \pm \epsilon$ and consider the randomness of the Rademacher vector a.

Note that $\langle a, h \rangle$ can be written as $a^\top (hh^\top) a = a^\top B a$, where $B := hh^\top$ satisfies $\|B\| = \|h\|_2^2$ and $\|B\|_{\mathrm{HS}}^2 = \|h\|_2^4$. For $r \in [m_L]$, we know that a_r is a centered subgaussian random variable with $\|a_r\|_{\psi_2} = 1$.

By Lemma A.9, we have

$$\Pr[|\langle a, h\rangle| \leq t] \leq \min\left\{\frac{C_1 t}{\|h\|_2}, \frac{C_2 t}{\widetilde{O}(m^{-0.2})\sqrt{k}}\right\} \leq \widetilde{O}(t).$$

By taking t to be $O(\delta/n)$, we have

$$\Pr[\forall i \in [n],\ \langle a, h_{i,L}\rangle^2 = \Omega(\delta^2/n^2)] \geq 1 - O(\delta).$$

Applying a union bound gives

$$\Pr[\min_{i\in[n]}\ (A_\ell)_{i,i} = \Omega(\delta^2 n^{-2} L^{-1})] \geq 1 - \delta/2.$$

By Lemma 15.20, with probability at least $1 - \delta/2$, we have:

$$\lambda_{\min}(H_{\ell-1}) \geq \Omega(\lambda)$$

for all $\ell \in [L]$.

Combine them together, we get that

$$\lambda_{\min}(G_\ell) \geq \Omega(\delta^2 \lambda n^{-2} L^{-1})$$

for all $\ell \in [L]$ with probability $1 - \delta$, which completes the proof of the lemma. $\square$

In order to bound $\lambda_{\min}(H_\ell)$, we first define the NTK kernel for multiple layer neural network.

Definition 15.6 (Multiple layer NTK kernel)**.** The NTK kernel $\mathbf{K}_\ell \in \mathbb{R}^{n\times n}$ for $\ell \in \{0, \ldots, L\}$ of an L-layer neural network are defined as follows:

- $(\mathbf{K}_0)_{i,j} := x_i^\top x_j$
- For $\ell > 0$, let $\Sigma_{\ell,i,j} := \begin{bmatrix} (\mathbf{K}_{\ell-1})_{i,i} & (\mathbf{K}_{\ell-1})_{i,j} \\ (\mathbf{K}_{\ell-1})_{j,i} & (\mathbf{K}_{\ell-1})_{j,j} \end{bmatrix} \in \mathbb{R}^{2\times 2}$ for any $(i,j) \in [n]\times[n]$. Then,

$$(\mathbf{K}_\ell)_{i,j} := \mathbb{E}_{(x_1,x_2)\sim\mathcal{N}(\mathbf{0},2\Sigma_{\ell-1,i,j})}[\phi(x_1)\phi(x_2)] \quad \forall \ell \in [L-1],$$
$$(\mathbf{K}_L)_{i,j} := \mathbb{E}_{(x_1,x_2)\sim\mathcal{N}(\mathbf{0},2\Sigma_{L-1,i,j})}[\phi'(x_1)\phi'(x_2)]$$

Let $\lambda_\ell := \lambda_{\min}(\mathbf{K}_\ell)$ to be the minimum eigenvalue of the NTK kernel $\mathbf{K}_\ell$.

In the following lemma, we generalize Lemma C.3 in [BPSW21] (also Lemma 3 in [CGH$^+$19]) into multiple layer neural networks.

Lemma 15.20. *For $\ell \in [L-1]$, let λ_ℓ denote the minimum eigenvalue of NTK defined for ℓ-th layer of neural networks. Suppose $m_\ell = \Omega(\lambda_\ell^{-2} n^2 \log(n/\delta))$, then with probability at least $1-\delta$, we have*

$$\lambda_{\min}(H_\ell) \geq \frac{3}{4}\lambda_\ell, \quad \forall \ell \in [L]$$

Proof. We will prove that $\|H_\ell - \mathbf{K}_\ell\|_\infty$ is small, which implies that $\lambda_{\min}(H_\ell)$ is close to λ_ℓ. The proof idea is similar to [DLL$^+$19] via induction on ℓ.

For $\ell = 1$, recall $(g_{1,i})_k = \sum_{b\in[m]}(W_1)_{k,b}(x_i)_b$ for $k \in [m]$. Hence, for any $k \in [m]$,

$$\begin{aligned}
\mathbb{E}[(g_{i,1})_k(g_{j,1})_k] = & \sum_{b,b'\in[m]} \mathbb{E}[(W_1)_{k,b}(W_1)_{k,b'}(x_i)_b(x_j)_{b'}] \\
= & \sum_{b\in[m]} \mathbb{E}[(W_1)_{k,b}^2]\cdot(x_i)_b(x_j)_b \qquad ((W_1)_{k,b} \sim \mathcal{N}(0, \tfrac{2}{m}).) \\
= & \frac{2}{m}\sum_{b\in[m]}(x_i)_b(x_j)_b \\
= & \frac{2}{m}x_i^\top x_j.
\end{aligned}$$

Then, we have

$$\begin{aligned}
\mathbb{E}[h_{i,1}^\top h_{j,1}] = & \sum_{k\in[m]} \mathbb{E}[(h_{i,1})_k(h_{j,1})_k] \\
= & \sum_{k\in[m]} \mathbb{E}[\phi((g_{i,1})_k)\phi((g_{j,1})_k)] \\
= & \sum_{k\in[m]} \mathbb{E}_{(u,v)\sim\mathcal{N}(0,\frac{2}{m}\Sigma_{1,i,j})}[\phi(u)\phi(v)] \\
= & \mathbb{E}_{(u,v)\sim\mathcal{N}(0,\frac{2}{m}\Sigma_{1,i,j})}[m\phi(u)\phi(v)] \\
= & \mathbb{E}_{(u',v')\sim\mathcal{N}(0,2\Sigma_{1,i,j})}[\phi(u')\phi(v')] \\
= & (\mathbf{K}_1)_{i,j}.
\end{aligned}$$

Next, we will show that $h_{i,1}^\top h_{j,1}$ concentrates around its expectation. First, for any $k \in [m]$,

$$|(h_{i,1})_k (h_{j,1})_k| \leq |(g_{i,1})_k (g_{j,1})_k| \leq |\langle (W_1)_{k,*}, x_i \rangle| \cdot |\langle (W_1)_{k,*}, x_j \rangle|.$$

Since $\langle (W_1)_{k,*}, x_i \rangle \sim \mathcal{N}(0, \frac{2\|x_i\|_2^2}{m})$, by the concentration of Gaussian distribution,

$$|\langle (W_1)_{k,*}, x_i \rangle| \leq \sqrt{c} \quad \forall k \in [m], i \in [n]$$

holds with probability at least $1 - mne^{-cm/4}$.

Conditioning on the above event, we have $|(h_{i,1})_k (h_{j,1})_k| \leq c$ for all $i, j \in [n]$ and $k \in [m]$. Then, by Hoeffding's inequality, we have for any $(i,j) \in [n] \times [n]$,

$$\begin{aligned} \Pr\left[|h_{i,1}^\top h_{j,1} - (\mathbf{K}_1)_{i,j}| \geq t\right] \leq &\ \exp\left(-\frac{t^2}{2m \cdot (2c)^2}\right) \\ = &\ \exp(-\Omega(t^2/(mc^2))). \end{aligned}$$

Hence, by union bound, we get that

$$\Pr[\max_{(i,j)\in[n]\times[n]} |h_{i,1}^\top h_{j,1} - (\mathbf{K}_1)_{i,j}| \leq t] \geq 1 - mn \cdot \exp(-\Omega(mc)) - n^2 \cdot \exp(-\Omega(t^2/(mc^2))).$$

If we choose $c := \frac{\log(mnL/\delta)}{m}$ and $t := m^{-1/2} \cdot \mathrm{polylog}(nL/\delta)$, we have with probability at least $1 - \frac{\delta}{L}$,

$$\max_{(i,j)\in[n]\times[n]} |h_{i,1}^\top h_{j,1} - (\mathbf{K}_1)_{i,j}| \leq \widetilde{O}(m^{-1/2}).$$

Let $h < L$. Suppose that for $\ell = 1, \ldots h$,

$$\max_{(i,j)\in[n]\times[n]} |h_{i,\ell}^\top h_{j,\ell} - (\mathbf{K}_\ell)_{i,j}| \leq \widetilde{O}(m^{-1/2}).$$

Consider $\ell = h + 1$. By a similar computation, we have

$$\mathbb{E}_{W_\ell}[(g_{i,\ell})_k (g_{j,\ell})_k] = \frac{2}{m} h_{i,\ell-1}^\top h_{j,\ell-1}.$$

Define a new covariance matrix

$$\widehat{\Sigma}_{\ell,i,j} := \begin{bmatrix} h_{i,\ell-1}^\top h_{i,\ell-1} & h_{i,\ell-1}^\top h_{j,\ell-1} \\ h_{j,\ell-1}^\top h_{i,\ell-1} & h_{j,\ell-1}^\top h_{j,\ell-1} \end{bmatrix} \quad \forall (i,j) \in [n] \times [n].$$

We have

$$\begin{aligned}\mathbb{E}_{W_\ell}[h_{i,\ell}^\top h_{j,\ell}] &= \sum_{k\in[m]} \mathbb{E}_{(u,v)\sim\mathcal{N}(0,\frac{2}{m}\widehat{\Sigma}_{\ell,i,j})}[\phi(u)\phi(v)] \\ &= \mathbb{E}_{(u',v')\sim\mathcal{N}(0,2\widehat{\Sigma}_{\ell,i,j})}[\phi(u')\phi(v')] \\ &:= (\widehat{\mathbf{K}}_\ell)_{i,j}.\end{aligned}$$

Hence, we have with probability at least $1-\frac{\delta}{L}$,

$$\max_{(i,j)\in[n]\times[n]} \left| h_{i,\ell}^\top h_{j,\ell} - (\widehat{\mathbf{K}}_\ell)_{i,j}\right| \le \widetilde{O}(m^{-1/2}). \tag{15.4}$$

It remains to upper bound the difference $\|\widehat{\mathbf{K}}_\ell - \mathbf{K}_\ell\|_\infty$.

$$\left\|\widehat{\mathbf{K}}_\ell - \mathbf{K}_\ell\right\|_\infty = \max_{(i,j)\in[n]\times[n]} \left|\mathbb{E}_{(u,v)\sim\mathcal{N}(0,2\widehat{\Sigma}_{\ell,i,j})}[\phi(u)\phi(v)] - \mathbb{E}_{(u,v)\sim\mathcal{N}(0,2\Sigma_{\ell,i,j})}[\phi(u)\phi(v)]\right|.$$

Recall that

$$\Sigma_{\ell,i,j} := \begin{bmatrix} (\mathbf{K}_{\ell-1})_{i,i} & (\mathbf{K}_{\ell-1})_{i,j} \\ (\mathbf{K}_{\ell-1})_{j,i} & (\mathbf{K}_{\ell-1})_{j,j}\end{bmatrix} \quad \forall (i,j)\in[n]\times[n],$$

and hence, by the induction hypothesis, we have

$$\|\widehat{\Sigma}_{\ell,i,j} - \Sigma_{\ell,i,j}\|_\infty \le \max_{(i,j)\in[n]\times[n]} \left| h_{i,\ell-1}^\top h_{j,\ell-1} - (\mathbf{K}_{\ell-1})_{i,j}\right| = \widetilde{O}(m^{-1/2}).$$

Notice that $\widehat{\Sigma}_{\ell,i,j}$ can be written as

$$\begin{bmatrix} \|h_{i,\ell-1}\|_2^2 & \cos(\theta_{\ell,i,j})\|h_{i,\ell-1}\|_2\|h_{j,\ell-1}\|_2 \\ \cos(\theta_{\ell,i,j})\|h_{i,\ell-1}\|_2\|h_{j,\ell-1}\|_2 & \|h_{j,\ell-1}\|_2^2\end{bmatrix}.$$

Moreover, when ϕ is the ReLU function, we have

$$\mathbb{E}_{(u,v)\sim\mathcal{N}(0,2\widehat{\Sigma}_{\ell,i,j})}[\phi(u)\phi(v)] = 2\|h_{i,\ell-1}\|_2\|h_{j,\ell-1}\|_2 \cdot F(\theta_{\ell,i,j}),$$

where

$$F(\theta) := \mathbb{E}_{(u,v)\sim\mathcal{N}(0,\Sigma(\theta))}[\phi(u)\phi(v)] \quad \text{with} \quad \Sigma(\theta) := \begin{bmatrix} 1 & \cos(\theta) \\ \cos(\theta) & 1\end{bmatrix}.$$

We note that $F(\theta)$ has the following analytic form:

$$F(\theta) = \frac{1}{2\pi}(\sin(\theta) + (\pi - \theta)\cos(\theta)) \in [0, 1/2]. \tag{15.5}$$

Similarly,

$$\mathbb{E}_{(u,v)\sim\mathcal{N}(0,2\Sigma_{\ell,i,j})}[\phi(u)\phi(v)] = 2\sqrt{(\mathbf{K}_{\ell-1})_{i,i}(\mathbf{K}_{\ell-1})_{j,j}} \cdot F(\tau_{\ell,i,j}),$$

where $\tau_{\ell,i,j} := \cos^{-1}\left(\frac{(\mathbf{K}_{\ell-1})_{i,j}}{\sqrt{(\mathbf{K}_{\ell-1})_{i,i}(\mathbf{K}_{\ell-1})_{j,j}}}\right)$. By the induction hypothesis, we have $(\mathbf{K}_\ell)_{i,j} \in h_{\ell,i}^\top h_{\ell,j} \pm \widetilde{O}(m^{-1/2})$ for all $i, j \in [n]$. By Lemma 15.27, we also have $\|h_{\ell,i}\|_2 \in 1 \pm \epsilon$ for all $\ell \in [L]$ and $i \in [n]$ with probability $1 - O(nL) \cdot e^{-\Omega(m\epsilon^2/L)}$. They implies that $\cos(\tau_{\ell,i,j}) \in \cos(\theta) \pm \widetilde{O}(m^{-1/2})$. Thus, by Taylor's theorem, it gives us

$$|F(\theta_{\ell,i,j}) - F(\tau_{\ell,i,j})| \leq \widetilde{O}(m^{-1/2}).$$

Therefore, we have

$$\begin{aligned}
&\left|\mathbb{E}_{(u,v)\sim\mathcal{N}(0,2\widehat{\Sigma}_{\ell,i,j})}[\phi(u)\phi(v)] - \mathbb{E}_{(u,v)\sim\mathcal{N}(0,2\Sigma_{\ell,i,j})}[\phi(u)\phi(v)]\right| \\
&= 2\left|\|h_{i,\ell-1}\|_2\|h_{j,\ell-1}\|_2 F(\theta_{\ell,i,j}) - \sqrt{(\mathbf{K}_{\ell-1})_{i,i}(\mathbf{K}_{\ell-1})_{j,j}} F(\tau_{\ell,i,j})\right| \\
&\leq \widetilde{O}(m^{-1/2}).
\end{aligned}$$

That is,

$$\|\widehat{\mathbf{K}_\ell} - \mathbf{K}_\ell\|_\infty \leq \widetilde{O}(m^{-1/2}). \tag{15.6}$$

Combining Eqs. (15.4) and (15.6) together, we get that

$$\max_{(i,j)\in[n]\times[n]} |h_{\ell,i}^\top h_{\ell,j} - (\mathbf{K}_\ell)_{i,j}| \leq \widetilde{O}(m^{-1/2})$$

holds with probability at least $1 - \frac{\delta}{L}$ for $\ell = h + 1$.

By induction, we have proved that for the first $L - 1$ layers, the intermediate correlation $h_{\ell,i}^\top h_{\ell,j}$ is close to the intermediate Gram matrix $(\mathbf{K}_\ell)_{i,j}$, i.e.,

$$\|H_\ell - K_\ell\| \leq \frac{\lambda_\ell}{4} \quad \forall \ell \in [L-1].$$

Hence, we get that for all $\ell \in [L-1]$,

$$\lambda_{\min}(H_\ell) \geq \frac{3}{4}\lambda_\ell$$

The lemma is then proved. $\square$

Lemma 15.21 (Bounds on the least eigenvalue at initialization for layer L)**.** *Suppose $m = \Omega(\lambda_L^{-2} n^2 \log(n/\delta))$, then we have*

$$\Pr[\lambda_{\min}(G_L) \geq \frac{3}{4}\lambda_L] \geq 1 - \delta.$$

Proof. Recall G_L is defined as

$$\begin{aligned}(G_L)_{i,j} &= \mathrm{vec}(\frac{\partial f(W, x_i)}{\partial W_L})^\top \mathrm{vec}(\frac{\partial f(W, x_j)}{\partial W_L}) \\ &= \mathrm{vec}(D_{i,L} a h_{i,L-1}^\top)^\top \mathrm{vec}(D_{j,L} a h_{j,L-1}^\top) \\ &= a^\top D_{i,L} D_{j,L} a \cdot h_{i,L-1}^\top h_{j,L-1},\end{aligned}$$

which has the same form as the correlation matrix of a two-layer over-parameterized neural network with input data $\{h_{L-1,i}\}_{i\in[n]}$. Define

$$(\widehat{\mathbf{K}}_L)_{i,j} := h_{i,L-1}^\top h_{j,L-1} \cdot \mathbb{E}_{w\sim\mathcal{N}(0,2I_m)}\left[\phi'(w^\top h_{i,L-1})\phi'(w^\top h_{j,L-1})\right].$$

Then, by the analysis of the two-layer case (see for example [SY19, DZPS19]), we have

$$\|G_L - \widehat{\mathbf{K}}_L\| \leq \frac{\lambda_L}{8},$$

if $m = \Omega(\lambda_L^{-2} n^2 \log(n/\delta))$, where $\lambda_L := \lambda_{\min}(\mathbf{K}_L)$. It remains to bound $\|\widehat{\mathbf{K}}_L - \mathbf{K}_L\|_\infty$. Equivalently, for any $(i,j) \in [n] \times [n]$,

$$\max_{(i,j)\in[n]\times[n]} \left|\mathbb{E}_{(u,v)\sim\mathcal{N}(0,2\widehat{\Sigma}_{L,i,j})}[\phi'(u)\phi'(v)] - \mathbb{E}_{(u,v)\sim\mathcal{N}(0,2\Sigma_{L,i,j})}[\phi'(u)\phi'(v)]\right|.$$

The expectation has the following analytic form:

$$\mathbb{E}_{(z_1,z_2)\sim\mathcal{N}(0,\Sigma)}[\phi'(z_1)\phi'(z_2)] = \frac{1}{4} + \frac{\sin^{-1}(\rho)}{2\pi} \quad \text{with} \quad \Sigma = \begin{bmatrix} p^2 & \rho pq \\ \rho pq & q^2 \end{bmatrix}.$$

By the analysis of the $(L-1)$-layer, we know that $|\rho_{L,i,j} - \widehat{\rho}_{L,i,j}| \leq \widetilde{O}(m^{-1/2})$, where $\rho_{L,i,j} := \cos(\tau_{L,i,j})$ and $\widehat{\rho}_{L,i,j} := \cos(\theta_{L,i,j})$. Also, notice that $\cos(\tau_{L,i,j}) = F(\tau_{L-1,i,j}) \in [0, 1/2]$ by Eq. (15.5). Hence, the derivative of the expectation is bounded, and by Taylor's theorem, we have

$$\|\widehat{\mathbf{K}}_L - \mathbf{K}_L\|_\infty \leq \widetilde{O}(m^{-1/2}).$$

It implies that $\|\widehat{\mathbf{K}}_L - \mathbf{K}_L\| \leq \frac{\lambda_L}{8}$, which further implies that

$$\|G_L - \mathbf{K}_L\| \leq \frac{\lambda_L}{4}.$$

Equivalently, we get that

$$\lambda_{\min}(G_L) \geq \frac{3}{4}\lambda_L$$

with probability at least $1-\delta$. $\square$

Remark 15.7. We observe a discrepancy of eigenvalue in our analysis: For last layer, the eigenvalue of our Gram matrix and the NTK is almost the same, while for intermediate layers, we can only provide a much weaker lower bound for Gram matrix. The main reason is by definition, the NTK for last layer is defined as the product of two derivatives of ReLU, which always have value 0 or 1. On the other hand, the NTKs for intermediate layers are defined in terms of the product of two ReLU's, which can have much larger magnitudes.

Due to such eigenvalue discrepancy, our algorithm focuses on *training the last layer*, since the training dynamic on intermediate layers has a much smaller magnitude. Hence, we present an algorithm that only trains the last layer while obtaining a good convergence result.

15.9.2 Bounds on the Least Eigenvalue during Optimization

In this section, we adapt the Lemma C.5 in [BPSW21] into the last layer of a multi-layer neural network. We make use of the result proved in [SYZ21].

Lemma 15.22 (Lemma C.2 in [SYZ21])**.** *Let $b > 0$ and $\widetilde{R} \leq 1/b$. Let $c > 0$ and $c' > 0$ denote two fixed constants. Suppose we have*

$$\|W_L - W_L(0)\| \leq \widetilde{R},$$

then we have

- $\|G_L(W) - G_L(W(0))\|_F \leq n\alpha$ *holds with probability at least* $1 - n^2\beta$.
- $\lambda_{\min}(G_L(W)) \geq \frac{3}{4}\lambda_L - n\alpha$ *holds with probability at least* $1 - n^2\beta - \delta$,

where $\alpha = \min\{c \cdot \exp(-b^2/2), 3\widetilde{R}\}$ *and* $\beta = \exp(-m \cdot \min\{c' \cdot \exp(-b^2/2), \widetilde{R}/10\})$.

Corollary 15.23. *Suppose we have*

- $\alpha = 3\widetilde{R}$ *and* $\widetilde{R} \leq O(\frac{\lambda_L}{n})$.
- $\alpha = c \cdot \exp(-b^2/2)$ *and* $\exp(-b^2/2) \leq O(\frac{\lambda_L}{n})$.

then we have $\lambda_{\min}(G_L(W)) \geq \frac{\lambda_L}{2}$.

Proof. We first note that to prove the corollary, it suffices to show that $n\alpha \leq \frac{\lambda_L}{4}$. We analyze two cases.

Case 1: $\alpha = 3R$**.** Suppose $\alpha = 3R$, then the condition translates to $3n\widetilde{R} \leq \frac{\lambda_L}{4}$ which indicates $\widetilde{R} \leq O(\frac{\lambda_L}{n})$.

Case 2: $\alpha = c \cdot \exp(-b^2/2)$**.** Suppose $\alpha = c \cdot \exp(-b^2/2)$, then we have $cn \cdot \exp(-b^2/2) \leq \frac{\lambda_L}{4}$ and $\exp(-b^2/2) \leq O(\frac{\lambda_L}{n})$. ☐

Remark 15.8. We note that the analysis of [SYZ21] focuses on the standard two-layer case of NTK, the reason we can leverage their result is that we can treat the NTK for last layer as a two-layer neural network where the inputs are $h_{i,L-1} \in \mathbb{R}^m$. One can also give include a direct proof of the multi-layer version, which agrees the above lemma and corollary.

15.10 Convergence Analysis of Our Algorithm

In this section, we present a convergence analysis of Algorithm 106. We show that as long as the neural network width is large enough, the convergence of Algorithm 106 is linear, and the weight matrix does not change too much.

15.10.1 Preliminary

We recall the initialization of our neural network.

Definition 15.7 (Initialization)**.** Let $m = m_\ell$ for all $\ell \in [L]$. Let $m_0 = d$. We assume weights are initialized as

- Each entry of weight vector $a \in \mathbb{R}^m$ is i.i.d. sampled from $\{-1, +1\}$ uniformly at random.
- Each entry of weight matrices $W_\ell \in \mathbb{R}^{m \times m}$ sampled from $\mathcal{N}(0, 2/m)$.

Remark 15.9. Later, we will also interpret W_L as sampled from $\mathcal{N}(0, 1)$ and then being scaled by $\sqrt{\frac{2}{m}}$.

We also restate the architecture of our neural network here.

Definition 15.8 (Architecture)**.** Our neural network is a standard L-layer feed-forward neural network, with the activation functions defined as a scaled version of shifted ReLU activation: $\phi(x) = \sqrt{c_b}\mathbf{1}[x > \sqrt{2/m}b]x$, where $c_b := (2(1 - \Phi(b) + b\phi(b)))^{-1/2}$. Here b is a threshold value we will pick later. At last layer, we use a scaled version of a vector with its entry being Rademacher random variables. We define the neural network function $f : \mathbb{R}^{m_0} \to \mathbb{R}$ as

$$f(W, x_i) = a^\top \phi(W_L \phi(W_{L-1} \phi(\dots \phi(W_1 x_i)))).$$

We measure the loss of the neural network via squared-loss function:

$$\mathcal{L}(W) = \frac{1}{2} \sum_{i=1}^{n} (f(x_i) - y_i)^2.$$

We use $f_t : \mathbb{R}^{d \times n} \to \mathbb{R}^n$ denote the prediction of our network:

$$f_t(X) = [f(W(t), x_1), \dots, f(W(t), x_n)]^\top.$$

We state two assumptions here.

Assumption 15.24 (Small Row Norm)**.** Let $t \in \{0, \dots, T\}$ and let $\widetilde{R} \le 1$ be a parameter. We assume

$$\|W_{L,r}(t) - W_{L,r}(0)\|_2 \le \widetilde{R}/\sqrt{m}, \quad \forall r \in [m].$$

Here, $W_{L,r} \in \mathbb{R}^m$ means the r-th row of matrix W_L.

Later, we will invoke this assumption by specifying the choice of $\widetilde{R}$.

Assumption 15.25 (Sparsity)**.** Let $t \in \{0, \dots, T\}$ and let $s \ge 1$ be an integer parameter. We assume

$$\|\Delta D_{i,\ell}\|_0 \le s, \forall \ell \in [L], i \in [n].$$

Later, we will invoke this assumption by specifying the choice of s.

15.10.2 Technical lemmas

We first show that during initialization, by using our shifted ReLU activation, the vector $h_{i,\ell}$ is sparse. Hence, the diagonal matrix $D_{i,\ell}$ is sparse as well.

Lemma 15.26 (Sparse initialization)**.** *Let $\sigma_b(x) = \max\{x - b, 0\}$ be the shifted ReLU activation with threshold $b > 0$. After initialization, with probability*

$$1 - nL \cdot e^{-\Omega(me^{-b^2 m/4})},$$

it holds for all $i \in [n]$ and $\ell \in [L]$,

$$\|h_{i,\ell}\|_0 \le O(m \cdot e^{-b^2 m/4}).$$

Proof. We fix $i \in [n]$ and $\ell \in [L]$, since we will union bound over all i and ℓ at last. Let $u_i \in \mathbb{R}^m$ be a fixed vector and $W_{\ell,r}$ to denote the r-th row of W_ℓ, then by the concentration of Gaussian, we have

$$\Pr[\sigma_b(\langle W_{\ell,r}, u_i\rangle) > 0] = \Pr_{z\sim\mathcal{N}(0,\frac{2}{m})}[z > b] \leq \exp(-b^2 m/4).$$

Let S be the following index set $S := \{r \in [m] : \langle W_{\ell,r}, u_i\rangle > b\}$, the above reasoning means that for the indicator random variable $\mathbf{1}[r \in S]$, we have

$$\mathbb{E}[\mathbf{1}[r \in S]] \leq \exp(-b^2 m/4).$$

Use Bernstein's inequality (Lemma A.4) we have that for all $t > 0$,

$$\Pr[|S| > k + t] \leq \exp(-\frac{t^2/2}{k + t/3}),$$

where $k := m \cdot \exp(-b^2 m/4)$. By picking $t = k$, we have

$$\Pr[|S| > 2k] \leq \exp(\frac{-3k}{8}).$$

Note that $|S|$ is essentially the quantity $\|h_{i,\ell}\|_0$, hence we can union bound over all ℓ and i and with probability at least

$$1 - nL \cdot \exp(-\Omega(m \cdot \exp(-b^2 m/4))),$$

we have $\|h_{i,\ell}\|_0 \leq 2m \cdot \exp(-b^2 m/4)$. $\square$

Remark 15.10. The above lemma shows that by using the shifted ReLU activation, we make sure that all $h_{i,\ell}$ are sparse after initialization. Specifically, we use $k := m \cdot \exp(-b^2 m/4)$ as a sparsity parameter. Later, we might rescale b so that the probability becomes $\exp(-b^2/2)$. We stress that such rescaling does not affect the sparsity of our initial vectors. If we rescale b and choose it as $\sqrt{2\alpha \log m}$, then $k = m^{1-\alpha}$ and hence with high probability, $\|h_{i,\ell}\|_0 \leq O(m^{1-\alpha})$.

As a direct consequence, we note that all initial $D_{i,\ell}$ are k-sparse as well.

We state a lemma that handles the ℓ_2 norm of $h_{i,\ell}$ when one uses *truncated Gaussian distribution* instead. Due to the length and the delicacy of the proof, we defer it to Section 15.11.

Lemma 15.27 (Restatement of Lemma 15.45)**.** *Let $b > 0$ be a fixed scalar. Let the activation function $\phi(x) := \sqrt{c_b}\mathbf{1}[x > \sqrt{2/m}b]x$, where $c_b := (2(1-\Phi(b)+b\phi(b)))^{-1/2}$. Let $\epsilon \in (0,1)$, then over the randomness of $W(0)$, with probability at least*

$$1 - O(nL) \cdot \exp(-\Omega(m \exp(-b^2/2)\epsilon^2/L^2)),$$

we have

$$\|h_{i,\ell}\|_2 \in [1-\epsilon, 1+\epsilon], \forall i \in [n], \ell \in [L].$$

The second lemma handles the consecutive product that appears naturally in the gradient computation. It is useful in analyzing the spectral property of the Gram matrix.

Lemma 15.28 (Variant of Lemma 7.3 in [AZLS19a])**.** *Suppose $m \geq \Omega(nL\log(nL))$, then over the randomness of initializations $W_1(0), \ldots, W_L(0) \in \mathbb{R}^{m\times m}$, for all $i \in [n]$ and $1 \leq a \leq b \leq L$,*

$$\Pr[\|W_b D_{i,b-1} W_{b-1} \ldots D_{i,a} W_a\| \leq O(\sqrt{L})] \geq 1 - e^{-\Omega(k/L^2)}.$$

The proof is similarly to the original proof of the corresponding lemma in [AZLS19a], however we replace the bound on $h_{i,\ell}$ with our Lemma 15.27. We highlight this does not change the bound, merely in expense of a worse probability.

The next several lemmas bound norms after small perturbation.

Lemma 15.29 (Lemma 8.2 in [AZLS19a])**.** *Suppose Assumption 15.24 is satisfied with $\widetilde{R} \leq O(\frac{1}{L^{9/2}\log^3 m})$. With probability at least $1 - e^{-\Omega(m\widetilde{R}^{2/3}L)}$,*

(a) $\Delta g_{i,\ell}$ can be written as $\Delta g_{i,\ell} = \Delta g_{i,\ell,1} + \Delta g_{i,\ell,2}$ where

  - $\|\Delta g_{i,\ell,1}\|_2 \le O(\widetilde{R}L^{3/2})$
  - $\|\Delta g_{i,\ell,2}\|_\infty \le O(\frac{\widetilde{R}L^{5/2}\sqrt{\log m}}{\sqrt{m}})$

(b) $\|\Delta D_{i,\ell}\|_0 \le O(m\widetilde{R}^{2/3}L)$ *and* $\|(\Delta D_{i,\ell})g_{i,\ell}\|_2 \le O(\widetilde{R}L^{3/2})$.

(c) $\|\Delta g_{i,\ell}\|_2, \|\Delta h_{i,\ell}\|_2 \le O(\widetilde{R}L^{5/2}\sqrt{\log m})$.

Remark 15.11. Lemma 15.29 establishes the connection between parameter $\widetilde{R}$ and s of Assumption 15.24 and 15.25. As long as $\widetilde{R}$ is small, then we have $s = O(m\widetilde{R}^{2/3}L)$. Such a relation enables us to pick R to our advantage and ensure the sparsity of $\Delta D_{i,\ell}$ is sublinear in m, and hence the update time per iteration is subquadratic in m.

15.10.3 Bounds on initialization

In the following lemma, we generalize Lemma C.2 in [BPSW21] into multiple-layer neural networks.

Lemma 15.30 (Bounds on initialization, multiple layer version of Lemma C.2 in [BPSW21])**.** *Suppose $m = \Omega(nL\log(nL))$, then we have the following*

- $\Pr[f(W, x_i) = \widetilde{O}(1),\ \forall i \in [n]] \ge 1 - e^{-\Omega(\log^2 n)}$.
- $\Pr[\|J_{L,0,i}\| = O(1),\ \forall i \in [n]] \ge 1 - O(nL) \cdot e^{-\Omega(k/L^2)}$.

Proof. We will prove the two parts of the statement separately.

Part 1: By definition, for any $i \in [n]$, we have

$$f(W, x_i) = a^\top \phi(W_L(\phi(\cdots \phi(W_1 x_i)))).$$

We shall make use of Lemma 15.27 here:

$$\Pr\left[\|h_{i,L}\|_2 \in [0.9, 1.1], \forall i \in [n]\right] \ge 1 - O(nL) \cdot \exp(-\Omega(k/L^2)).$$

Recall that $a \in \mathbb{R}^m$ has each of its entry being a Rademacher random variable, hence it's 1-subgaussian. Use the concentration of subgaussian (Lemma A.5), we know that

$$\Pr[|a^\top h_{i,L}| \geq 1.1t] \leq 2\exp(-\frac{t^2}{2}),$$

setting $t = O(\log^2 n)$, and union bound over all $i \in [n]$, we conclude our desired result.

Part 2: For the last layer, we consider W_L is initialized as follows: each entry is first sampled from $\mathcal{N}(0,1)$, then we scale down W_L by $\frac{\sqrt{2}}{\sqrt{m}}$. This means we can write the output of last layer as $\frac{\sqrt{2}}{\sqrt{m}} W_L h_{i,L-1}$, and therefore, the gradient is $\frac{\sqrt{2}}{\sqrt{m}} D_{i,L} h_{i,L-1} a^\top$. Hence,

$$\begin{aligned} \|J_{L,0,i}\| &= \frac{1}{\sqrt{m}} \|h_{i,L-1} a^\top D_{i,L}\| \\ &\leq \frac{\sqrt{2}}{\sqrt{m}} \|h_{i,L-1}\|_2 \cdot \|D_{i,L} a\|_2 \\ &= O(1). \end{aligned}$$

The last step follows from the fact that $\|D_{i,L} a\|_2 \leq O(\sqrt{m})$ and $\|h_{i,L-1}\|_2 \leq 1.1$ with probability at least $1 - O(nL) \cdot \exp(-\Omega(k/L^2))$. $\square$

15.10.4 Bounds on small perturbation

In the following, we generalize the Lemma C.4 in [BPSW21] into multiple layer neural network. We use the interpretation that W_L is generated from $\mathcal{N}(0,1)$ and scaled by $\sqrt{\frac{2}{m}}$ in our proof.

Lemma 15.31 (multiple layer version of Lemma C.4 in [BPSW21])**.** *Suppose $m = \Omega(nL\log(nL))$, then over the random initialization of*

$$W(0) = \{W_1(0), W_2(0), \cdots W_L(0)\},$$

the following holds with probability at least $1 - nL \cdot e^{-\log^2 m}$, for any set of weight W_L satisfying for each $r \in [m]$,

$$\|W_{L,r} - W_{L,r}(0)\|_2 \leq R/\sqrt{m},$$

- $\|W_L - W_L(0)\|_F \leq R$.
- $\|J_{W_L,x_i} - J_{W_L(0),x_i}\|_2 = \widetilde{O}(R^{1/2}/m^{1/4})$.
- $\|J_{W_L} - J_{W_L(0)}\|_F = \widetilde{O}(n^{1/2}R^{1/2}/m^{1/4})$.
- $\|J_{W_L}\|_F = \widetilde{O}(n^{1/2})$.

Proof. **Part 1.** Note that

$$\begin{aligned}\|W_L - W_L(0)\|_F^2 = &\sum_{r=1}^{m} \|W_{L,r} - W_{L,r}(0)\|_2^2 \\ \leq &\ m \cdot R^2/m \\ = &\ R^2.\end{aligned}$$

Taking square root yields our desired result.

Part 2. To simplify the notation, we ignore the subscripts i below. We have

$$\begin{aligned}\|J_{W_L,x} - J_{W_L(0),x}\|^2 &= \frac{2}{m}\|(D_L(0) + \Delta D_L)ah_L^\top - D_L(0)ah_L^\top\|^2 \\ &= \frac{2}{m}\|\Delta D_L a h_L^\top\|^2 \\ &= \frac{2}{m}\|\Delta D_L a h_L^\top\|_F^2 \\ &= \frac{2}{m}\sum_{r\in[m]} a_r^2 \cdot h_{L,r}^2 \cdot |\mathbf{1}[\langle W_{L,r}, h_L\rangle \geq b] - \mathbf{1}[\langle W_{L,r}(0), h_L\rangle \geq b]| \\ &= O(\frac{1}{m})\sum_{r\in[m]} |\mathbf{1}[\langle W_{L,r}, h_L\rangle \geq b] - \mathbf{1}[\langle W_{L,r}(0), h_L\rangle \geq b]|.\end{aligned}$$

where we use $\Delta D_L a h_L^\top$ is a rank 1 matrix in the third step.

Let $s_r := |\mathbf{1}[\langle W_{L,r}, h_L\rangle \geq b] - \mathbf{1}[\langle W_{L,r}(0), h_L\rangle \geq b]|$ and define the event E_r as

$$E_r = \Big\{\|W_{L,r} - W_{L,r}(0)\|_2 \leq R/\sqrt{m}, \quad \mathbf{1}[\langle W_{L,r}, h_L\rangle \geq b] \neq \mathbf{1}[\langle W_{L,r}(0), h_L\rangle \geq b]\Big\}.$$

It is not hard to see that event E_r happens if and only if

$$W_{L,r}(0)^\top h_L \in [b - \|h_L\|_2 R/\sqrt{m}, b + \|h_L\|_2 R/\sqrt{m}].$$

By the anti-concentration of Gaussian distribution (Lemma A.7), we have

$$\mathbb{E}[s_r] = \Pr[E_r = 1] \le \frac{4}{5} R/\sqrt{m}.$$

We have

$$\begin{aligned}\Pr\Big[\sum_{r=1}^m s_r \ge (t + \frac{4}{5})\|h_L\| R\sqrt{m}\Big] \le \ & \Pr\Big[\sum_{r=1}^m (s_r - \mathbb{E}[s_r]) \ge t\|h_L\|_2 R\sqrt{m}\Big] \\ & \le 2\exp(-\frac{2t^2R^2\|h_L\|_2^2 m^2}{m^2}) \\ & = 2\exp(-2t^2R^2\|h_L\|_2^2) \\ & \le 2\exp(-t^2),\end{aligned}$$

where the first step follows from our above analysis, we use Lemma A.3 in the second step, and we use both $\|h_L\|_2^2 \ge 0.5$ and $R \ge 1$ in the final step.

Set $t = \log m$ and by union bound over i, we have with probability at least $1 - n \cdot e^{-\log^2 m}$,

$$\begin{aligned}\|J_{W_L,x_i} - J_{W_L(0),x_i}\|^2 = & \ \frac{1}{m}\sum_{r=1}^m s_r \\ \le & \ \frac{1}{m}\widetilde{O}(R\sqrt{m}) \\ = & \ \widetilde{O}(\frac{R}{\sqrt{m}}).\end{aligned}$$

Taking square root yields our desired result.

Part 3. Note that the squared Frobenious norm is just the sum of all squared ℓ_2 norm of rows, hence

$$\|J_{W_L} - J_{W_L(0)}\|_F \le \widetilde{O}(n^{1/2}R^{1/2}/m^{1/4}).$$

Part 4. We will prove by triangle inequality:

$$\begin{aligned}\|J_{W_L}\|_F \le & \ \|J_{W_L(0)}\|_F + \|J_{W_L} - J_{W_L(0)}\|_F \\ \le & \ \widetilde{O}(n^{1/2}) + \widetilde{O}(n^{1/2}R^{1/2}/m^{1/4}) \\ = & \ \widetilde{O}(n^{1/2}).\end{aligned}$$

Note that, in the final step, we use both the choice of R (see Def. 15.10) and m (see Def. 15.11). $\square$

15.10.5 Putting it all together

In this section, we will prove the following core theorem that analyzes the convergence behavior of Algorithm 106:

Theorem 15.32 (Formal version of Theorem 15.1)**.** *Suppose the neural network width satisfies $m = \Omega(\lambda_L^{-2} n^2 L^2)$, then over the randomness of the initialization of the neural network and the randomness of the algorithm, Algorithm 106 satisfies*

$$\Pr[\|f_{t+1} - y\|_2 \leq \frac{1}{3}\|f_t - y\|_2] \geq 1 - \exp(-\Omega(\log^2 n)).$$

Before moving on, we introduce several definitions and prove some useful facts related to them.

Definition 15.9 (function J)**.** We define

$$\mathsf{J}_\ell(Z_1, \ldots, Z_L)_i := \ D_{i,\ell}(Z_\ell) \prod_{k=\ell+1}^{L} Z_k^\top D_{i,k}(Z_k) a(h_i(Z_1, \ldots, Z_{\ell-1}))^\top \quad \in \mathbb{R}^{m_\ell \times m_{\ell-1}}$$

where

$$\begin{aligned} D_{i,\ell}(Z_\ell) &:= \mathrm{diag}(\phi'(Z_\ell h_i(Z_1, \ldots, Z_{\ell-1}))), && \in \mathbb{R}^{m_\ell \times m_\ell} \\ h_i(Z_1, \ldots, Z_{\ell-1}) &:= \phi(Z_{\ell-1}(\phi(Z_{\ell-2} \cdots (\phi(Z_1 x_i))))) && \in \mathbb{R}^{m_{\ell-1}} \end{aligned}$$

Fact 15.33. *Let J_ℓ denote the function be defined as Definition 15.9. For any $t \in \{0, \ldots, T\}$, we have*

$$f_{t+1} - f_t = \left(\int_0^1 \mathsf{J}_L((1-s)W(t) + sW(t+1)) \mathsf{d}s \right)^\top \cdot \mathrm{vec}(W_L(t+1) - W_L(t)),$$

Proof. For $i \in [n]$, consider the i-th coordinate.

$$\begin{aligned}
(f_{t+1} - f_t)_i = & \int_0^1 f((1-s)W(t) + sW(t+1), x_i)' \mathsf{d}s \\
= & \int_0^1 \left(\frac{\partial f}{\partial W_L}((1-s)W(t) + sW(t+1), x_i) \right)^\top \cdot \mathrm{vec}(W_L(t+1) - W_L(t)) \mathsf{d}s \\
= & \left(\int_0^1 \mathsf{J}_L((1-s)W(t) + sW(t+1))_i \mathsf{d}s \right)^\top \cdot \mathrm{vec}(W_L(t+1) - W_L(t)),
\end{aligned}$$

Thus, we complete the proof. $\square$

Fact 15.34. *For any $t \in \{0, \dots, T\}$, we have $\mathsf{J}_\ell(W_1(t), \dots, W_L(t)) = J_{\ell,t}$.*

Proof. In order to simplify the notation, we drop the term t below.

We note that for $i \in [n]$, the i-th row of matrix $J_{\ell,t}$ is defined as

$$D_{i,\ell}(\prod_{k=\ell+1}^{L} W_k^\top D_{i,k}) a h_{i,\ell-1}^\top,$$

where

$$\begin{aligned}
D_{i,\ell} = & \mathrm{diag}(\phi'(W_\ell h_{i,\ell-1})), \\
h_{i,\ell-1} = & \phi(W_{\ell-1}(\phi(W_{\ell-2} \dots (\phi(W_1 x_i))))),
\end{aligned}$$

this is essentially the same as $h_i(W_1, \dots, W_{\ell-1})$ and $D_{i,\ell}(W_\ell)$. This completes the proof. $\square$

We state the range we require for parameter $\widetilde{R}$ and R:

Definition 15.10. We choose $\widetilde{R}$ so that

$$\frac{2}{\sqrt{m}} \cdot \frac{n}{\lambda_L} \leq \widetilde{R} \leq \min\{\frac{1}{L^{4.5} \log^3 m}, \frac{\lambda_L}{n}\}.$$

Recall that R is the scale-up version of $\widetilde{R}$, hence

$$\frac{n}{\lambda_L} \leq R \leq \min\{\frac{1}{L^{4.5} \log^3 m}, \frac{\lambda_L}{n}\} \cdot \sqrt{m}.$$

Remark 15.12. Recall that the sparsity parameter s is directly related to $\widetilde{R}$: $s = O(m\widetilde{R}^{2/3}L)$, hence to ensure the sparsity is small, we shall pick $\widetilde{R}$ as small as possible.

Next, we pick the value of m:

Definition 15.11. We choose m to be

$$m \geq \Omega(n^4 L \lambda_L^{-4}).$$

We use induction to prove the following two claims recursively.

Definition 15.12 (Induction hypothesis 1)**.** Let $t \in [T]$ be a fixed integer. We have

$$\|W_{L,r}(t) - W_{L,r}(0)\|_2 \leq R/\sqrt{m}$$

holds for any $r \in [m]$.

Definition 15.13 (Induction Hypothesis 2)**.** Let $t \in [T]$ be a fixed integer. We have

$$\|f_t - y\|_2 \leq \frac{1}{3}\|f_{t-1} - y\|_2.$$

Suppose the above two claims hold up to t, we prove they continue to hold for time $t+1$. The second claim is more delicate, we are going to prove it first and we define

$$J_{\ell,t,t+1} := \int_0^1 \mathsf{J}_\ell\Big((1-s)W_t + sW_{t+1}\Big)\mathsf{d}s,$$

where J_ℓ is defined as Definition 15.9.

Lemma 15.35. *Let* $g_L^\star := (J_{L,t}J_{L,t}^\top)^{-1}(f_t - y)$*. We have*

$$\begin{aligned}\|f_{t+1} - y\|_2 \leq\ & \|f_t - y - J_{L,t}J_{L,t}^\top g_{L,t}\|_2 \\ & + \|(J_{L,t} - J_{L,t,t+1})J_{L,t}^\top g_L^\star\|_2 \\ & + \|(J_{L,t} - J_{L,t,t+1})J_{\ell,t}^\top(g_{L,t} - g_L^\star)\|_2. \end{aligned} \tag{15.7}$$

Proof. Consider the following computation:

$$\begin{aligned}
&\|f_{t+1} - y\|_2 \\
=& \|f_t - y + (f_{t+1} - f_t)\|_2 \\
=& \|f_t - y + J_{L,t,t+1} \cdot \mathrm{vec}(W_{L,t+1} - W_{L,t})\|_2 \\
=& \|f_t - y - J_{L,t,t+1} \cdot J_{L,t}^\top g_{L,t}\|_2 \\
=& \|f_t - y - J_{L,t}J_{L,t}^\top g_{L,t} + J_{L,t}J_{L,t}^\top g_{L,t} - J_{L,t,t+1}J_{L,t}^\top g_{L,t}\|_2 \\
\leq& \|f_t - y - J_{L,t}J_{L,t}^\top g_{L,t}\|_2 + \|(J_{L,t} - J_{L,t,t+1})J_{L,t}^\top g_{L,t}\|_2 \\
\leq& \|f_t - y - J_{L,t}J_{L,t}^\top g_{L,t}\|_2 + \|(J_{L,t} - J_{L,t,t+1})J_{L,t}^\top g_L^\star\|_2 + \|(J_{L,t} - J_{L,t,t+1})J_{L,t}^\top (g_{L,t} - g_L^\star)\|_2,
\end{aligned}$$

The second step follows from the definition of $J_{L,t,t+1}$ and simple calculus. □

Claim 15.36 (1st term in Eq. (15.7))**.** *We have*

$$\|f_t - y - J_{L,t}J_{L,t}^\top g_{L,t}\|_2 \leq \frac{1}{9}\|f_t - y\|_2.$$

Proof. We have

$$\begin{aligned}
\|f_t - y - J_{L,t}J_{L,t}^\top g_{L,t}\|_2 &\leq \epsilon_0\|f_t - y\|_2 \\
&\leq \frac{1}{9}\|f_t - y\|_2, \qquad (15.8)
\end{aligned}$$

since $g_{L,t}$ is an ϵ_0 ($\epsilon_0 \leq \frac{1}{9}$) approximate solution to the regression problem

$$\min_g \|J_{L,t}J_{L,t}^\top g - (f_t - y)\|_2.$$

□

Claim 15.37 (2nd term in Eq. (15.7))**.** *We have*

$$\|(J_{L,t} - J_{L,t,t+1})J_{L,t}^\top g_L^\star\|_2 \leq \frac{1}{9}\|f_t - y\|_2.$$

Proof. We bound the second term in Eq. (15.7) as follows:

$$\begin{aligned}
\|(J_{L,t} - J_{L,t,t+1})J_{L,t}^\top g_L^\star\|_2 &\leq \|J_{L,t} - J_{L,t,t+1}\| \cdot \|J_{L,t}^\top g_L^\star\|_2 \\
&= \|J_{L,t} - J_{L,t,t+1}\| \cdot \|J_{L,t}^\top (J_{L,t}J_{L,t}^\top)^{-1} \cdot (f_t - y)\|_2 \\
&\leq \|J_{L,t} - J_{L,t,t+1}\| \cdot \|J_{L,t}^\top (J_{L,t}J_{L,t}^\top)^{-1}\| \cdot \|f_t - y\|_2. \qquad (15.9)
\end{aligned}$$

We bound these term separately.

For the first term in Eq. (15.9),

$$\begin{aligned}
\|J_{L,t} - J_{L,t,t+1}\| &= \left\| \mathsf{J}_L(W_t) - \int_0^1 \mathsf{J}_L((1-s)W_t + sW_{t+1})\mathsf{d}s \right\| \\
&\leq \int_0^1 \|\mathsf{J}_L(W_t) - \mathsf{J}_L((1-s)W_t + sW_{t+1})\| \, \mathsf{d}s \\
&\leq \int_0^1 \|\mathsf{J}_L(W_t) - \mathsf{J}_L(W_0)\| + \|\mathsf{J}_L(W_0) - \mathsf{J}_L((1-s)W_t + sW_{t+1})\| \, \mathsf{d}s \\
&\leq \|\mathsf{J}_L(W_t) - \mathsf{J}_L(W_0)\| + \int_0^1 \|\mathsf{J}_L(W_0) - \mathsf{J}_L((1-s)W_t + sW_{t+1})\| \, \mathsf{d}s \\
&\leq \widetilde{O}(n^{1/2} R^{1/2} / m^{1/4}), \qquad (15.10)
\end{aligned}$$

where by Fact 15.34, we know $\|\mathsf{J}_L(W_t) - \mathsf{J}_L(W_0)\| = \|J_{W_L(t)} - J_{W_L(0)}\| \leq \widetilde{O}(n^{1/2} R^{1/2} / m^{1/4})$, we use Lemma 15.31 in the last inequality.

For the term $\int_0^1 \|\mathsf{J}_L(W_0) - \mathsf{J}_L((1-s)W_t + sW_{t+1})\| \, \mathsf{d}s$, we analyze the following:

$$\begin{aligned}
& \|(1-s) \cdot \mathrm{vec}(W_L(t)) + s \cdot \mathrm{vec}(W_L(t+1)) - \mathrm{vec}(W_L(0))\|_2 \\
\leq & (1-s) \cdot \|\mathrm{vec}(W_L(t)) - \mathrm{vec}(W_L(0))\|_2 + s \cdot \|\mathrm{vec}(W_L(t+1)) - \mathrm{vec}(W_L(0))\|_2 \\
= & (1-s) \cdot \|W_L(t) - W_L(0)\|_F + s \cdot \|W_L(t+1) - W_L(0)\|_F \\
\leq & O(R).
\end{aligned}$$

This means the perturbation of $(1-s)W_L(t) + sW_L(t+1)$ with respect to $W_L(0)$ is small, hence $\|\mathsf{J}_L(W_0) - \mathsf{J}_L((1-s)W_t + sW_{t+1})\| = \widetilde{O}(n^{1/2} R^{1/2} / m^{1/4})$.

Furthermore, we have

$$\|J_{L,t}^\top (J_{L,t} J_{L,t}^\top)^{-1}\| = \frac{1}{\sigma_{\min}(J_{L,t}^\top)} \leq \sqrt{2/\lambda_L}, \qquad (15.11)$$

where the second inequality follows from $\sigma_{\min}(J_{L,t}) = \sqrt{\lambda_{\min}(J_{L,t} J_{L,t}^\top)} \geq \sqrt{\lambda_L / 2}$ (see Lemma 15.23).

Combining Eq. (15.9), (15.10) and (15.11), we have

$$\begin{aligned}
\|(J_{L,t} - J_{L,t,t+1}) J_{L,t}^\top g_L^\star\|_2 &\leq \widetilde{O}(n^{1/2} R^{1/2} / m^{1/4}) \cdot \lambda_L^{-1/2} \cdot \|f_t - y\|_2 \\
&\leq \frac{1}{9} \|f_t - y\|_2, \qquad (15.12)
\end{aligned}$$

where the last step follows from choice of m (Definition 15.11). $\square$

Claim 15.38 (3rd term in Eq. (15.7))**.** *We have*

$$\|(J_{L,t} - J_{L,t,t+1})J_{L,t}^\top (g_{L,t} - g_L^\star)\|_2 \leq \frac{1}{9}\|f_t - y\|_2$$

Proof. We can show

$$\|(J_{L,t} - J_{L,t,t+1})J_{L,t}^\top (g_{L,t} - g_L^\star)\|_2 \leq \|J_{L,t} - J_{L,t,t+1}\| \cdot \|J_{L,t}^\top\| \cdot \|g_{L,t} - g_L^\star\|_2. \quad (15.13)$$

Moreover, one has

$$\begin{aligned}\frac{\lambda}{2}\|g_{L,t} - g_L^\star\|_2 &\leq \lambda_{\min}(J_{L,t}J_{L,t}^\top) \cdot \|g_{L,t} - g_L^\star\|_2 \\ &\leq \|J_{L,t}J_{L,t}^\top g_{L,t} - J_{L,t}J_{L,t}^\top g_L^\star\|_2 \\ &= \|J_{L,t}J_{L,t}^\top g_{L,t} - (f_t - y)\|_2 \\ &\leq \frac{\sqrt{\lambda_L/n}}{2} \cdot \|f_t - y\|_2. \end{aligned} \quad (15.14)$$

The first step comes from $\lambda_{\min}(J_{L,t}J_{L,t}^\top) = \lambda_{\min}(G_{L,t}) \geq \lambda_L/2$ (see Lemma 15.23). The last step follows from $g_{L,t}$ is an ϵ_0 ($\epsilon_0 \leq \sqrt{\lambda_L/n}$) approximate solution to the regression.

Consequently, we have

$$\begin{aligned}\|(J_{L,t} - J_{L,t,t+1})J_{L,t}^\top (g_{L,t} - g_L^\star)\|_2 &\leq \|J_{L,t} - J_{L,t,t+1}\| \cdot \|J_{L,t}^\top\| \cdot \|g_{L,t} - g_L^\star\|_2 \\ &\leq \widetilde{O}(n^{1/2}R^{1/2}/m^{1/4}) \cdot \widetilde{O}(n^{1/2}) \cdot \frac{2}{\sqrt{n\lambda_L}} \cdot \|f_t - y\|_2 \\ &= \widetilde{O}(\frac{n^{1/2}R^{1/2}}{m^{1/4}\lambda_L^{1/2}}) \cdot \|f_t - y\|_2.\end{aligned}$$

Note that, for the 2nd step, it follows from Eq. (15.10) and (15.14) and the fact that $\|J_{L,t}\| \leq O(\sqrt{n})$ (see Lemma 15.31).

Finally, we have

$$\begin{aligned}\|(J_{L,t} - J_{L,t,t+1})J_{L,t}^\top (g_{L,t} - g_L^\star)\|_2 &\leq \widetilde{O}(\frac{n^{1/2}R^{1/2}}{m^{1/4}\lambda_L^{1/2}}) \cdot \|f_t - y\|_2 \quad (15.15) \\ &\leq \frac{1}{9}\|f_t - y\|_2. \quad (15.16)\end{aligned}$$

The last step follows from choice of m (Definition 15.11).

$\square$

Lemma 15.39 (Putting it all together)**.** *We have*

$$\|f_{t+1} - y\|_2 \leq \frac{1}{3}\|f_t - y\|_2. \tag{15.17}$$

Proof. Combining Eq. (15.7), (15.8), (15.12), and (15.15), we have proved the second claim, i.e.,

$$\|f_{t+1} - y\|_2 \leq \frac{1}{3}\|f_t - y\|_2.$$

$\square$

15.10.6 Bounds on the movement of weights

Lemma 15.40. *Let R be chosen as in Definition 15.10, then the following holds:*

$$\|W_{L,r}(t+1) - W_{L,r}(0)\|_2 \leq R/\sqrt{m}.$$

Proof. First, we have

$$\begin{aligned}
\|g_{L,t}\|_2 &\leq \|g_L^\star\|_2 + \|g_{L,t} - g_L^\star\|_2 \\
&= \|(J_{L,t}J_{L,t}^\top)^{-1}(f_t - y)\|_2 + \|g_{L,t} - g_L^\star\|_2 \\
&\leq \|(J_{L,t}J_{L,t}^\top)^{-1}\| \cdot \|(f_t - y)\|_2 + \|g_{L,t} - g_L^\star\|_2 \\
&\leq \frac{1}{\lambda_L} \cdot \|f_t - y\|_2 + \frac{1}{\sqrt{n\lambda_L}} \cdot \|f_t - y\|_2 \\
&\lesssim \frac{1}{\lambda_L} \cdot \|f_t - y\|_2 \qquad (15.18)
\end{aligned}$$

where the third step is owing to Eq. (15.14), and the final step is due to the fact that $1/\sqrt{n\lambda_L} \leq 1/\lambda_L$.

Then

$$\begin{aligned}
\|W_{L,r}(k+1) - W_{L,r}(k)\|_2 &= \Big\| \sum_{i=1}^n \frac{1}{\sqrt{m}} a_r h_{i,L-1}^\top \mathbf{1}[\langle W_{L,r}(k), h_{i,L-1}\rangle \geq 0] g_{L,k,i} \Big\|_2 \\
&\leq O(\frac{1}{\sqrt{m}}) \sum_{i=1}^n |g_{L,k,i}| \\
&\leq O(\frac{\sqrt{n}}{\sqrt{m}}) \cdot \|g_{L,k}\|_2 \\
&\leq O(\frac{\sqrt{n}}{\sqrt{m}}) \cdot \frac{1}{\lambda_L} \cdot \|f_k - y\|_2 \\
&\leq O(\frac{n^{1/2}}{2^k \lambda_L m^{1/2}}) \cdot \|f_0 - y\|_2 \\
&\leq \widetilde{O}(\frac{n}{2^k \lambda_L m^{1/2}}).
\end{aligned}$$

The first step follows from the update rule, the second step is by triangle inequality, the third step uses the fact the ℓ_1 norm of a vector is upper bounded by $\sqrt{n}$ times the ℓ_2 norm, the fourth step is by Eq. (15.18), and the last step is by each entry of f_0 and y is of order $\widetilde{O}(1)$.

Consequently, we have

$$\begin{aligned}
\|W_{L,r}(t+1) - W_{L,r}(0)\|_2 &\leq \sum_{k=0}^t \|W_{L,r}(k+1) - W_{L,r}(k)\|_2 \\
&\leq \sum_{k=0}^t \widetilde{O}(\frac{n}{2^k \lambda_L m^{1/2}}) \\
&\leq \widetilde{O}(\frac{n}{\lambda_L m^{1/2}}).
\end{aligned}$$

By the choice of R (Definition 15.10), we know this is upper bounded by $R/\sqrt{m}$. This concludes our proof.

$\square$

15.11 Bounds on the Intermediate Layer Output with Shifted ReLU

In this section, we prove a technical lemma (Lemma 15.27) involving truncated gaussian distribution, which correlates to the shifted ReLU activation we use.

Definition 15.14 (Truncated Gaussian distribution)**.** Suppose $X \sim \mathcal{N}(0, \sigma^2)$. Let $b \in \mathbb{R}$. Then, we say a random variable Y follows from a truncated Gaussian distribution $\mathcal{N}_b(0, \sigma^2)$ if $Y = X | X \geq b$. The probability density function for $\mathcal{N}_b(0, \sigma^2)$ is as follows:

$$f(y) = \frac{1}{\sigma(1 - \Phi(b/\sigma))} \cdot \frac{1}{\sqrt{2\pi}} e^{-y^2/(2\sigma^2)} \quad y \in [b, \infty),$$

where $\Phi(\cdot)$ is the standard Gaussian distribution's CDF.

Fact 15.41 (Properties of truncated Gaussian distribution)**.** *For $b \in \mathbb{R}$, suppose $X \sim \mathcal{N}_b(0, \sigma^2)$. Let $\beta := b/\sigma$. Then, we have*

- $\mathbb{E}[X] = \frac{\sigma\phi(\beta)}{1-\Phi(\beta)}$, *where* $\phi(x) := \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$.
- $\mathbf{Var}[X] = \sigma^2(1 + \beta\phi(\beta)/(1 - \Phi(\beta)) - (\phi(\beta)/(1 - \Phi(\beta)))^2)$.
- $X/\sigma \sim \mathcal{N}_{b/\sigma}(0, 1)$.
- *When* $\sigma = 1$, X *is* $C(b + 1)$*-subgaussian, where* $C > 0$ *is an absolute constant.*

Lemma 15.42 (Concentration inequality for b-truncated chi-square distribution)**.** *For $b \in \mathbb{R}$, $n > 0$, let $X \sim \chi^2_{b,n}$; that is, $X = \sum_{i=1}^n Y_i^2$ where $Y_1, \ldots, Y_n \sim \mathcal{N}_b(0, 1)$ are independent b-truncated Gaussian random variables. Then, there exist two constants C_1, C_2 such that for any $t > 0$,*

$$\Pr\left[\left|X - n(1 + \frac{b\phi(b)}{1 - \Phi(b)})\right| \geq nt\right] \leq \exp\left(-C_1 nt^2/b^4\right) + \exp\left(-C_2 nt/b^2\right).$$

In particular, we have

$$\Pr\left[|X - n(1 + b(b+1))| \geq t\right] \leq \exp\left(-C_1 t^2/(nb^4)\right) + \exp\left(-C_2 t/b^2\right).$$

Proof. Since we know that $Y_i \sim \mathcal{N}_b(0,1)$ is $C(b+1)$-subgaussian, it implies that Y_i^2 is a sub-exponential random variable with parameters $(4\sqrt{2}C^2(b+1)^2, 4C^2(b+1)^2)$. Hence, by the standard concentration of sub-exponential random variables, we have

$$\Pr\left[\left|\sum_{i=1}^n Y_i^2 - n\,\mathbb{E}[Y_i^2]\right| \geq nt\right] \leq \begin{cases} 2\exp\left(-\frac{nt^2}{2\cdot 32C^4(b+1)^4}\right) & \text{if } nt \leq 8C^2(b+1)^2 \\ 2\exp\left(-\frac{nt}{2\cdot 4C^2(b+1)^2}\right) & \text{otherwise} \end{cases}$$
$$\leq 2\exp\left(-C_1 nt^2/b^4\right) + 2\exp\left(-C_2 nt/b^2\right).$$

$\square$

Fact 15.43. *Let $h \in \mathbb{R}^p$ be fixed vectors and $h \neq 0$, let $b > 0$ be a fixed scalar, $W \in \mathbb{R}^{m\times p}$ be random matrix with i.i.d. entries $W_{i,j} \sim \mathcal{N}(0, \frac{2}{m})$ and vector $v \in \mathbb{R}^m$ defined as $v_i = \phi((Wh)_i) = \mathbf{1}[(Wh)_i \geq b](Wh)_i$. Then*

- *$|v_i|$ follows i.i.d. from the following distribution: with probability $1-e^{-b^2m/(4\|h\|^2)}$, $|v_i| = 0$, and with probability $e^{-b^2m/(4\|h\|^2)}$, $|v_i|$ follows from truncated Gaussian distribution $\mathcal{N}_b(0, \frac{2}{m}\|h\|_2^2)$.*
- *$\frac{m\|v\|_2^2}{2\|h\|_2^2}$ is in distribution identical to $\chi^2_{b',\omega}$ (b'-truncated chi-square distribution of order ω) where ω follows from binomial distribution $\mathcal{B}(m, e^{-b^2m/(4\|h\|^2)})$ and $b' = \frac{\sqrt{m/2}}{\|h\|_2}b$.*

Proof. We assume each vector W_i is generated by first generating a gaussian vector $g \sim \mathcal{N}(0, \frac{2}{m}I)$ and then setting $W_i = \pm g$ where the sign is chosen with half-half probability.

Now, $|\langle W_i, h\rangle| = |\langle g, h\rangle|$ only depends on g, and is in distribution identical to $\mathcal{N}_b(0, \frac{2}{m}\|h\|_2^2)$.

Next, after the sign is determined, the indicator $\mathbf{1}[(W_ih)_i \geq b]$ is 1 with probability $e^{-b^2m/(4\|h\|^2)}$ and 0 with probability $1 - e^{-b^2m/(4\|h\|^2)}$.

Therefore, $|v_i|$ satisfies the aforementioned distribution.

As for $\|v\|_2^2$, letting $\omega \in \{0,1,\ldots,m\}$ be the variable indicates how many indicators are 1, then $\omega \sim \mathcal{B}(m, e^{-b^2 m/(4\|h\|^2)})$ and $\frac{m\|v\|_2^2}{2\|h\|_2^2} \sim \chi^2_{b',\omega}$, where $b' = \frac{\sqrt{m/2}}{\|h\|_2} b$. $\square$

Fact 15.44 (Gaussian tail bound)**.** *For any $b > 0$, we have*

$$\frac{e^{-b^2/2}}{C(b+1)} \leq 1 - \Phi(b) \leq e^{-b^2/2},$$

where C is an absolute constant.

We prove a truncated Gaussian version of Lemma 7.1 of [AZLS19a].

Lemma 15.45. *Let $b > 0$ be a fixed scalar. Let the activation function be defined as*

$$\phi(x) := \sqrt{c_b}\mathbf{1}[x > \sqrt{2/m}b]x,$$

where

$$c_b := (2(1 - \Phi(b) + b\phi(b)))^{-1}.$$

Let $\epsilon \in (0,1)$, then over the randomness of $W(0)$, with probability at least

$$1 - O(nL) \cdot \exp(-\Omega(m \exp(-b^2/2)\epsilon^2/L^2)),$$

we have

$$\|h_{i,\ell}\|_2 \in [1-\epsilon, 1+\epsilon], \forall i \in [n], \ell \in [L].$$

Proof. We only prove for a fixed $i \in [n]$ and $\ell \in \{0,1,2,\ldots,L\}$ because we can apply union bound at the end. Below, we drop the subscript i for notational convenience, and write $h_{i,\ell}$ and x_i as h_ℓ and x respectively.

According to Fact 15.43, fixing any $h_{\ell-1} \neq 0$ and letting W_ℓ be the only source of randomness, we have

$$\frac{m}{2}\|h_\ell\|_2^2 \sim \chi^2_{b/\|h\|_2,\omega}, \quad \text{with} \quad \omega \sim \mathcal{B}(m, 1 - \Phi(b')),$$

where $b' := b/\|h_{\ell-1}\|_2$.

We first consider the $\ell = 1$ case. Then, we have $\|h_{\ell-1}\|_2 = 1$, and $b' = b$. Let $P_b := 1 - \Phi(b)$. By Chernoff bound, for any $\delta \in (0, 1)$, we have

$$\Pr[\omega \in (1 \pm \delta) m P_b] \geq 1 - \exp(-\Omega(\delta^2 P_b m)).$$

In the following proof, we condition on this event. By Fact 15.44,

$$\omega \in (1 \pm \delta) P_b m \iff \omega \in \left[(1-\delta) \frac{e^{-b^2/2}}{C(b+1)} m, (1+\delta) \exp(-b^2/2) m \right].$$

By Lemma 15.42, we have

$$\Pr\left[\left| \frac{m}{2} \|h_1\|_2^2 - \omega \left(1 + \frac{b\phi(b)}{P_b} \right) \right| > t \right] \leq \exp\left(-\Omega(t^2/(\omega b^4)) \right) + \exp\left(-\Omega(t/b^2) \right)$$

Note that

$$\omega \left(1 + \frac{b\phi(b)}{P_b} \right) \in (1 \pm \delta) m P_b + (1 \pm \delta) m P_b \cdot \frac{b\phi(b)}{P_b} = (1 \pm \delta)(P_b + b\phi(b)) \cdot m.$$

Let $c_b^{-1} := 2(P_b + b\phi(b))$ be the normalization constant. Then, we have

$$\Pr[|c_b \|h_1\|_2^2 - (1 \pm \delta)| > 2tc_b/m] \leq \exp\left(-\Omega(t^2/(\omega b^4)) \right) + \exp\left(-\Omega(t/b^2) \right).$$

We want $2tc_b/m = O(\delta)$, i.e., $t = O(\delta c_b^{-1} m)$. Then, we have $\omega t = m^{\Omega(1)} > b^2$. Hence, by Lemma 15.42, we actually have

$$\Pr[|c_b \|h_1\|_2^2 - (1 \pm \delta)| > O(\delta)] \leq \exp\left(-\Omega(\delta m/(c_b b^2)) \right).$$

By taking $\delta = \epsilon/L$, we get that

$$\|h_1\|_2^2 \in [1 - \epsilon/L, 1 + \epsilon/L]$$

holds with probability

$$\geq 1 - \exp(-\Omega(\epsilon^2 P_b m/L^2)) - \exp\left(-\Omega(\epsilon m/(c_b b^2 L)) \right) \geq 1 - \exp(-\Omega(\epsilon^2 P_b m/L^2)),$$

where the last step follows from $\frac{1}{c_b b^2} = \frac{P_b + b\phi(b)}{b^2} = \Theta(P_b)$.

We can inductively prove the $\ell > 1$ case. Since the blowup of the norm of h_1 is $1 \pm \epsilon/L$, the concentration bound is roughly the same for h_ℓ for $\ell \geq 2$. Thus, by carefully choosing the parameters, we can achieve $\|h_\ell\|_2^2 \in [(1-\epsilon/L)^\ell, (1+\epsilon/L)^\ell]$ with high probability.

In this end, by a union bound over all the layers $\ell \in [L]$ and all the input data $i \in [n]$, we get that

$$\|h_{i,\ell}\|_2 \in [1-\epsilon, 1+\epsilon]$$

with probability at least

$$1 - O(nL)\exp(-\Omega(\epsilon^2 P_b m / L^2)),$$

which completes the proof of the lemma.

$\square$

Chapter 16: Privacy Distributed Learning: A Theoretical Analysis of InstaHide's Security

16.1 Introduction

Collaboratively training neural networks based on sensitive data is appealing in many AI applications, such as healthcare, fraud detection, and virtual assistants. How to train neural networks without compromising data confidentiality and prediction accuracy has become a common research goal in both academia and industry. [HSLA20b] recently proposed an approach called InstaHide for image classification. The key idea is to train the model on a dataset where (1) each image is a mix of k_{priv} private images and k_{pub} public images, and (2) each pixel is independently sign-flipped after the mixing. Instahide shows promising prediction accuracy on MNIST, CIFAR-10, CIFAR-100, and ImageNet datasets. TextHide [HSC+20] applies InstaHide's idea to text datasets and achieves promising results on natural language processing tasks.

To understand the security aspect of InstaHide in realistic deployment scenarios, InstaHide authors present an InstaHide challenge[1] that involves $n_{\mathsf{priv}} = 100$ private images, ImageNet dataset as the public images, $m = 5000$ sample images (each image is a combination of $k_{\mathsf{priv}} = 2$ private images and $k_{\mathsf{pub}} = 4$ public images and the sign of each pixel is randomly flipped). The challenge is to recover a private image given the set of sample images.

[CLSZ21] is a theoretical work that formulates the InstaHide attack problem as a recovery problem. It also provides an algorithm to recover a private image assuming each private and public image is a random Gaussian image (i.e., each pixel is an i.i.d. draw from $\mathcal{N}(0, 1)$). The algorithm shows that $O(n_{\mathsf{priv}}^{k_{\mathsf{priv}} - 2/(k_{\mathsf{priv}}+1)})$ sample images are sufficient to recover one private image. [CDG+20] provides the first heuristic-based

[1]`https://github.com/Hazelsuko07/InstaHideChallenge.`

practical attack for the InstaHide challenge ($k_{\mathsf{priv}} = 2$), and it can generate images that are visually similar to the private images in the InstaHide challenge dataset.

With the same formulation in [CLSZ21], we achieve a better upper bound on the number of samples needed to recover private images when $k_{\mathsf{priv}} = 2$. Our new algorithm can recover all the private images using only $n_{\mathsf{priv}} \log(n_{\mathsf{priv}})$ samples.[2] This significantly improves the state-of-the-art theoretical results [CLSZ21] that requires $n_{\mathsf{priv}}^{4/3}$ samples to recover a single private image. However, our running time is exponential in the number of private images (n_{priv}) and polynomial in the number of public images (n_{pub}), where the running time of the algorithm in [CLSZ21] is polynomial in n_{priv} and n_{pub}. In addition, we provide a four-step framework to compare our attacks with the attacks presented in [CDG+20] and [CLSZ21]. We hope our framework can inspire more efficient attacks on InstaHide-like approaches and can guide the design of future-generation deep learning algorithms on sensitive data.

16.1.1 Our result

[CLSZ21] formulates the InstaHide attack problem as a recovery problem that given sample access to oracle that can generate as much as InstaHide images you want, there are two goals : 1) sample complexity, minimize the number InstaHide images being used, 2) running time, use those InstaHide images to recover the original images as fast as possible.

Let private and public data vectors are Gaussians. Let S_{pub} denote the set of public images with $|S_{\mathsf{pub}}| = n_{\mathsf{pub}}$, let S_{priv} denote the set of private images with $|S_{\mathsf{priv}}| = n_{\mathsf{priv}}$. The model that produces InstaHide image can be desrcibed as follows:

- Pick k_{pub} vectors from public data set and k_{priv} vectors from private data set.
- Normalize k_{pub} vectors by $1/\sqrt{k_{\mathsf{pub}}}$ and normalize k_{priv} vectors by $1/\sqrt{k_{\mathsf{priv}}}$.

[2]For the worst case distribution, $\Omega(n_{\mathsf{priv}})$ is a trivial sample complexity lower bound.

- Add $k_{\mathsf{pub}} + k_{\mathsf{priv}}$ vectors together to obtain a new vector, then flip each coordinate of that new vector independently with probability $1/2$.

We state our result as follows:

Theorem 16.1 (Informal version of Theorem 16.3)**.** *Let $k_{\mathsf{priv}} = 2$. If there are n_{priv} private vectors and n_{pub} public vectors, each of which is an i.i.d. draw from $\mathcal{N}(0, \mathsf{Id}_d)$, then as long as $d = \Omega(poly(k_{\mathsf{pub}}) \log(n_{\mathsf{pub}} + n_{\mathsf{priv}}))$, there is some $m = O(n_{\mathsf{priv}} \log n_{\mathsf{priv}})$ such that, given a sample of m random synthetic vectors independently generated as above, one can exactly recover all the private vectors in time*

$$O(dm^2 + dn_{\mathsf{pub}}^2 + n_{\mathsf{pub}}^{2\omega+1} + mn_{\mathsf{pub}}^2) + d2^{O(m)}$$

with high probability.

Notations. For any positive integer n, we use $[n]$ to denote the set $\{1, 2, \cdots, n\}$. For a set S, we use $supp(S)$ to denote the support of S. For a vector x, we use $\|x\|_2$ to denotes entry-wise ℓ_2 norm. For two vectors a and b, we use $a \circ b$ to denote a vector where i-th entry is $a_i b_i$. For a vector a, we use $|a|$ to denote a vector where the i-th entry is $|a_i|$.

16.1.2 Comparison to recent attacks

Refs	**Recover**	k_{priv}	**Samples**	**Step 1**	**Step 2**	**Step 3**	**Step 4**
[CLSZ21]	one	≥ 2	$m \geq n^{k_{\mathsf{priv}} - 2/(k_{\mathsf{priv}}+1)}$	dm^2	$dn_{\mathsf{pub}}^2 + n_{\mathsf{pub}}^{2\omega+1}$	m^2	$2^{k_{\mathsf{priv}}^2}$
Ours	all	$= 2$	$m \geq n_{\mathsf{priv}} \log n_{\mathsf{priv}}$	dm^2	$dn_{\mathsf{pub}}^2 + n_{\mathsf{pub}}^{2\omega+1}$	mn_{priv}	$2^m \cdot n_{\mathsf{priv}}^2 d$

Table 16.1: A summary of running times in different steps between ours and [CLSZ21]. This table only compares the theoretical result. Let k_{priv} denote the number of private images we select in InstaHide image. Let d denote the dimension of image. Let n_{pub} denote the number of images in public dataset. Let n_{priv} denote the number of images in private dataset. We provide a computational lower bound for Step 4 in Appendix 16.6. There is no algorithm that solves Step 4 in $2^{o(n_{\mathsf{priv}})}$ time under Exponential Time Hypothesis (ETH) (Theorem 16.13).

Our attack algorithm (Algorithm 111) contains four steps for $k_{\mathsf{priv}} = 2$. We can prove $m = O(n_{\mathsf{priv}} \log(n_{\mathsf{priv}}))$ suffices. Our algorithm shares similarities as two recent attack results : one is a practical attack [CDG+20], the other is a theoretical attack [CLSZ21]. In the next a few paragraphs, we describe our attack algorithm in four major steps. For each step, we also give a comparison with the corresponding step in [CDG+20, CLSZ21].

- **Step 1.** Section 16.4.1. Get the Gram matrix $\mathbf{M} \in \mathbb{R}^{m \times m}$ for all images.
  - For this step, [CDG+20]'s attack is using a pre-trained neural network on public dataset to construct the Gram matrix.
  - For this step, we use [CLSZ21]'s attack as a black-box. It takes $O(m^2 d)$ time.
- **Step 2.** Section 16.4.2. Get all public image coefficient via using a SDP solver [JKL+20], and substract public part from Gram matrix.
  - For this step, [CDG+20]'s attack is 1) they treat public images as noise, 2) they don't need to take care of the public images' labels, since current InstaHide Challenge doesn't provide label for public images.
  - For this step, we use [CLSZ21]'s attack as a black box. The time of this step has two parts : 1) formulate the matrix, it takes dn_{pub}^2, 2) solving a SDP with $n_{\mathsf{pub}}^2 \times n_{\mathsf{pub}}^2$ size matrix variable and $O(n_{\mathsf{pub}}^2)$ constraints, thus it takes $n_{\mathsf{pub}}^{2\omega+1}$ time [JKL+20], where ω is the exponent of matrix multiplication.
- **Step 3.** Section 16.4.3. Recover $\mathbf{W} \in \mathbb{R}^{m \times n_{\mathsf{priv}}}$ from observation $\mathbf{M}$ ($\mathbf{M} = \mathbf{W}\mathbf{W}^\top$), this step takes $O(m \cdot n_{\mathsf{priv}}^2)$ time.
  - For this step, [CDG+20]'s attack is using K-means to figure out cliques (Figure 16.1) and then do min-cost max flow problem to figure out the correspondence between InstaHide image and original image (Figure 16.2, 16.3).

  - For this step, [CLSZ21]'s attack is using a clever idea called "floral matrix". We use the fact that $k_{\mathsf{priv}} = 2$ is a graph (while $k_{\mathsf{priv}} \geq 3$ is corresponding to hypergraph), exploring the nice combinatorial property on a graph.

- **Step 4.** Section 16.4.4. Solve d independent ℓ_2-regression problems. Given $\mathbf{W} \in \mathbb{R}^{m \times n_{\mathsf{priv}}}$ and $\mathbf{Y} \in \mathbb{R}^{m \times d}$. For each $i \in [d]$, let $\mathbf{Y}_{*,i} \in \mathbb{R}^m$ denote the i-th column of $\mathbf{Y}$, we need to solve the following ℓ_2 regression

$$\min_{z \in \mathbb{R}^{n_{\mathsf{priv}}}} \| |\mathbf{W}z| - |\mathbf{Y}_{*,i}| \|_2.$$

  The classical ℓ_2 regression can be solved in an efficient way in both theory and practice. However, here we don't know the random signs, and there are 2^m possibilities. We are unaware any provable algorithm without guessing all the 2^m possibilities. Thus the running time of our algorithm in this step is dominated by 2^m.

  - For this step, [CDG+20]'s attack is a heuristic algorithm that uses gradient descent.
  - For this step, [CLSZ21]'s attack is doing the exact same thing as us. However, their goal is just recover one private image which means $m = O(k^2)$. They only need to guess 2^{k^2} possibilities.

16.2 Summary of the Attack by Carlini et al.

This section summarizes the result of Carlini et al, which is an attack of InstaHide when $k_{\mathsf{priv}} = 2$. [CDG+20]. We first briefly describe the current version of InstaHide Challenge. Suppose there are n_{priv} private images, the InstaHide authors [HSLA20b] first choose a parameter T, this can be viewed as the number of iterations in the deep learning training process. For each $t \in [T]$, [HSLA20b] draws a random permutation $\pi_t : [n_{\mathsf{priv}}] \to [n_{\mathsf{priv}}]$. Each InstaHide image is constructed from a private image i, another private image $\pi_t(i)$ and also some public images. Therefore, there are $T \cdot n_{\mathsf{priv}}$ InstaHide images in total. Here is a trivial observation: each private image shown up in exactly $2T$ InstaHide images (because $k_{\mathsf{priv}} = 2$). The model in [CLSZ21] is a different one: each InstaHide image is constructed from two random private images and some random public images. Thus, the observation that each private image appears exactly $2T$ does not hold. In the current version of InstaHide Challenge, the InstaHide authors create the InstaHide labels (a vector that lies in $\mathbb{R}^L$ where the L is the number of classes in image classification task) in a way that the label of an InstaHide image is a linear combination of labels (i.e., one-hot vectors) of the private images and not the public images. This is also a major difference compared with [CLSZ21]. [CDG+20] won't be confused about, for the label of an InstaHide image, which coordinates of the label vector are from the private images and which are from the public images.

- **Step 1.** Recover a similarity[3] matrix $\mathbf{M} \in \{0, 1, 2\}^{m \times m}$.
  - Train a deep neural network based on all the public images, and use that neural network to construct the similarity matrix $\mathbf{M}$.
- **Step 2.** Treat public image as noise.
- **Step 3. Clustering.** This step is divided into 3 substeps.

  The first substep uses the similarity matrix $\mathbf{M}$ to construct Tn_{priv} clusters of InstaHide images based on each InstaHide image such that the images inside one cluster shares a common original image.

  The second substep run K-means on these clusters, to group clusters into n_{priv} groups such that each group corresponds to one original image.

  The third substep construct a min-cost flow graph to compute the two original images that each InstaHide image is mixed from.

[3]In [CDG+20], they call it similarity matrix, in [CLSZ21] they call it Gram matrix. Here, we follow [CDG+20] for convenient.

- **Grow clusters.** Figure 16.1 illustrates an example of this step. For a subset S of InstaHide images ($S \subset [m]$), we define $\textsc{insert}(S)$ as

$$\textsc{Insert}(S) = S \cup \arg\max_{i \in [m]} \sum_{j \in S} \mathbf{M}_{i,j}$$

  For each $i \in [m]$, we compute set $S_i \subset [m]$ where $S_i = \textsc{insert}^{(T/2)}(\{i\})$.

- **Select cluster representatives.** Figure 16.1 illustrates an example of this step. Define distance between clusters as

$$\mathrm{dist}(i,j) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|}.$$

  Run k-means using metric $\mathrm{dist} : [m] \times [m] \to \mathbb{R}$ and $k = n_{\mathsf{priv}}$. Result is n_{priv} groups $C_1, \dots, C_{n_{\mathsf{priv}}} \subseteq [m]$. Randomly select a representative $r_i \in C_i$, for each $i \in [n_{\mathsf{priv}}]$.

- **Computing assignments.** Construct a min-cost flow graph as Figure 16.2, with weight matrix $\widetilde{\mathbf{W}} \in \mathbb{R}^{m \times n_{\mathsf{priv}}}$ defined as follows:

$$\widetilde{\mathbf{W}}_{i,j} = \frac{1}{|S_{r_j}|} \sum_{k \in S_{r_j}} \mathbf{M}_{i,k}.$$

  for $i \in [m], j \in [n_{\mathsf{priv}}]$. After solving the min-cost flow (Figure 16.3), construct the assignment matrix $\mathbf{W} \in \mathbb{R}^{m \times n_{\mathsf{priv}}}$ such that $\mathbf{W}_{i,j} = 1$ if the edge from i to j has flow, and 0 otherwise.

- **Step 4.** Recover original image. From Step 3, we have the unweighted assignment matrix $\mathbf{W} \in \{0,1\}^{m \times n_{\mathsf{priv}}}$. Before we recover the original image, we need to first recover the weight of mixing, which is represented by the weighted assignment matrix $\mathbf{U} \in \mathbb{R}^{m \times n_{\mathsf{priv}}}$. To recover weight, we first recover the label for each cluster group, and use the recovered label and the mixed label to recover the weight.

  - First, we recover the label for each cluster, for all $i \in [n_{\mathsf{priv}}]$. Let L denote the number of classes in the classification task of InstaHide application. For

$j \in [m]$, let $y_j \in \mathbb{R}^L$ be the label of j.

$$\mathrm{label}(i) = \bigcap_{j \in [m], \mathbf{W}_{j,i}=1} \mathit{supp}(y_j) \in [L].$$

Then, for $i \in [m]$ and $j \in [n_{\mathsf{priv}}]$ such that $\mathbf{W}_{i,j} = 1$, define $\mathbf{U}_{i,j} = y_{i,\mathrm{label}(j)}$ for $|\mathit{supp}(y_i)| = 2$ and $\mathbf{U}_{i,j} = y_{i,\mathrm{label}(j)}/2$ for $|\mathit{supp}(y_i)| = 1$.

Here, $\mathbf{W} \in \{0,1\}^{m \times n_{\mathsf{priv}}}$ is the unweighted assignment matrix and $\mathbf{U} \in \mathbb{R}^{m \times n_{\mathsf{priv}}}$ is the weighted assignment matrix. For $\mathbf{W}_{i,j} = 0$, let $\mathbf{U}_{i,j} = 0$.

– Second, for each pixel $i \in [d]$, we run gradient descent to find the original images. Let $\mathbf{Y} \in \mathbb{R}^{m \times d}$ be the matrix of all $\mathsf{InstaHide}$ images, $\mathbf{Y}_{*,i}$ denote the i-th column of $\mathbf{Y}$.[4]

$$\min_{z \in \mathbb{R}^{n_{\mathsf{priv}}}} \| |\mathbf{U}z| - |\mathbf{Y}_{*,i}| \|_2.$$

16.3 Preliminaries

We use the same setup as [CLSZ21].

Definition 16.1 (Image matrix notation, Definition 2.2 in [CLSZ21])**.** Let image matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$ be a matrix whose columns consist of vectors $x_1, \ldots, x_n \in \mathbb{R}^d$ corresponding to n images each with d pixels taking values in $\mathbb{R}$. It will also be convenient to refer to the rows of $\mathbf{X}$ as $p_1, \ldots, p_d \in \mathbb{R}^n$.

We define public set and private set as follows,

Definition 16.2 (Public/private notation, Definition 2.3 in [CLSZ21])**.** Let $S_{\mathsf{pub}} \subset [n]$ be some subset. We will refer to S_{pub} and $S_{\mathsf{priv}} = [n] \backslash S_{\mathsf{pub}}$ as the set of public and private images respectively, and given a vector $w \in \mathbb{R}^n$, we will refer to $\mathit{supp}(w) \cap S_{\mathsf{pub}}$ and $\mathit{supp}(w) \cap S_{\mathsf{priv}}$ as the public and private coordinates of w respectively.

[4]The description of the attack in [CDG+20] recovers original images by using gradient descent for $\min_{z \in \mathbb{R}^{n_{\mathsf{priv}}}} \|\mathbf{U}|z| - |\mathbf{Y}_{*,i}|\|_2$, which we believe is a typo.

We define synthetic images as follows,

Definition 16.3 (Synthetic images, Definition 2.4 in [CLSZ21])**.** Given sparsity levels $k_{\mathsf{pub}} \leq |S_{\mathsf{pub}}|, k_{\mathsf{priv}} \leq |S_{\mathsf{priv}}|$, image matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$ and a selection vector $w \in \mathbb{R}^n$ for which $[w]_{S_{\mathsf{pub}}}$ and $[w]_{S_{\mathsf{priv}}}$ are k_{pub}- and k_{priv}-sparse respectively, the corresponding synthetic image is the vector $y^{\mathbf{X},w} = |\mathbf{X}w| \in \mathbb{R}^d$ where $|\cdot|$ denotes entrywise absolute value. We say that $\mathbf{X} \in \mathbb{R}^{d \times n}$ and a sequence of selection vectors $w_1, \ldots, w_m \in \mathbb{R}^n$ give rise to a synthetic dataset $\mathbf{Y} \in \mathbb{R}^{m \times d}$ consisting of the images $(y^{\mathbf{X},w_1}, \ldots, y^{\mathbf{X},w_m})^\top$.

We define Gaussian images,

Definition 16.4 (Gaussian images, Definition 2.5 in [CLSZ21])**.** We say that $\mathbf{X}$ is a random *Gaussian image matrix* if its entries are sampled i.i.d. from $\mathcal{N}(0, 1)$.

We define the distribution over selection vectors,

Definition 16.5 (Distribution over selection vectors, Definition 2.6 in [CLSZ21])**.** Let $\mathcal{D}$ be the distribution over selection vectors defined as follows. To sample once from $\mathcal{D}$, draw random subset $T_1 \subset S_{\mathsf{pub}}, T_2 \subseteq S_{\mathsf{priv}}$ of size k_{pub} and k_{priv} and output the unit vector whose i-th entry is $1/\sqrt{k_{\mathsf{pub}}}$ if $i \in T_1$, $1/\sqrt{k_{\mathsf{priv}}}$ if $i \in T_2$, and zero otherwise.[5]

We define pub and priv operators as follows,

Definition 16.6 (Public/private operators)**.** We define function $\mathsf{pub}(\cdot)$ and $\mathsf{priv}(\cdot)$ such that for vector $w \in \mathbb{R}^n$, $\mathsf{pub}(w) \in \mathbb{R}^{n_{\mathsf{pub}}}$ will be the vector where the coordinates of w corresponding to private subset S_{priv} are deleted, and $\mathsf{priv}(w) \in \mathbb{R}^{n_{\mathsf{priv}}}$ will be the vector where the coordinates of w corresponding to public subset S_{pub} are deleted.

[5]Note that any such vector does not specify a convex combination, but this choice of normalization is just to make some of the analysis later on somewhat cleaner, and our results would still hold if we chose the vectors in the support of $\mathcal{D}$ to have entries summing to 1.

For subset $\widetilde{S} \subset S$ we will refer to $vec(\widetilde{S}) \in \mathbb{R}^n$ as the vector that $vec(\widetilde{S})_i = 1$ if $i \in \widetilde{S}$ and $vec(\widetilde{S})_i = 0$ otherwise. We define the public and private components of $\mathbf{W}$ and $\mathbf{Y}$ for convenient.

Definition 16.7 (Public and private components of image matrix and selection vectors)**.** For a sequence of selection vectors $w_1, \ldots, w_m \in \mathbb{R}^n$ we will refer to $\mathbf{W} = (vec(supp(w_1)), \ldots, vec(supp(w_m)))^\top \in \{0,1\}^{m \times n}$ as the mixup matrix.

Specifically, we will refer to $\mathbf{W}_{\mathsf{pub}} \in \{0,1\}^{m \times n_{\mathsf{pub}}}$ as the public component of mixup matrix and $\mathbf{W}_{\mathsf{priv}} \in \{0,1\}^{m \times n_{\mathsf{priv}}}$ as the private component of mixup matrix, i.e.,

$$\mathbf{W}_{\mathsf{pub}} = \begin{bmatrix} \mathsf{pub}(\mathbf{W}_{1,*}) \\ \vdots \\ \mathsf{pub}(\mathbf{W}_{m,*}) \end{bmatrix}, \quad \mathbf{W}_{\mathsf{priv}} = \begin{bmatrix} \mathsf{priv}(\mathbf{W}_{1,*}) \\ \vdots \\ \mathsf{priv}(\mathbf{W}_{m,*}) \end{bmatrix}.$$

We will refer to $\mathbf{X}_{\mathsf{pub}} \in \mathbb{R}^{d \times n_{\mathsf{pub}}}$ as public component of image matrix where columns of $\mathbf{X} \in \mathbb{R}^{d \times n}$ corresponding to private set S_{priv} are deleted, and $\mathbf{X}_{\mathsf{priv}} \in \mathbb{R}^{d \times n_{\mathsf{priv}}}$ as private component of image matrix where columns of $\mathbf{X} \in \mathbb{R}^{d \times n}$ corresponding to public set S_{pub} are deleted.

Furthermore we define $\mathbf{Y}_{\mathsf{pub}} \in \mathbb{R}^{m \times d}$ as public contribution to InstaHide images and $\mathbf{Y}_{\mathsf{priv}} \in \mathbb{R}^{m \times d}$ as private contribution to InstaHide images:

$$\mathbf{Y}_{\mathsf{pub}} = \frac{1}{\sqrt{k_{\mathsf{pub}}}} \mathbf{W}_{\mathsf{pub}} \mathbf{X}_{\mathsf{pub}}^\top, \quad \mathbf{Y}_{\mathsf{priv}} = \frac{1}{\sqrt{k_{\mathsf{priv}}}} \mathbf{W}_{\mathsf{priv}} \mathbf{X}_{\mathsf{priv}}^\top.$$

Instead of considering only one private image recovery as [CLSZ21, Problem 1], here we consider a harder question which requires to recover all the private images.

Problem 16.2 (Private image recovery)**.** Let $\mathbf{X} \in \mathbb{R}^{d \times n}$ be a Gaussian image matrix. Given access to the public images $\{x_s\}_{s \in S_{\mathsf{pub}}}$ and the synthetic dataset $(y^{\mathbf{X}, w_1}, \ldots, y^{\mathbf{X}, w_m})$, where $w_1, \ldots, w_m \sim \mathcal{D}$ are unknown selection vectors, output a set of vectors $\{\widetilde{x}_s\}_{s \in S_{\mathsf{priv}}}$ for which there exists a one-to-one mapping ϕ from $\{\widetilde{x}_s\}_{s \in S_{\mathsf{priv}}}$ to $\{x_s\}_{s \in S_{\mathsf{priv}}}$ satisfying $|\phi(\widetilde{x}_s)_j| = |(x_s)_j|$, $\forall j \in [d]$.

Algorithm 111 Recovering All Private Images when $k_{\mathsf{priv}} = 2$

```
 1: procedure RecoverAll($\mathbf{Y}$)    ▷ Theorem 16.3, Theorem 16.1
 2:     ▷ InstaHide dataset $\mathbf{Y} = (y^{\mathbf{X},w_1}, \ldots, y^{\mathbf{X},w_m})^\top \in \mathbb{R}^{m \times d}$
 3:     ▷ Step 1. Retrieve Gram matrix
 4:     $\mathbf{M} \leftarrow \frac{1}{k_{\mathsf{priv}}+k_{\mathsf{pub}}} \cdot \textsc{GramExtract}(\mathbf{Y}, \frac{1}{2(k_{\mathsf{pub}}+k_{\mathsf{priv}})})$    ▷ Algorithm 1 in [CLSZ21]
 5:     ▷ Step 2. Subtract Public images from Gram matrix
 6:     for $i \in [m]$ do
 7:         $S_i \leftarrow \textsc{LearnPublic}(\{(p_j)_{S_{\mathsf{pub}}}, y_j^{\mathbf{X},w_i})\}_{j\in[d]})$    ▷ Algorithm 2 in [CLSZ21]
 8:     end for
 9:     $\mathbf{W}_{\mathsf{pub}} \leftarrow (\mathsf{pub}(vec(S_1)), \ldots, \mathsf{pub}(vec(S_m)))^\top$    ▷ $\mathbf{W}_{\mathsf{pub}} \in \{0,1\}^{m \times n_{\mathsf{pub}}}$
10:     $\mathbf{M}_{\mathsf{priv}} \leftarrow k_{\mathsf{priv}} \cdot (\mathbf{M} - \frac{1}{k_{\mathsf{pub}}} \mathbf{W}_{\mathsf{pub}} \mathbf{W}_{\mathsf{pub}}^\top)$
11:     ▷ Step 3. Assign original images
12:     $\mathbf{W}_{\mathsf{priv}} \leftarrow \textsc{AssigningOriginalImages}(\mathbf{M}_{\mathsf{priv}}, n_{\mathsf{priv}})$    ▷ Algorithm 112
13:     ▷ Step 4. Solving system of equations.
14:     $\mathbf{Y}_{\mathsf{pub}} = \frac{1}{\sqrt{k_{\mathsf{pub}}}} \mathbf{W}_{\mathsf{pub}} \mathbf{X}_{\mathsf{pub}}^\top$    ▷ $\mathbf{X}_{\mathsf{pub}} \in \mathbb{R}^{d \times n_{\mathsf{pub}}}$, $\mathbf{Y}_{\mathsf{pub}} \in \mathbb{R}^{m \times d}$, $\mathbf{W}_{\mathsf{pub}} \in \{0,1\}^{m \times n_{\mathsf{pub}}}$
15:     $\widetilde{X} \leftarrow \textsc{SolvingSystemofEquations}(\mathbf{W}_{\mathsf{priv}}, \sqrt{k_{\mathsf{priv}}} \mathbf{Y}_{\mathsf{pub}}, \sqrt{k_{\mathsf{priv}}} \mathbf{Y})$    ▷ Algorithm 113
16:     return $\widetilde{X}$
17: end procedure
```

16.4 Recovering All Private Images when $k_{\mathsf{priv}} = 2$

In this section, we prove our main algorithmic result:

Theorem 16.3 (Main result). *Let $S_{\mathsf{pub}} \subset [n]$, and let $n_{\mathsf{pub}} = |S_{\mathsf{pub}}|$ and $n_{\mathsf{priv}} = |S_{\mathsf{priv}}|$. Let $k_{\mathsf{priv}} = 2$. Let $k = k_{\mathsf{priv}} + k_{\mathsf{pub}}$. If $d \geq \Omega\big(poly(k_{\mathsf{pub}}, k_{\mathsf{priv}}) \log(n_{\mathsf{pub}} + n_{\mathsf{priv}})\big)$ and $m \geq \Omega\big(k^{poly(k_{\mathsf{priv}})} n_{\mathsf{priv}} \log n_{\mathsf{priv}}\big)$, then with high probability over $\mathbf{X}$ and the sequence of randomly chosen selection vectors $w_1, \ldots, w_m \sim \mathcal{D}$, there is an algorithm which takes as input the synthetic dataset $\mathbf{Y}^\top = (y^{\mathbf{X},w_1}, \ldots, y^{\mathbf{X},w_m}) \in \mathbb{R}^{d \times m}$ and the columns of $\mathbf{X}$ indexed by S_{pub}, and outputs n_{priv} images $\{\widetilde{x}_s\}_{s \in S_{\mathsf{priv}}}$ for which there exists one-to-one mapping ϕ from $\{\widetilde{x}_s\}_{s \in S_{\mathsf{priv}}}$ to $\{x_s\}_{s \in S_{\mathsf{priv}}}$ satisfying $|\phi(\widetilde{x}_s)_j| = |(x_s)_j|$ for all $j \in [d]$. Furthermore, the algorithm runs in time*

$$O(m^2 d) + O(d n_{\mathsf{pub}}^2) + O(n_{\mathsf{pub}}^{2\omega+1}) + O(m n_{\mathsf{priv}}^2) + 2^m \cdot m n_{\mathsf{priv}}^2 d.$$

16.4.1 Retrieving Gram matrix

In this section, we present the algorithm for retrieving the Gram matrix.

Theorem 16.4 (Retrieve Gram matrix, [CLSZ21])**.** *Let $n = n_{\mathsf{pub}} + n_{\mathsf{priv}}$. Suppose $d = \Omega(\log(m/\delta)/\eta^4)$. For random Gaussian image matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$ and arbitrary $w_1, \dots, w_m \in \mathbb{S}^{d-1}_{\geq 0}$, let Σ^* be the output of* GramExtract *when we set $\eta = 1/2k$. Then with probability $1-\delta$ over the randomness of $\mathbf{X}$, we have that $\Sigma^* = k \cdot \mathbf{W}\mathbf{W}^\top \in \mathbb{R}^{m\times m}$. Furthermore,* GramExtract *runs in time $O(m^2 d)$.*

We briefly describe how they achieved this. Without loss of generality, we may assume $S_{\mathsf{priv}} = [n]$, since once we determine the support of public images S_{pub}, we can easily subtract the contribution of them. Consider a matrix $\mathbf{Y} \in \mathbb{R}^{m\times d}$ whose rows are $y^{\mathbf{X},w_1}, \dots, y^{\mathbf{X},w_m}$. Then, it can be written as

$$\mathbf{Y} = \begin{bmatrix} \langle p_1, w_1 \rangle & \cdots & \langle p_d, w_1 \rangle \\ \vdots & \ddots & \vdots \\ \langle p_1, w_m \rangle & \cdots & \langle p_d, w_m \rangle. \end{bmatrix}$$

Since $\mathbf{X}$ is a Gaussian matrix, we can see that each column of $\mathbf{Y}$ is the absolute value of an independent draw of $\mathcal{N}(0, \mathbf{W}\mathbf{W}^\top)$. They defined this distribution as $\mathcal{N}^{\mathsf{fold}}(0, \mathbf{W}\mathbf{W}^\top)$, and they can prove that the covariance matrix of $\mathcal{N}^{\mathsf{fold}}(0, \mathbf{W}\mathbf{W}^\top)$ can be directly related $\mathbf{W}\mathbf{W}^\top$. Then, the task becomes estimating the covariance matrix of $\mathcal{N}^{\mathsf{fold}}(0, \mathbf{W}\mathbf{W}^\top)$ from d independent samples (columns of $\mathbf{Y}$), which can be done by computing the empirical estimates.

16.4.2 Remove public images

In this section, we present the algorithm of subtracting public images from Gram matrix. Formally, given any synthetic image $y^{\mathbf{X},w}$ we recover the entire support of $[w]_{S_{\mathsf{pub}}}$ (essentially $supp([w]_{S_{\mathsf{pub}}})$).

Theorem 16.5 (Subtract public images from Gram matrix, [CLSZ21])**.** *Let $n = n_{\mathsf{priv}} + n_{\mathsf{pub}}$. For any $\delta \geq 0$, if $d = \Omega(poly(k_{\mathsf{pub}})/\log(n/\delta))$, then with probability at least $1-\delta$ over the randomness of $\mathbf{X}$, we have that the coordinates output by* LearnPublic *are exactly equal to $supp([w]_{S_{\mathsf{pub}}})$. Furthermore,* LearnPublic *runs* *in time $O(dn_{\mathsf{pub}}^2 + n_{\mathsf{pub}}^{2\omega+1})$, where $\omega \approx 2.373$ is the exponent of matrix multiplication [Wil12].*

Note that this problem is closely related to the Gaussian phase retrieval problem. However, we can only access the public subset of coordinates for any image vector p_i. We denote these partial vectors as $\{[p_i]_{S_{\mathsf{pub}}}\}_{i\in[d]}$. The first step they did is to construct a matrix $\widetilde{\mathbf{M}} \in \mathbb{R}^{n_{\mathsf{pub}}\times n_{\mathsf{pub}}}$:

$$\widetilde{\mathbf{M}} = \frac{1}{d}\sum_{i=1}^{d}((y_i^{\mathbf{X},w})^2 - 1)\cdot([p_i]_{S_{\mathsf{pub}}}[p_i]_{S_{\mathsf{pub}}}^\top - \mathbf{I}).$$

They can prove that when p_i's are i.i.d standard Gaussian vectors, the expectation of $\widetilde{\mathbf{M}}$ is $\mathbf{M} = \frac{1}{2}[w]_{S_{\mathsf{pub}}}[w]_{S_{\mathsf{pub}}}^\top$. However, when $d \ll n$, $\widetilde{\mathbf{M}}$ is not a sufficiently good spectral approximation of $\mathbf{M}$, which means we cannot directly use the top eigenvector of $\widetilde{\mathbf{M}}$. Instead, they showed that with high probability, $[w]_{S_{\mathsf{pub}}}$ can be approximated by the top eigenvector of the solution of the following semi-definite programming (SDP):

$$\max_{Z\succeq 0}\ \langle\widetilde{\mathbf{M}}, Z\rangle\ s.t.\ tr[Z] = 1, \sum_{i,j=1}^{n_{\mathsf{pub}}}|Z_{i,j}| \le k_{\mathsf{pub}}.$$

Hence, the time complexity of this step is $O(dn_{\mathsf{pub}}^2 + n_{\mathsf{pub}}^{2\omega+1})$, where the first term is the time cost for constructing $\widetilde{\mathbf{M}}$ and the second term is the time cost for SDP [JKL+20].

16.4.3 Assigning encoded images to original images

We are now at the position of recovering $\mathbf{W}_{\mathsf{priv}} \in \mathbb{R}^{m\times n_{\mathsf{priv}}}$ from private Gram matrix $\mathbf{M}_{\mathsf{priv}} \in \mathbb{R}^{m\times m}$. Recall that $\mathbf{M}_{\mathsf{priv}} = \mathbf{W}_{\mathsf{priv}}\mathbf{W}_{\mathsf{priv}}^\top \in \mathbb{R}^{m\times m}$ where $\mathbf{W}_{\mathsf{priv}} \in \{0,1\}^{m\times n_{\mathsf{priv}}}$ is the mixup matrix with column sparsity k_{priv}. By recovering mixup matrix $\mathbf{W}$ from private Gram matrix $\mathbf{M}$ the attacker maps each synthetic image $y^{\mathbf{X},w_i}, i \in [m]$ to two original images $x_{i_1}, \ldots, x_{i_{k_{\mathsf{priv}}}}$ (to be recovered in the next step) in the private data set, where $k_{\mathsf{priv}} = 2$.

On the other hand, in order to recover original image x_i from private data set, the attacker needs to know precisely the set of synthetic images $y^{\mathbf{X},w_i}, i \in [m]$

Algorithm 112 Assigning Original Images

```
1: procedure AssigningOriginalImages($\mathbf{M}_{\mathsf{priv}}, n_{\mathsf{priv}}$)
2:         ▷ $\mathbf{M}_{\mathsf{priv}} \in \mathbb{R}^{m \times m}$ is Private Gram matrix, $n_{\mathsf{priv}}$ is the number of private images
3:     $\mathbf{M}_G \leftarrow \mathbf{M}_{\mathsf{priv}} - \mathbf{I}$
4:     if $n_{\mathsf{priv}} < 5$ then
5:         for $H \in \{0,1\}^{n_{\mathsf{priv}} \times n_{\mathsf{priv}}}$ do
6:             $\mathbf{M}_H \leftarrow$ adjacency matrix of the line graph of $H$
7:             if $\mathbf{M}_H = \mathbf{M}_G$ then
8:                 $\widetilde{\mathbf{W}} \leftarrow \widetilde{\mathbf{W}} \cup \{\mathbf{W}_H\}$        ▷ $\mathbf{W}_H$ is the incidence matrix of $H$
9:             end if
10:        end for
11:        return $\widetilde{\mathbf{W}}$
12:    end if
13:    Reconstruct $G$ from $\mathbf{M}_G$        ▷ By Theorem 16.9
14:    return $\mathbf{W}$        ▷ The incidence matrix of $G$
15: end procedure
```

generated by x_i. Therefore this step is crucial to recover the original private images from InstaHide images.

Definition 16.8 (Distinguishable). For matrix $\mathbf{M} \in \mathbb{R}^{m \times m}$, we say $\mathbf{M}$ is distinguishable if there exists unique solution $\mathbf{W} = (w_1, \ldots, w_m)^\top$ (up to permutation of rows) to the equation $\mathbf{W}\mathbf{W}^\top = \mathbf{M}$ such that $w_i \in \mathit{supp}(\mathcal{D}_{\mathsf{priv}})$ for all $i \in [m]$.

Theorem 16.6 (Assign InstaHide images to the original images). *When $m = \Omega(n_{\mathsf{priv}} \log n_{\mathsf{priv}})$, let $\mathbf{W}_{\mathsf{priv}} = (w_1, \ldots, w_m)^\top$ where $w_i, i \in [m]$ are sampled from distribution $\mathcal{D}_{\mathsf{priv}}$ and $\mathbf{M}_{\mathsf{priv}} = \mathbf{W}_{\mathsf{priv}}\mathbf{W}_{\mathsf{priv}}^\top \in \mathbb{R}^{m \times m}$. Then with high probability $\mathbf{M}_{\mathsf{priv}}$ is distinguishable and algorithm* AssigningOriginalImages *inputs private Gram matrix $\mathbf{M}_{\mathsf{priv}} \in \{0,1,2\}^{m \times m}$ correctly outputs $\mathbf{W}_{\mathsf{priv}} \in \{0,1\}^{m \times n_{\mathsf{priv}}}$ with row sparsity $k_{\mathsf{priv}} = 2$ such that $\mathbf{W}_{\mathsf{priv}}\mathbf{W}_{\mathsf{priv}}^\top = \mathbf{M}_{\mathsf{priv}}$. Furthermore* AssigningOriginalImages *runs in time $O(mn_{\mathsf{priv}})$.*

16.4.4 Solving a large system of equations

In this section, we solve the step 4, recovering all of the private images by solving an ℓ_2-regression problem. Formally, given mixup coefficients $\mathbf{W}_{\mathsf{priv}}$ (for private images) and contributions to InstaHide images from public images $\mathbf{Y}_{\mathsf{pub}}$ we recover all private images $\mathbf{X}_{\mathsf{priv}}$ (up to absolute value).

Theorem 16.7 (Solve ℓ_2-regression with hidden signs)**.** *Given $\mathbf{W}_{\mathsf{priv}} \in \mathbb{R}^{m \times n_{\mathsf{priv}}}$ and $\mathbf{Y}_{\mathsf{pub}}, \mathbf{Y} \in \mathbb{R}^{m \times d}$. For each $i \in [d]$, let $\mathbf{Y}_{*,i} \in \mathbb{R}^m$ denote the i-th column of $\mathbf{Y}$ and similarily for $\mathbf{Y}_{\mathsf{pub}_{*,i}}$, the following ℓ_2 regression*

$$\min_{z_i \in \mathbb{R}^{n_{\mathsf{priv}}}} \| |\mathbf{W}_{\mathsf{priv}} z_i + \mathbf{Y}_{\mathsf{pub}_{*,i}}| - \mathbf{Y}_{*,i} \|_2.$$

for all $i \in [d]$ can be solve by SolvingSystemOfEquations *in time $O(2^m \cdot m n_{\mathsf{priv}}^2 \cdot d)$. Furthermore, with probability* 1 SolvingSystemOfEquations *outputs $\widetilde{X}$ whose columns constitute a set of vectors $\{\widetilde{x}_s\}_{s \in S_{\mathsf{priv}}}$ for which there exists a one-to-one mapping ϕ from $\{\widetilde{x}_s\}_{s \in S_{\mathsf{priv}}}$ to $\{x_s\}_{s \in S_{\mathsf{priv}}}$ satisfying $|\phi(\widetilde{x}_s)_j| = |(x_s)_j|$ for all $j \in [d]$.*

Proof. Suppose $\mathbf{W}_{\mathsf{priv}} = \begin{bmatrix} w_1 & w_2 & \cdots & w_m \end{bmatrix}^\top$. Then, the ℓ_2-regression we considered actually minimizes

$$\sum_{j=1}^m (|w_j^\top z_i + \mathbf{Y}_{\mathsf{pub}_{j,i}}| - \mathbf{Y}_{j,i})^2 = \sum_{j=1}^m (w_j^\top z_i + \mathbf{Y}_{\mathsf{pub}_{j,i}} - \sigma_j \cdot \mathbf{Y}_{j,i})^2,$$

where $\sigma_j \in \{-1, 1\}$ is the sign of $w_j z_i^*$ for the minimizer z_i^*.

Therefore, in Algorithm 113, we enumerate all possible $\sigma \in \{\pm 1\}^m$. Once σ is fixed, the optimization problem becomes the usual ℓ_2-regression, which can be solved in $O(n_{\mathsf{priv}}^\omega + m n_{\mathsf{priv}}^2)$ time. Since we assume $m = \Omega(n_{\mathsf{priv}} \log(n_{\mathsf{priv}}))$ in the previous step, the total time complexity is $O(2^m \cdot m n_{\mathsf{priv}}^2)$.

We can repeat this process for all $i \in [d]$ and solve all z_i's.

□

16.5 Missing proofs for Theorem 16.6

For simplicity, let $\mathbf{W}$ denote $\mathbf{W}_{\mathsf{priv}}$ and $\mathbf{M}$ denote $\mathbf{M}_{\mathsf{priv}}$ in this section.

Algorithm 113 Solving a large system of equations

```
1: procedure SolvingSystemOfEquations($\mathbf{W}_{\mathsf{priv}}, \mathbf{Y}_{\mathsf{pub}}, \mathbf{Y}$)
2:                                   ▷ $\mathbf{W}_{\mathsf{priv}} \in \mathbb{R}^{m \times n_{\mathsf{priv}}}, \mathbf{Y}_{\mathsf{pub}} \in \mathbb{R}^{m \times d}, \mathbf{Y} \in \mathbb{R}^{m \times d}$
3:     for $i = 1 \to d$ do
4:         $\widetilde{x}_i \leftarrow \emptyset$
5:         for $\sigma \in \{-1, +1\}^m$ do
6:             $z \leftarrow \min_{z \in \mathbb{R}^{n_{\mathsf{priv}}}} \|\mathbf{W}_{\mathsf{priv}} z + \mathbf{Y}_{\mathsf{pub}_{*,i}} - \sigma \circ \mathbf{Y}_{*,i}\|_2$
7:             if $\mathrm{sign}(\mathbf{W}_{\mathsf{priv}} z + \mathbf{Y}_{\mathsf{pub}_{*,i}}) = \sigma$ then
8:                 $\widetilde{x}_i \leftarrow \widetilde{x}_i \cup z$
9:             end if
10:        end for
11:    end for
12:    $\widetilde{X} \leftarrow \{\widetilde{x}_1, \cdots, \widetilde{x}_d\}$
13:    return $\widetilde{X}$
14: end procedure
```

16.5.1 A graph problem ($k_{\mathsf{priv}} = 2$)

In this section, we interpret the $k_{\mathsf{priv}} = 2$ case as a graph problem. We consider graph $G = (V, E), |V| = n_{\mathsf{priv}}$ and $|E| = m$ where each $v_i \in V$ corresponds to an original image in private data set and each $e = (v_i, v_j) \in E$ correspond to an encrypted image generated from two original images corresponding to v_i and v_j. We define the Gram matrix of graph $G = (V, E)$, denoted by $\mathbf{M}_G \in \{0, 1, 2\}^{m \times m}$ where $m = |E|$, to be $\mathbf{M}_G = \mathbf{W}\mathbf{W}^\top - \mathbf{I}$ where $\mathbf{W} \in \{0, 1\}^{m \times n_{\mathsf{priv}}}$ is the incidence matrix of G. That is, $(\mathbf{M}_G)_{i,j} = |e_i \cap e_j|$ for edges $e_i, e_j \in E$.[6] We can see that $\mathbf{M}_G$ is actually the line graph $L(G)$ of the graph G. We similarly call a graph G distinguishable if there exists no other graph G' such that G and G' have the same Gram matrix (up to permutations of edges), namely $\mathbf{M}_G = \mathbf{M}_{G'}$ (for some ordering of edges). To put it into another word, if we know $\mathbf{M}_G$, we can recover G uniquely. Therefore, recovering $\mathbf{W}$ from $\mathbf{M}$ can be viewed as recovering graph G from its Gram matrix $\mathbf{M}_G \in \mathbb{R}^{m \times m}$, and a graph is distinguishable if and only if its Gram matrix $\mathbf{M}_G$ is distinguishable.

This problem was studied since 1970s and fully resolved by Whitney [Whi92].

[6]With high probability, W will not have multi-edge. So, most entries of M will be in $\{0, 1\}$.

Theorem 16.8 ([Whi92])**.** *Suppose G and H are connected simple graphs and $L(G) \cong L(H)$. If G and H are not K_3 and $K_{1,3}$, then $G \cong H$. Furthermore, if $|V(G)| \geq 5$, then an isomorphism of $L(G)$ uniquely determines an isomorphism of G.*

In other words, this theorem claims that given $\mathbf{M} = \mathbf{W}\mathbf{W}^\top$, if the underlying $\mathbf{W}$ is not the incident matrix of K_3 or $K_{1,3}$, $\mathbf{W}$ can be uniquely identified up to permutation. Theorem 16.8 can also be generalized to the case when G has multi-edges [Zve97].

On the other hand, a series of work [Rou73, Leh74, Sys82, DS95, LTVM15] showed how to efficiently reconstruct the original graph from its line graph:

Theorem 16.9 ([LTVM15])**.** *Given a graph L with m vertices and t edges, there exists an algorithm that runs in time $O(m+t)$ to decide whether L is a line graph and output the original graph G. Furthermore, if L is promised to be the line graph of G, then there exists an algorithm that outputs G in time $O(m)$.*

With Theorem 16.8 and Theorem 16.9, Theorem 16.6 follows immediately:

Proof of Theorem 16.6. First, since $m = \Omega(n_{\mathsf{priv}} \log(n_{\mathsf{priv}}))$, a well-known fact in random graph theory by Erdős and Rényi [ER60] showed that the graph G with incidence matrix $\mathbf{W}$ will almost surely be connected. Then, we compute $\mathbf{M}_G = \mathbf{M} - \mathbf{I}$, the adjacency matrix of the line graph $L(G)$. Theorem 16.8 implies that G can be uniquely recovered from $\mathbf{M}_G$ as long as n_{priv} is large enough. Finally, We can reconstruct G from $\mathbf{M}_G$ by Theorem 16.9.

For the time complexity of Algorithm 112, the reconstruction step can be done in $O(m)$ time. Since we need to output the matrix $\mathbf{W}$, we will take $O(mn_{\mathsf{priv}})$-time to construct the adjacency matrix of G. Here, we do not count the time for reading the whole matrix $\mathbf{M}$ into memory.

□

16.5.2 General case ($k_{\mathsf{priv}} > 2$)

The characterization of $\mathbf{M}$ and $\mathbf{W}$ as the line graph and incidence graph can be generalized to $k_{\mathsf{priv}} > 2$ case, which corresponds to hypergraphs.

Suppose $\mathbf{M} = \mathbf{W}\mathbf{W}^\top$ with $k_{\mathsf{priv}} = k > 2$. Then, $\mathbf{W}$ can be recognized as the incidence matrix of a k-uniform hypergraph G, i.e., each hyperedge contains k vertices. $\mathbf{M}_G = \mathbf{M} - \mathbf{I}$ corresponds to adjacency matrix of the line graph of hypergraph G: $(\mathbf{M}_G)_{i,j} = |e_i \cap e_j|$ for e_i, e_j being two hyperedges. Now, we can see that each entry of $\mathbf{M}_G$ is in $\{0, \ldots, k\}$.

Unfortunately, the identification problem becomes very complicated for hypergraphs. Lovász [Lov77] stated the problem of characterizing the line graphs of 3-uniform hypergraphs and noted that Whitney's isomorphism theorem (Theorem 16.8) cannot be generalized to hypergraphs. Hence, we may not be able to uniquely determine the underlying hypergraph and we should just consider a more basic problem:

Problem 16.10 (Line graph recognition for hypergraph)**.** Given a simple graph $L = (V, E)$ and $k \in \mathbb{N}$, decide if L is the line graph of a k-uniform hypergraph G.

Even for the recognition problem, it was proved to be NP-complete for fixed $k \geq 3$ [LT93, PRT81]. However, Problem 16.10 becomes tractable if we add more constraints to the underlying hypergraph G. First, suppose G is a linear hypergraph, i.e., the intersection of two hyperedges is at most one. If we further assume the minimum degree of G is at least 10, i.e., each vertex are in at least 10 hyperedges, there exists a polynomial-time algorithm for the decision problem. Similar result also holds for $k > 3$ [JKL97]. Let the edge-degree of a hyperedge be the number of triangles in the hypergraph containing that hyperedge. [JKL97] showed that assuming the minimum edge-degree of G is at least $2k^2 - 3k + 1$, there exists a polynomial-time algorithm to decide whether L is the line graph of a linear k-uniform hypergraph. Furthermore, in the yes case, the algorithm can also reconstruct the underlying hypergraph. We also note that without any constraint on minimum degree or edge-degree, the complexity of recognizing line graphs of k-uniform linear hypergraphs is still unknown.

16.6 Computational Lower Bound

The goal of this section is to prove that the ℓ_2-regression with hidden signs is actually a very hard problem, even for approximation (Theorem 16.13), which implies that Algorithm 113 cannot be significantly improved. For simplicity we consider $S_{\mathsf{pub}} = \emptyset$.

We will reduce the MAX-CUT problem to the ℓ_2-regression. A MAX-CUT instance is a graph $G = (V, E)$ with n vertices and m edges. The goal is to find a subset of vertices $S \subseteq V$ such that the number of edges between S and $V \backslash S$ is maximized, i.e., $\max_{S \subseteq V} |E(S, V \backslash S)|$. We can further assume G is 3-regular, that is, each vertex has degree 3.

We first state an NP-hardness of approximation result for 3-regular MAX-CUT.

Theorem 16.11 (Imapproximability of 3-regular MAX-CUT, [BK99]). *For every $\epsilon > 0$, it is NP-hard to approximate 3-regular MAX-CUT within a factor of $r + \epsilon$, where $r \approx 0.997$.*

If we assume the Exponential Time Hypothesis (ETH), which a plausible assumption in theoretical computer science, we can get stronger lower bound for MAX-CUT.

Definition 16.9 (Exponential Time Hypothesis (ETH), [IP01]). There exists a constant $\epsilon > 0$ such that the time complexity of n-variable 3SAT is at least $2^{\epsilon n}$.

Theorem 16.12 ([FLP16]). *Assuming ETH, there exists a constant $0 < r' < 1$ such that no $2^{o(n)}$-time algorithm can r'-approximate the MaxCut of an n-vertex, 5-regular graph.*

With Theorem 16.11 and Theorem 16.12, we can prove the following inapproximability result for the ℓ_2-regression problem with hidden signs.

Theorem 16.13 (Lower bound of ℓ_2-regression with hidden signs)**.** *There exists a constant $\epsilon > 0$ such that it is* NP*-hard to $(1+\epsilon)$-approximate*

$$\min_{z\in\mathbb{R}^n} \||Wz| - y\|_2, \tag{16.1}$$

where $W \in \{0,1\}^{m\times n}$ is row 2-sparse and $y \in \{0,1\}^m$.

Furthermore, assuming ETH*, there exists a constant ϵ' such that no $2^{o(n)}$-time algorithm can ϵ'-approximate Eq. (16.1).*

Proof. Given a 3-regular $\mathsf{MAX\text{-}CUT}$ instance G, we construct an ℓ_2-regression instance (W, y) with $W \in \{0,1\}^{m'\times n}$ and $y \in \{0,1\}^{m'}$ where $m' = m + cn = (1 + 3c/2)m$ and $c = 10^6$ as follows.

- For each $i \in [m]$, let the i-th edge of G be $e_i = \{u, v\}$. We set $W_{i,*}$ to be all zeros except the u-th and v-th coordinates being one. That is, we add a constraint $|z_u + z_v|$. And we set $y_i = 0$.
- For each $j \in [n]$, we set $W_{m+c(j-1)+1,*}, \ldots, W_{m+cj,*}$ to be all zero vectors except the j-th entry being one. That is, we add c constraints of the form $|z_j|$. And $y_{m+c(j-1)+1} = \cdots = y_{m+cj} = 1$.

Completeness. Let opt be the optimal value of max-cut of G and let S_{opt} be the optimal subset. Then, for each $u \in S_{\mathsf{opt}}$, we set $z_u = 1$; and for $u \notin S_{\mathsf{opt}}$, we set $z_u = -1$. For the first type constraints $|z_u + z_v|$, if u and v are cut by S_{opt}, then $|z_u + z_v| = 0$; otherwise $|z_u + z_v| = 2$. For the second type constraints $|z_j|$, all of them are satisfied by our assignment. Thus, $\|Wz - y\|_2^2 = 4(m - \mathsf{opt})$.

Soundness. Let η be a constant such that $r < \eta < 1$, where r is the approximation lower bound in Theorem 16.11. Let $\delta = \frac{1-\eta}{10c}$. We will show that, if there exits a z such that $\|Wz - y\|_2^2 \leq \delta m'$, then we can recover a subset S with cut-size ηm.

It is easy to see that the optimal solution lies in $[-1, 1]^n$. Since for $z \notin [-1, 1]^n$, we can always transform it to a new vector $z' \in [-1, 1]^n$ such that $\|Wz' - y\|_2 \leq \|Wz - y\|_2$.

Suppose $z \in \{-1, 1\}^n$ is a Boolean vector. Then, we can pick $S = \{i \in [n] : z_i = 1\}$. We have the cut-size of S is

$$\begin{aligned}
|E(S, V \backslash S)| &\geq m - \delta m'/4 \\
&= m - \delta(1 + 3c/2)m/4 \\
&= (1 - \delta/4 - 3c\delta/8)m \\
&\geq \eta m,
\end{aligned}$$

where the last step follows from $\delta \leq \frac{8(1-\eta)}{2+6c}$.

For general $z \in [-1, 1]^n$, we first round z by its sign: let $\overline{z}_i = \text{sign}(z_i)$ for $i \in [n]$. We will show that

$$\|W\overline{z} - y\|_2^2 - \|Wz - y\|_2^2 \leq \frac{48}{c} m$$

which implies

$$\begin{aligned}
\|W\overline{z} - y\|_2^2 &= \|Wz - y\|_2^2 + (\|W\overline{z} - y\|_2^2 - \|Wz - y\|_2^2) \\
&\leq \delta m' + \frac{48}{c} m.
\end{aligned}$$

Then, we have the cut-size of S is

$$\begin{aligned}
|E(S, V \backslash S)| &\geq m - (\delta m' - 48m/c)/4 \\
&= (1 - \delta/4 - 3c\delta/8 - 12/c)m \\
&\geq \eta m,
\end{aligned}$$

where the last step follows from $\delta \le \frac{8(1-\eta-12/c)}{2+6c}$.

Let $\Delta_i := |\overline{z}_i - z_i| = 1 - |z_i| \in [0,1]$. We have

$$\begin{aligned}
\|W\overline{z} - y\|_2^2 - \|Wz - y\|_2^2 = & \sum_{i=1}^{m} (\overline{z}_{u_i} + \overline{z}_{v_i})^2 - (z_{u_i} + z_{v_i})^2 + c \cdot \sum_{j=1}^{n} (|\overline{z}_j| - 1)^2 - (|z_j| - 1)^2 \\
= & \sum_{i=1}^{m} (\overline{z}_{u_i} + \overline{z}_{v_i})^2 - (z_{u_i} + z_{v_i})^2 - c \cdot \sum_{j=1}^{n} (|z_j| - 1)^2 \\
= & \sum_{i=1}^{m} (\overline{z}_{u_i} + \overline{z}_{v_i})^2 - (z_{u_i} + z_{v_i})^2 - c \cdot \sum_{j=1}^{n} \Delta_j^2 \\
\le & \sum_{i=1}^{m} 4|\Delta_{u_i} + \Delta_{u_j}| - c \cdot \sum_{j=1}^{n} \Delta_j^2 \\
= & \sum_{i=1}^{n} 12\Delta_i - c\Delta_i^2 \\
\le & \frac{72}{c} n \\
= & \frac{48}{c} m,
\end{aligned}$$

where the first step follows by the construction of W and y. The second step follows from $|\overline{z}_j| = 1$ for all $j \in [n]$. The third step follows from the definition of Δ_j. The forth step follows from $|z_{u_i} + z_{u_j}| \in [0,2]$. The fifth step follows from the degree of the graph is 3. The fifth step follows from the minimum of the quadratic function $12x - cx^2$ in $[0,1]$ is $\frac{72}{c}$. The last step follows from $m = 3n/2$.

Therefore, by the completeness and soundness of reduction, if we take $\epsilon \in (0,\delta)$, Theorem 16.11 implies that it is NP-hard to $(1+\epsilon)$-approximate the ℓ_2-regression, which completes the proof of the first part of the theorem.

For the furthermore part, we can use the same reduction for a 5-regular graph. By choosing proper parameters (c and δ), we can use Theorem 16.12 to rule out $2^{o(n)}$-time algorithm for $O(1)$-factor approximation. We omit the details since they are almost the same as the first part. $\square$

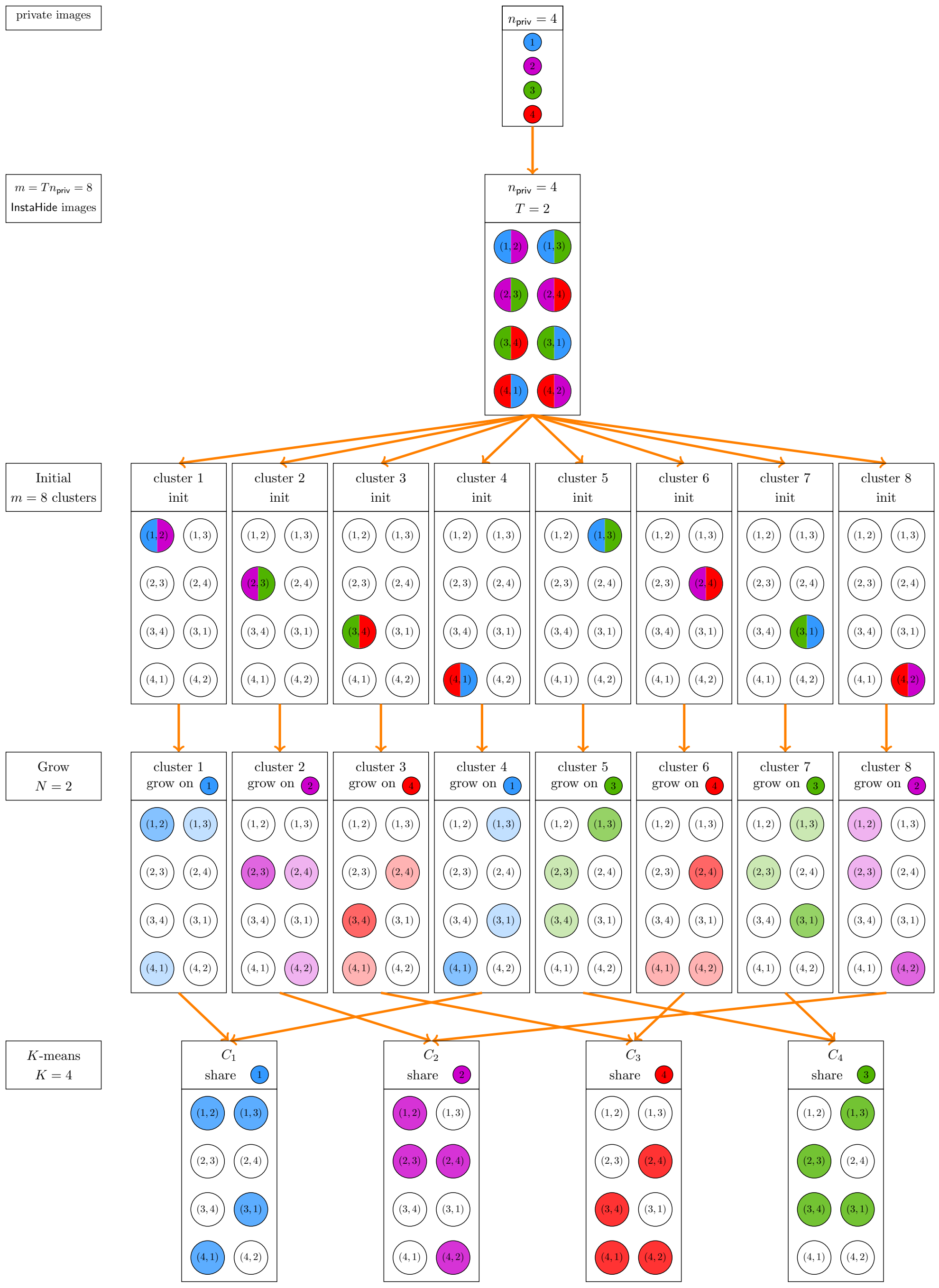


Figure 16.1: An example about the cluster step in [CDG+20] for $T = 2$ and $n_{\mathsf{priv}} = 4$. First, starting from each InstaHide image (top), the algorithm grows cluster S_i with size 3 (middle). Then, we use K-means for $K = 4$ to compute 4 groups $C_1, \dots, C_4$ (bottom), these groups each correspond to one original image.

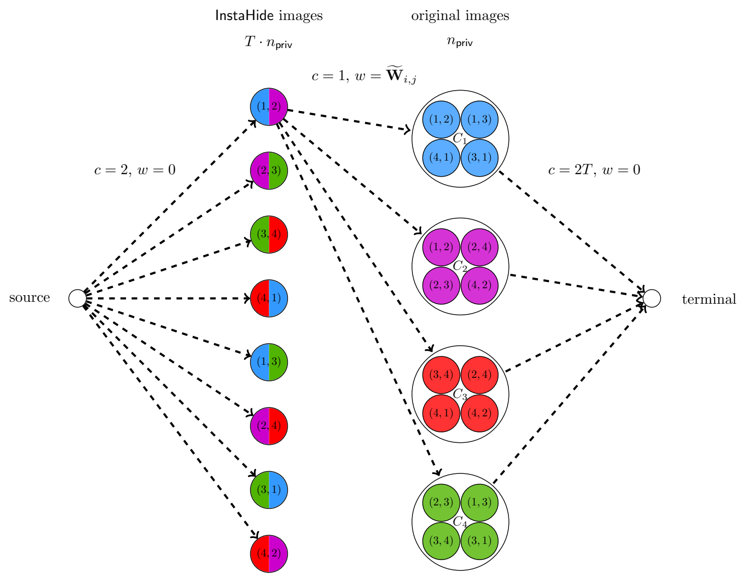


Figure 16.2: The construction of the graph for min-cost max flow. c denotes the flow capacity of the edge, and w denote the weight of the edge. The graph contains $T \cdot n_{\mathsf{priv}}$ nodes for each InstaHide images, n_{priv} nodes for each original images, a source and a terminal. There are three types of edges: i) (left) from the source to each InstaHide image node, with flow capacity 2 and weight 0; ii) (middle) from each InstaHide image node i to each original image node j, with flow capacity 1 and weight $\widetilde{\mathbf{W}}_{i,j}$; iii) (right) from each original image node to the terminal, with flow capacity $2T$ and weight 0.

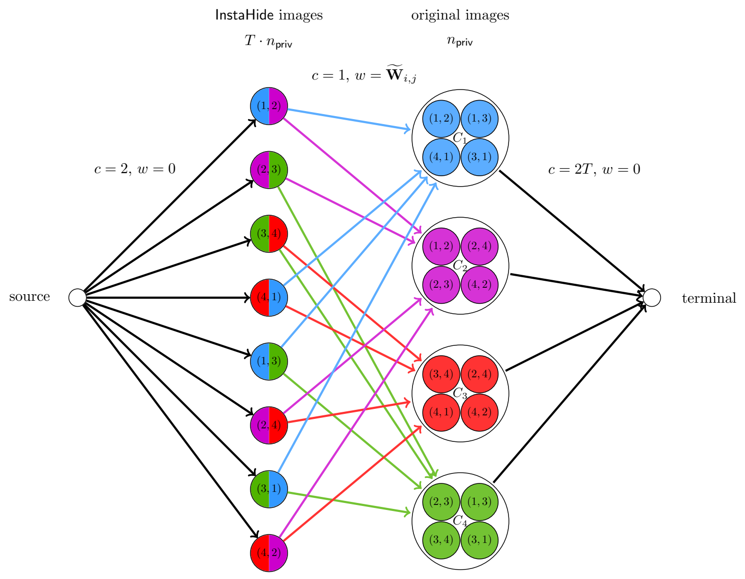


Figure 16.3: The result of solving the min-cost flow in Figure 16.2. Each InstaHide image is assigned to two clusters, which ideally correspond to two original images. In reality, a cluster may not contain all InstaHide images that share the same original image.

Chapter 17: Symmetric Boolean Factor Analysis with Application to Private Learning

17.1 Introduction

Nonnegative Matrix Factorization (NMF) [AGKM12, Moi13, RSW16, SWZ17, SWZ19] is a fundamental problem with a wide range of applications including image segmentation [LS99, SL11], document clustering [XLG03], financial analysis [dFDRC08], music transcription [SB03], and communication complexity [AUY83, Nis91]. Roughly, given an $n_1 \times n_2$ matrix $\mathbf{M}$ and a rank parameter $r > 0$, the goal is to find $n_1 \times r$ matrix $\mathbf{W}_1$ and an $n_2 \times r$ matrix $\mathbf{W}_2$ for which $\mathbf{W}_1\mathbf{W}_2^\top$ best approximates $\mathbf{M}$.

As we discuss in Section 17.1.2, a number of different variants of NMF have been studied in this literature, e.g. 1) constraining the factorization to be *symmetric* in the sense that $\mathbf{W}_1 = \mathbf{W}_2$ [DHS05, ZS05, YHD+12, YGL+13, CRDH08, KG12, HXZ+11, WLW+11, KDP12, ZWA13], 2) constraining the factors to be binary-valued [BBB+19, FGL+19, CIK17, KPRW19, ZLDZ07, RPG16], and 3) constraining them to be sparse [Hoy04, Gil12, KP08, SO06, GC05, ZFRK10]. It turns out that in many situations, it is fruitful to combine all of these desiderata. In this chapter, we study the following hybrid of these many variants of NMF.

Suppose we are given a symmetric nonnegative matrix $\mathbf{M} \in \mathbb{R}^{m\times m}_{\geq 0}$, as well as parameters $k, r \in \mathbb{N}$. Consider the following optimization problem

$$\min_{\mathbf{W}\in\mathcal{S}_{m,r,k}} \|\mathbf{M} - \mathbf{W}\mathbf{W}^\top\|_0 \qquad \text{for} \qquad \mathcal{S}_{m,r,k} = \{\mathbf{W} \in \{0,1\}^{m\times r} : \|\mathbf{W}_{j,*}\|_0 = k \,\forall\, j \in [m]\}, \tag{17.1}$$

where $\|\cdot\|_0$ denotes the number of nonzero entries, where matrix multiplication is either over the reals or over the Boolean semiring[1]. We will refer to this problem as SSBMF (sparse symmetric Boolean matrix factorization) in the sequel.

[1]In the Boolean semiring, addition is given by logical OR, and multiplication is given by logical AND

The particular set $\mathcal{S}_{m,r,k}$ we optimize over turns out to have a number of natural motivations in graph clustering, combinatorics of hypergraphs, and more recently, ML security.

Clique Decomposition For instance, consider the problem of identifying community structure in a social network. In the real world, there will be overlaps between communities, and a natural goal might be to identify a collection of them that covers the graph but such that every node only occurs in a limited number of communities. In (17.1), we can think of $\mathbf{M}$ as the adjacency matrix of some graph G. Now note that over the Boolean semiring, if we had $\mathbf{M} = \mathbf{W}\mathbf{W}^\top$, then $\mathbf{W}$ would encode a *clique cover* of G where every vertex belongs to a small number of cliques. Indeed, we can think of the parameter r as the total number of cliques in the cover and k as the number of cliques to which any one of the m vertices belongs, in which case the nonzero entries in the j-th column of $\mathbf{W}$ indicate which vertices belong to the j-th clique.

Line Graphs of Hypergraphs In combinatorics, there is a large body of work on the problem of recovering hypergraphs from their *line graphs* [Rou73, Leh74, Lov77, Sys82, Whi92, DS95, JKL97, MT97, SST05, LTVM15], which turns out to be equivalent to the version of (17.1) when the input matrix $\mathbf{M}$ admits an exact factorization. We can regard any $\mathbf{W} \in \mathcal{S}_{m,r,k}$ as the incidence matrix of a k-uniform hypergraph H with m hyperedges and r vertices, so if we work over the Boolean semiring, $\mathbf{M} \triangleq \mathbf{W}\mathbf{W}^\top$ is exactly the adjacency matrix for the line graph of H, namely the graph whose vertices correspond to hyperedges of H and whose edges correspond to pairs of hyperedges which overlap on at least one vertex of H.

When $k = 2$, Whitney's isomorphism theorem [Whi92] characterizes which graphs are uniquely identified by their line graphs; in our notation, this theorem characterizes when one can uniquely identify $\mathbf{W}$ (up to permutation) from $\mathbf{M} = \mathbf{W}\mathbf{W}^\top$. In this case, [Rou73, Leh74, Sys82, DS95, LTVM15] have given efficient algorithms for recovering $\mathbf{W}$ from $\mathbf{M}$. Unfortunately, for $k > 2$, no analogue of Whitney's theorem exists [Lov77]. In fact, even determining whether a given graph is the line graph of some k-uniform hypergraph is NP-complete [PRT81]. Rephrased in the language of (17.1), this implies the following worst-case hardness result for sparse symmetric Boolean NMF:

Theorem 17.1 ([PRT81])**.** *If $\mathsf{P} \neq \mathsf{NP}$, no polynomial-time algorithm can take an arbitrary matrix $\mathbf{M}$ and decide whether* (17.1) *has zero objective value.*

A number of results for reconstructing a hypergraph H from its line graph are known under additional assumptions on H [JKL97, MT97, SST05]. In a similar spirit, in this chapter we ask whether there are natural *average-case* settings where one could hope to do this.. One such setting is suggested by the following recent application of SSBMF to ML security.

Attacks on InstaHide Interestingly, SSBMF has arisen implicitly in a number of works [CLSZ21, CDG$^+$20, HST$^+$20] attacking InstaHide, a recently proposed scheme for privately training neural networks on image data [HSLA20b]. At a high level, the premise for InstaHide was to train on a synthetic dataset essentially consisting of random linear combinations of k images from the private dataset [ZCDLP18], rather than on the private dataset itself.

As we discuss in Section 17.5, attacking this scheme amounts to a certain natural variant of the classic k-sum problem, and the aforementioned attacks reduce from solving this problem to solving an instance of SSBMF. Roughly speaking, they show how to form the Gram matrix $\mathbf{M}$ whose rows and columns are indexed by images in the synthetic dataset such that the (i, j)-th entry of $\mathbf{M}$ is 1 if the sets of private images that give rise to synthetic images i and j overlap, and 0 otherwise. Similar to the clique cover example above, we can then think of the optimal $\mathbf{W}$ in (17.1) as encoding which private images were used to generate the synthetic images.

We emphasize that the attacks on InstaHide [CLSZ21, CDG+20, HST+20] were able to devise efficient algorithms for SSBMF precisely because the instances that arose there were average-case: concretely, the rows of the optimal $\mathbf{W}$ were random k-sparse binary vectors.

That said, the key drawback of these algorithms is that they either lack provable guarantees [CDG+20], require extremely large m [CLSZ21], or only apply to $k = 2$ [HST+20].

17.1.1 Our results

Motivated by the average-case version of SSBMF that arises in the above security application, as well as the shortcomings in the aforementioned works in this setting, in this chapter we focus on the following distributional assumption:

Assumption 17.2. Every row of $\mathbf{W} \in \{0,1\}^{m\times r}$ is an independent, uniformly random k-sparse bitstring, and the algorithm takes as input the matrix $\mathbf{M} = \mathbf{W}\mathbf{W}^\top$ over the Boolean semiring.

Remark 17.1. Note that an algorithm that works under Assumption 17.2 can easily be modified to handle the setting where $\mathbf{M} = \mathbf{W}\mathbf{W}^\top$ over the integers/reals instead of the Boolean semiring. The reason is that the matrix $\mathbf{M}'$ whose (i, j)-th entry is $\mathbb{1}\mathbf{M}_{ij} > 0$ satisfies $\mathbf{M}' = \mathbf{W}\mathbf{W}^\top$ over the Boolean semiring. So in the "realizable" setting of Assumption 17.2, working over the Boolean semiring is more general than working over the integers/reals.

Our main result is to give a polynomial-time algorithm for (17.1) in this setting:

Theorem 17.3 (Average-case guarantee, informal version of Theorem 17.25)**.** *Fix any integer $2 \le k \le r$, failure probability $\delta \in (0, 1)$, and suppose $m \ge \widetilde{\Omega}(rk \log(1/\delta))$. Suppose $\mathbf{W}, \mathbf{M}$ satisfy Assumption 17.2, where matrix multiplication is over the Boolean semiring. There is an algorithm which takes as input $\mathbf{M}$, runs in $O(m^{\omega+1})$ time*[2] *and,*

[2] $\omega \approx 2.373$ is the exponent of matrix multiplication.

with probability $1-\delta$ over the randomness of $\mathbf{W}$, outputs a matrix $\widehat{\mathbf{W}} \in \{0,1\}^{m\times r}$ whose columns are a permutation of those of $\mathbf{W}$.

We stress that the minimum m for which Theorem 17.3 holds is a fixed polynomial in r, k; in contrast, the only known provable result to work in this setting for general k [CLSZ21] used a rather involved combinatorial argument to "partially" factorize $\mathbf{M}$ when $m = \Omega(n^{k-2/k})$.

Furthermore, we emphasize that our dependence on r is *nearly optimal*:

Remark 17.2. The connection to line graphs makes clear why m must be at least $\Omega(r\log r)$ for unique recovery of $\mathbf{W}$ (up to permutation) from $\mathbf{M} = \mathbf{W}\mathbf{W}^\top$ to be possible, even for $k=2$. In this case, $\mathbf{M}$ is simply the adjacency matrix for the line graph of a random (multi)graph G given by sampling m edges with replacement. It is well known that such a graph is w.h.p. not connected if $m = o(r\log r)$ [ER60], so by Whitney's theorem there will be multiple non-isomorphic graphs for which $\mathbf{M}$ is the adjacency matrix of their line graph.

We remark that in the language of line graphs, Theorem 17.3 says that *whereas it is* NP*-complete to even recognize whether a given graph is a line graph of some hypergraph in the worst case, reconstructing a* random *hypergraph from its line graph is tractable.*

To prove Theorem 17.3, we design an algorithm that first bootstraps the "pairwise" information present in $\mathbf{M}$ into third-order information in the form of the tensor $\mathbf{T} = \sum_{i=1}^r \mathbf{W}_i^{\otimes 3}$, where $\mathbf{W}_i$ is the i-th column of $\mathbf{W}$ (see Section 17.2.1 of the technical overview for a summary of how $\mathbf{T}$ is constructed from $\mathbf{M}$). Once $\mathbf{T}$ has been constructed, we would like to invoke standard algorithms for tensor decomposition to recover the $\mathbf{W}_i$'s.

However, the main technical hurdle we must overcome before we can apply tensor decomposition is to prove that $\mathbf{W}$ is full column rank with high probability (see Theorem 17.9). As we discuss in Section 17.2.2 of the overview, this can be done with a straightforward net argument if m scales at least quadratically in r. As we are interested in getting an optimal dependence on m in terms of r however, the bulk of the technical content in this chapter goes into showing this holds even when m scales near-linearly in r. To achieve this, our analysis appeals to an array of technical tools from discrete probability, e.g. estimates for binary Krawtchouk polynomials, a group-theoretic Littlewood-Offord inequality, and modern tools [FKS20] for showing nonsingularity of random *square* Boolean matrices.

Connection to the k-sum problem As we alluded to above, the aforementioned attacks on InstaHide elucidated a simple connection between SSBMF and the following natural variant of k-sum that we call *batched-k-vector sum*, or BkV-SUM for short, for which Theorem 17.3 can also be leveraged to obtain average-case guarantees.

Suppose there is a database $\mathbf{X}$ which is a list of d-dimensional vectors $x_1, \cdots, x_r$. For a fixed integer k, we are given vectors $y_1, \cdots, y_m$, where for each $j \in [m]$, we promise that there is a set S_j with size k for which $y_j = \sum_{i \in S_j} x_i$. Given $y_1, \cdots, y_m$, our goal is to recover sets $S_1, \cdots, S_m$, even if $\mathbf{X}$ is unknown to us. We refer to Definition 17.4 for a formal definition of BkV-SUM.

[CLSZ21, CDG+20] show that when x_i are Gaussian vectors or real images, it is possible to construct a *similarity oracle* that, given any y_i, y_j, returns the number of x's they share (i.e., $|S_i \cap S_j|$). In the presence of such an oracle, it is easy to see that there is a reduction from BkV-SUM to SSBMF.

For reconstruction attacks however, we would like even finer-grained information, e.g. the actual entries of the vectors x_i used to generate the y_i's. We give an algorithm for this that, given the information obtained by SSBMF, recovers all of the "heavy" coordinates of the x_i's.

Theorem 17.4 (Informal version of Theorem 17.29). *Fix any $k \in \mathbb{N}$ and failure probability $\delta \in (0,1)$, and suppose $m \geq \widetilde{\Omega}(rk \log(d/\delta))$. Given a synthetic dataset of m d-dimensional vectors generated by batched k-vector sum, together with its similarity* *oracle, there is an $O(m^{\omega+1} + d \cdot r \cdot m)$-time algorithm for approximately recovering the magnitudes of the "heavy" coordinates of every vector in the original dataset $\mathbf{X} = \{x_1, \ldots, x_r\}$. Here, a coordinate of a vector is "heavy" if its magnitude is $\Omega(k)$ times the average value of any original vector in that coordinate.*

In fact, this theorem applies even if we only get access to the *entrywise absolute values* of the y_i's, yielding the first provable attack on InstaHide which is polynomial in all parameters m, d, k, r (see Section 17.5.3 for details).

Worst-Case Guarantee Finally, we observe for the worst-case version of (17.1), it is quite straightforward to get a quasipolynomial-time approximation algorithm by setting up an appropriate CSP and applying known solvers [MM15]:

Theorem 17.5 (Worst-case guarantee, informal version of Theorem 17.32)**.** *Given a symmetric matrix $\mathbf{M} \in \mathbb{Z}^{m \times m}$, rank parameter $r \leq m$, and accuracy parameter $\epsilon \in (0,1]$, there is an algorithm that runs in $m^{O(\epsilon^{-1}k^2 \log r)}$ time and outputs $\widehat{\mathbf{W}} \in \mathcal{S}_{m,r,k}$ such that*

$$\|\mathbf{M} - \widehat{\mathbf{W}}\widehat{\mathbf{W}}^\top\|_0 \leq \min_{\mathbf{W} \in \mathcal{S}_{m,r,k}} \|\mathbf{M} - \mathbf{W}\mathbf{W}^\top\|_0 + \epsilon m^2,$$

where matrix multiplication can be over $\mathbb{Z}$ or over the Boolean semiring.

17.1.2 Related work

Nonsingularity of Boolean Matrices The nonsingularity of random matrices with binary-valued entries has been well-studied in random matrix theory, especially in the setting where each entry is an i.i.d. random draw from $\{\pm 1\}$ (see for example, [CV10, RV08, CV08]). However, in our setting, there exists some dependence within a row since we require that the row sum equals to k. While there have been a number of works under this setting (for example [Ngu13, FJLS20, Jai21, AHP20, FKS20]), they all studied the case when k is large (i.e., when $k = \Omega(\log r)$).

For $k = O(1)$, there was a long-standing conjecture from [Vu08] that the adjacency matrix of a random k-regular graph is singular with probability $o(1)$ when $3 \leq k < r$. A recent work [Hua18] fully resolved this conjecture using a local CLT and large deviation estimate. In contrast, our $\mathbf{W}$ corresponds to the adjacency matrix of a bipartite graph with only *left-regularity*. Furthermore, our proof uses only elementary techniques, and our emphasis is on showing that for somewhat tall rectangular matrices, the columns are linearly dependent with probability $1/\operatorname{poly}(r)$ for any constant $k \geq 1$.

Fountain Codes Another line of work that studies the nonsingularity of random matrices with discrete-valued entries is that of *fountain codes* [Mac05, Lub02]. While an exposition of this literature would take us too far afield, at a high level many of these works are interested in establishing that for an $m \times r$ matrix whose rows are independently sampled from a particular distribution over $\mathbb{F}_q^r$ (ideally supported over sparse vectors to allow for fast decoding), the so-called *MDS* property holds. This is the property that any $r \times r$ submatrix of the matrix is full rank. Note that this is an even stronger property than linear independence of the columns. While these works have considered distributions somewhat similar to ours (e.g. [AD14b] consider a distribution over vectors of sparsity *at most* k where one samples k coordinates from $[r]$ with replacement, assigns those entries to be random from $\mathbb{F}_q$, and sets all other entries to be zero), to our knowledge they all require the sparsity of the rows to be at least logarithmic in r.

Comparison to [ADM$^+$18] Perhaps the work most closely related to ours is that of [ADM$^+$18]. In our notation, they consider the following problem. There is an unknown matrix $\mathbf{W} \in \mathbb{R}^{m \times r}$, and for a fixed parameter $\ell \in \mathbb{N}$, they would like to recover $\mathbf{W}$ given the ℓ-th order tensor $\sum_i \mathbf{W}_i^{\otimes \ell}$. They show that when the columns of $\mathbf{W}$ are sampled from a sufficiently anti-concentrated distribution and $r \leq m^{\lfloor \frac{\ell-1}{2} \rfloor}$, then they can recover the $\mathbf{W}_i$'s from the tensor.

While this appears on the surface to be quite related both to SSBMF and to our choice of algorithm, we emphasize a crucial distinction. In SSBMF we work with $\ell = 2$ as we are simply given the matrix $\mathbf{W}\mathbf{W}^\top = \sum_i \mathbf{W}_i^{\otimes 2}$, and for this choice of ℓ, the guarantees of [ADM$^+$18] are vacuous as $\lfloor \frac{\ell-1}{2} \rfloor = 0$. There is a good reason for this: their algorithm is based on directly applying tensor decomposition to the tensor $\sum_i \mathbf{W}_i^{\otimes \ell}$ that is given as input. In contrast, a key innovation in our work is to "bootstrap" a higher-order tensor out of only the lower-order information given by $\mathbf{M} = \mathbf{W}\mathbf{W}^\top$.

Of course, the other difference between [ADM$^+$18] and our work is that they work in a smoothed analysis setting, while we work in a random setting. As a result, even though a key step in both our analysis and theirs is to show that under our respective distributional assumptions, the columns of $\mathbf{W}$ are full rank with high probability, our techniques for doing so are very different. We also remark that while the smoothed analysis setting is qualitatively more flexible, their analysis suffers the same drawback as the abovementioned works on nonsingularity of Boolean matrices and fountain codes, namely it cannot handle the setting of Assumption 17.2 if the rows of $\mathbf{W}$ are $o(\log r)$-sparse.

Standard NMF Lastly, we give an overview of the large literature on nonnegative matrix factorization and its many variants. The most standard setting of NMF is $\min_{\mathbf{U},\mathbf{V} \geq 0} \|\mathbf{M} - \mathbf{W}_1 \mathbf{W}_2^\top\|$, where $\mathbf{W}_1, \mathbf{W}_2$ range over all possible $m \times r$ matrices with nonnegative entries. This has been the subject of a significant body of theoretical and applied work, and we refer to the survey [Gil12] for a comprehensive overview of this literature.

Symmetric NMF When we constrain $\mathbf{W}_1$ and $\mathbf{W}_2$ to be equal, we obtain the question of *symmetric* NMF, which is closely related to kernel k-means [DHS05] and has been studied in a variety of contexts like graph clustering, topic modeling, and computer vision [ZS05, YHD$^+$12, YGL$^+$13, CRDH08, KG12, WLW$^+$11, ZWA13]. Symmetric NMF has received comparatively less attention but is nevertheless a popular clustering technique [HXZ+11, KDP12, DHS05] where one takes the input matrix $\mathbf{M}$ to be some similarity matrix for a collection of data points and interprets the factorization $\mathbf{W}$ as specifying a soft clustering of these points into r groups, where $\mathbf{W}_{i,\ell}$ is the "probability" that point i lies in cluster ℓ.

While there exist efficient provable algorithms for asymmetric NMF under certain separability assumptions [AGKM12, Moi13], the bulk of the work on symmetric NMF has been focused on designing iterative solvers for converging to a stationary point [HXZ+11, KDP12, VGL+16, LHW17, ZLLL18]; we refer to the recent work of [DBd19] for one such result and an exposition of this line of work.

Binary Matrix Factorization In NMF, when we constrain $\mathbf{W}_1, \mathbf{W}_2$ to be matrices with binary-valued entries, the problem becomes *binary matrix factorization* [ZLDZ07, CIK17, BBB+19, FGL+19, KPRW19, KT21b, KT19], which is connected to a diverse array of problems like LDPC codes [RPG16], optimizing passive OLED displays [KPRW19], and graph partitioning [CIK17].

With the exception of [ZWA13, KT21b] which considered community detection with overlapping communities, most works on binary matrix factorization focus on the asymmetric setting. Over the reals, this is directly related to the bipartite clique partition problem [Orl77, CHHK14, CIK17]. [KPRW19] gave the first constant-factor approximation algorithm for this problem that runs in time $2^{O(r^2 \log r)} \operatorname{poly}(m)$. Our Theorem 17.5 also extends to this asymmetric setting (see Theorem 17.31); see Remark 17.3 for a comparison.

On the other hand, over the Boolean semiring, this problem is directly related to the *bipartite clique cover* problem. Also called *Boolean factor analysis*, the *discrete basis problem*, or *minimal noise role mining*, it has received substantial attention in the context of topic modeling, database tiling, association rule mining, etc. [SBM03, ŠH06, BV10, MMG+08, VAG07, LVAH12, MSVA16]. The best algorithm in this case, due to [FGL+19], runs in time $2^{2^{O(r)}} \cdot m^2$, matching the lower bound of [CIK17]. [EU18] considered the decision version of this problem, whose goal is to decide the minimum clique cover size r, and proved that it is NP-hard if $k \geq 5$. A more general version of this problem was studied by [AGSS12] with applications to community detection. Our worst-case algorithm also applies to this setting (see Corollary 17.33). In particular, even without the sparsity condition, the running time still matches the lower bound (see Remark 17.4).

Over the reals, [KT21b] gives sufficient conditions on $\mathbf{W} \in \{0,1\}^{m \times r}$ for which one can recover the matrix $\mathbf{W}\mathbf{W}^\top$, namely that the set of Hadamard products between columns of $\mathbf{W}$ spans a $(\binom{r}{2}+1)$-dimensional space. In this case, they give an SDP-based algorithm for recovering $\mathbf{W}$. While it is conceivable that for sufficiently large m, $\mathbf{W}$ also satisfy this property with high probability under our Assumption 17.2, in our setting an even simpler condition suffices, namely that $\mathbf{W}$ has full column rank. As we will see, even this turns out to be quite nontrivial to show when m is small.

More generally, we refer to the comprehensive survey of [MN20] for other results on BMF.

Sparse NMF In sparse NMF, one enforces that the factors $\mathbf{W}_1, \mathbf{W}_2$ must be sparse matrices [Hoy04, Gil12] as this can lead to more interpretable results, e.g. in speech separation [SO06], cancer diagnosis [GC05] and facial expression recognition [ZFRK10]. Hoyer [Hoy04] initiated the study of this problem based on the observation that standard NMF usually outputs a sparse solution and modeled Sparse NMF by adding a sparsity (L_1-norm divided by L_2-norm) penalty term to the minimization. Gillis [Gil12] studied this problem from a geometric perspective via data preprocessing. Apart from this regularization approach, there are also works that deal with exact sparsity constraints [PP12, CG19].

Miscellaneous Notation

We use $\mathbb{F}_q$ to denote finite field and sometimes use $\mathbb{F}$ to denote $\mathbb{F}_2$. For any nonnegative function f, we use $\widetilde{O}(f)$ to denote $f \operatorname{poly}(\log f)$ and use $\widetilde{\Omega}(f)$ to denote $f / \operatorname{poly}(\log f)$.

For a vector $x \in \mathbb{R}^n$, we use $\operatorname{supp}(x)$ to denote the nonzero indices, $\|x\|_0$ to denote the number of nonzero entries, and $\|x\|_p$ to denote its ℓ_p norm. We use $\vec{1}_d$ to denote the all-ones vector in $\mathbb{R}^d$ and suppress the subscript when the context is clear.

For a matrix A, we use A^+ to denote the pseudo-inverse of matrix A. We use $\|A\|_F$ to denote the Frobenius norm of matrix A, $\|A\|_1 = \sum_{i,j} |A_{i,j}|$ to denote its entrywise ℓ_1 norm, $\|A\|_0$ to denote the number of nonzero entries, and $\|A\|_\infty$ to denote $\max_{i,j} |A_{i,j}|$.

Given r, k, we let $\binom{r}{[k]}$ denote the collection of all subsets of $[r]$ of size k.

17.2 Technical Overview

As the technical details for Theorems 17.4 and 17.5 are more self-contained, in this section we focus on highlighting the key technical ingredients for our main result, Theorem 17.3.

The starting point is the following thought experiment. If instead of getting access to the matrix $\mathbf{M} = \sum_{i=1}^r \mathbf{W}_i^{\otimes 2}$ over the Boolean semiring, suppose we had the *tensor*

$$\mathbf{T} = \sum_{i=1}^r \mathbf{W}_i^{\otimes 3} \tag{17.2}$$

over $\mathbb{Z}$. Provided the columns $\mathbf{W}_i$ of $\mathbf{W}$ are linearly independent over $\mathbb{R}$, then we can run a standard tensor decomposition algorithm to recover $\mathbf{W}_1, \ldots, \mathbf{W}_r$ up to permutation. As such, there are two technical steps:

- Step 1. bootstrapping the tensor $\mathbf{T}$ given only $\mathbf{M}$,
- Step 2. showing linear independence of the $\mathbf{W}_i$.

17.2.1 Bootstrapping the tensor

The key insight is that although the similarity matrix $\mathbf{M}$ only gives access to "second-order" information about correlations between the entries of $\mathbf{W}$, we can bootstrap third-order information in the form of $\mathbf{T} \triangleq \sum_{i=1}^{r} \mathbf{W}_i^{\otimes 3}$, where $\mathbf{W}_i$ is the i-th column of $\mathbf{W}$ and the tensor is defined over $\mathbb{R}$ rather than the Boolean semiring.

Given sets $S_1, \ldots, S_c \subset [r]$, let $\mu(S_1, \ldots, S_c)$ denote the probability that a randomly chosen subset T of $[r]$ of size k does not intersect any of them. It is easy to see that

$$\mu(S_1, \ldots, S_c) = \binom{r - |S_1 \cup \cdots \cup S_c|}{k} \Big/ \binom{r}{k} \triangleq \mu_{|S_1 \cup \cdots \cup S_c|}. \tag{17.3}$$

The following fact implies that μ is monotonically decreasing in $|S_1 \cup \cdots \cup S_c|$ and, importantly, that the different μ_t's are well-separated.

Fact 17.6 (Informal version of Fact 17.14). *There exist absolute constants $C, C' > 0$ for which the following holds. If $r \geq C \cdot k^2$, then for any $0 \leq t \leq 3k$, we have that $\mu_t \geq \mu_{t+1} + C'k/r$ and $\mu_t \geq 1 - O(tk/r)$.*

Hence, we first estimate μ by computing the fraction of columns of $\mathbf{M}$ which are simultaneously zero in rows corresponding to the sets $S_1, \ldots, S_c$. Provided that the number of columns m is sufficiently large, we can estimate this probability to within error $O(k/r)$, and then invert along μ to exactly recover $|S_1 \cup \cdots \cup S_c|$, from which we obtain $\mathbf{T}_{a,b,c}$ via:

$$\mathbf{T}_{a,b,c} = |S_1 \cap S_2 \cap S_3| = |S_1 \cup S_2 \cup S_3| - |S_1 \cup S_2| - |S_2 \cup S_3| - |S_1 \cup S_3| + 3k,$$

for any $a, b, c \in [m]$ with the corresponding subsets S_1, S_2, S_3.

Lemma 17.7 (Informal version of Lemma 17.15). *If $m \geq \widetilde{\Omega}(rk \log(1/\delta))$, then with probability at least $1 - \delta$ over the randomness of $\mathbf{M}$, there is an algorithm (Algorithm 114)for computing $\mathbf{T}_{a,b,c}$ for any $a, b, c \in [m]$ in time $O(m^{\omega+1})$.*

We defer a full proof of Lemma 17.7 to Section 17.4.2. As for the runtime, we can compute each slice of $\mathbf{T}$ by setting up an appropriate matrix multiplication. Then, it suffices to apply the following standard guarantee for (noiseless) tensor decomposition:

Lemma 17.8 (Jennrich's algorithm, see e.g. Lemma 17.16)**.** *Given a collection of linearly independent vectors $w_1, \dots, w_r \in \mathbb{R}^m$, there is an algorithm that takes any tensor $\mathbf{T} = \sum_{i=1}^r w_i^{\otimes 3}$, runs in time $O(m^\omega)$, and outputs a list of vectors $\widehat{w}_1, \dots, \widehat{w}_r$ for which there exists permutation π satisfying $\widehat{w}_i = w_{\pi(i)}$ for all $i \in [r]$.*

The full algorithm for recovering $\mathbf{W}$ from $\mathbf{M}$ is given in Algorithm 114 below.

Algorithm 114 Recovering $\mathbf{W}$ from $\mathbf{M}$

```
1: procedure TensorRecover($\mathbf{M}$)    ▷ $\mathbf{M} \in \{0,1\}^{m \times m}$ s.t. $\mathbf{M} = \mathbf{W}\mathbf{W}^\top$ over Boolean semiring for some $\mathbf{W} \in \mathcal{S}_{m,r,k}$
2:                                                    ▷ Form the tensor $\mathbf{T}$
3:     for $(a,b,c) \in [m] \times [m] \times [m]$ do
4:         Let $\mu_{abc}$ be the fraction of $\ell \in [m]$ for which $\mathbf{M}_{a,\ell} = \mathbf{M}_{b,\ell} = \mathbf{M}_{c,\ell}$ are all zero. Define $\mu_{ab}, \mu_{ac}, \mu_{bc}$ analogously    ▷ (Lemma 17.7)
5:         Let $t_{abc}$ be the nonnegative integer $t$ for which $\mu_t$ (see Eq. (17.3)) is closest to $\mu_{abc}$. Define $t_{ab}, t_{ac}, t_{bc}$ analogously
6:         $\mathbf{T}_{a,b,c} \leftarrow t_{abc} - t_{ab} - t_{ac} - t_{bc} + 3k$
7:     end for
8:                                                    ▷ Run Jennrich's on $\mathbf{T}$
9:     Randomly sample unit vectors $v_1, v_2 \in \mathbb{S}^{m-1}$.
10:    $\mathbf{M}_1 \leftarrow \mathbf{T}(\mathrm{Id}, \mathrm{Id}, v_1)$, $\mathbf{M}_2 \leftarrow \mathbf{T}(\mathrm{Id}, \mathrm{Id}, v_2)$
11:    Let $\widetilde{w}_1, \dots, \widetilde{w}_r \in \mathbb{R}^m$ denote the left eigenvectors outside the kernel of $\mathbf{M}_1\mathbf{M}_2^+$
12:    Round $\{\widetilde{w}_i\}$ to Boolean vectors $\{v_i\}$; let $\widehat{\mathbf{W}} \in \{0,1\}^{m \times r}$ consist of $\{w_i\}$
13:    return $\widehat{\mathbf{W}}$    ▷ Matrix $\widehat{W} \in \mathcal{S}_{m,r,k}$ which is equal to $\mathbf{W}$ up to column permutation
14: end procedure
```

17.2.2 Linear independence

To use Jennrich's, it remains to show that the columns of $\mathbf{W}$ are indeed linearly independent with high probability. This is the technical heart of our analysis. We show:

Theorem 17.9 (Linear independence of $\mathbf{W}$, informal version of Theorem 17.17)**.** *Let $\mathbf{W} \in \{0,1\}^{m\times r}$ be a random matrix whose rows are i.i.d. random vectors each following a uniform distribution over $\{0,1\}^r$ with exactly k ones. For constant $k \geq 1$ and $m = \Omega(\max(r, (r/k)\log r))$, the r columns of $\mathbf{W}$ are linearly independent in $\mathbb{R}$ with probability at least $1 - \frac{1}{\mathrm{poly}(r)}$.*

Note that by a simple net argument, when $m = \widetilde{\Omega}(r^2)$ one can show that $\mathbf{W}$ is not only full rank, but polynomially well-conditioned. But as our emphasis is on having the sample complexity m depend near-optimally on the rank parameter r, we need to be much more careful.

When k is odd, it turns out to be more straightforward to prove Theorem 17.9. In this case, to show linear independence of the $\mathbf{W}_i$'s over $\mathbb{R}$, we first observe that it suffices to show linear independence over $\mathbb{F}_2$. For a given $u \in \mathbb{F}_2^r$, one can explicitly compute the probability that $\mathbf{W}u = 0$, and by giving fine enough estimates for these probabilities based on bounds for binary Krawtchouk polynomials and taking a union bound, we conclude that the columns of W are linearly independent with high probability in this case.

This proof strategy breaks down for even k because in this case $\mathbf{W}$ is not full-rank over $\mathbb{F}_2$: the columns of $\mathbf{W}$ add up to zero over $\mathbb{F}_2$. Instead, we build on ideas from [FKS20], which studies the square matrix version of this problem and upper bounds the probability that there exists some x for which $\mathbf{W}x = 0$ and for which the most frequent entry in x occurs a fixed number of times. Intuitively, if the most frequent entry does not occur too many times, then the probability that $\mathbf{W}x = 0$ is very small. Otherwise, even if the probability is large, there are "few" such vectors. Unfortunately, [FKS20] requires $k \geq \Omega(\log r)$, and in order to handle the practically relevant regime of $k = O(1)$, we need to adapt their techniques and exploit the fact that $\mathbf{W}$ is a slightly tall matrix in our setting.

In particular, we leverage a certain group-theoretic Littlewood-Offord inequality [JŠ19] which may be of independent interest to the TCS community. More specifically, we consider two scenarios under which there is a vector x such that $\mathbf{W}x = 0$. The first one is that there is a vector x with $\mathbf{W}x = 0$ which has some zero entries, in which case we can actually still reduce to the $\mathbb{F}_2$ case as in the case of odd k. The second one is that there is a vector x with $\mathbf{W}x = 0$ where all entries of x are non-zero. In this case, since $\mathbf{1} \in \ker(\mathbf{W})$ in $\mathbb{F}_2$, we cannot consider linear independence over $\mathbb{F}_2$, and our main workaround is instead to work over the cyclic group $\mathbb{Z}_q$ where $q \in [k-1, 2k]$ is a prime.

For this latter case, we further divide into two more cases: (1) the most frequent entry in x occurs a large number of times; (2) x does not have such an entry. By a union bound over both kinds of vectors x, it suffices to show anti-concentration in the sense that we want to upper bound $\Pr_w[\langle w, x\rangle = a \pmod q]$ for a fixed x in either case. In case (1), we can use the upper bound from [FKS20]. In case (2), [FKS20] used a Littlewood-Offord-type inequality [Erd45] to upper bound this probability, which does not apply in our case because it works for real numbers and we cannot apply union bound for an infinite number of vectors. Instead, we use a group-theoretic Littlewood-Offord inequality [JŠ19] and follow the idea of [FKS20] to transform the uniform k-sparse distribution to an i.i.d. Bernoulli distribution, completing the proof.

We defer the details of Theorem 17.9 to Section 17.4.4. Altogether, this allows us to conclude Theorem 17.3, and the formal proof is in Section 17.4.5.

Roadmap In Section 17.3 we provide technical preliminaries, basic definitions, and tools from previous work. In Section 17.4 we prove our average-case guarantee, Theorem 17.3. In Section 17.5 we describe the connection between SSBMF, BkV-SUM, and private neural network training in greater detail and then prove Theorem 17.4. In Appendix 17.6 we prove our worst-case guarantee, Theorem 17.5.

17.3 Preliminaries

17.3.1 Notations

We introduce some notations and definitions we will use throughout this chapter.

For a positive integer n, we use $[n]$ to denote the set $\{1, 2, \cdots, n\}$. For a vector x, we use $\|x\|_2$ to denote its ℓ_2 norm. We use $\|x\|_1$ to denote its ℓ_1 norm.

For a matrix A, we use $\|A\|$ to denote its spectral norm. We use $\|A\|_0$ to denote the number of non-zero entries in A. We use $\|A\|_F$ to denote the Frobenius norm of A. For a square and full rank matrix A, we use A^{-1} to denote its inverse.

17.3.2 Basic Definitions

We provide a definition for Boolean semiring.

Definition 17.1. The Boolean semiring is the set $\{0, 1\}$ equipped with addition corresponding to logical OR (i.e. $x + y = 0$ if $x = y = 0$ and $x + y = 1$ otherwise) and multiplication corresponding to logical AND (i.e. $x \cdot y = 1$ if $x = y = 1$ and $x \cdot y = 0$ otherwise).

We define Bernoulli random variable as follows:

Fact 17.10. *For any $0 < c < 1$ and $0 \leq p \leq \epsilon$, one can estimate the mean of a Bernoulli random variable $\mathrm{Ber}(p)$ to error $c \cdot \epsilon$ with probability $1 - \delta$ using $O(c^{-2}\epsilon^{-1})$ samples.*

17.3.3 Discrete probability tools

Definition 17.2. Given a vector $v \in \mathbb{N}^r$, define a *fibre* of a vector to be a set of all indices whose entries are equal to a particular value.

Lemma 17.11 ([FKS20])**.** *Let $w \in \{0, 1\}^r$ be a random vector with exactly k ones where $k < r/2$. Let $q \geq 2$ be an integer and consider a fixed vector $v \in \mathbb{Z}_q^r$ whose* *largest fibre has size $r-s$. Then, for any $a \in \mathbb{Z}_q$ we have* $\Pr[\langle w, v\rangle \equiv a \pmod q] \leq P_s$ *for some $P_s = 2^{-O(sk/r)}$ if $sk = o(r)$, and $P_s = 2^{-\Omega(1)}$ if $sk = \Omega(r)$.*

Lemma 17.12 ([FKS20])**.** *For $x \in \mathbb{R}^r$ whose largest fibre has size $r-s$, and let $w \in \{0,1\}^r$ be a random vector with k ones. Then*

$$\max_{a\in\mathbb{R}} \Pr[\langle x, w\rangle = a] = O\big(\sqrt{r/(sk)}\big).$$

In the proof of Proposition 17.21, we needed the following estimate for the binary Krawtchouk polynomial:

Lemma 17.13 ([Pol19])**.** *For $k \leq 0.16r$, $\lambda \leq \frac{r}{2}$, we have*

$$|K_k^r(\lambda)| \leq \binom{r}{k} \cdot \left(1 - \frac{2k}{r}\right)^{\lambda}.$$

17.4 Average-Case Algorithm

In Section 17.4.1 we prove that the non-intersection probabilities μ_t defined in Section 17.2.1 are well-separated, which we previously stated informally as Fact 17.6. In Section 17.4.2 we show how to exploit this fact to construct the tensor $\mathbf{T}$ defined in (17.2). In Section 17.4.3 we give a sufficient condition for the tensor $\mathbf{T}$ to be decomposable.

Section 17.4.4 is the main technical component of this chapter where we show that under Assumption 17.2, the columns of $\mathbf{W}$ are linearly independent (over $\mathbb{R}$) with high probability so that we can actually apply tensor decomposition. In Section 17.4.4.1 we observe that one way to show linear independent of the columns of $\mathbf{W}$ over $\mathbb{R}$ is to show linear independence over $\mathbb{F}_2$. In Section 17.4.4.2 we handle the case of odd k by using this observation together with a helper lemma from Section 17.4.4.3 involving estimates of binary Krawtchouk polynomials. For even k, we cannot reduce to showing linear independence over $\mathbb{F}_2$, so in Section 17.4.4.4 we handle this case using a combination of ideas from [FKS20] and a group-theoretic Littlewood-Offord inequality [JŠ19].

Finally, in Section 17.4.5 we put all the ingredients together to complete the proof of Theorem 17.3.

17.4.1 Non-intersection probabilities μ_t well-separated

We begin by showing that the non-intersection probabilities μ_t defined in (17.3) are monotonically decreasing and well-separated. For ease of reading, we recall the definition of μ_t here. Given sets $S_1, \ldots, S_c \subset [r]$, let $\mu(S_1, \ldots, S_c)$ denote the probability that a randomly chosen subset T of $[r]$ of size k does not intersect any of them. Then

$$\mu(S_1, \ldots, S_c) = \binom{r - |S_1 \cup \cdots \cup S_c|}{k} \bigg/ \binom{r}{k} \triangleq \mu_{|S_1 \cup \cdots \cup S_c|}. \tag{17.4}$$

Fact 17.14 (Formal version of Fact 17.6)**.** *There exist absolute constants $C, C' > 0$ for which the following holds. If $r \geq C \cdot k^2$, then for any $0 \leq t \leq 3k$, we have that $\mu_t \geq \mu_{t+1} + C'k/r$ and $\mu_t \geq 1 - O(tk/r)$.*

Proof. For any $0 \leq t \leq r$, note that

$$\begin{aligned}
\frac{\binom{r-t}{k}}{\binom{r}{k}} - \frac{\binom{r-t-1}{k}}{\binom{r}{k}} &= \frac{r-t-1}{r} \cdot \frac{r-t-2}{r-1} \cdots \frac{r-t-k+1}{r-k+2} \cdot \left(\frac{r-t}{r-k+1} - \frac{r-t-k}{r-k+1}\right). \\
&\geq \left(1 - \frac{t+1}{r-k+2}\right)^{k-1} \cdot \frac{k}{r-k+1} \\
&\geq \left(1 - \frac{(t+1)(k-1)}{r-k+2}\right) \cdot \frac{k}{r-k+1} \geq \Omega(k/r),
\end{aligned}$$

where in the last step we used the assumptions that $r \geq \Omega(k^2)$ and $t \leq O(k)$. For the second part of the claim, we similarly have that

$$\begin{aligned}
\frac{\binom{r-t}{k}}{\binom{r}{k}} &= \frac{r-t}{r} \cdot \frac{r-t-1}{r-1} \cdots \frac{r-t-k+1}{r-k+1} \\
&\geq \left(1 - \frac{t}{r-k+1}\right)^{k} \\
&\geq 1 - \frac{tk}{r-k+1} \geq 1 - O(tk/r).
\end{aligned}$$

$\square$

17.4.2 Constructing a tensor

We now use Fact 17.14 to show how to construct the tensor $\mathbf{T}$ defined in (17.2), namely $\mathbf{T} \triangleq \sum_{i=1}^{r} \mathbf{W}_i^{\otimes 3}$.

Lemma 17.15 (Constructing a tensor, formal version of Lemma 17.7)**.** *If $m \geq \widetilde{\Omega}\left(rk\log(1/\delta)\right)$, then with probability at least $1-\delta$ over the randomness of $\mathbf{M}$, there is an algorithm for computing $\mathbf{T}_{a,b,c}$ for any $a,b,c \in [m]$ in time $O(m^{\omega+1})$.*

Proof. If entries a,b,c correspond to subsets S_1, S_2, S_3 of $[m]$, then

$$\mathbf{T}_{a,b,c} = |S_1 \cap S_2 \cap S_3| = |S_1 \cup S_2 \cup S_3| - |S_1 \cup S_2| - |S_2 \cup S_3| - |S_1 \cup S_3| + 3k. \quad (17.5)$$

By Fact 17.10 and the second part of Fact 17.14 applied to $t = |S_i \cup S_j|$ or $t = |S_1 \cup S_2 \cup S_3|$, we conclude that any $\mu_{|S_i \cup S_j|}$ or $\mu_{|S_1 \cup S_2 \cup S_3|}$ can be estimated to error $C'k/2r$ with probability $1 - \delta/m^3$ provided that

$$m \geq \Omega\left(\frac{t^2 r}{k}\log(m^3/\delta)\right).$$

Note that $t \leq 3k$, so this holds by the bound on m in the hypothesis of the lemma. By the first part of Fact 17.14, provided these quantities can be estimated within the desired accuracy, we can exactly recover every $|S_i \cup S_j|$ as well as $|S_1 \cup S_2 \cup S_3|$. So by (17.5) and a union bound over all (a,b,c), with probability at least $1-\delta$ we can recover every entry of $\mathbf{T}$.

It remains to show that $\mathbf{T}$ can be computed in the claimed time. Note that naively, each entry would require $O(m)$ time to compute, leading to $O(m^4)$ runtime. We now show how to do this more efficiently with fast matrix multiplication. Fix any $a \in [m]$ and consider the a-th slice of $\mathbf{T}$. Recall that for every $b,c \in [m]$, we would like to compute the number of columns $\ell \in [m]$ for which $\mathbf{M}_{a,\ell}, \mathbf{M}_{b,\ell}, \mathbf{M}_{c,\ell}$ are all zero (note that we can compute the other relevant statistics like the number of ℓ for which $\mathbf{M}_{a,\ell}$ and $\mathbf{M}_{b,\ell}$ are both zero in total time $O(m^3)$ across all a,b even naively).

We can first restrict our attention to the set L_a of ℓ for which $\mathbf{M}_{a,\ell} = 0$, which can be computed in time $O(m)$. Let $\mathbf{M}_a$ denote the matrix given by restricting $\mathbf{M}$ to the columns indexed by L_a and subtracting every resulting entry from 1. By design, $(\mathbf{M}_a\mathbf{M}_a^\top)_{b,c}$ is equal to the number of $\ell \in L_a$ for which $\mathbf{M}_{b,\ell} = \mathbf{M}_{c,\ell} = 0$, and the matrix $\mathbf{M}_a\mathbf{M}_a^\top$ can be computed in time $O(m^\omega)$. The claimed runtime follows. $\square$

17.4.3 Tensor decomposition

We can invoke standard guarantees for tensor decomposition to decompose $\mathbf{T}$, the key sufficient condition being that the components $\{\mathbf{W}_1, \dots, \mathbf{W}_r\}$ be linearly independent over $\mathbb{R}$.

Lemma 17.16 (Tensor decomposition, formal version of Lemma 17.8)**.** *Given a collection of linearly independent vectors $w_1, \dots, w_r \in \mathbb{R}^m$, there is an algorithm that takes any tensor $\mathbf{T} = \sum_{i=1}^r w_i^{\otimes 3}$, runs in time $O(m^\omega)$, and outputs a list of vectors $\widehat{w}_1, \dots, \widehat{w}_r$ for which there exists permutation π satisfying $\widehat{w}_i = w_{\pi(i)}$ for all $i \in [r]$.*

Proof. The algorithm is simply to run Jennrich's algorithm ([HAR70, LRA93]), but we include a proof for completeness. Pick random $v_1, v_2 \in \mathbb{S}^{m-1}$ and define $\mathbf{M}_1 \triangleq \mathbf{T}(\mathrm{Id}, \mathrm{Id}, v_1)$ and $\mathbf{M}_2 \triangleq \mathbf{T}(\mathrm{Id}, \mathrm{Id}, v_2)$. If $\mathbf{W} \in \mathbb{R}^{m\times r}$ is the matrix whose columns consist of $w_1, \dots, w_r$, then we can write $\mathbf{M}_a = \sum_{i=1}^r \langle w_i, v_a\rangle w_i w_i^\top = \mathbf{W}\mathbf{D}_a\mathbf{W}^\top$, where $\mathbf{D}_a \triangleq \mathrm{diag}(\langle w_1, v_a\rangle, \dots, \langle w_r, v_a\rangle)$. As a result, $\mathbf{M}_a\mathbf{M}_b^+ = \mathbf{W}\mathbf{D}_a\mathbf{D}_b^+\mathbf{W}^+$. This gives an eigendecomposition of $\mathbf{M}_a\mathbf{M}_b^+$ because the entries of $\mathbf{D}_a\mathbf{D}_b^+$ are distinct almost surely because $\{w_i\}$ are linearly independent. We conclude that the nontrivial eigenvectors of $\mathbf{M}_a\mathbf{M}_b^+$ are precisely the vectors $\{w_i\}$ up to permutation as claimed. Forming $\mathbf{M}_1$ and $\mathbf{M}_2$ takes $O(m^2)$ time, and forming $\mathbf{M}_a\mathbf{M}_b^+$ and computing its eigenvectors takes $O(m^\omega)$ time. $\square$

17.4.4 Linear independence of W

We now turn to proving the main technical result in this chapter, namely that for $\mathbf{W}$ satisfying Assumption 17.2, the columns of $\mathbf{W}$ are linearly independent over $\mathbb{R}$ with high probability as soon as m is near-linear in r.

Theorem 17.17 (Linear independence of $\mathbf{W}$, formal version of Theorem 17.9)**.** *Let $\mathbf{W} \in \{0,1\}^{m \times r}$ be a random matrix whose rows are i.i.d. random vectors each following a uniform distribution over $\{0,1\}^r$ with exactly k ones. For constant $k \geq 1$ and $m = \Omega(\max(r, (r/k)\log r))$, the r columns of $\mathbf{W}$ are linearly independent in $\mathbb{R}$ with probability at least $1 - \frac{1}{\mathrm{poly}(r)}$.*

17.4.4.1 Reduction to Finite Fields

We treat the cases of odd and even k separately. For the former, we will show that the columns of $\mathbf{W}$ are linearly independent over $\mathbb{F}_2$ with high probability, which by the following is sufficient for linear independence over $\mathbb{R}$:

Lemma 17.18 (Reduction from $\mathbb{F}_2$ to $\mathbb{R}$)**.** *Let $\mathbf{W} \in \{0,1\}^{m \times r}$. If the columns of $\mathbf{W}$ are linearly independent in $\mathbb{F}_2$, then they are also linearly independent in $\mathbb{R}$.*

Proof. We prove a contrapositive statement, i.e., if the columns are linearly dependent in $\mathbb{R}$, then they are still dependent in $\mathbb{F}_2$.

Let $w_1, \ldots, w_n$ be the columns of $\mathbf{W}$. If they are linearly dependent in $\mathbb{R}$, then there exists $c_1, \ldots, c_r \in \mathbb{R}$ such that $\sum_{i=1}^r c_i w_i = 0$. By Gaussian elimination, it is easy to see that $c_1, \ldots, c_r \in \mathbb{Q}$. By multiplying some common factor, we can get r integers $c_1', \ldots, c_r' \in \mathbb{Z}$ such that not all of them are even numbers and

$$c_1' w_1 + c_2' w_2 + \cdots + c_r' w_r = 0.$$

Then, apply (mod 2) to both sides of the equation and for any $j \in [m]$, the j-th entry of the resulting vector is

$$\sum_{i=1}^r c_i' \mathbf{W}_{ij} \pmod 2 = \bigoplus_{i=1}^r (c_i' \mod 2) \cdot \mathbf{W}_{ij} = 0, \tag{17.6}$$

where the first step follows from $\mathbf{W}_{ij} \in \{0,1\}$.

Define a vector $a \in \mathbb{F}_2^r$ such that $a_i := c_i' \mod 2$ for $i \in [r]$. Then, $a \neq 0$ and Eq. (17.6) implies that $\mathbf{W}a = 0$ in $\mathbb{F}_2$, which means the columns of $\mathbf{W}$ are linearly dependent in $\mathbb{F}_2$. And the claim hence follows. $\square$

17.4.4.2 Linear independence for odd k

We are now ready to handle the case of odd k. The following lemma proves the $\mathbb{F}_2$ case.

Lemma 17.19 (Linear independence in $\mathbb{F}_2$)**.** *Give two fixed positive integers n and r, for any odd positive integer k, if $m = \Omega(\max(r, (r/k)\log r))$, then with probability at least $1 - \frac{1}{\mathrm{poly}(r)}$, the columns of $m \times r$ matrix $\mathbf{W}$ are linearly independent in $\mathbb{F}_2$.*

Proof. To show that the columns of $\mathbf{W}$ are linearly independent, equivalently, we can show that $\ker(\mathbf{W}) = \emptyset$ with high probability; that is, for all $x \in \mathbb{F}_2^r \backslash \{0\}$, $\mathbf{W}x \neq 0$.

For $1 \leq \lambda \leq r$, let $P_\lambda := \Pr[\mathbf{W}u = 0]$ for $u \in \mathbb{F}_2^r$ and $|u| = \lambda$. Note that this probability is the same for all weight-λ vectors. Then, we have

$$\begin{aligned}\Pr[\ker(\mathbf{W}) \neq \emptyset] &\leq \sum_{\lambda=1}^{r} P_\lambda \cdot \left|\left\{u \in \mathbb{F}_2^r \middle| |u| = \lambda\right\}\right| \\ &\leq \sum_{\lambda=1}^{r} P_\lambda \cdot \binom{r}{\lambda}.\end{aligned}$$

Fix $\lambda \in [r]$ and $u \in \mathbb{F}_2^r$ with weight λ. Since the rows of $\mathbf{W}$ are independent, we have

$$P_\lambda = \Pr[\mathbf{W}u = 0] = \prod_{i=1}^{m} \Pr[\langle w_i, u\rangle = 0] = (\Pr[\langle w, u\rangle = 0])^m,$$

where w is a uniformly random vector in the set $\{u \in \mathbb{F}_2^r | |u| = k\}$. It's easy to see that k should be an odd number; otherwise, $\mathbf{W}\mathbf{1} = 0$.

By Proposition 17.21, we have

$$\begin{aligned}
\Pr[\ker \mathbf{W} \neq \emptyset] &\leq \sum_{\lambda=1}^{r/2} \left(\frac{1}{2}+\frac{1}{2}\left(1-\frac{2k}{r}\right)^{\lambda}\right)^{m} \cdot \binom{r}{\lambda} + \left(\frac{1}{2}\right)^{m} \cdot \binom{r}{\lambda} \\
&\leq 2^{r-1-m} + \sum_{\lambda=1}^{r/2} \left(\frac{1}{2}+\frac{1}{2}\left(1-\frac{2k}{r}\right)^{\lambda}\right)^{m} \cdot \binom{r}{\lambda} && (\textstyle\sum_{i=0}^{r/2}\binom{r}{i} = 2^{r-1}.) \\
&\leq 2^{r-1-m} + \sum_{\lambda=1}^{r/2} \left(\frac{1+k\lambda/r}{1+2k\lambda/r}\right)^{m} \cdot \binom{r}{\lambda} && ((1-\tfrac{2k}{r})^{\lambda} \leq \tfrac{1}{1+(2k\lambda)/r}.) \\
&\leq 2^{r-1-m} + \sum_{\lambda=1}^{r/2} \exp\left(-\frac{km\lambda/r}{1+k\lambda/r}\right) \cdot \binom{r}{\lambda} && (1-x \leq e^{-x}.) \\
&\leq 2^{r-1-m} + \sum_{\lambda=1}^{r/2} \left(\exp\left(-\frac{km/r}{1+k\lambda/r} + \log(er/\lambda)\right)\right)^{\lambda} && (\tbinom{r}{\lambda} \leq (er/\lambda)^{\lambda}.)
\end{aligned}$$

Suppose $m > (r/k)\log(er/\lambda) + \lambda \log(er/\lambda)$ for $0 < \lambda < r/2$, then we have

$$\begin{aligned}
-\frac{km/r}{1+k\lambda/r} + \log(er/\lambda) &< -\frac{\log(er/\lambda) + k\lambda/r \log(er/\lambda)}{1+k\lambda/r} + \log(er/\lambda) \\
&= -\log(er/\lambda) + \log(er/\lambda) \\
&= 0.
\end{aligned}$$

Note that when $0 < \lambda < r/2$, $\lambda \log(er/\lambda) \leq \Omega(r)$. Also, $(r/k)\log(er/\lambda) \leq \Omega((r/k)\log r)$. Hence, if we take $m = \Omega(r + (r/k)\log r)$, we have

$$\left(\exp\left(-\frac{km/r}{1+k\lambda/r} + \log(er/\lambda)\right)\right)^{\lambda} \leq \exp\left(\frac{-\lambda \cdot \Omega(\log(r))}{1+k\lambda/r}\right) \leq \frac{1}{\mathrm{poly}(r)}.$$

Therefore,

$$\begin{aligned}
\Pr[\ker \mathbf{W} \neq \emptyset] &\leq 2^{r-1-m} + \sum_{\lambda=1}^{r/2} \left(\exp\left(-\frac{km/r}{1+k\lambda/r} + \log(er/\lambda)\right)\right)^{\lambda} \\
&\leq 2^{-\Omega((r/k)\log r)} + \sum_{\lambda=1}^{r/2} \frac{1}{\mathrm{poly}(r)} \\
&= \frac{1}{\mathrm{poly}(r)},
\end{aligned}$$

and the lemma is then proved. $\square$

Combining Lemma 17.18 and Lemma 17.19, we immediately have the following corollary:

Corollary 17.20 (Linear independence in $\mathbb{R}$ for odd k)**.** *For $r \geq 0$, odd constant $k \geq 1$, when $m = \Omega((r/k)\log r) \geq r$, with probability at least $1 - \frac{1}{\mathrm{poly}(r)}$, the columns of matrix $\mathbf{W}$ are linearly independent in $\mathbb{R}$.*

17.4.4.3 Helper lemma

In the proof of Lemma 17.19 above, we required the following helper lemma:

Proposition 17.21. *For $1 \leq \lambda \leq \frac{r}{2}$, we have either*

$$P_\lambda \leq \left(\frac{1}{2} + \frac{1}{2}\left(1 - \frac{2k}{r}\right)^\lambda\right)^m \quad \text{and} \quad P_{r-\lambda} \leq \left(\frac{1}{2}\right)^m,$$

or

$$P_\lambda \leq \left(\frac{1}{2}\right)^m \quad \text{and} \quad P_{r-\lambda} \leq \left(\frac{1}{2} + \frac{1}{2}\left(1 - \frac{2k}{r}\right)^\lambda\right)^m.$$

Proof. We can write the probability of $\langle w, u\rangle = 0$ as follows:

$$\Pr[\langle w, u\rangle = 0] = \binom{r}{k}^{-1} \cdot \sum_{i=0,\ i \text{ even}}^{k-1} \binom{\lambda}{i}\binom{r-\lambda}{k-i}.$$

We first consider the following sum with alternating signs:

$$K_k^r(\lambda) := \sum_{i=0}^{k} \binom{\lambda}{i}\binom{r-\lambda}{k-i}(-1)^i,$$

which is the binary Krawtchouk polynomial.

Then, for $1 \leq \lambda \leq \frac{r}{2}$, we have

$$\begin{aligned}\Pr[\langle w, u\rangle = 0] &= \binom{r}{k}^{-1} \sum_{i=0,\ i \text{ even}}^{k-1} \binom{\lambda}{i}\binom{r-\lambda}{k-i} \\ &= \frac{1}{2} + \frac{K_k^r(\lambda)}{2\binom{r}{k}}.\end{aligned}$$

By symmetry, for $\frac{r}{2} < \lambda < r$,

$$\begin{aligned}\Pr[\langle w, u\rangle = 0] &= \binom{r}{k}^{-1} \sum_{i=0,\ i \text{ even}}^{k-1} \binom{\lambda}{i}\binom{r-\lambda}{k-i} \\ &= \binom{r}{k}^{-1} \sum_{i=1,\ i \text{ odd}}^{k} \binom{r-\lambda}{i}\binom{\lambda}{k-i} \\ &= \frac{1}{2} - \frac{K_k^r(r-\lambda)}{2\binom{r}{k}}.\end{aligned}$$

By Lemma 17.13, we know that for $1 \le \lambda \le \frac{r}{2}$, $|K_k^r(\lambda)| \le \binom{r}{k}\left(1 - \frac{2k}{r}\right)^{\lambda}$.

Therefore, if $K_k^r(\lambda) > 0$, then

$$\begin{aligned}P_\lambda \le\ & \Pr[\langle w, u\rangle = 0]^m \le \left(\frac{1}{2} + \frac{1}{2}\left(1 - \frac{2k}{r}\right)^{\lambda}\right)^m, \\ P_{r-\lambda} \le\ & \left(\frac{1}{m}\right)^m.\end{aligned}$$

If $K_k^r(\lambda) < 0$, then

$$\begin{aligned}P_{r-\lambda} &\le \left(\frac{1}{2} + \frac{1}{2}\left(1 - \frac{2k}{r}\right)^{\lambda}\right)^m, \\ P_\lambda &\le \left(\frac{1}{m}\right)^m.\end{aligned}$$

The proposition hence follows. $\square$

17.4.4.4 Linear independence for even k

Next, we turn to the case of even k. When k is even, we cannot use Lemma 17.18 because the matrix is not linearly independent in $\mathbb{F}_2$. Instead, we use a variant of the proof in [FKS20] to show that in this case, the linear independence of the columns of $\mathbf{W}$ still holds with high probability:

Lemma 17.22. *Give two fixed positive integers n and r, for any even positive integer k, if $m = \Omega(r + (r/k)\log r)$, then with probability at least $1 - \frac{1}{\mathrm{poly}(r)}$, the columns of $m \times r$ matrix $\mathbf{W}$ are linearly independent in $\mathbb{R}$.*

Proof. We first show that it suffices to consider the case when $\mathbf{W} \in \mathbb{Z}^{m \times r}$ is an integral matrix. Suppose the columns of $\mathbf{W}$ are not linearly independent, i.e., there exists $x \in \mathbb{Q}^r$ such that $\mathbf{W}x = 0$. Then, we will be able to multiply by an integer and then divide by a power of two to obtain a vector $c \in \mathbb{Z}^r$ with at least one odd entry such that $\langle w_i, c\rangle = 0$ for all $i \in [m]$ where $w_i^\top$ denotes the i-th row of $\mathbf{W}$.

To upper bound the column-singular probability, we have

$$\begin{aligned}\Pr[\mathbf{W} \text{ is column-singular}] \leq\ & \Pr[\mathbf{W}x = 0 \text{ for some } x \text{ with } |\operatorname{supp}(x)| < r] \\ & + \Pr[\mathbf{W}x = 0 \text{ for some } x \text{ with } |\operatorname{supp}(x)| = r].\end{aligned}$$

For the first term, it can be upper bounded by

$$\Pr[\langle w_i, x\rangle \equiv 0 \pmod 2\ \forall i \in [r] \text{ for some } x \neq \mathbf{1}],$$

where $\mathbf{1}$ is an length-r all ones vector.

Note that the reduction (Lemma 17.18) fails only when all of the entries of c are odd because $\mathbf{W1} = 0$ in $\mathbb{F}_2$ when k is even. Hence, We can use almost the same calculation in Lemma 17.19 to upper bound this probability by $r^{-\Omega(1)}$ when $m = \Omega((r/k)\log r)$.

For the second term, let $\mathcal{S}_r$ denote the set of vectors with support size r. Define a *fibre* of a vector to be a set of all indices whose entries are equal to a particular value. And define a set $\mathcal{P}$ to be

$$\mathcal{P} := \{c \in \mathbb{Z}^r : c \text{ has largest fibre of size at most } (1-\delta)r\}$$

Then, we have

$$\begin{aligned}\Pr[\mathbf{W}x = 0 \text{ for some } x \text{ with } |\operatorname{supp}(x)| = r] \leq\ & \Pr[\exists x \in \mathcal{S}_r \backslash \mathcal{P} : \langle w_i, x\rangle = 0\ \forall i \in [m]] \\ & + \Pr[\exists x \in \mathcal{S}_r \cap \mathcal{P} : \langle w_i, x\rangle = 0\ \forall i \in [m]] \\ \leq\ & B_1 + B_2.\end{aligned}$$

For B_1, let $q = k - 1$. It suffices to prove that there is no non-constant vector $c \in \mathbb{Z}_q^r$ with largest fibre of size at least $(1-\delta)r$ such that $\langle w_i, c\rangle \equiv 0 \pmod{q}$ for all $i \in [m]$. Then, by Lemma 17.11, let $t = O(r/k)$ and the probability can be upper bounded by

$$\begin{aligned}
B_1 &\le \sum_{s=1}^{\delta r} \binom{r}{s} q^{s+1} (P_s)^{m_1} \\
&\le \sum_{s=1}^{t} 2^{s \log r + (s+1)\log q - O(sk/r)\cdot m} + \sum_{s=t+1}^{\delta r} 2^{s \log r + (s+1) \log q - \Omega(m)} \\
&\le \delta r \cdot 2^{-\Omega(\log r)} \\
&= r^{-O(1)},
\end{aligned}$$

if we take $m_1 = \Omega((r/k)\log r)$ and $\delta < 1$.

For B_2, we need to apply a union bound:

$$B_2 \le |\mathcal{S}_r \cap \mathcal{P}| \cdot \left(\max_{x \in \mathcal{P}} \Pr[\langle w, x\rangle = 0]\right)^m.$$

In fact, it suffices to consider the vectors in $\mathbb{Z}_q^r$, where q is a prime in $[k-1, 2k]$. We need the following group-theoretic Littlewood-Offord inequality:

Lemma 17.23 (Corollary 1 in [JŠ19])**.** *Let $q \ge 2$ be a prime. Let $x \in \mathbb{Z}_q^n$ with $|\operatorname{supp}(x)| = n$. Let $w_i \in \{-1, 1\}_{\mathbb{Z}_q}$ be a Bernoulli random variable for $i \in [n]$. Then,*

$$\sup_{g \in \mathbb{Z}_q} \Pr[\langle x, w\rangle = g] \le 3 \max\left\{\frac{1}{q}, \frac{1}{\sqrt{n}}\right\}.$$

Then, similar to the proof of Lemma 4.2 in [FKS20], Lemma 17.23 implies the following claim:

Claim 17.24. *Let $0 < k \le r/2$, and consider a vector $x \in \mathbb{Z}_q^r$ (q prime) whose largest fibre has size $r - s$, and let $w \in \{0,1\}^r$ be a random vector with exactly k ones. Then,*

$$\sup_{g \in \mathbb{Z}_q} \Pr[\langle x, w\rangle = g] = O\left(\max\{1/q, \sqrt{r/(ks)}\}\right).$$

In our case, $s = \delta r$ and $\sqrt{r/(ks)} = (\delta k)^{-1/2}$. That is,

$$\max_{x \in \mathcal{P}} \Pr[\langle w, x\rangle = 0] \le O\left(\frac{1}{\sqrt{\delta k}}\right)$$

Then, we apply union bound for B_2:

$$\begin{aligned} B_2 &\le (q-1)^r \cdot \left(\frac{1}{\sqrt{\delta k}}\right)^m \\ &\le (2k)^r \cdot (\delta k)^{-m/2} \\ &= \exp(O(r\log(k) - m\log(\delta k))) \\ &= \exp(-\Omega(r \log k)) \\ &\le \frac{1}{\mathrm{poly}(r)}, \end{aligned}$$

if we take δ to be a constant in $(0,1)$ and $m = \Omega(r)$.

Combining them together, we get that

$$\Pr[\mathbf{W} \text{ is column-singular}] \le \frac{1}{\mathrm{poly}(r)}$$

if $m = \Omega(\max\{r, (r/k)\log r\})$. □

17.4.5 Putting everything together

In this section we formally show how to conclude our main average-case guarantee:

Theorem 17.25 (Average-case guarantee, formal version of Theorem 17.3)**.** *Fix any integer $2 \le k \le r$, failure probability $\delta \in (0,1)$, and suppose $m \ge \widetilde{\Omega}(rk\log(1/\delta))$. Let $\mathbf{W} \in \{0,1\}^{m\times r}$ be generated by the following random process: for every $i \in [m]$, the i-th row of $\mathbf{W}$ is a uniformly random k-sparse binary vector. Define $\mathbf{M} \triangleq \mathbf{W}\mathbf{W}^\top$ where matrix multiplication is over the Boolean semiring. There is an algorithm which runs in $O(m^{\omega+1})$ time and, with probability $1-\delta$ over the randomness of $\mathbf{W}$, outputs a matrix $\widehat{\mathbf{W}} \in \{0,1\}^{m\times r}$ whose columns are a permutation of those of $\mathbf{W}$.*[3]

[3] $\omega \approx 2.373$ is the exponent of matrix multiplication.

Proof. By Lemma 17.15, as long as m satisfies the bound in the hypothesis, with probability at least $1-\delta$ one can successfully form the tensor $\mathbf{T} = \sum_{i=1}^{r} \mathbf{W}_i^{\otimes 3}$ in time $O(m^{\omega+1})$. By Theorem 17.17, the columns of $\mathbf{W}$ are linearly independent with high probability. By Lemma 17.16, one can therefore recover the columns of $\mathbf{W}$ up to permutation. $\square$

17.5 Connections Between BkV-SUM, SSBMF, InstaHide

In this section, we begin by formally defining BkV-SUM and describing the connection to SSBMF elucidated in [CLSZ21, CDG+20]. In Section 17.5.3 we then prove Theorem 17.4 by describing an algorithm that takes a solution to the SSBMF instance corresponding to a BkV-SUM instance and extracts more fine-grained information about the latter, specifically certain coordinates of the unknown database generating the BkV-SUM instance. As we will explain, the motivation for this particular recovery guarantee comes squarely from designing better attacks on InstaHide.

17.5.1 Connection to batched k-vector sum

We first describe an extension of the well-studied k-sum problem [Kar72, Eri95, Pat10, AL13, Abb19] to a "batched" setting. Recall that the classic k-sum problem asks: given a collection of r numbers, determine whether there exists a subset of size k summing to zero. One can consider an analogous question where we instead have a collection of r vectors in $\mathbb{R}^d$, and more generally, instead of asking whether some subset of size k sums to the zero vector, we could ask the following search problem which has been studied previously [BIWX11, CP14, ALW14]:

Definition 17.3 (k-Vector-Sum)**.** Given a database $\mathbf{X}$ consisting of d-dimensional vectors $x_1, \cdots, x_r$, for a fixed vector y and an integer k, we promise that there is a set $S \subset [n]$ such that $|S| = k$ and $y = \sum_{i \in S} x_i$. We can observe y and have access to database $\mathbf{X}$, our goal is to recover set S.

Even when $d = 1$, the Exponential Time Hypothesis implies that no algorithm can do better than $r^{o(k)}$, and this essentially remains true even for vectors over finite fields of small characteristic [BIWX11].

We consider two twists on this question. First, instead of a single vector y, imagine we got a *batch* of multiple vectors $y_1, \ldots, y_m$, each of which is the sum of some k vectors in the database, and the goal is to figure out the constituent vectors for each y_i. Second, given that there might be some redundant information among the y_i's, it is conceivable that under certain assumptions on the database $\mathbf{X}$, we could even hope to solve this problem without knowing $\mathbf{X}$. These considerations motivate the following problem:

Definition 17.4 (BkV-SUM)**.** Given *unknown* database $\mathbf{X}$ which is a list of vectors $x_1, \cdots, x_r \in \mathbb{R}^d$ and a fixed integer k. For a set of vector $y_1, \cdots, y_m \in \mathbb{R}^d$, for each $j \in [m]$, we promise that there is a set $S_j \subset [n]$ such that $|S_j| = k$ and $y_j = \sum_{i \in S_j} x_i$. Given $y_1, \cdots, y_m$, our goal is to recover sets $S_1, \cdots, S_m$.

As we discuss in Section 17.5.3, BkV-SUM is closely related to existing attacks on a recently proposed scheme for privately training neural networks called InstaHide.

17.5.2 Similarity oracle

Taking a step back, note that BkV-SUM could even be harder than kV-SUM from a worst-case perspective, e.g. if $y_1 = \cdots = y_m$. The workaround we consider is motivated by the aforementioned applications of BkV-SUM to InstaHide [CLSZ21, CDG+20]. It turns out that in these applications, the unknown database $\mathbf{X}$ possesses additional properties that allow one to construct the following oracle, e.g. by training an appropriate neural network classifier [CDG+20]:

Definition 17.5 (Similarity oracle)**.** Recall the notation of Definition 17.4. Given vectors $\{y_1, \ldots, y_m\}$ generated by BkV-SUM, let $\mathcal{O}$ denote the oracle for which $\mathcal{O}(i, j) = |S_i \cap S_j| \neq \emptyset]$ for all $i, j \in [m]$. Also define the *selection matrix* $\mathbf{W} \in \{0, 1\}^{m \times r}$ to be the matrix whose i-th row is the indicator vector for subset S_i.

The following simple observation tells us that given an instance of BkV-SUM together with a similarity oracle, we can immediately reduce to an instance of SSBMF:

Fact 17.26. *If the matrix* $\mathbf{M} \in \mathbb{R}^{m\times m}$ *has entries* $\mathbf{M}_{i,j} = \mathcal{O}(i,j)$*, then it satisfies* $\mathbf{M} = \mathbf{W}\mathbf{W}^\top$.

17.5.3 An improved attack on InstaHide

Given a similarity oracle, we can in fact do more than just recover $S_1,\ldots,S_m$: provided that d is sufficiently large, we can use the matrix $\mathbf{W}$ we have recovered as well as the vectors $y_1,\ldots,y_m$ to solve a collection of linear systems to recover $x_1,\ldots,x_m$. In this section, we show a stronger guarantee: recovery is possible even if we only have access to the *entrywise absolute values* of the y_i's. To motivate this result, we first spell out how exactly InstaHide [HSLA20b] relates to the BkV-SUM problem considered in this section.

Definition 17.6. From a *private dataset* $\mathbf{X} = \{x_1,\ldots,x_r\}$, InstaHide generates a *synthetic dataset* by sampling $\mathbf{W} \in \{0,1\}^{m\times r}$ according to Assumption 17.2 and outputting the vectors $z_1,\ldots,z_m$ given by $z_i = |\mathbf{W}_i\mathbf{X}|$, where $|\cdot|$ denotes entrywise absolute value, $\mathbf{W}_i$ is the i-th row of $\mathbf{W}$, and, abusing notation, $\mathbf{X}$ denotes the $r\times d$ matrix whose j-th row is x_j. In light of Definition 17.4, let $S_i \subset [r]$ denote the support of the i-th row of $\mathbf{W}$.

Note that the synthetic dataset $\{z_1,\ldots,z_m\}$ in Definition 17.6 is simply given by the entrywise absolute values of the vectors $\{y_1,\ldots,y_m\}$ in Definition 17.4 if the size-k subsets $S_1,\ldots,S_m$ there were chosen uniformly at random.

It was shown in [HSLA20b] that by training a neural network on the synthetic dataset generated from a private image dataset, one can still achieve good classification accuracy, and the hope was that by taking entrywise absolute values, one could conceal information about the private dataset. This has since been refuted empirically by [CDG+20], and provably by [CLSZ21, HST+20] for extremely small values of k, but a truly polynomial-time, provable algorithm for recovering private images from synthetic ones generated by InstaHide had remained open, even given a similarity oracle.

Here we close this gap by showing how to efficiently recover most of the private dataset by building on our algorithm for SSBMF. The pseudocode is given in Algorithm 115.

Algorithm 115 Recovering heavy coordinates of $\mathbf{M}$

```
1: procedure GetHeavyCoordinates($\mathbf{M}$)    ▷ Similarity matrix $\mathbf{M}$ for synthetic images generated from unknown private dataset $\mathbf{X}$
2:     $\mathbf{W} \leftarrow$TensorRecover($\mathbf{M}$)
3:     for $j \in [d]$ do
4:         Let $z \in \mathbb{R}^m$ have $i$-th entry equal to the $j$-th coordinate of synthetic image $i$
5:         Form the vector $\widetilde{p}' \triangleq \frac{1}{m}\sum_{i=1}^m \left(w_i - \frac{k-1}{r-2}\vec{1}\right) \cdot z_i^2$, where $w_i$ is the $i$-th row of $\mathbf{W}$
6:         Set the $j$-th row of $\widehat{\mathbf{X}}$ to be $\widetilde{p}' \cdot \frac{r(r-1)}{k(r-2k+1)}$
7:     end for
8:     return $\widehat{\mathbf{X}}$    ▷ Matrix $\widehat{\mathbf{X}}$ which approximates heavy entries of $\mathbf{X}$ (see Theorem 17.29)
9: end procedure
```

The following lemma is the main ingredient for analyzing Algorithm 115, whose main guarantees were informally stated in Theorem 17.4. It essentially says that given $\mathbf{W}$ and the ℓ-th coordinates of all $z_1, \ldots, z_m$, one can approximately reconstruct the ℓ-th coordinates of all $x_1, \ldots, x_r$ which are sufficiently "heavy." Roughly, an index $i \in [r]$ is "heavy" if the magnitude of the ℓ-th coordinate of x_i is roughly k times larger than the average value in that coordinate across all $x_1, \ldots, x_r$.

Lemma 17.27. *For any absolute constant $\eta > 0$, there is an absolute constant $c > 0$ for which the following holds as long as $m \geq \Omega(\log(d/\delta))$. There is an algorithm* GetHeavyCoordinates *that takes as input $\mathbf{W} \in \{0,1\}^{m \times r}$ satisfying Assumption 17.2 and vector $z \in \mathbb{R}^m$ satisfying $|\mathbf{W}p| = z$ for some vector $p \in \mathbb{R}^r$, runs in* *time $O(r \cdot m)$, and outputs $\widehat{p}$ such that for every $i \in [r]$ for which $|p_i| \geq (ck/r) \cdot \overline{p}$, we have that $\widetilde{p}_i = p_i \cdot (1 \pm \eta)$.*

We will need the following basic calculation. Henceforth, given a vector p, let $\overline{p}$ denote the sum of its entries.

Fact 17.28. *For any vector $p \in \mathbb{R}^r$,*

$$\mathbb{E}_S[\langle e_S, p\rangle^2] = \frac{k(r-k)}{r(r-1)}\|p\|_2^2 + \frac{k(k-1)}{r(r-1)}\overline{p}^2 \tag{17.7}$$

where the expectation is over a random size-k subset $S \subset [r]$.

Proof. Let ξ_i denote the indicator for the event that $i \in S$ so that

$$\begin{aligned}
\mathbb{E}_S[\langle e_S, p\rangle^2] &= \mathbb{E}_S\left[\sum_{i,j\in[r]} \xi_i\xi_j p_i p_j\right] \\
&= \sum_{i\in[r]} p_i^2 \cdot \mathbb{E}[\xi_i] + \sum_{i\neq j} p_i p_j \, \mathbb{E}[\xi_i\xi_j] \\
&= \left(\frac{k}{r} - \frac{k(k-1)}{r(r-1)}\right)\|p\|_2^2 + \frac{k(k-1)}{r(r-1)}\overline{p}^2 \\
&= \frac{k(r-k)}{r(r-1)}\|p\|_2^2 + \frac{k(k-1)}{r(r-1)}\overline{p}^2.
\end{aligned}$$

as claimed. □

We are now ready to prove Lemma 17.27.

Proof of Lemma 17.27. Define the vector

$$\widetilde{p} \triangleq \mathbb{E}_S\left[\langle e_S, p\rangle^2 \cdot e_S\right], \tag{17.8}$$

where the expectation is over a random subset $S \subset [r]$ of size k, and $e_S \in \{0,1\}^k$ is the indicator vector for the subset S. The i-th entry of $\widetilde{p}$ is given by

$$\begin{aligned}\widetilde{p}_i &= \binom{r}{k}^{-1} \cdot \sum_{S \in \binom{r-1}{[k-1]}} (\langle e_S, p\rangle + p_i)^2 \\ &= \binom{r}{k}^{-1} \cdot \sum_{S \in \binom{r-1}{[k-1]}} p_i^2 + 2p_i \cdot p_S + p_S^2 \qquad (p_S \triangleq \textstyle\sum_{j\in S} p_j.) \\ &= \binom{r}{k}^{-1} \cdot \left(p_i^2 \cdot \binom{r-1}{k-1} + 2p_i \cdot \sum_{S \in \binom{r-1}{[k-1]}} p_S + \sum_{S \in \binom{r-1}{[k-1]}} p_S^2 \right)\end{aligned}$$

For the second term, we have

$$\sum_{S \in \binom{r-1}{[k-1]}} p_S = \sum_{j \in [r]-\{i\}} p_j \cdot \binom{r-2}{k-2}.$$

For the third term, we have

$$\begin{aligned}\sum_{S \in \binom{r-1}{[k-1]}} p_S^2 &= \sum_{S \in \binom{r-1}{[k-1]}} \sum_{j,\ell \in S} p_j p_\ell \\ &= \sum_{S \in \binom{r-1}{[k-1]}} \sum_{j \in S} p_j^2 + \sum_{S \in \binom{r-1}{[k-1]}} \sum_{j \neq \ell} p_j p_\ell \\ &= \sum_{j \in [r]-\{i\}} p_j^2 \cdot \binom{r-2}{k-2} + \sum_{j,\ell \in [r]-\{i\}, j\neq \ell} p_j p_\ell \cdot \binom{r-3}{k-3}.\end{aligned}$$

Hence, the i-th entry of $\mathbb{E}_S[\langle e_S, p\rangle^2 \cdot e_S]$ is

$$\begin{aligned}\Big(\mathbb{E}_S\left[e_S \cdot \langle e_S, p\rangle^2\right]\Big)_i &= p_i^2 \cdot \frac{k(r-2k+1)}{r(r-1)} + \|p\|_2^2 \cdot \frac{k(k-1)(r-k)}{r(r-1)(r-2)} \\ &\quad + 2p_i\overline{p} \cdot \frac{k(k-1)}{r(r-1)} + \overline{p}^2 \cdot \frac{k(k-1)(k-2)}{r(r-1)(r-2)}.\end{aligned}$$

We conclude by Fact 17.28 that the i-th entry of $\mathbb{E}_S\left[\langle e_S, p\rangle^2 \cdot \left(e_S - \frac{k-1}{r-2} \cdot \vec{1}\right)\right]$ is bounded by

$$\begin{aligned}\left(\mathbb{E}_S\left[\left(e_S - \frac{k-1}{r-2}\vec{1}\right) \cdot \langle e_S, p\rangle^2\right]\right)_i &= p_i^2 \cdot \frac{k(r-2k+1)}{r(r-1)} + 2p_i\overline{p} \cdot \frac{k(k-1)}{r(r-1)} \\ &\quad + \overline{p}^2 \cdot \frac{k(k-1)(2k-3)}{r(r-1)(r-2)}\end{aligned}$$

We do not have exact access to $\widetilde{p}$, but we may form the unbiased estimator

$$\widetilde{p}' \triangleq \frac{1}{m} \sum_{i=1}^{m} \left(w_i - \frac{k-1}{r-2} \vec{1} \right) \cdot z_i^2, \tag{17.9}$$

where w_i is the i-th row of $\mathbf{W}$. For any $i \in [m]$, each coordinate of $w_i \cdot z_i^2$ is bounded within the interval $[-\|z\|_\infty^2, \|z\|_\infty^2]$, so by Chernoff, provided that $m \geq \Omega(\log(d/\delta)/\epsilon^2)$, we ensure that $\widetilde{p}'_i \in \widetilde{p}_i(1 \pm \epsilon)$ for all i with probability at least $1 - \delta$. Now consider the following estimator for p_i^2:

$$\widehat{q}_i \triangleq \widetilde{p}'_i \cdot \frac{r(r-1)}{k(r-2k+1)}. \tag{17.10}$$

We can thus upper bound the error of $\widehat{q}_i$ relative to p_i^2 by

$$\frac{\overline{p}}{p_i} \cdot O\left(\frac{k}{r}\right) + \left(\frac{\overline{p}}{p_i}\right)^2 \cdot O\left(\frac{k^2}{r^2}\right) \pm \epsilon \cdot O\left(1 + \frac{\overline{p}}{p_i}\frac{k}{r} + \frac{\overline{p}^2}{p_i^2}\frac{k^2}{r^2}\right).$$

If we assume that $|p_i| \geq \Omega(k/r)\overline{p}$ and $\epsilon = O(1)$ for appropriately chosen constant factors, then we have that $\widehat{q}_i \in p_i^2 \cdot (1 \pm \eta)$ as desired. $\square$

We are now ready to prove the main guarantee for our attack on InstaHide, originally stated informally in Theorem 17.29.

Theorem 17.29 (Formal version of Theorem 17.4)**.** *For any absolute constant $\eta > 0$, there is an absolute constant $c > 0$ for which the following holds. Fix any integer $k \geq 2$, failure probability $\delta \in (0,1)$, and suppose $m \geq \widetilde{\Omega}(rk \log(d/\delta))$. Given a synthetic dataset of size m generated by* BkV-SUM *from a matrix $\mathbf{X}$, together with its similarity oracle, there is an $O(m^{\omega+1} + d \cdot r \cdot m)$-time algorithm which outputs a matrix $\widehat{\mathbf{X}}$ such that for any $(i,j) \in [r] \times [d]$ satisfying $|\mathbf{X}_{i,j}| \geq (ck/r)\sum_{i' \in [r]} |\mathbf{X}_{i',j}|$, we have that $|\widehat{\mathbf{X}}_{i,j}| = |\mathbf{X}_{i,j}| \cdot (1 \pm \eta)$.*

Proof. By Theorem 17.25 and the assumed lower bound on m, we can exactly recover the selection matrix $\mathbf{W}$ (up to some column permutation) in time $O(m^{\omega+1})$. Using Lemma 17.27, for every pixel index $j \in [d]$ we can run GetHeavyCoordinates($\mathbf{M}$) to recover the pixels in position j which are heaviest among the r private images in time $O(m \cdot r)$, yielding the desired guarantee. $\square$

17.6 Worst-Case Algorithm

In this section, we give a worst-case quasi-polynomial algorithm for sparse boolean matrix factorization problem. It turns out that our techniques here can handle both SSBMF as well as an asymmetric variant, In Section 17.6.1 we define this variant and give some background on constraint satisfaction problems (CSPs). Section 17.6.2 gives the algorithm for the asymmetric and symmetric version of the problem by exhibiting a reduction to 2-CSP. Section 17.6.3 extends the algorithm to the Boolean semiring.

17.6.1 CSP preliminaries

We first define a general (asymmetric) version of SSBMF as follows:

Definition 17.7 (Sparse Boolean matrix factorization (Sparse BMF))**.** Given an $m \times m$ matrix $\mathbf{M}$ where each entry is in $\{0, 1, \cdots, k\}$. Suppose matrix $\mathbf{M}$ can be factorized into two matrices $\mathbf{U} \in \{0, 1\}^{m \times r}$ and $\mathbf{V} \in \{0, 1\}^{r \times m}$, where each row of $\mathbf{U}$ is k-sparse and each column of $\mathbf{V}$ is k-sparse.

The task is to find a row k-sparse matrix $\widehat{\mathbf{U}} \in \{0, 1\}^{m \times r}$ and a column k-sparse matrix $\widehat{\mathbf{V}} \in \{0, 1\}^{r \times m}$ such that $\mathbf{M} = \widehat{\mathbf{U}}\widehat{\mathbf{V}}$.

We can also define the sparse Boolean matrix factorization as an optimization problem.

Definition 17.8 (Sparse BMF, optimization version)**.** Given an $m \times m$ matrix $\mathbf{M}$ where each entry is in $\{0, 1, \cdots, k\}$. The goal is to find a row k-sparse matrix $\widehat{\mathbf{U}} \in \{0, 1\}^{m \times r}$ and a column k-sparse matrix $\widehat{\mathbf{V}} \in \{0, 1\}^{r \times m}$ such that the number of different entries $\|\mathbf{M} - \widehat{\mathbf{U}}\widehat{\mathbf{V}}\|_0$ is minimized.

We now recall the definition of 2-CSPs:

Definition 17.9 (Max 2-CSP)**.** A 2-CSP problem is defined by the tuple $(\Sigma, V, E, \mathcal{C})$. Σ is an alphabet set of size q, V is a variable set of size n, $E \subseteq V \times V$ is the constraint set. V and E define an underlying graph of the 2-CSP instance, and $\mathcal{C} = \{C_e\}_{e\in E}$ describes the constraints. For each $e \in E$, C_e is a function $\Sigma \times \Sigma \to \{0,1\}$. The goal is to find an assignment $\sigma : V \to \Sigma$ with maximal *value*, defined to be the number of satisfied edges $e = (u,v) \in E$ (i.e., for which $C_e(\sigma(u), \sigma(v)) = 1$).

We will use the following known algorithm for solving "dense" 2-CSP instances:

Theorem 17.30 ([DM18])**.** *Define the density δ of a 2-CSP instance to be $\delta \triangleq |E|/\binom{|V|}{2}$. For any $0 < \epsilon \le 1$, there is an approximation algorithm that, given any δ-dense 2-CSP instance with optimal value* OPT*, runs in time $(nq)^{O(\epsilon^{-1}\cdot\delta^{-1}\cdot\log q)}$ and outputs an assignment with value $\mathrm{OPT} - \epsilon|E|$, where $n = |V|$ and $q = |\Sigma|$.*

17.6.2 From factorization to CSPs

We give a reduction that can reduce the general sparse BMF problem to a Max 2-CSP problem, and then use a quasi-polynomial time 2-CSP solver to find an approximation solution.

Theorem 17.31 (QPTAS for asymmetric sparse BMF, formal version of Theorem 17.5)**.** *Given $m, k, r \ge 0$ and an $m \times m$ matrix $\mathbf{M}$ as the input of an instance of sparse Boolean matrix factorization problem. Let* OPT *be the optimal value of the problem, i.e.,* $\mathrm{OPT} := \min_{\mathbf{U},\mathbf{V}} \|\mathbf{M} - \mathbf{U}\mathbf{V}\|_0$*, where $\mathbf{U}, \mathbf{V}$ satisfy the sparsity constraints of the problem.*

For any $1 \ge \epsilon > 0$, there exists an algorithm that runs in

$$m^{O(\epsilon^{-1} k \log r)} r^{O(\epsilon^{-1} k^2 \log r)}$$

time and finds a row k-sparse matrix $\widehat{\mathbf{U}}$ and a column k-sparse matrix $\widehat{\mathbf{V}}$ satisfying

$$\|\mathbf{M} - \widehat{\mathbf{U}}\widehat{\mathbf{V}}\|_0 \le \mathrm{OPT} + \epsilon m^2.$$

Proof. For the input matrix $\mathbf{M}$, let $\mathbf{U}$ and $\mathbf{V}$ be the ground-truth of the factorization. Let $(b_1^\top, \ldots, b_m^\top)$ be the rows of $\mathbf{U}$ and $(c_1, \ldots, c_m)$ be the columns of $\mathbf{V}$. We construct a 2-CSP instance $\mathcal{F}_A$ that finds $\mathbf{U}$ and $\mathbf{V}$ as follows:

- Let $\Sigma = \left\{(q_1, \ldots, q_r) \mid q_i \in \{0,1\} \; \forall i \in [r] \text{ and } \sum_{i\in[r]} q_i = k\right\}$ be the alphabet.
- The underlying graph is a bipartite graph. The left-side vertices $V_L = [m]$ corresponding to the rows of $\mathbf{U}$. The right-side vertices $V_R = [m]$ corresponding to the columns of $\mathbf{V}$.
- For $e = (u, v) \in V_L \times V_R$, define the constraint C_e to be: for all $(p_1, \ldots, p_r), (q_1, \ldots, q_r) \in \Sigma \times \Sigma$,

$$C_e((p_1, \ldots, p_r), (q_1, \ldots, q_r)) = 1 \iff \sum_{i=1}^{r} p_i q_i = A_{u,v}.$$

Note that $\mathcal{F}_A$ has value OPT. We can create an assignment from the ground-truth such that $\sigma(u) = b_u$ for $u \in V_L$ and $\sigma(v) = c_v$ for $v \in V_R$. By definition of the sparse Boolean matrix factorization problem, this is a legal assignment. Also, since the number of $A_{u,v} = \langle b_u, c_v \rangle$ for all $(u, v) \in [m] \times [m]$ is $m^2 - \mathrm{OPT}$, we can see that all such edges are satisfied by this assignment.

Then, we can run the QPTAS algorithm (Theorem 17.30) on $\mathcal{F}_A$ and obtain an assignment that at most $\mathrm{OPT} - \epsilon|E|$ constraints are unsatisfied, which means the number of different entries between $\mathbf{M}$ and $\widehat{\mathbf{U}}\widehat{\mathbf{V}}$ is at most $\mathrm{OPT} - \epsilon m^2$.

The alphabet size of $\mathcal{F}_A$ is $\binom{r}{k}$. The reduction time is $O(m^2 r^k)$ and the 2-CSP solving time is $(mr^k)^{O(\epsilon^{-1}\log(r^k))}$ by Theorem 17.30 since the density of a complete bipartite graph is $\delta = \frac{1}{2}$. The theorem is then proved. □

Remark 17.3. We briefly compare this to the guarantee of [KPRW19], who obtained a

$$2^{O(r^2 \log r)} \operatorname{poly}(m)$$

constant-factor approximation algorithm. By introducing a sparsity constraint on the rows of $\mathbf{U}, \mathbf{V}$ through our new parameter k, we circumvent the exponential dependence on r, at the cost of running in time quasipolynomial in m. In particular, our guarantee dominates when the rank parameter r is at least roughly $\widetilde{\Omega}(\sqrt{\log m})$, though strictly speaking our guarantee is incomparable because we aim for an additive approximation and only measure error in L_0 rather than Frobenius norm.

A similar reduction can be used to prove a worst-case guarantee for SSBMF, stated informally in Theorem 17.5.

Theorem 17.32 (Formal version of Theorem 17.5)**.** *Given $m, k, r \geq 0$ and a symmetric $m \times m$ matrix $\mathbf{M}$ as the input of a (worst-case) instance of* SSBMF*. Let* OPT *be the optimal value of the problem, i.e.,* $\mathrm{OPT} := \min_{\mathbf{W}} \|\mathbf{M} - \mathbf{W}\mathbf{W}^\top\|_0$*, where $\mathbf{W}$ is a row k-sparse matrix in $\{0,1\}^{m \times r}$. For any accuracy $\epsilon \in (0,1)$, there is an algorithm running in time*

$$m^{O(\epsilon^{-1} k \log r)} r^{O(\epsilon^{-1} k^2 \log r)}$$

which finds a row k-sparse matrix $\widehat{\mathbf{W}}$ satisfying

$$\|\mathbf{M} - \widehat{\mathbf{W}}\widehat{\mathbf{W}}^\top\|_0 \leq \mathrm{OPT} + \epsilon m^2.$$

Proof. The construction of the 2-CSP instance $\mathcal{F}_A$ is almost the same as in the proof of Theorem 17.31, except that in this case, the underlying graph is a complete graph, where the vertices $V = [m]$ correspond to the rows of $\mathbf{W}$. Then, each constraint $C_{(u,v)}$ checks whether $\langle b_u, b_v \rangle = \mathbf{M}_{u,v}$. The correctness of the reduction follows exactly the proof of Theorem 17.31 and we omit it here. The density of $\mathcal{F}_A$ in this case is 1, and hence the running time of the algorithm is $(mr^k)^{O(\epsilon^{-1} \log(r^k))}$. $\square$

17.6.3 Extension to the Boolean semiring

A direct corollary of Theorem 17.32 is that the sparse BMF over the Boolean semiring can also be solved in quasi-polynomial time.

Corollary 17.33. *Given $m, k, r \geq 0$ and a symmetric Boolean $m \times m$ matrix $\mathbf{M}$. Let* OPT *be the optimal value of the problem, i.e.,* $\mathrm{OPT} := \min_{\mathbf{W}} \|\mathbf{M} - \mathbf{W}\mathbf{W}^\top\|_0$, *where* $\mathbf{W}$ *is a row k-sparse matrix in $\{0,1\}^{m \times r}$ and the matrix multiplication is over the Boolean semiring, i.e., $a + b$ is $a \vee b$ and $a \cdot b$ is $a \wedge b$. For any accuracy parameter $\epsilon \in (0,1)$, there exists an algorithm that runs in*

$$m^{O(\epsilon^{-1} k \log r)} r^{O(\epsilon^{-1} k^2 \log r)}$$

time and finds a row k-sparse matrix $\widehat{\mathbf{W}}$ satisfying

$$\|\mathbf{M} - \widehat{\mathbf{W}}\widehat{\mathbf{W}}^\top\|_0 \leq \mathrm{OPT} + \epsilon m^2.$$

Proof. The construction can be easily adapted to the case when matrix multiplication is over the Boolean semiring, where $a + b$ becomes $a \vee b$ and $a \cdot b$ becomes $a \wedge b$. We can just modify the constraints of the 2-CSP instance in the reduction and it is easy to see that the algorithm still works. $\square$

Remark 17.4. Factorizing Boolean matrices with Boolean arithmetic is equivalent to the bipartite clique cover problem. It was proved by [CIK17] that the time complexity lower bound for the exact version of this problem is $2^{2^{\Omega(r)}}$. Since the approximation error is ϵm^2, when $\epsilon < \frac{1}{m^2}$, the output of our algorithm is the exact solution. Further, if we do not have the row sparsity condition, i.e., $k = r$, then the time complexity becomes

$$2^{O(m^2 (\log m) \cdot r^2 \log r)}.$$

In the realm of parameterized complexity (see for example [CFK+15]), due to the kernelization in [FMPS09], we may assume $m \leq 2^r$ and the running time of our algorithm is $2^{\widetilde{O}(2^{2r} \cdot r^3)}$, which matches the lower bound of this problem.

Chapter 18: Copyright Protection in the Quantum Era

18.1 Introduction

In the past decade, deep learning has achieved remarkable success across various domains. However, the rapidly increasing size of models has presented a challenge as training them without a GPU cluster has become nearly impossible. Moreover, newer large language models like GPT-4 require substantial computational resources just for doing inference. Consequently, there are two main approaches to accessing these deep learning models.

The first approach (which is widely used) involves AI companies deploying their models on the cloud, allowing clients to use them remotely. While this method provides convenience, it raises concerns regarding data exchange between servers and clients, potentially compromising privacy.

The second approach is for AI companies to directly send the models to clients with sufficient computational resources, enabling all computations to be performed locally. However, it raises a silent question:

how can AI companies prevent clients from copying the models and distributing them to unauthorized users?

This question motivates the study of copy-protection in cryptography, which seeks for protocols to distribute deep learning models (or general programs) to clients and safeguarding against unauthorized copying and distribution. Nonetheless, most classical approaches are heuristic in nature, as it is inherently impossible to prevent a malicious user from copying the code of a program.

Quantum copy-protection, proposed by Aaronson [Aar09], aims to use the unclonability of quantum states to achieve programs that cannot be copied. That is, the program f is given as a quantum state $|\psi_f\rangle$. $|\psi_f\rangle$ allows for computing f on arbitrary inputs; meanwhile, it is infeasible to copy the state $|\psi_f\rangle$, or even convert $|\psi_f\rangle$ into two arbitrary states that both allow for computing f. The quantum no-cloning theorem shows that quantum states, in general, cannot be copied. Copy protection takes this much further, augmenting the unclonable state with the ability to evaluate programs.

Progress on quantum copy-protection has unfortunately been slow. On the negative side, copy-protection for general programs is impossible. As explained by Aaronson [Aar09], any *learnable* program—that is, a program whose description can be learned from just its input/output behavior—cannot be copy-protected. Indeed, given the (copy-protected) code for the program, an attacker can just query the code on several inputs and learn the original program from the results. The attacker can then copy the original program indefinitely. A more recent result of Ananth and La Placa [AP20] shows, under certain computational assumptions, that certain contrived unlearnable programs cannot be copy-protected.

On the positive side, Aaronson demonstrates a quantum oracle[1] relative to which copy-protection exists for any unlearnable program. Due to the negative result above, this scheme cannot be instantiated in general. Worse, even for programs that are not subject to the impossibility result, it remains unclear how even heuristically to instantiate the scheme. Very recently, Ananth and La Placa [AP20] build a version of copy-protection, which they call software leasing, which guarantees a sort of copy *detection* mechanism. Unfortunately, their work explicitly allows copying the functionality and only ensures that such copying can be detected. Also, their construction only works for a certain class of "evasive" functions, which accept a hidden sparse set of inputs. The work of Ben-David and Sattath [BDS16] and more recently Amos et al. [AGKZ20] can be seen as copy-protecting specific cryptographic functionalities.

[1]That is, an oracle that implements a quantum operation.

18.1.1 This work

In this chapter, we give new general results for copy-protection. Our two main results are:

- Any unlearnable functionality can be copy-protected, relative to a *classical* oracle.
- Any functionality that can be *watermarked* in a certain sense, can be copy-*detected* assuming just the existence of public-key quantum money.

Both of our results are very general, applying to a wide variety of learning and watermarking settings, including settings where functionality preservation is not required. Along the way to obtaining our results, we give new definitions for security of copy-protection (as well as copy-detection and watermarking), which provide stronger guarantees.

Our first result improves Aaronson [Aar09] to use a classical oracle, which can then heuristically be instantiated using candidate post-quantum obfuscation (e.g. [BGMZ18, BDGM20]), resulting in a concrete candidate copy-protection scheme. Of course, the impossibility of Ananth and La Placa [AP20] means the resulting scheme cannot be secure in the standard model for arbitrary programs. Still, it can be conjectured to be secure for programs not subject to the impossibility.

Our second result complements Ananth and La Placa [AP20]'s positive result for copy-detecting certain evasive functions by copy-detecting arbitrary watermarkable functions. For our purposes, watermarkable functions are those that can have a publicly observable "mark" embedded into the program, such that it is infeasible to remove the mark without destroying the functionality. We note that the results (and techniques) are incomparable to [AP20]. First, watermarkable functions are never evasive, so the class of functions considered are disjoint. Second, our security guarantee is much stronger than theirs, which we discuss in Section 18.1.2.

Taken together, we believe our results strongly suggest that watermarkable functions may be copy-protectable. Concretely, the impossibility result of Ananth and La Placa also applies to copy detection, and our second result shows that watermarkable functions, therefore, circumvent the impossibility. Based on this, we conjecture that our first result, when instantiated with candidate obfuscators, is a secure copy-protection scheme for watermarkable functions. We leave proving or disproving our conjecture as an interesting direction for future work.

As a consequence, our scheme offers a solution for copy-protecting deep learning models (relative to a classical oracle). The security provided by our quantum copy-protection scheme guarantees that for a malicious user in the real-world (with limited computational power), the only way to copying the model is to *training it from scratch*! This presents a significant hurdle, particularly for large models, as the cost associated with training from scratch is prohibitively high.

18.1.2 Technical overview

Definitional Work. We first investigate the definition of quantum copy-protection. We find that existing definitions and other straightforward attempts have several limitations. We therefore carefully develop a strong and general definition of copy-protection to resolve these limitations. In particular, our definition captures attacks where (1) the program is meaningfully copied even if the functionality is technically different, and (2) the program is copied only with a small but detectable probability.

Consider the following attempt of defining quantum copy-protection: we say an adversary successfully pirates a quantum program for computing a function f if it outputs two quantum programs σ_1, σ_2, each of them able to compute f correctly on a large fraction of inputs. Now consider applying this definition to the case where f is a signing algorithm with a particular signing key hard-coded, and suppose there are many valid signatures for each message. Consider a hypothetical adversary who "splits" the program into two pieces, each computing valid signatures; but neither computing the same signature that f produces. Such programs are "good enough" for forging signatures, and the ability to copy a signature-producing program in this way would naturally be considered an attack. However, the usual notions of security for copy-protection do not apply to such programs.

Another example is the copy-protection of public-key encryption. Let f be a decrypting algorithm with a particular decryption key hard-coded. Suppose the split two program pieces only work correctly on a sparse set: they can only decrypt correctly on ciphertexts of m_0, m_1; for ciphertexts of other messages, the output is arbitrary. This splitting attack does not violate the security notion either since both functions produced by the adversary differ from the original program on most inputs. But again, such programs are "good enough" for breaking the semantic security of the encryption scheme, and therefore would reasonably count as an attack.

Similar definitional issues are discussed in the context of watermarking [GKM+19] but have not been explored in the setting of copy-protection. Inspired by the watermarking case, our solution is to define "compute f correctly" by a general relation. The relation takes some random coins r, the function f (with some additional information about f hard-coded in the circuit); it samples an input and runs the (quantum) program on that classical input; finally, it checks the output of the quantum program, testing (in superposition) if the output z together with f, r is in the relation. As an example, if f is a signing circuit (with the signing key hard-coded), the relation is defined as: use random coins r to generate a random message m, run the quantum program on m and test in superposition if it is a valid signature, by applying the verification algorithm $\mathsf{Ver}(\mathsf{vk}, m, \cdot\,; r)$.

Therefore, we propose a general definition that can capture a broader class of attacks, especially in the context of cryptographic functionalities.

Unfortunately, another uniquely quantum issue arises when trying to formulate a definition. We intuitively want to consider a program to be a valid copy if it computes f correctly on a non-trivial fraction of the domain. Unfortunately, there is no physical way to actually test if a program represented as a quantum state satisfies the property when an algorithm only receives a single copy of the program, especially in game-based security definitions.

Generally, any attempt at assigning a non-trivial property to quantum states (e.g., "valid program" vs. "invalid program") will be physically meaningless. Indeed, given any valid program P_1 and any invalid program P_2 (regardless of the meaning of "valid"), what is the uniform superposition $|P_1\rangle + |P_2\rangle$ of the two programs? Is it valid or invalid? Regardless of which, because the three programs are not orthogonal quantum states, no measurement can determine all three states' validity.

At a more operational level, the classical way to test for correctly computing f is to evaluate the function on all points and report the fraction of inputs where the program computed correctly. Alternatively, one can efficiently *estimate* the fraction of inputs that are computed correctly by simply testing a polynomial number of random points. Regardless, testing involves running the program on multiple points.

In the setting of quantum programs, however, the uncertainty principle implies that the moment one tests the first input, the quantum program state is irreversibly altered, potentially affecting the subsequent evaluations of the program. Thus, the fraction of inputs computed correctly is ever-changing, and simply evaluating the program on several points will not give a meaningful indication of the program's validity at any single point in time.

To illustrate further issues, consider the adversary which takes its quantum program P and simply produces $\frac{1}{\sqrt{2}}(|P\rangle|D\rangle|0\rangle + |D\rangle|P\rangle|1\rangle)$ where D is a dummy program that outputs junk. The two halves of this bipartite system each have probability 1/2 of outputting the right answer on a random input. And yet, this "attack" is rather useless and should not be considered a break.

On the other hand, consider a hypothetical attacker which produces $\frac{1}{\sqrt{2}}|P\rangle|P\rangle + \frac{1}{\sqrt{2}}|D\rangle|D\rangle$. The two halves of this bipartite system each separately has probability

1/2 of outputting the right answer on a random input. However, if we measure both halves, there is a 1/2 chance of obtaining two copies of P, which each answers correctly with probability 1. Therefore, this attack should likely be considered a break.

Thus, we see that any characterization of program validity which just tests the program on a random input cannot distinguish cases that should be considered breaks from those that are not. On the other hand, if we test multiple random inputs, we run into the problem that testing each input causes the program state to change, meaning we may not get meaningful results.

Our solution will be to use recent ideas from Zhandry [Zha20], who considered similar issues in the context of traitor tracing. At a high level, the issue above is that we are trying to assign a property to a quantum state (whether the state is a good program), but this property is non-physical and does not make sense for mixed or entangled states. Instead, we want "a program is good" to be a measurement that can be applied to the state. We would naturally also want the measurement to be projective, so that if a program is once tested to be "good", it will always be "good".

Let $\mathcal{M} = (M_0, M_1)$ be binary positive operator-valued measures (POVMs) that represents choosing random coins and testing if the quantum program computes correctly with respect to the random coins. For a mixed quantum program state σ, the probability the program evaluates correctly relative to this test is given as $\mathsf{Tr}[M_0\sigma]$. Let $\mathcal{M}'$ be the (inefficient) projective measurement $\{P_p\}_{p\in[0,1]}$, projecting onto the eigenspaces of M_0, where p ranges over the corresponding eigenvalues of M_0[2] Zhandry showed that the measurement below results in an equivalent POVM as $\mathcal{M}$:

- Apply the projective measurement $\mathcal{M}'$, and obtain p;
- Output 0 with probability p, and output 1 with probability $1 - p$.

[2]Since $M_0 + M_1$ is the identity, M_1 shares the same eigenvectors, with eigenvalue $1 - p$.

Intuitively, $\mathcal{M}'$ will project a state to an eigenvector with eigenvalue p. The leftover state computes correctly on p-fraction of all inputs.

Therefore, we say a quantum program σ is tested to be γ-good, if the measurement $\mathcal{M}'$ has outcome $p \geq \gamma$. We say an adversary successfully pirates a quantum program for computing f if the two programs are both tested to be γ-good with non-negligible probability. We will show an efficient algorithm that approximates the measurement. Thus, our new definition provides an operational game-based security definition that resolves all the issues we mentioned above. Besides, although the definition may be laborious, this definition implies the previous definitions in [Aar09, CMP20] and etc, and we find proving security with this definition is considerably easier. Using similar ideas, we also define quantum unlearnability of programs and quantum copy-detection.

Our Copy-Protection Scheme. We give a quantum copy-protection construction for all unlearnable functions based on (1) classical oracles, and (2) subspace states, or more abstractly, any tokenized signature scheme [BDS16].

Note the difference between classical and quantum oracles: a classical oracle is a classical functionality that can be (superposition) queried in a black-box way, while a quantum oracle is a quantum unitary operation used as a black-box. It is more feasible to implement classical oracles heuristically considering existing candidates of (post-quantum) obfuscations for classical circuits [BDGM20]. Therefore our construction is a significant improvement from the result in [Aar09].

A tokenized signature generates a signature token $|\mathsf{sig}\rangle$ which we call a signing token. A signer who gets one copy of the signing token can sign a single bit b of her choice. $\mathsf{Sign}(b, |\mathsf{sig}\rangle)$ outputs a classical signature whose correctness guarantee is the same as classical signatures: namely, verification will accept the result as a signature on b. Importantly, the signing procedure is a unitary and will produce a superposition of all valid signatures of b; to obtain a classical signature, a measurement to the state is necessary, leading to a collapse of the token state. Thus, a signature token $|\mathsf{sig}\rangle$ can only be used to produce one classical signature of a single bit, and any attempt to produce a classical signature of the other bit would fail. [BDS16] formalizes this idea and constructs a tokenized signature scheme relative to a classical oracle (a subspace membership oracle).

The high-level idea of our copy-protection scheme is that it requires an authorized user to query an oracle twice on signatures of bits 0 and 1. Let f be the function we want to copy-protect. Define the following classical circuits:

$$\mathcal{O}_1(x, \mathsf{sig}) = \begin{cases} H(x) & \text{if } \mathsf{Ver}(\mathsf{vk}, 0, \mathsf{sig}) = 1 \\ \perp & \text{otherwise} \end{cases},$$

$$\mathcal{O}_2(x, \mathsf{sig}) = \begin{cases} f(x) \oplus H(x) & \text{if } \mathsf{Ver}(\mathsf{vk}, 1, \mathsf{sig}) = 1 \\ \perp & \text{otherwise} \end{cases}.$$

Here H is a random function. The copy-protected program of f is a signature token $|\mathsf{sig}\rangle$ and obfuscations of $\mathcal{O}_1, \mathcal{O}_2$, which we will heuristically treat as oracles to $\mathcal{O}_1, \mathcal{O}_2$. We denote this program as $(|\mathsf{sig}\rangle, \mathcal{O}_1, \mathcal{O}_2)$.

To obtain $f(x)$, a user has to query on signatures of both bits and get $H(x)$ and $H(x) \oplus f(x)$. Note that even if one can only produce one of the classical signatures with token $|\mathsf{sig}\rangle$, a user can still query both oracles $\mathcal{O}_1, \mathcal{O}_2$ multiple times. To obtain $H(x)$, a user can compute the superposition of all valid signatures of 0 by applying a unitary and feed the quantum state together with x to $\mathcal{O}_1$. It then measures the output register. The user never actually measures the signature. Because the output register contains a unique output $H(x)$, by Gentle Measurement Lemma [Aar04], it can rewind the quantum state back to $|\mathsf{sig}\rangle$. Thus, our copy-protection scheme allows a copy-protected program to be evaluated on multiple inputs multiple times.

We next show how to prove anti-piracy security. Let σ_1, σ_2 be two (potentially entangled) program states pirated by an adversary, which makes oracle access to both $\mathcal{O}_1, \mathcal{O}_2$ and breaks the anti-piracy security. Let $\mathcal{O}_\perp$ be an oracle that always outputs $\perp$. If σ_1 never queries the oracle $\mathcal{O}_2$, we know the two programs $(\sigma_1, \mathcal{O}_1, \mathcal{O}_2)$

and $(\sigma_1, \mathcal{O}_1, \mathcal{O}_\perp)$ will have identical output distributions. Moreover, $(\sigma_1, \mathcal{O}_1, \mathcal{O}_\perp)$ can be simulated even without querying f because $\mathcal{O}_1$ is simply a random oracle (on valid inputs). Therefore, the program can be used to break the unlearnability of f. Similarly, if σ_2 never queries the oracle $\mathcal{O}_1$, the program $(\sigma_2, \mathcal{O}_\perp, \mathcal{O}_2)$ can be used to break the unlearnability of f.

Since f is unlearnable, the above two cases can not happen. We show that under this case, we can extract signatures of both 0 and 1. Intuitively, since $(\sigma_1, \mathcal{O}_1, \mathcal{O}_2)$ makes queries to $\mathcal{O}_2$, we can run the program on random inputs and measure a random query to $\mathcal{O}_2$, thereby extracting a signature of 1. Similarly, it holds for $(\sigma_2, \mathcal{O}_1, \mathcal{O}_2)$ and one could extract a signature of 0. Unfortunately, this intuition does not quite work since σ_1 and σ_2 are potentially entangled. It means there can be correlations between the outcomes of the measurements producing the two signatures: perhaps, if the measurement on $(\sigma_1, \mathcal{O}_1, \mathcal{O}_2)$ produces a valid signature on 1, then the measurement on $(\sigma_2, \mathcal{O}_1, \mathcal{O}_2)$ is guaranteed to fail to produce a signature. We show by a delicate argument that, in fact, adversaries cannot cheat using such correlations. Intuitively, although σ_1, σ_2 are entangled, we show there exists an efficient measurement: by applying this measurement to (σ_1, σ_2), with non-negligible probability, the resulting programs (σ_1', σ_2') have the following properties:

- They are both "good" programs. Thus, we can extract a signature of 1 in σ_1'.
- The resulting program σ_2'' after applying any measurement on σ_1' is still "good". Similarly, we can extract a signature of 0 in σ_2''.

Note that the above argument does not directly work for the original programs (σ_1, σ_2).

Our Copy-Detection Scheme. Inspired by [AP20], we propose a weaker definition called copy-detection, which has an additional checking procedure. A user can publicly verify a program's validity by running this checking procedure. The security guarantees that, given one copy of the program, no adversary can produce two programs such that both programs pass the checking procedure and both are 'functionally correct' (as in the copy-protection definition) — in other words, *honest* users can always identify the pirate. Looking ahead, we note that copy-detection is similar to secure software leasing (SSL, [AP20]), with the major differences are (1) the checking procedure is public, (2) 'functionally correct' in the security of copy-detection is average-case while that in the security of SSL is worst-case.

We construct a copy-detection scheme for any function family that can be watermarked. A watermarking scheme roughly consists of the following procedure: Mark takes a circuit and a message, and outputs a circuit embedded with that mark; $\mathsf{Extract}$ takes a marked circuit and produces the embedded mark. A watermarking scheme requires: (1) the watermarked circuit $\tilde{f} = \mathsf{Mark}(f, m)$ should preserve its intended functionality as f; (2) any efficient adversary given a marked $\tilde{f}$, can not generate a new marked circuit with a different mark (or remove the mark) while preserving its functionality. Watermarking primitives have been studied in previous works, including [CHN$^+$18, KW17, QWZ18, KW19, GKM$^+$19].

Our construction also requires a public key quantum money scheme. It consists two procedures: a generation procedure and a verification procedure. The generation procedure outputs a quantum banknote $|\$\rangle$. Verification is public, takes a quantum money banknote, and outputs either a (classical) serial number of that banknote or $\perp$ indicating it is an invalid banknote. The security requires no efficient adversary could use a quantum banknote $|\$\rangle$ to prepare two quantum banknotes $|\$_1\rangle\,|\$_2\rangle$ such that both banknotes pass the verification and their serial numbers are equal to that of $|\$\rangle$. We note that this version of quantum money corresponds to a "mini-scheme" as defined by [AC12].

The copy-detection scheme takes a function f, samples a banknote $|\$\rangle$ with serial number s, lets $\tilde{f} \leftarrow \mathsf{Mark}(f, s)$ and outputs a copy-detection program as $(\tilde{f}, |\$\rangle)$.

To evaluate the function, it simply runs the classical program $\tilde{f}$. To check a program is valid, it extracts the serial number from the money state and compares it with the mark of the program.

The security requires that no efficient adversary could produce $\tilde{f}_1, |\$_1\rangle$ and $\tilde{f}_2, |\$_2\rangle$ such that two programs pass the check and both classical circuits preserve the functionality. Let s be the serial number of $|\$\rangle$, s_b be the serial number of $|\$_b\rangle$ for $b = 1, 2$. To pass the check, there are two possible cases:

- $s_1 = s_2 = s$. In this case, $|\$_1\rangle |\$_2\rangle$ breaks the security of the quantum money scheme because one successfully duplicates a banknote with the same serial number.
- At least one of $s_b \neq s$. Because the mark of $\tilde{f}_b$ is also equal to s_b, one of $\tilde{f}_b$ breaks the security of the watermarking scheme, as it preserves the functionality, while having a different mark from s.

We show that the above construction and proof apply to a wide range of watermarking primitives.

Copy-Protection in the Standard Model? The security of our copy-protection scheme requires treating the obfuscated programs as oracles. While we prove security for all unlearnable programs, we cannot expect such security to hold in the standard model: as shown in [AP20], there are unlearnable functions that can not be copy-protected or even copy-detected. On the other hand, watermarkable programs are a natural class of programs that are necessarily immune to the style of counter-example of Barak et al. [BGI+01], on which the copy-protection impossibility is based. Namely, the counter-example works by giving programs that are unlearnable, but such that having any (even approximate [BP15]) code for the program lets you recover the original program. Such programs *cannot* be watermarkable, as the adversary can always recover the original program from the (supposedly) watermarked program.

Thus, we broadly conjecture that all watermarkable functions can be copy-protected. Our copy-detection result gives some evidence that this may be feasible. Concretely, we conjecture that our copy-protection construction is secure for any watermarkable program when the oracles are instantiated with post-quantum obfuscation constructions. We leave justifying either the broad or concrete conjectures as fascinating open questions.

18.1.3 Other related works

Quantum Copy Protection Quantum copy-protection was proposed by Aaronson in [Aar09]; this paper gave two candidate schemes for copy-protecting point functions without security proofs and showed that any functions that are not quantum learnable could be quantum copy-protected relative to a quantum oracle (an oracle which could perform an arbitrary unitary).

[AP20] gave a conditional impossibility of general copy-protection: they construct a quantum unlearnable circuit using the quantum FHE scheme and compute-and-compare obfuscation [WZ17, GKW17], which is not copy-protectable once a QPT adversary has non-black-box access to the program. [AP20] also gave a new definition that is weaker than the standard copy-protection security, called Secure Software Leasing (SSL) and an SSL construction for a subclass of evasive functions, namely, searchable compute-and-compare circuits.

[BL19] and [GZ20] introduced two new notions respectively, unclonable encryption/ decryption schemes; [CMP20] gave a construction for copy-protecting point functions in the quantum random oracle model with techniques inspired by [BL19] and the construction can be extended to copy-protecting compute-and-compare circuits. [BJL$^+$21] then constructed information-theoretic SSL for point functions.

Quantum Money Quantum money was first proposed by Wiesner in around 1970; [Wie83] gave the first private-key quantum money scheme based on conjugate coding. Aaronson [Aar09] gave a first public-key quantum money scheme. The explicit scheme was broken by Lutomirski et al. [LAF$^+$09]. Later, [AC12] proposed a secure public-key quantum money scheme relative to a classical oracle. Zhandry [Zha19] put forward an even stronger notion, quantum lightning; [Zha19] also instantiated the quantum money scheme of [AC12] with post-quantum indistinguishability obfuscation. Other public-key quantum money schemes include [Kan18, FGH$^+$12] and private-key quantum money scheme from [JLS18]. One other variant is classically verifiable quantum money [Gav12, RS19].

One-time Programs and One-time Memory Another idea of copy-protecting softwares is through one-time program, introduced in [GKR08]. One-time programs can be executed on only one single input and nothing other than the result of this computation is leaked. Quantum one-time programs are further studied in [BGS13] and [LSZ20].

18.1.4 Concurrent and independent work

Very recently, [KNY20] presents a secure software leasing for a subclass of evasive functions and PRFs, using watermarking and two-tier quantum-lightning, which can be built from the LWE assumption. Their main observation is that the full power of public-key quantum money is not needed in the verification of SSL, and they introduce a new primitive in between public-key and private-key quantum money, which they call two-tier quantum lightning. While their construction can be built from LWE alone, our construction aims at a more generalized definition in terms of successful piracy and functionality-preserving; our copy detection construction also works for a broader class of cryptographic functionalities such as encryption and signature.

18.2 Preliminaries

We denote by λ the security parameter, and when inputted into an algorithm, λ will be represented in unary. We say a function $\epsilon(x)$ is *negligible* if for all inverse polynomials $1/p(x)$, $\epsilon(x) < 1/p(x)$ for all large enough x. We denote a negligible function by $\mathsf{negl}(x)$. We use QPT to denote quantum polynomial time.

18.2.1 Quantum computation

We give some basic definitions of quantum computation and quantum information in Appendix B.

Definition 18.1 (Trace distance)**.** Let $\rho, \sigma \in \mathbb{C}^{2^n \times 2^n}$ be the density matrices of two quantum states. The trace distance between ρ and σ is

$$\|\rho - \sigma\|_{\mathrm{tr}} := \frac{1}{2}\sqrt{\mathrm{Tr}[(\rho - \sigma)^\dagger(\rho - \sigma)]},$$

Here, we only state a key lemma for our construction: the Gentle Measurement Lemma proposed by Aaronson [Aar04], which gives a way to perform measurements without destroying the state.

Lemma 18.1 (Gentle Measurement Lemma [Aar04])**.** *Suppose a measurement on a mixed state ρ yields a particular outcome with probability $1 - \epsilon$. Then after the measurement, one can recover a state $\tilde{\rho}$ such that $\|\tilde{\rho} - \rho\|_{\mathrm{tr}} \leq \sqrt{\epsilon}$. Here $\|\cdot\|_{\mathrm{tr}}$ is the trace distance (defined in Definition 18.1).*

18.2.2 Quantum oracle algorithm

We consider the quantum query model in this chapter, which gives quantum circuits access to some oracles.

Definition 18.2 (Classical Oracle)**.** A classical oracle $\mathcal{O}$ on input query x is a unitary transformation of the form $U_f\,|x, y, z\rangle \rightarrow |x, y + f(x), z\rangle$ for classical function $f : \{0,1\}^n \rightarrow \{0,1\}^m$. Note that a classical oracle can be queried in quantum superposition.

In the rest of the chapter, we refer to the word “oracle” as “classical oracle”. A quantum oracle algorithm with oracle access to $\mathcal{O}$ is a sequence of unitary U_i and oracle access to $\mathcal{O}$ (or U_f). Thus, the query complexity of a quantum oracle algorithm is defined as the number of $\mathcal{O}$ access.

In the analysis of the security of the copy-protection scheme in Section 18.5.2, we will use the following theorem from [BBBV97] to bound the change in adversary’s state when we change the oracle’s input-output behavior at places where the adversary hardly ever queries on.

Theorem 18.2 ([BBBV97]). *Let $|\phi_i\rangle$ be the superposition of quantum Turing machine $\mathcal{M}$ with oracle $\mathcal{O}$ on input x at time i. Define $W_y(|\phi_i\rangle)$ to be the sum of squared magnitudes in $|\phi_i\rangle$ of configurations of $\mathcal{M}$ which are querying the oracle on string y. For $\epsilon > 0$, let $F \subseteq [0, T-1] \times \Sigma^*$ be the set of time-string pairs such that $\sum_{(i,y)\in F} W_y(|\phi_i\rangle) \leq \epsilon^2/T$.*

Now suppose the answer to each query $(i, y) \in F$ is modified to some arbitrary fixed $a_{i,y}$ (these answers need not be consistent with an oracle). Let $|\phi'_i\rangle$ be the superposition of $\mathcal{M}$ on input x at time i with oracle $\mathcal{O}$ modified as stated above. Then $\||\phi_T\rangle - |\phi'_T\rangle\|_{\mathrm{tr}} \leq \epsilon$.

18.2.3 Direct-product problem and quantum signature tokens

In this section, we will define direct-product problems, which are key components of the quantum signature token scheme by Ben-David and Sattath [BDS16] and also our quantum copy-protection scheme.

Definition 18.3 (Dual Subspace). Given a subspace S of a vector space V, let $S^\perp$ be the orthogonal complement of S: the set of $y \in V$ such that $x \cdot y = 0$ for all $x \in S$. It is not hard to show: $S^\perp$ is also a subspace of V; $(S^\perp)^\perp = S$.

Definition 18.4 (Subspace Membership Oracles). A subspace membership oracle for a subspace $A \subseteq \mathbb{F}^n$, denoted as U_A, on input vector v, will output 1 if $v \in A$, $v \neq 0$ and output 0 otherwise.

Definition 18.5 (Subspace State)**.** For a subspace $A \subseteq \mathbb{F}^n$, the state $|A\rangle$ is defined as $\frac{1}{\sqrt{|A|}} \sum_{v \in A} |v\rangle$, which is a uniform superposition of all vectors in A.

Direct-Product Problem Our construction relies on the following problem called the "Direct-Product Problem" in [AC12]: for any QPT adversary $\mathcal{A}$, given one copy of $|A\rangle$ and oracle access to $U_A, U_{A^\perp}$, the problem is to finds two *non-zero* vectors such that $u \in A$ and $v \in A^\perp$.

Ben-David and Sattath [BDS16] proved the hardness of the direct-product problem for the construction of quantum signature tokens. More precisely, a signature token is a subspace state $|A\rangle$ in their construction. All vectors in $A \backslash \{0\}$ are signatures for bit 0 and all vectors in $A^\perp \backslash \{0\}$ are signatures for bit 1. Therefore, to generate valid signatures for both 0 and 1, it is required to solve the "Direct-Product Problem". We believe that our copy-protection scheme works for general signature token schemes. To keep the statements and proofs simple, we focus on the construction in [BDS16].

Theorem 18.3 ([BDS16])**.** *Let $\epsilon > 0$ be such that $1/\epsilon = o(2^{n/2})$. Let A be a random subspace $\mathbb{F}^n$, and $\dim(A) = n/2$. Given one copy of $|A\rangle$ and access to both subspace membership oracles of U_A and $U_{A^\perp}$, an adversary needs $\Omega(\sqrt{\epsilon} 2^{n/4})$ queries to output a pair of non-zero vectors (u, v) such that $u \in A$ and $v \in A^\perp$ with probability at least ϵ.*

We will refer to the direct-product problem as a security game, which is defined as follows:

Definition 18.6 (Direct-Product Game)**.** A direct-product game consists of the following steps:

Setup Phase: the challenger takes in a security parameter λ, samples a random $\lambda/2$-dimensional subspace A from $\mathbb{F}^\lambda$; then prepares the membership oracle U_A for A, $U_{A^\perp}$ for the dual subspace $A^\perp$ and a quantum state $|A\rangle$.

Query Phase: the challenger sends $|A\rangle$ to the adversary; the adversary can query $U_A, U_{A^\perp}$ for polynomially many times.

Output Phase: the adversary outputs two vectors (u, v).

The challenger checks if $u \in A \setminus \{0\}, v \in A^\perp \setminus \{0\}$. If this is satisfied, then the adversary wins the game.

Theorem 18.3 shows that for any QPT adversary, the winning probability of the direct-product game is negligible.

18.2.4 Testing quantum programs: measurement implementation

In classical cryptographic security games, the challenger typically gets some information from the adversary and checks if this information satisfies certain properties.

However, in the quantum world, when a challenger tries to decide if a quantum adversary has produced a state with certain properties, especially in the context of security games for properties related to unclonability, classical definitions of "testing properties" may result in various failures as discussed in [Zha20]. Such an issue has also been discussed in the introduction.

To deal with this issue, [Zha20] formalized a measurement procedure for testing an adversary's state. This is best understood with an example.

Consider a security game where the adversary needs to produce a state that can decrypt a challenge ciphertext. First, the challenger's behavior is abstracted into the following:

- Encrypt a random message bit m to get c using randomness rand, note that randomness rand is used to sample m and random coins for encryption;
- Run the adversary's state on the resulting ciphertext c;

- Output 1 or 0 depending on whether the adversary's state correctly decrypts or not.

Fixing the ciphertext c, the procedure of outputting 1 or 0 depending on whether the adversary's state correctly decrypts c can be described as a projective measurement $\mathcal{P}_{m,c} = (P_{m,c}, Q_{m,c})$ where $P_{m,c}$ corresponds to output 1, $Q_{m,c}$ corresponds to output 0 and $(P_{m,c}, Q_{m,c})$ can be efficiently implemented given subscript m, c. The challenger uses m, c as a control to decide which projective measurement to be applied to the state.

More generally, let $\mathcal{R}$ be the set of randomness, $\mathcal{I}$ be the control set (similar to the role of m, c in the above example). Let D be a function from $\mathcal{R}$ to $\mathcal{I}$. For a uniform randomness rand, $D(\mathsf{rand})$ defines a distribution over $\mathcal{I}$. Therefore we will use the word "distribution" for D in the rest of the discussion. For every control (or index) $i \in \mathcal{I}$, we have a projective measurement $\mathcal{P}_i = (P_i, Q_i)$. Let $\mathcal{P} = \{\mathcal{P}_i = (P_i, Q_i)\}$ be a collection of binary projective measurements. We define a mixture of projective measurement $\mathcal{P}_D$ as follows.

Definition 18.7 (Mixture of Projective Measurement $\mathcal{P}_D$)**.** For $\mathcal{P} = \{P_i, Q_i\}_{i \in \mathcal{I}}$ and $D : \mathcal{R} \to \mathcal{I}$, a mixture of projective measurement $\mathcal{P}_D = (P_D, Q_D)$ is a POVM defined as the following:

$$P_D = \sum_{i \in \mathcal{I}} \Pr[i \leftarrow D(R)]\, P_i, \qquad Q_D = \sum_{i \in \mathcal{I}} \Pr[i \leftarrow D(R)]\, Q_i,$$

where R is a uniform random variable in $\mathcal{R}$.

In other words, $\mathcal{P}_D$ is implemented in the following way: sample randomness $\mathsf{rand} \leftarrow \mathcal{R}$, compute index/control $i \leftarrow D(\mathsf{rand})$ and apply projective measurement $\mathcal{P}_i = (P_i, Q_i)$.

Thus, for any quantum state ρ, $\mathrm{Tr}[P_D \rho]$ is the probability that a random sampled projective measurement $\mathcal{P}_i = (P_i, Q_i)$ (according to the distribution D) applies on ρ and outputs 1.

Definition 18.8 (Projective Implementation)**.** Let $\mathcal{P} = (P, Q)$ be a binary outcome POVM. Let $\mathcal{D}$ be a finite set of distributions $(p, 1-p)$ over outcomes $\{0, 1\}$. Let $\mathcal{E} = \{E_p\}_{(p,1-p)\in\mathcal{D}}$ be a projective measurement with index set $\mathcal{D}$. Consider the following measurement experiment:

- Measure under the projective measurement $\mathcal{E}$ and obtain a distribution $(p, 1-p)$ over $\{0, 1\}$;
- Output a bit according to the distribution: output 1 with probability p and output 0 with probability $1 - p$.

We say the measurement $\mathcal{E}$ is a projective implementation of $\mathcal{P}$ if the above experiment and $\mathcal{P}$ produce identical outcomes on any quantum states. We denote $\mathcal{E}$ by $\mathsf{ProjImp}(\mathcal{P})$.

Note that if the collapsed state is an eigenvector of P corresponding to eigenvalue p, then it is also an eigenvector of Q corresponding to eigenvalue $1 - p$.

Lemma 18.4 (A variation of Lemma 1 in [Zha20])**.** *Any binary outcome POVM $\mathcal{P} = (P, Q)$ has a unique projective measurement $\mathsf{ProjImp}(\mathcal{P})$.*

In this chapter, we propose the following new definition corresponding to $\mathsf{ProjImp}$.

Definition 18.9 (Threshold Implementation)**.** A threshold implementation with parameter γ of a binary POVM $\mathcal{P} = (P, Q)$ is a variant of projective implementation $\mathsf{ProjImp}(\mathcal{P})$, denoted as $(\mathsf{TI}_\gamma(\mathcal{P}), \mathbf{I} - \mathsf{TI}_\gamma(\mathcal{P}))$:

- Measure under the projective measurement $\mathcal{E}$ ($\mathsf{ProjImp}(\mathcal{P})$) and obtain a distribution $(p, 1-p)$ over $\{0, 1\}$;
- Output a bit according to the distribution $(p, 1-p)$: output 1 if $p \geq \gamma$, or 0 otherwise.

Remark 18.1. For any quantum state ρ, the threshold implementation outputs 1 with probability $\mathrm{Tr}[\mathsf{TI}_\gamma(\mathcal{P})\rho]$, and 0 with probability $1 - \mathrm{Tr}[\mathsf{TI}_\gamma(\mathcal{P})\rho]$.

Remark 18.2. For a binary outcome measurement $\mathcal{P} = (P, Q)$, we usually say "perform measurement P on ρ" if $\mathcal{P}$ was performed on ρ. For example, if we say a threshold implementation $\mathsf{TI}(\mathcal{P}_D)$ on a quantum state ρ outputs 1, we refer to apply $(\mathsf{TI}_\gamma(\mathcal{P}_D), \mathbf{I} - \mathsf{TI}_\gamma\mathcal{P}_D)$ on ρ and the outcome is 1.

18.2.4.1 Approximating Projective Implementation

Before describing the theorem of the approximation algorithm, we give two definitions that characterize how good an approximation projective implementation is, which were first introduced in [Zha20].

Definition 18.10 (Shift Distance)**.** For two distributions D_0, D_1, the shift distance with parameter ϵ is defined as $\Delta^{\epsilon}_{\mathsf{Shift}}(D_0, D_1)$, which is the smallest quantity δ such that for all $x \in \mathbb{R}$:

$$\Pr[D_0 \le x] \le \Pr[D_1 \le x + \epsilon] + \delta,$$
$$\Pr[D_1 \le x] \le \Pr[D_0 \le x + \epsilon] + \delta.$$

For two real-valued measurements $\mathcal{M}$ and $\mathcal{N}$ over the same quantum system, the shift distance between $\mathcal{M}$ and $\mathcal{N}$ with parameter ϵ is defined as,

$$\Delta^{\epsilon}_{\mathsf{Shift}}(\mathcal{M}, \mathcal{N}) := \sup_{|\psi\rangle} \Delta^{\epsilon}_{\mathsf{Shift}}\left(\mathcal{M}(|\psi\rangle), \mathcal{N}(|\psi\rangle)\right).$$

Definition 18.11 ((ϵ, δ)-Almost Projective)**.** A real-valued quantum measurement $\mathcal{M}$ is said to be (ϵ, δ)-almost projective if for all quantum state $|\psi\rangle$, apply $\mathcal{M}$ twice in a row to $|\psi\rangle$, obtaining outcomes X and Y. Then we have $\Pr[|X - Y| \le \epsilon] \ge 1 - \delta$.

The following theorem gives a way to approximate any projective implementation:

Theorem 18.5 (Theorem 2 in [Zha20]). *Let D be any probability distribution over some control set $\mathcal{I}$ and $\mathcal{P}$ be a collection of projective measurements. For any $0 < \epsilon, \delta < 1$, there exists an algorithm of measurement $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}$ that satisfies the followings:*

- $\Delta^{\epsilon}_{\mathsf{Shift}}(\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}, \mathsf{ProjImp}(\mathcal{P}_D)) \leq \delta$.
- $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}$ *is (ϵ, δ)-almost projective.*
- *The expected running time of $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}$ is $T_{\mathcal{P},D} \cdot \textsf{poly}(1/\epsilon, \log(1/\delta))$ where $T_{\mathcal{P},D}$ is the combined running time of D, the procedure mapping i to (P_i, Q_i) and the run-time of measurement (P_i, Q_i).*

18.3 Learning Game Definitions

In this section, we define unlearnability, copy-protection, copy-detection, and watermarking with respect to a function family and a testing distribution.

We assume a function f is sampled uniformly at random from a function family $\mathcal{F}_\lambda$. To test the correctness of a quantum program (for computing f), it samples an input x from a testing distribution D_λ, runs the quantum program on x, and checks if the output is $f(x)$.

We will give the generalized definitions (for unlearnability, copy-protection, copy-detection, and watermarking) in Appendix G.3, which allow for more general sampling procedures and functionality testing. Since our constructions naturally extend to these settings, we leave all the discussions about definitions and proofs in the appendix.

Definition 18.12 (Quantum Program with Classical Inputs and Outputs). A quantum program with classical inputs is a pair of quantum state ρ and unitaries $\{U_x\}_{x\in[N]}$ (where $[N]$ is the domain), such that the state of the program evaluated on input x is equal to $U_x \rho U_x^\dagger$. To obtain an output, it measures the first register of $U_x \rho U_x^\dagger$. Moreover, $\{U_x\}_{x\in[N]}$ has a compact classical description which means "applying U_x" can be efficiently computed given x.

Notation-wise, the input and output space N, M are functions of λ.

18.3.1 Unlearnability

First, we define γ-goodness testing with respect to a fixed function f and a testing distribution D (over inputs).

Definition 18.13 (γ-Goodness Test with respect to f, D)**.** Let $(\rho, \{U_x\}_{x\in[N]})$ be a quantum program for computing a classical function $f : [N] \to [M]$. Let D be a testing distribution over the input space $[N]$.

- Define $\mathcal{P}_x = (P_x, Q_x)$ be the following projective measurement:
  - On input x, it runs U_x on the quantum state ρ;
  - It measures whether the output register is equal to $f(x)$; output 1 if yes, and 0 otherwise.

  Let $\mathcal{P} = \{\mathcal{P}_x\}$ be a collection of projective measurements.
- D is the distribution that samples an input: given randomness rand, output $x = D(\mathsf{rand})$.
- Let $\mathcal{P}_D = (P_D, Q_D)$ be the mixture of projective measurement defined in Definition 18.7.
- We say a quantum program is tested **γ-good** with respect to f, D, if the threshold implementation $\mathsf{TI}_\gamma(\mathcal{P}_D)$ outputs 1.

We then define a learning game for a function family $\mathcal{F}$ and a set of testing distribution $\mathcal{D}$. Note that we assume for a fixed security parameter λ, f is drawn *uniformly at random* from $\mathcal{F}_\lambda$ and the testing distribution D_f is efficiently sampleable given the description f.

Definition 18.14 (Learning Game for $\mathcal{F}, \mathcal{D}$)**.** A learning game for a function family $\mathcal{F} = \{\mathcal{F}_\lambda : [N] \to [M]\}$, a distribution family $\mathcal{D} = \{D_f\}$, a threshold $\gamma \in (0,1)$, and an adversary $\mathcal{A}$ is denoted as $\mathsf{LearningGame}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}(1^\lambda)$, which consists of the following steps:

1. **Sampling Phase**: At the beginning of the game, the challenger takes a security parameter λ and samples a function $f \leftarrow \mathcal{F}_\lambda$ uniformly at random;
2. **Query Phase**: $\mathcal{A}$ then gets oracle access to f;
3. **Output Phase**: Finally, $\mathcal{A}$ outputs a quantum program $(\rho, \{U_x\}_{x\in[N]})$.

The game outputs 1 if and only if the program is tested γ-good with respect to f, D_f.

Definition 18.15 (Quantum Unlearnability of $\mathcal{F}$ with Testing Distribution $\mathcal{D}$)**.** A family of functions $\mathcal{F}$ with respect to $\mathcal{D}$ is called γ quantum unlearnable if for any QPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that the following holds for all λ:

$$\Pr\left[b = 1,\, b \leftarrow \mathsf{LearningGame}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}(1^\lambda)\right] \leq \mathsf{negl}(\lambda)$$

18.3.2 Copy-protection

Definition 18.16 (Quantum Copy-Protection)**.** A quantum copy-protection scheme for $\mathcal{F}, \mathcal{D}$ consists of the following procedures:

$\mathsf{Setup}(1^\lambda) \to \mathsf{sk}$: the setup algorithm takes in a security parameter λ in unary and generates a secret key sk.

$\mathsf{Generate}(\mathsf{sk}, f) \to (\rho_f, \{U_{f,x}\}_{x\in[N]})$: on input $f \in \mathcal{F}_\lambda$ and secret key sk, the vendor generates a quantum program $(\rho_f, \{U_{f,x}\}_{x\in[N]})$.

$\mathsf{Compute}(\rho_f, \{U_{f,x}\}_{x\in[N]}, x) \to y$: given a quantum program, a user can compute the function $f(x)$ on input x by applying $U_{f,x}$ on ρ_f and measuring the first register of the state.

Efficiency: Setup, $\mathsf{Compute}$ and $\mathsf{Generate}$ should run in $\mathsf{poly}(\lambda)$ time.

Correctness: There exists a negligible function $\mathsf{negl}(\cdot)$ such that: all $\lambda \in \mathbb{N}$, every $f \in \mathcal{F}_\lambda$, all $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda)$, all $(\rho_f, \{U_{f,x}\}_{x\in[N]}) \leftarrow \mathsf{Generate}(\mathsf{sk}, f)$, for all $x \in [N]$, apply $U_{f,x}$ on ρ_f and measure the first register, with probability at least $1 - \mathsf{negl}(\lambda)$, the output is a fixed value $z_{f,x}$; moreover, $z_{f,x} = f(x)$.

Security: The γ-anti-piracy security defined below.

Note that correctness ensures that the copy-protected program can be evaluated polynomially many times.

Definition 18.17 (γ-Anti-Piracy Security Game)**.** An anti-piracy security game for $\mathcal{F}, \mathcal{D}$ and adversary $\mathcal{A}$ is denoted as $\mathsf{CopyProtectionGame}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}(1^\lambda)$, which consists of the following steps:

1. **Setup Phase**: At the beginning of the game, the challenger takes a security parameter λ and obtains a secret key $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda)$.
2. **Sampling Phase**: A function f is sampled uniformly at random, $f \leftarrow \mathcal{F}_\lambda$.
3. **Query Phase**: $\mathcal{A}$ makes a single query to the challenger and obtains a copy-protection program for f: $(\rho_f, \{U_{f,x}\}_{x\in[N]}) \leftarrow \mathsf{Generate}(\mathsf{sk}, f)$.
4. **Output Phase**: Finally, $\mathcal{A}$ outputs a (possibly mixed and entangled) state σ over two registers R_1, R_2 and two sets of unitaries $(\{U_{R_1,x}\}_x, \{U_{R_2,x}\}_x)$ They can be viewed as programs $\mathsf{P}_1 = (\sigma[R_1], \{U_{R_1,x}\}_{x\in[N]})$ and $\mathsf{P}_2 = (\sigma[R_2], \{U_{R_2,x}\}_{x\in[N]})$.

The game outputs 1 if and only if both programs $\mathsf{P}_1, \mathsf{P}_2$ are tested to be γ-good with respect to $f, \mathcal{D}_f$.

Definition 18.18 (γ-Anti-Piracy-Security)**.** A copy-protection scheme for $\mathcal{F}$ and $\mathcal{D}$ has γ-anti-piracy security, if for any QPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that the following holds for all $\lambda \in \mathbb{N}$:

$$\Pr\left[b = 1, b \leftarrow \mathsf{CopyProtectionGame}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}(1^\lambda)\right] \le \mathsf{negl}(\lambda) \tag{18.1}$$

18.3.3 Copy-detection

A copy-detection scheme is very similar to the copy-protection scheme, except it has an additional procedure Check which applies a projective measurement and checks if the quantum state is valid.

Definition 18.19 (Quantum Copy-Detection)**.** A quantum copy-detection scheme for $\mathcal{F}, \mathcal{D}$ consists of the following procedures:

$\mathsf{Setup}(1^\lambda)$, $\mathsf{Generate}(\mathsf{sk}, f)$ and $\mathsf{Compute}(\rho_f, \{U_{f,x}\}_{x\in[N]}, x)$ are the same as those in Definition 18.16, except Setup additionally samples a public key for Check.

$\mathsf{Check}(\mathsf{pk}, \rho_f, \{U_{f,x}\}_{x\in[N]}) \to b, \rho'$: on input a quantum program, it applies a binary projective measurement (P_0, P_1) on ρ_f that depends on $\{U_{f,x}\}_{x\in[N]}$; it outputs the outcome b and the leftover state ρ'.

Correctness ($\mathsf{Generate}$): The same as the security of Definition 18.16.

Correctness (Check): For every efficient $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that, all $\lambda \in \mathbb{N}$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$, every $f \in \mathcal{F}_\lambda$, all $(\rho_f, \{U_{f,x}\}_{x\in[N]}) \leftarrow \mathsf{Generate}(\mathsf{sk}, f)$, $\mathsf{Check}(\mathsf{pk}, \rho_f, \{U_{f,x}\}_{x\in[N]})$ outputs 1 with probability at least $1 - \mathsf{negl}(\lambda)$.

Security: The γ-copy-detection security defined below.

Definition 18.20 (γ-Copy-Detection Security Game)**.** A copy-detection security game for $\mathcal{F}, \mathcal{D}$ and adversary $\mathcal{A}$ is denoted as $\mathsf{CopyDetectionGame}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}(1^\lambda)$, which consists of the following steps:

1. **Setup Phase**: At the beginning of the game, the challenger takes a security parameter λ and obtains keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$.
2. **Sampling Phase**: A function f is sampled uniformly at random, $f \leftarrow \mathcal{F}_\lambda$.

3. **Query Phase**: $\mathcal{A}$ makes a single query to the challenger and obtains a quantum program for f: $(\rho_f, \{U_{f,x}\}_{x\in[N]}) \leftarrow \mathsf{Generate}(\mathsf{sk}, f)$.

4. **Output Phase**: Finally, $\mathcal{A}$ outputs a (possibly mixed and entangled) state σ over two registers R_1, R_2 and two sets of unitaries $(\{U_{R_1,x}\}_x, \{U_{R_2,x}\}_x)$ They can be viewed as programs $\mathsf{P}_1 = (\sigma[R_1], \{U_{R_1,x}\}_{x\in[N]})$ and $\mathsf{P}_2 = (\sigma[R_2], \{U_{R_2,x}\}_{x\in[N]})$.

The game outputs 1 if and only if

- Apply Check on P_i respectively and both outcomes are 1. Let P_i' be the collapsed program conditioned on outcomes are 1.
- *Both* programs $\mathsf{P}_1', \mathsf{P}_2'$ are tested γ-good with respect to f, D_f.

Definition 18.21 (γ-Copy-Detection-Security)**.** A copy-detection scheme for $\mathcal{F}, \mathcal{D}$ has γ-copy-detection security, if for any QPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that the following holds for all $\lambda \in \mathbb{N}$:

$$\Pr\left[b = 1, b \leftarrow \mathsf{CopyDetectionGame}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}(1^\lambda)\right] \leq \mathsf{negl}(\lambda) \tag{18.2}$$

18.3.4 Watermarking primitives with public extraction

In this section, we formalize watermarking. We will give the generalized notations in Appendix G.3.4.

Definition 18.22 (Watermarking Primitives for $\mathcal{F}, \mathcal{D}$)**.** A watermarking scheme for $\mathcal{F}, \mathcal{D}$ consists of the following classical algorithms:

$\mathsf{Setup}(1^\lambda)$: it takes as input a security parameter 1^λ and outputs keys $(\mathsf{xk}, \mathsf{mk})$. xk is the extracting key and mk is the marking key. We only consider the publicly extractable watermarking schemes. Thus, xk is always public.

$\mathsf{Mark}(\mathsf{mk}, f, \tau)$: it takes a circuit f and a message $\tau \in \mathcal{M}_\lambda$, outputs a marked circuit $\widetilde{f}$.

$\mathsf{Extract}(\mathsf{xk}, f')$: it takes a circuit f' and outputs a message in $\{\perp\} \cup \mathcal{M}_\lambda$.

It satisfies the following properties.

Definition 18.23 (Correctness of Mark (Functionality Preserving))**.** For for every efficient algorithm $\mathcal{A}$, there exists a negligible function negl, for all $(\mathsf{xk}, \mathsf{mk}) \leftarrow \mathsf{Setup}(1^\lambda)$, and every $\tau \in \mathcal{M}_\lambda$, all λ,

$$\Pr\left[\tilde{f}(x) = f(x) \;:\; \substack{f \leftarrow \mathcal{F}_\lambda \\ \tilde{f} \leftarrow \mathsf{Mark}(\mathsf{mk}, f, \tau) \\ x \leftarrow D_f}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Definition 18.24 (Correctness of $\mathsf{Extract}$)**.** For every efficient algorithm $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$, for all $(\mathsf{xk}, \mathsf{mk}) \leftarrow \mathsf{Setup}(1^\lambda)$, and every $\tau \in \mathcal{M}_\lambda$, all λ,

$$\Pr\left[\tau \neq \mathsf{Extract}(\mathsf{xk}, \widetilde{f}) \;:\; \substack{f \leftarrow \mathcal{F}_\lambda \\ \widetilde{f} \leftarrow \mathsf{Mark}(\mathsf{mk}, f, \tau)}\right] \leq \mathsf{negl}(\lambda).$$

Definition 18.25 (γ-Unremovability with respect to $\mathcal{F}, \mathcal{D}$)**.** Consider the following game, denoted as $\mathsf{WaterMarkingGame}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}$:

1. **Setup**: The challenger samples $(\mathsf{xk}, \mathsf{mk}) \leftarrow \mathsf{Setup}(1^\lambda)$. $\mathcal{A}$ then gets xk.
2. **Sampling Phase**: A function f is sampled uniformly at random in $\mathcal{F}_\lambda$.
3. **Query Phase**: $\mathcal{A}$ has classical access to $\mathsf{Mark}(\mathsf{mk}, f, \cdot)$ at any time. Define Q be the set of messages that $\mathcal{A}$ has queried on.
4. **Output Phase**: Finally, the algorithm outputs a circuit f^*.

The adversary wins the game if and only if

$$\mathsf{Extract}(\mathsf{xk}, f^*) \notin Q \;\wedge\; \Pr_{x \leftarrow D_f}[f^*(x) = f(x)] \geq \gamma$$

We say a watermarking scheme has γ-unremovability with respect to $\mathcal{F}, \mathcal{D}$, if for all QPT $\mathcal{A}$, it wins the above game with negligible probability in λ.

Remark 18.3. Here, we only consider a weaker security notion where a quantum adversary only has classical oracle access in the query phase. We claim it is practical and good enough in most of the settings since the watermarking key mk is only in the hands of the challenger: whenever adversary queries $\mathsf{Mark}(\mathsf{mk}, f, \cdot)$, the challenger can always measure this query.

Remark 18.4. Watermarking primitives should also satisfy 'meaningfulness' property [GKM+19] but since we do not use this property in our construction, we omit it here.

18.4 Approximating Threshold Implementation

By applying $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}$ and checking if the outcome is greater than or smaller than γ, we get an approximated threshold implementation $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma}$. Here, we use $(\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma}, \mathbf{I} - \mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma})$ to denote this binary POVM.

Theorem 18.5 gives the following theorem on approximating threshold implementation:

Theorem 18.6. *For any $\epsilon, \delta, \gamma, \mathcal{P}, D$, the algorithm of measurement $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma}$ that satisfies the followings:*

- *For all quantum state ρ, $\mathrm{Tr}[\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma-\epsilon} \cdot \rho] \geq \mathrm{Tr}[\mathsf{TI}_{\gamma}(\mathcal{P}_D) \cdot \rho] - \delta$.*
- *By symmetry, for all quantum state ρ, $\mathrm{Tr}[\mathsf{TI}_{\gamma-\epsilon}(\mathcal{P}_D) \cdot \rho] \geq \mathrm{Tr}[\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma} \cdot \rho] - \delta$.*
- *For all quantum state ρ, let ρ' be the collapsed state after applying $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma}$ on ρ (conditioned on outcome 1). Then, $\mathrm{Tr}[\mathsf{TI}_{\gamma-2\epsilon}(P_D) \cdot \rho'] \geq 1 - 2\delta$.*
- *The expected running time is the same as $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}$.*

Intuitively the theorem says that if a quantum state ρ has weight p on eigenvectors with eigenvalues at least γ, the measurement $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma-\epsilon}$ with probability at least $p - \delta$ will produce a collapsed state which has weight $1 - 2\delta$ on eigenvectors with eigenvalues at least $\gamma - 2\epsilon$. Also note that the running time is proportional to

$\mathsf{poly}(1/\epsilon, 1/(\log\delta))$, which is a polynomial in λ as long as ϵ is any inverse polynomial and δ is any inverse sub-exponential function. The proof of the above theorem is in Appendix G.2.1.

We can also consider approximating the measurements on a bipartite (possibly entangled) quantum state. In this case, we will prove a similar statement as Theorem 18.6.

Lemma 18.7. *Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be two collections of projective measurements and D_1 and D_2 be any probability distributions defined on the index set of $\mathcal{P}_1$ and $\mathcal{P}_2$ respectively. For any $0 < \epsilon, \delta, \gamma < 1$, the algorithms $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_1,D_1,\gamma}$ and $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_2,D_2,\gamma}$ satisfy the followings:*

- *For any bipartite (possibly entangled, mixed) quantum state $\rho \in \mathscr{H}_{\mathcal{L}} \otimes \mathscr{H}_{\mathcal{R}}$,*
$$\operatorname{Tr}\left[\left(\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_1,D_1,\gamma-\epsilon} \otimes \mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_2,D_2,\gamma-\epsilon}\right)\rho\right] \geq \operatorname{Tr}\left[\left(\mathsf{TI}_{\gamma}(\mathcal{P}_{D_1}) \otimes \mathsf{TI}_{\gamma}(\mathcal{P}_{D_2})\right)\rho\right] - 2\delta.$$
- *For any (possibly entangled, mixed) quantum state ρ, let ρ' be the collapsed state after applying $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_1,D_1,\gamma} \otimes \mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_2,D_2,\gamma}$ on ρ (and normalized). Then,*
$$\operatorname{Tr}\left[\left(\mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_1}) \otimes \mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_2})\right)\rho'\right] \geq 1 - 4\delta.$$

We defer the proof of the above Lemma to Appendix G.2.2.

18.5 Quantum Copy-Protection Scheme

Let λ be the security parameter. Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ be a class of circuits. We assume $\mathcal{F}$ is quantum unlearnable with respect to $\mathcal{D}$ (see Definition 18.15) and can be computed by polynomial-sized classical circuits. The construction for quantum copy-protection of function class $\mathcal{F}_\lambda$ is defined in Fig. 18.1.

Note that this construction works for general quantum unlearnable function families as well. By simply changing the notation in the proof to that in the general quantum unlearnability case, we prove it for general quantum unlearnable function families. More discussion will be given in the appendix.

$\mathsf{Setup}(1^\lambda) \to \mathsf{sk}$: The setup algorithm takes in security parameter 1^λ.

- Pick a uniformly random subspace $A \subseteq \mathbb{F}^\lambda$ of dimension $\lambda/2$.
- Output $\mathsf{sk} = A$, where A is described by a set of orthogonal basis vectors.

$\mathsf{Generate}(\mathsf{sk}, f \in \mathcal{F}_\lambda)$: The $\mathsf{Generate}$ algorithm receives $\mathsf{sk} = A$ and a function f from $\mathcal{F}_\lambda$.

- Prepare a subspace state on n qubits corresponding to A, $|A\rangle = \frac{1}{\sqrt{|A|}} \sum_{v\in A} |v\rangle$.
- Generate oracles $U_A, U_{A^\perp}$ which compute subspace membership functions for subspace A and its dual subspace $A^\perp$ respectively.
- Generate oracles $\mathcal{O}_1, \mathcal{O}_2$ such that

$$\mathcal{O}_1(x,v) = \begin{cases} f(x) \oplus g(x) & \text{if } v \in A \text{ and } v \neq 0, \\ \perp & \text{otherwise.} \end{cases}$$

$$\mathcal{O}_2(x,v) = \begin{cases} g(x) & \text{if } v \in A^\perp \text{ and } v \neq 0, \\ \perp & \text{otherwise.} \end{cases}$$

  where g is a uniformly random function, with the same input and output length as f.

- Finally, the $\mathsf{Generate}$ algorithm outputs a quantum program $\big(\rho = |A\rangle\langle A|, \{U_x\}_{x\in[N]}\big)$, which describes the following procedure:
  - On input x, prepare the state $|0\rangle\langle 0| \otimes |x\rangle\langle x| \otimes \rho$ and make an oracle query U_A and measure the first register (output register) to get y_1; the remaining state is $|x\rangle\langle x| \otimes \rho'$.
  - Apply QFT on the third register ρ' to get ρ''.
  - Prepare the state $|0\rangle\langle 0| \otimes |x\rangle\langle x| \otimes \rho''$ and make an oracle query $U_{A^\perp}$ and measure the first register to get y_2.
  - Output $y_1 \oplus y_2$.

The description of $\{U_x\}_{x\in[N]}$ requires the oracle of $U_A, U_{A^\perp}$ (or the VBB obfuscations).

Figure 18.1: Quantum copy-protection scheme.

Oracle Heuristics In practice we use a quantum-secure PRF [Zha12] to implement function g; and we use quantum-secure (classical) VBB obfuscation to implement each of $(\mathcal{O}_1, \mathcal{O}_2, U_A, U_{A^\perp})$. We can replace VBB obfuscation programs with oracles that only allow black-box access by the security of VBB obfuscation; afterwards, we can also replace PRF g with a real random function by the property of PRF. The heuristic analysis is straightforward and we omit them here.

18.5.1 Correctness and efficiency

Correctness For the quantum program $\left(\rho = |A\rangle\langle A|, \{U_x\}_{x\in[N]}\right)$ produced by the Generate algorithm, it performs the following computation:

1. Make an oracle query $\mathcal{O}_1$ on the state $|0\rangle|x\rangle|A\rangle$, the resulting state is statistically close to $|y_1\rangle|x\rangle|A\rangle$. Note that $|A\rangle$ with overwhelming probability $1-1/|A|$ contains a non-zero vector in A. It measures y_1, which is $y_1 = f(x) \oplus g(x)$.

2. It then prepares a state by applying QFT on the third register and the resulting state is is statistically close to $|0\rangle|x\rangle\left|A^\perp\right\rangle$. It makes an oracle query $\mathcal{O}_2$ on the state $|0\rangle|x\rangle\left|A^\perp\right\rangle$, the resulting state is statistically close to $|y_2\rangle|x\rangle\left|A^\perp\right\rangle$ where $y_2 = g(x)$.

Therefore, with overwhelming probability, the output is $y_1 \oplus y_2 = f(x)$.

Efficiency In Generate algorithm, as shown in [AC12], given the basis of A, the subspace state $|A\rangle$ can be prepared in polynomial time using QFT. For the oracles $\mathcal{O}_1, \mathcal{O}_2$, it only needs to check the membership of A and $A^\perp$ and compute functions f and g. f can be prepared in polynomial time by definition. As we discussed above, we can prepare the function g as a PRF. Therefore, the oracles $\mathcal{O}_1, \mathcal{O}_2$ can be generated in polynomial time. The Compute algorithm is clearly efficient.

18.5.2 Anti-piracy security

We show that for a *quantum unlearnable* families of functions $\mathcal{F}$ with respect to $\mathcal{D}$ defined in Definition 18.15, the quantum copy-protection scheme has anti-piracy security against any quantum polynomial-time adversaries. More formally:

Theorem 18.8 (Main Theorem). *Let $\mathcal{F}$ be a function families that is γ-quantum-unlearnable with respect to distribution $\mathcal{D}$ (γ is a non-negligible function of λ). The above copy-protection scheme for $\mathcal{F}, \mathcal{D}$ has $(\gamma(\lambda) - 1/\mathsf{poly}(\lambda))$-anti-piracy security, for all polynomial poly.*

In order to describe the quantum query behavior of quantum programs made to oracles, we give the following definitions and notations.

We recall that in Definition 18.13, a QPT adversary $\mathcal{A}$ in the anti-piracy security game $\mathsf{CopyProtectionGame}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}(1^\lambda)$, will produce a state σ over registers R_1, R_2 and unitaries $\{U_{R_1,x}\}_{x\in[N]}, \{U_{R_2,x}\}_{x\in[N]}$, the challenger will then perform γ-goodness test on σ using threshold implementations $\mathsf{TI}_\gamma(P_{R_1,f})$ and $\mathsf{TI}_\gamma(P_{R_2,f})$. For simplicity we will describe the unitary ensembles $\{U_{R_1,x}\}_{x\in[N]}$, $\{U_{R_2,x}\}_{x\in[N]}$ as U_{R_1}, U_{R_2} and describe threshold implementations $\mathsf{TI}_\gamma(P_{R_1,f})$, $\mathsf{TI}_\gamma(P_{R_2,f})$ as $\mathsf{TI}_{R_1,\gamma}, \mathsf{TI}_{R_2,\gamma}$. Similarly, let $\mathsf{ATI}_{R_1,\gamma-\epsilon}$ and $\mathsf{ATI}_{R_2,\gamma-\epsilon}$ denote the approximation threshold implementation $\mathsf{ATI}^{\epsilon,\delta}_{R_1,\gamma-\epsilon}$ and $\mathsf{ATI}^{\epsilon,\delta}_{R_2,\gamma-\epsilon}$ respectively, for some inverse polynomial ϵ and inverse subexponential function δ (in other words, $\log(1/\delta)$ is polynomial in λ).

In this particular construction, $\mathcal{A}$'s behavior can be described as follows: $\mathcal{A}$ "splits" the copy-protection state ρ into two potentially entangled states $\sigma[R_1], \sigma[R_2]$. $\mathcal{A}$ prepares $(\sigma[R_1], U_{R_1})$ with oracle access to $(\mathcal{O}_1, \mathcal{O}_2)$ as pirate program P_1; and prepares $(\sigma[R_2], U_{R_2})$ with oracle access $(\mathcal{O}_1, \mathcal{O}_2)$ as pirate program P_2. Therefore, $\mathsf{TI}_{R_b,\gamma}$ and $\mathsf{ATI}_{R_b,\gamma-\epsilon}$ both make oracle queries to $\mathcal{O}_1, \mathcal{O}_2$.

We can assume the joint state of R_1, R_2 has been purified and the overall state is a pure state over register R_1, R_2, R_3 where P_1 has only access to R_1 and P_2 has only access to R_2.

Quantum Query Weight Let σ be any quantum state of R_1, R_2, R_3. We consider the program P_1. P_1 has access to register R_1 and oracle access to $\mathcal{O} = (\mathcal{O}_1, \mathcal{O}_2)$. We denote $|\phi_i\rangle$ to be the overall state of registers R_1, R_2, R_3 before P_1 makes i-th query to $\mathcal{O}_1$, *when it applies* $\mathsf{ATI}_{R_1,\gamma-\epsilon}$ *on* $\sigma[R_1]$.

$$|\phi_i\rangle = \sum_{x,v,z} \alpha_{x,v,z} |x, v, z\rangle .$$

where (x, v) is the query to oracle $\mathcal{O}_1$ and z is working space of P_1, the registers of R_2, R_3. Note that when $\mathsf{ATI}_{R_1,\gamma-\epsilon}$ is applied on $\sigma[R_1]$, it in fact applies some unitary and eventually makes a measurement, during which the unitary makes queries to oracles $\mathcal{O}_1, \mathcal{O}_2$. Therefore such a query weight is well-defined.

We denote by $W_{1,A,i}$ to be the sum of squared amplitudes in $|\phi_i\rangle$, which are querying $\mathcal{O}_1$ on input (x, v) such that $v \in A \setminus \{0\}$:

$$W_{1,A,i} = \sum_{x,v,z:v\in A\setminus\{0\}} |\alpha_{x,v,z}|^2$$

Then we sum up all the squared amplitudes $W_{1,A,i}$ in all the queries made by P_1 to $\mathcal{O}_1$, where $v \in A \setminus \{0\}$. We denote this sum as $W_{1,A} = \sum_{i\in[\ell_1]} W_{1,A,i}$, where $\ell_1 = \ell_1(\lambda)$ is the number of queries made by P_1 to $\mathcal{O}_1$.

Similarly, we write $W_{1,A^\perp} = \sum_{i\in[\ell_2]} W_{1,A^\perp,i} = \sum_{i\in[\ell_2]} \sum_{x,v,z:v\in A^\perp\setminus\{0\}} |\alpha_{x,v,z}|^2$ to be the sum of squared amplitudes in $|\phi_i\rangle$ where $v \in A^\perp \setminus \{0\}$, in the ℓ_2 queries made by P_1 to $\mathcal{O}_2$.

Accordingly, for the other program P_2 and threshold implementation $\mathsf{ATI}_{R_2,\gamma-\epsilon}$, we denote these sums of squared amplitudes as $W_{2,A} = \sum_{i\in[m_1]} W_{2,A,i}$ and $W_{2,A^\perp} = \sum_{i\in[m_2]} W_{2,A^\perp,i}$, where m_1, m_2 are the number of queries made by P_2 to oracles $\mathcal{O}_1, \mathcal{O}_2$ respectively.

Case One. Fixing a function f, let $(\sigma, U_{R_1}, U_{R_2})$ be the two programs output by the adversary which are both tested γ-good with respect to f, D_f with some non-negligible probability.

Let $\mathcal{O}_\perp$ be an oracle that always outputs $\perp$. We hope one of the following events will happen:

1. The program $(\sigma[R_1], U_{R_1})$ with oracle access to $\mathcal{O}_1, \mathcal{O}_\perp$ is tested $(\gamma - 2\epsilon)$-good with respect to f, D_f, with non-negligible probability.

2. The program $(\sigma[R_2], U_{R_2})$ with oracle access to $\mathcal{O}_\perp, \mathcal{O}_2$ is tested $(\gamma - 2\epsilon)$-good with respect to f, D_f, with non-negligible probability.

Let $\widetilde{\mathsf{ATI}}_{R_1,\gamma-\epsilon}$ be the same as $\mathsf{ATI}_{R_1,\gamma-\epsilon}$ except with oracle access to $\mathcal{O}_1, \mathcal{O}_\perp$ and $\widetilde{\mathsf{ATI}}_{R_2,\gamma-\epsilon}$ be the same as $\mathsf{ATI}_{R_2,\gamma-\epsilon}$ except with oracle access to $\mathcal{O}_\perp, \mathcal{O}_2$. Similarly, let $\widetilde{\mathsf{TI}}_{R_b,\gamma-2\epsilon}$ be the same threshold implementation as $\mathsf{TI}_{R_b,\gamma-2\epsilon}$ except with oracle access to $\mathcal{O}_1, \mathcal{O}_\perp$ and $\mathcal{O}_\perp, \mathcal{O}_2$ respectively.

Since $(\sigma[R_1], U_{R_1})$ and $(\sigma[R_2], U_{R_2})$ are both γ-good with respect to $f, \mathcal{D}_f$ with non-negligible probability, for some non-negligible function $\beta(\cdot)$,

$$\mathrm{Tr}[(\mathsf{TI}_{R_1,\gamma} \otimes \mathsf{TI}_{R_2,\gamma}) \cdot \sigma] \geq \beta(\lambda)$$

From the property of the approximated threshold implementation (Lemma 18.7),

$$\mathrm{Tr}[(\mathsf{ATI}_{R_1,\gamma-\epsilon} \otimes \mathsf{ATI}_{R_2,\gamma-\epsilon}) \cdot \sigma] \geq \beta(\lambda) - 2\delta$$

Thus, for any $b \in \{1, 2\}$, we have $\mathrm{Tr}[\mathsf{ATI}_{R_b,\gamma-\epsilon} \cdot \sigma[R_b]] \geq \beta(\lambda) - 2\delta$. Since δ is negligible, both probabilities are still non-negligible.

We then define the following two events:

E_1 : Let E_1 be the event denotes $\mathrm{Tr}[\widetilde{\mathsf{ATI}}_{R_1,\gamma-\epsilon} \cdot \sigma[R_1]]$ is non-negligible. If E_1 happens, by Theorem 18.6,

$$\mathrm{Tr}[\widetilde{\mathsf{TI}}_{R_1,\gamma-2\epsilon} \cdot \sigma[R_1]] \geq \mathrm{Tr}[\widetilde{\mathsf{ATI}}_{R_1,\gamma-\epsilon} \cdot \sigma[R_1]] - \delta$$

which is still non-negligible. In other words, $(\sigma[R_1], U_{R_1})$ with oracle access to $\mathcal{O}_1, \mathcal{O}_\perp$ is tested $(\gamma - 2\epsilon)$-good with respect to f, D_f with non-negligible probability.

E_2 : Similarly, define E_2 as the program $(\sigma[R_2], U_{R_2})$ with oracle access to $\mathcal{O}_\perp, \mathcal{O}_2$ is $(\gamma - 2\epsilon)$-good with respect to f, D_f with non-negligible probability.

Case Two. Fixing a function f, let $(\sigma, U_{R_1}, U_{R_2})$ be the two programs output by the adversary which are both γ-good with respect to f, D_f, with non-negligible probability.

If $\mathsf{E}_1 \vee \mathsf{E}_2$ does not happen, we are in the case $\bar{\mathsf{E}}_1 \wedge \bar{\mathsf{E}}_2$. By definition, there exist negligible functions $\mathsf{negl}_1, \mathsf{negl}_2$ such that

$$\mathrm{Tr}[\widetilde{\mathsf{ATI}}_{R_1,\gamma-\epsilon} \cdot \sigma[R_1]] \leq \mathsf{negl}_1(\lambda) \qquad \mathrm{Tr}[\widetilde{\mathsf{ATI}}_{R_2,\gamma-\epsilon} \cdot \sigma[R_2]] \leq \mathsf{negl}_2(\lambda)$$

We look at the following thought experiments:

1. We apply $\mathsf{ATI}_{R_1,\gamma-\epsilon} \otimes \mathsf{ATI}_{R_2,\gamma-\epsilon}$ on σ, by Lemma 18.7, there exists a non-negligible function $\beta(\cdot)$ such that

$$\mathrm{Tr}\left[(\mathsf{ATI}_{R_1,\gamma-\epsilon} \otimes \mathsf{ATI}_{R_2,\gamma-\epsilon}) \cdot \sigma\right] \geq \beta(\lambda) - 2\delta.$$

2. We apply $\mathsf{ATI}_{R_1,\gamma-\epsilon} \otimes \widetilde{\mathsf{ATI}}_{R_2,\gamma-\epsilon}$ on σ. We have,

$$\mathrm{Tr}\left[(\mathsf{ATI}_{R_1,\gamma-\epsilon} \otimes \widetilde{\mathsf{ATI}}_{R_2,\gamma-\epsilon}) \cdot \sigma\right] \leq \mathrm{Tr}\left[(I \otimes \widetilde{\mathsf{ATI}}_{R_2,\gamma-\epsilon}) \cdot \sigma\right] \leq \mathsf{negl}_2(\lambda).$$

3. Note that in 1 and 2, the only difference is the oracle access: in 1, it has oracle access to $\mathcal{O}_1, \mathcal{O}_2$; in 2, it has oracle access to $\mathcal{O}_\perp, \mathcal{O}_2$. Let σ' be the state which we apply $(\mathsf{ATI}_{R_1,\gamma-\epsilon} \otimes I)$ on σ and obtain a outcome 1, which happens with non-negligible probability. Let $W_{2,A}$ be the query weight defined on the state σ'. We know that $W_{2,A}$ can not be negligible otherwise by Theorem 18.2 (BBBV), the probability difference in 1 and 2 can not be non-neglibile.

   Define M_{R_2} be the operator that measures a random query of $\mathsf{ATI}_{R_2,\gamma-\epsilon}$ to $\mathcal{O}_1$ and the query (x, v) satisfies $v \in A \setminus \{0\}$. By the above discussion, there exists a non-negligible function $\beta_1(\cdot)$,

$$\mathrm{Tr}\left[(\mathsf{ATI}_{R_1,\gamma-\epsilon} \otimes M_{R_2}) \cdot \sigma\right] \geq \beta_1(\lambda).$$

4. We apply $\widetilde{\mathsf{ATI}}_{R_1,\gamma-\epsilon} \otimes M_{R_2}$ on σ. We have,

$$\mathrm{Tr}\left[(\widetilde{\mathsf{ATI}}_{R_1,\gamma-\epsilon} \otimes M_{R_2}) \cdot \sigma\right] \leq \mathrm{Tr}\left[(\widetilde{\mathsf{ATI}}_{R_1,\gamma-\epsilon} \otimes I) \cdot \sigma\right] \leq \mathsf{negl}_1(\lambda).$$

5. By a similar argument of 3, let M_{R_1} be the operator that measures a random query of $\mathsf{ATI}_{R_1,\gamma-\epsilon}$ to $\mathcal{O}_2$ and the query (x, v) satisfies $v \in A^\perp \setminus \{0\}$. There exists a non-negligible function $\beta_2(\cdot)$,

$$\mathrm{Tr}\left[(M_{R_1} \otimes M_{R_2}) \cdot \sigma\right] \geq \beta_2(\lambda).$$

Thus, in the case, one can extract a pair of vectors $(u, v) \in (A \setminus \{0\}) \times (A^\perp \setminus \{0\})$ with non-negligible probability. To conclude it, we have the following lemma,

Lemma 18.9. *Fixing a function f, let $(\sigma, U_{R_1}, U_{R_2})$ be the two programs output by the adversary which are both γ-good with respect to f, D_f, with non-negligible probability. If $\mathsf{E}_1 \vee \mathsf{E}_2$ does not happen, by randomly picking and measuring a query of $\mathsf{ATI}_{R_1,\gamma-\epsilon}$ to $\mathcal{O}_2$ and a query of $\mathsf{ATI}_{R_2,\gamma-\epsilon}$ to $\mathcal{O}_1$, one can obtain a pair of vectors $(u, v) \in (A \setminus \{0\}) \times (A^\perp \setminus \{0\})$ with non-negligible probability.*

Then we show a reduction to violate unlearnability in case of E_1 or E_2 and a reduction to violate direct product hardness in case of $\bar{\mathsf{E}}_1 \wedge \bar{\mathsf{E}}_2$. We have the following lemmas:

Lemma 18.10. *Let $\Pr[\mathsf{E}_1]$ be the probability of E_1 taken over all randomness of $\mathsf{CopyProtectionGame}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}(1^\lambda)$. If $\Pr[\mathsf{E}_1]$ is non-negligible, there exists an adversary $\mathcal{A}_1$ that wins $\mathsf{LearningGame}^{\mathcal{A}_1}_{\mathcal{F},\mathcal{D},\gamma-2\epsilon}(1^\lambda)$ with non-negligible probability.*

Proof. The challenger in the copy-protection security game plays as the quantum unlearnability adversary $\mathcal{A}_1$ for function $f \leftarrow \mathcal{F}$, given only black-box access to f; we denote this black box as oracle $\mathcal{O}_f$, which on query $|x, z\rangle$, answers the query with $|x, f(x) + z\rangle$.

Next, we show that $\mathcal{A}_1$ can simulate the copy-protection security game for $\mathcal{A}$ using the information given and uses $\mathcal{A}$ to quantumly learn f. $\mathcal{A}_1$ samples random $\lambda/2$-dimensional subspace A over $\mathbb{F}$ and prepares the membership oracles (two unitaries) $U_A, U_A^\perp$ as well as state $|A\rangle$.

Using $U_A, U_A^\perp$ and given oracle access to f in the unlearnability game, $\mathcal{A}_1$ simulates the copy-protection oracles $\mathcal{O}_1, \mathcal{O}_2$ for $\mathcal{A}$ in the query phase of anti-piracy game.

There is one subtlety in the proof: $\mathcal{A}_1$ needs to simulate the oracles in the anti-piracy game slightly differently: $\mathcal{A}_1$ simulates the oracles with their functionalities partially swapped:

$$\mathcal{O}_1'(x, v) = \begin{cases} g(x) & \text{if } v \in A \text{ and } v \neq 0, \\ \perp & \text{otherwise.} \end{cases}$$
$$\mathcal{O}_2'(x, v) = \begin{cases} f(x) \oplus g(x) & \text{if } v \in A^\perp \text{ and } v \neq 0, \\ \perp & \text{otherwise.} \end{cases}$$

That is, a random function $g(x)$ is output when queried on $u \in A\backslash\{0\}$, and $f(x) \oplus g(x)$ is output when queried on $u \in A^\perp \setminus \{0\}$. The distributions of $\mathcal{O}_1, \mathcal{O}_2$ and $\mathcal{O}_1', \mathcal{O}_2'$ are identical. Note that $g(x)$ can be simulated by a quantum secure PRF or a $2t$-wise independent hash function where t is the number of oracle queries made by $\mathcal{A}$ [Zha12].

In the output phase, $\mathcal{A}$ outputs $(\sigma, U_{R_1}, U_{R_2})$ and sends to $\mathcal{A}_1$. $\mathcal{A}_1$ simply outputs $(\sigma[R_1], U_{R_1})$ with oracle access to $\mathcal{O}_1', \mathcal{O}_\perp$. The program does not need access to oracle f because $\mathcal{O}_1'$ is only about $g(\cdot)$ and $\mathcal{O}_\perp$ is a dummy oracle. If E_1 happens, the program is a $(\gamma - 2\epsilon)$-good with non-negligible probability, by the definition of E_1. Because $\Pr[E_1]$ is also non-negligible, $\mathcal{A}_1$ breaks $(\gamma - 2\epsilon)$-quantum-unlearnability of $\mathcal{F}, \mathcal{D}$. □

Lemma 18.11. *Let* $\Pr[\mathsf{E}_2]$ *be the probability of* E_2 *taken over all randomness of* $\mathsf{CopyProtectionGame}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}(1^\lambda)$. *If* $\Pr[\mathsf{E}_2]$ *is non-negligible, there exists an adversary* $\mathcal{A}_2$ *that wins* $\mathsf{LearningGame}^{\mathcal{A}_2}_{\mathcal{F},\mathcal{D},\gamma-2\epsilon}(1^\lambda)$ *with non-negligible probability.*

Proof Sketch. The proof is almost identical to the proof for Lemma 18.11 except oracles $\mathcal{O}_1, \mathcal{O}_2$ are simulated in the same way as that in the construction. $\mathcal{O}_1(x, v)$ outputs $f(x) \oplus g(x)$ if $v \in A \setminus \{0\}$, and otherwise outputs $\perp$. Similarly, $\mathcal{O}_2(x, v)$ outputs $g(x)$ if $v \in A^\perp \setminus \{0\}$, and otherwise outputs $\perp$ $\square$

As discussed above, if $\Pr[\mathsf{E}_1 \vee \mathsf{E}_2]$ is non-negligible, we can break the quantum unlearnability. Otherwise, $\Pr[\bar{\mathsf{E}}_1 \wedge \bar{\mathsf{E}}_2]$ is overwhelming. We show that in the case, one can use the adversary $\mathcal{A}$ to break the direct-product problem Theorem 18.3.

Lemma 18.12. *Let* $\Pr[\bar{\mathsf{E}}_1 \wedge \bar{\mathsf{E}}_2]$ *be the probability taken over all randomness of the game* $\mathsf{CopyProtectionGame}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}(1^\lambda)$*. If* $\Pr[\bar{\mathsf{E}}_1 \wedge \bar{\mathsf{E}}_2]$ *is non-negligible, there exists an adversary* $\mathcal{A}_3$ *that breaks the direct-product problem.*

Proof. The challenger in the copy-protection security game plays as the adversary in breaking direct-product problem, denoted as $\mathcal{A}_3$. In the reduction, $\mathcal{A}_3$ is given the access to membership oracles $U_A, U_A^\perp$ and one copy of $|A\rangle$.

Next, we show that $\mathcal{A}_3$ can simulate the anti-piracy security game for $\mathcal{A}$ using the information given and uses $\mathcal{A}$ to obtain the two vectors. $\mathcal{A}_3$ samples $f \leftarrow \mathcal{F}$, and simulates a γ-anti-piracy game, specifically simulating the copy-protection oracle $\mathcal{O}_1, \mathcal{O}_2$ for adversary $\mathcal{A}$. In the output phase, $\mathcal{A}$ outputs $(\sigma, U_{R_1}, U_{R_2})$.

$\mathcal{A}_1$ upon taking the output, it randomly picks and measures a query of $\mathsf{ATI}_{R_1,\gamma-\epsilon}$ to $\mathcal{O}_2$ and a query of $\mathsf{ATI}_{R_2,\gamma-\epsilon}$ to $\mathcal{O}_1$, and obtain a pair of vectors (u, v). If $\bar{\mathsf{E}}_1 \wedge \bar{\mathsf{E}}_2$ happens. By Lemma 18.9, (u, v) breaks the direct-product problem with non-negligible probability. Since $\Pr[\bar{\mathsf{E}}_1 \wedge \bar{\mathsf{E}}_2]$ is non-negligible, the overall probability is non-negligible.

$\square$

Note that the proof does not naturally extend to q-collusion resistant anti-piracy. We leave this as an interesting open problem.

$\mathsf{Setup}(1^\lambda)$: it runs $\mathsf{WM.Setup}(1^\lambda)$ to get xk, mk, let $\mathsf{sk} = \mathsf{mk}$ and $\mathsf{pk} = \mathsf{xk}$.

$\mathsf{Generate}(\mathsf{sk}, f)$:

- it runs $\mathsf{QM.Gen}(1^\lambda)$ to get a money state $|\$\rangle$ and a serial number s (by applying $\mathsf{QM.Ver}$ to the banknote);
- let $\widetilde{f} = \mathsf{WM.Mark}(\mathsf{mk}, f, s)$ which is classical;
- it outputs the quantum state $\rho_f = (\widetilde{f}, |\$\rangle)$, and $\{U_{f,x}\}_{x \in [N]}$;
- let $\{U_{f,x}\}_{x \in [N]}$ describe the following unitary: on input a quantum state ρ, treat the first register as a classical function g, compute $g(x)$ in superposition.

$\mathsf{Check}(\mathsf{pk}, (\rho_f, \{U_{f,x}\}_{x \in [N]}))$:

- it parses and measures the first register, which is $(f', |\$'\rangle)$;
- it checks if $\mathsf{QM.Ver}(|\$'\rangle)$ is valid and it gets the serial number s';
- it then checks if $s' = \mathsf{WM.Extract}(\mathsf{pk} = \mathsf{xk}, f')$;
- if all the checks pass, it outputs 1; otherwise, it outputs 0.

Figure 18.2: Quantum copy-detection scheme.

18.6 Quantum Copy-Detection

18.6.1 Construction

We construct a copy-detection scheme for a watermarkable function family $\mathcal{F}$ with respect to an input distribution $\mathcal{D}$. Let QM and WM be a public key quantum money scheme (see Appendix G.1.1) and a publicly extractable watermarking scheme for $\mathcal{F}, \mathcal{D}$, whose serial number space $\mathcal{S}_\lambda$ of QM is a subset of the message space $\mathcal{M}_\lambda$ of WM. We construct a copy-detection scheme in Fig. 18.2. The general scheme and full proofs are in Appendix G.5.

18.6.2 Efficiency and correctness

First, for all $\lambda \in \mathbb{N}$, all efficient $\mathcal{A}$, every $f \in \mathcal{F}_\lambda$, the copy-detection program is $(\rho_f, \{U_{f,x}\}_{x\in[N]})$. We have $\mathsf{Compute}(\rho_f, \{U_{f,x}\}_{x\in[N]}, x) = \tilde{f}(x)$, where $\tilde{f} = \mathsf{WM.Mark}(\mathsf{mk}, f, s)$ for some serial number s. From the correctness of WM, it satisfies the correctness of copy-detection.

The correctness of Check comes from the correctness of $\mathsf{WM.Extract}$ and **unique serial number** property of QM. Check is a projection since $\mathsf{QM.Ver}$ is also a projection. Efficiency is straightforward.

18.6.3 Security

Theorem 18.13. *Assume* QM *is a quantum money scheme and* WM *for* $\mathcal{F}, \mathcal{D}$ *with* γ*-unremovability, the above copy-detection scheme for* $\mathcal{F}, \mathcal{D}$ *has* γ*-copy-detection-security.*

Proof. Let $\mathcal{A}$ be a QPT algorithm that tries to break the security of the copy-detection scheme. Let $(\sigma, U_{R_1}, U_{R_2})$ be the programs output by $\mathcal{A}$ which wins the game $\mathsf{CopyDetectionGame}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}$. To win the game, the program $(\sigma, U_{R_1}, U_{R_2})$ should pass the following two tests:

1. Apply the projective measurement (defined by $\mathsf{Check}(\mathsf{pk}, \cdot)$) on both $\sigma[R_1]$ and $\sigma[R_2]$, and both outcomes are 1.
2. Let σ' be the state that passes step 1. Then both programs $(\sigma'[R_1], U_{R_1})$, $(\sigma'[R_2], U_{R_2})$ are tested to be γ-good with non-negligible probability.

In our construction, Check first measures the program registers. The resulting state is $\tilde{f}_1, \tilde{f}_2, \sigma$, where $\tilde{f}_1, \tilde{f}_2$ are supposed to be classical (marked) circuits that computes f and σ are (possibly entangled) states that are supposed to be quantum money for each of the program.

Next, Check applies $\mathsf{QM.Ver}$ on both registers of σ and computes serial numbers. Define S_b be the random variable of $\mathsf{QM.Ver}$ applying on $\sigma[R_b]$ representing the serial number of ρ_b. Define S be the random variable of $\mathsf{QM.Ver}(|\$\rangle)$ representing the serial number of the quantum money state in the $\mathsf{Generate}$ procedure.

Define E be the event that both $\mathsf{WM.Extract}(\mathsf{xk}, \tilde{f}_b) = S_b$ and at least one of S_1, S_2 is not equal to S. Define E' be the event that both S_1, S_2 are equal to S and both $\mathsf{WM.Extract}(\mathsf{xk}, \tilde{f}_b) = S_b$. If $\tilde{f}_1, \tilde{f}_2, \sigma$ passes the step 1, exactly one of E and E' happens.

In step 2, it simply tests if $\tilde{f}_1$ and $\tilde{f}_2$ are γ-good with respect to f, D_f. Since $\tilde{f}_1, \tilde{f}_2$ are classical circuits, it is equivalent to check whether they work correctly on at least γ fraction of all inputs. If it passes step 2, we have for all $b \in \{1, 2\}$, $\Pr_{x \leftarrow D_\lambda}[\tilde{f}_b(x) = f(x)] \geq \gamma$.

Therefore, the probability of $\mathcal{A}$ breaks the security game is indeed,

$$\begin{aligned}
&\Pr_{(\tilde{f}_1,\tilde{f}_2,\sigma)}\left[\forall b, \Pr_{x\leftarrow D_\lambda}[\tilde{f}_b(x) = f(x)] \geq \gamma\right] \\
&= \Pr_{(\tilde{f}_1,\tilde{f}_2,\sigma)}\left[(E \vee E') \wedge \forall b, \Pr_{x\leftarrow D_\lambda}[\tilde{f}_b(x) = f(x)] \geq \gamma\right] \\
&\leq \Pr_{(\tilde{f}_1,\tilde{f}_2,\sigma)}\left[E \wedge \forall b, \Pr_{x\leftarrow D_\lambda}[\tilde{f}_b(x) = f(x)] \geq \gamma\right] + \Pr_{(\tilde{f}_1,\tilde{f}_2,\sigma)}[E']
\end{aligned}$$

Note that the probability is taken over the randomness of $\mathsf{CopyDetectionGame}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}$. Next we are going to show both probabilities are negligible, otherwise we can break the quantum money scheme or watermarking scheme.

Claim 18.14. $\Pr_{(\tilde{f}_1,\tilde{f}_2,\sigma)}[E'] \leq \mathsf{negl}(\lambda)$.

Proof. It corresponds to the security game of the quantum money scheme. Assume $\Pr[E']$ is non-negligible, we can construct an adversary $\mathcal{B}$ for the quantum money scheme with non-negligible advantage. Given a quantum money state $|\$\rangle$, the algorithm $\mathcal{B}$ simulates the challenger for the copy-detection and can successfully 'copy' a money state. $\square$

Claim 18.15. $\Pr_{(\tilde{f}_1,\tilde{f}_2,\sigma)}\left[E \wedge \forall b, \Pr_{x\leftarrow D_\lambda}[\tilde{f}_b(x) = f(x)] \geq \gamma\right] \leq \textsf{\textit{negl}}(\lambda).$

Proof. It corresponds to the security game of the underlying watermarking scheme. Since if E happens, at least one of the circuit has different mark than s and it satisfies the correctness requirement. □

Thus, the probability of $\mathcal{A}$ breaks the game is negligible. □

Part IV

Concentration and Discrepancy

Chapter 19: Hyperbolic Polynomials I: Concentration and Anti-Concentration

19.1 Introduction

The study of concentration of sums of independent random variables dates back to Central Limit Theorems, and hence to de Moivre and Laplace, while modern concentration bounds for sums of random variables were probably first established by Bernstein [Ber24] in 1924. An extremely popular variant now known as Chernoff bounds was introduced by Rubin and published by Chernoff [Che52] in 1952.

Hyperbolic polynomials are real, multivariate homogeneous polynomials $p(x) \in \mathbb{R}[x_1, \ldots, x_n]$, and we say that $p(x)$ is hyperbolic in direction $e \in \mathbb{R}^n$ if the univariate polynomial $p(te - x) = 0$ for any x has only real roots as a function of t (counting multiplicities). The study of hyperbolic polynomials was first proposed by Gårding in [Går51] and has been extensively studied in the mathematics community [Går59, Gül97, BGLS01, Ren06]. Some examples of hyperbolic polynomials are as follows:

- Let $h(x) = x_1 x_2 \cdots x_n$. It is easy to see that $h(x)$ is hyperbolic with respect to any vector $e \in \mathbb{R}^n_+$.
- Let $X = (x_{i,j})_{i,j=1}^n$ be a symmetric matrix where $x_{i,j} = x_{j,i}$ for all $1 \leq i, j \leq n$. The determinant polynomial $h(x) = \det(X)$ is hyperbolic with respect to $\widetilde{I}$, the identity matrix I packed into a vector. Indeed, $h(t\widetilde{I} - x) = \det(tI - X)$, the characteristic polynomial of the symmetric matrix X, has only real roots by the spectral theorem.
- Let $h(x) = x_1^2 - x_2^2 - \cdots - x_n^2$. Then, $h(x)$ is hyperbolic with respect to $e = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^\top$.

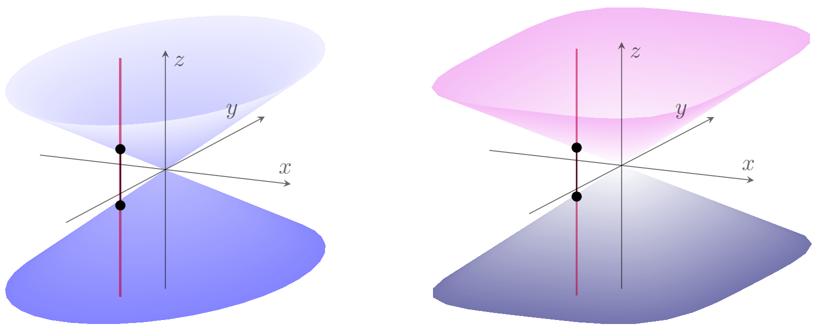


Figure 19.1: The function on the left is $h(x, y, z) = z^2 - x^2 - y^2$, which is hyperbolic with respect to $e = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top$, since any line in this direction always has two intersections, corresponding to the two real roots of $h(-x, -y, t - z) = 0$. The function on the right is $g(x, y, z) = z^4 - x^4 - y^4$, which is *not* hyperbolic with respect to e, since it only has 2 intersections but the degree is 4.

Inspired by the eigenvalues of matrix, we can define the hyperbolic eigenvalues of a vector x as the real roots of $t \mapsto h(te - x)$, that is, $\lambda_{h,e}(x) = (\lambda_1(x), \ldots, \lambda_d(x))$ such that $h(te - x) = h(e) \prod_{i=1}^d (t - \lambda_i(x))$ (see Fact 19.5). In other words, the hyperbolic eigenvalues of x are the zero points of the hyperbolic polynomial restricted to a real line through x. In this chapter, we assume that h and e are fixed and we just write $\lambda(x)$ omitting the subscript. Furthermore, similar to the spectral norm of matrix, the *hyperbolic spectral norm* of a vector x can be defined as

$$\|x\|_h = \max_{i \in [d]} |\lambda_i(x)|. \tag{19.1}$$

In this chapter, we study the concentration phenomenon of the roots of hyperbolic polynomials. More specifically, we consider the hyperbolic spectral norm of the sum of randomly signed vectors, i.e., $\| \sum_{i=1}^n r_i x_i \|_h$, where $r \in \{-1, 1\}^n$ are uniformly random signs and $\{x_1, x_2, \cdots, x_n\}$ are any fixed vectors in $\mathbb{R}^m$. This kind of summation has been studied in the following cases:

1. **Scalar case:** $x_i \in \{-1, 1\}$ and the norm is just the absolute value, i.e., $| \sum_{i=1}^n r_i x_i |$, the scalar version Chernoff bound [Che52] shows that

$$\Pr_{r \sim \{-1,1\}^n} \left[\left| \sum_{i=1}^n r_i x_i \right| > t \right] \leq 2 \exp\left(-t^2/(2n)\right),$$

corresponding to the case when $h(x) = x$ for $x \in \mathbb{R}$ and the hyperbolic direction $e = 1$.

2. **Matrix case:** x_i are d-by-d symmetric matrices and the norm is the spectral norm, i.e., $\| \sum_{i=1}^n r_i x_i \|$, the matrix Chernoff bound [Tro15] shows that

$$\Pr_{r \sim \{-1,1\}^n} \left[\left\| \sum_{i=1}^n r_i x_i \right\| > t \right] \leq 2d \cdot \exp\left(- \frac{t^2}{2 \left\| \sum_{i=1}^n x_i^2 \right\|} \right),$$

corresponding to $h(x) = \det(X)$ and $e = I$.

We try to generalize these results to the hyperbolic spectral norm for any hyperbolic polynomial h, which is recognized as an interesting problem in this field by James Renegar [Ren19b].

19.1.1 Our results

In this chapter, we can prove the following "Chernoff-type" concentration for hyperbolic spectral norm. We show that, when adding uniformly random signs to n vectors, the hyperbolic spectral norm of their summation will concentrate with an exponential tail.

Theorem 19.1 (Nearly optimal hyperbolic Chernoff bound for Rademacher sum)**.** *Let h be an m-variate, degree-d hyperbolic polynomial with respect to a direction $e \in \mathbb{R}^m$. Let $1 \leq s \leq d$, $\sigma > 0$. Given $x_1, x_2, \cdots, x_n \in \mathbb{R}^m$ such that rank$(x_i) \leq s$ for all $i \in [n]$ and $\sum_{i=1}^n \|x_i\|_h^2 \leq \sigma^2$, where rank$(x)$ is the number of nonzero hyperbolic eigenvalues of x. Then, we have*

$$\mathbb{E}_{r \sim \{\pm 1\}^n} \left[\left\| \sum_{i=1}^n r_i x_i \right\|_h \right] \leq 2\sqrt{\log(s)} \cdot \sigma.$$

Furthermore, for every $t > 0$, and for some fixed constant $c > 0$,

$$\Pr_{r \sim \{\pm 1\}^n} \left[\left\| \sum_{i=1}^n r_i x_i \right\|_h > t \right] \leq 2 \exp\left(- \frac{ct^2}{\sigma^2 \log(s+1)} \right).$$

We discuss the optimality of Theorem 19.1 in different cases:

- **Degree-1 case:** When the hyperbolic polynomial's degree $d = s = 1$, the hyperbolic polynomial is $h(z) = z$. Then, we have $\|x\|_h = |x|$ and we get the the Hoeffding's inequality [Hoe63]:

$$\Pr_{r\sim\{\pm1\}^n}\Big[\Big|\sum_{i=1}^n r_i x_i\Big| > t\Big] \leq \exp\Big(-\Omega\Big(t^2/(\sum_{i=1}^n x_i^2)\Big)\Big).$$

  It implies that our result is optimal in this case.

- **A special degree-2 case:** $h(z) = z_1^2 - z_2^2 - \cdots - z_m^2$. Let $v_1, \ldots, v_n$ be any $(d-1)$-dimensional vectors. Then, we define $x_i := \begin{bmatrix} 0 & v_i \end{bmatrix} \in \mathbb{R}^d$ for $i \in [n]$. We know that $\|x_i\|_h = \|v_i\|_2$, and Theorem 19.1 gives the following result:

$$\Pr_{r\sim\{\pm1\}^n}\Big[\Big\|\sum_{i=1}^n r_i v_i\Big\|_2 > t\Big] \leq \exp(-\Omega(t^2/\sigma^2)),$$

  where $\sigma^2 := \sum_{i=1}^n \|v_i\|^2$, which recovers the dimension-free vector-valued Bernstein inequality [Min17].

- **Constant degree case:** When $d > 1$ is a constant, consider h being the determinant polynomial of d-by-d matrix. Since $s \leq d = O(1)$, we can show that $\sigma = (\sum_{i=1}^n \|x_i\|^2)^{1/2} = \Theta(\|\sum_{i=1}^n x_i^2\|^{1/2})$, and Theorem 19.1 exactly recovers the matrix Chernoff bound [Tro15], which implies that our result is also optimal in this case.

- **Constant rank case:** When all the vectors have constant hyperbolic rank, we still take $h = \det(X)$, but $X_1, \ldots, X_n$ are constant rank matrices with arbitrary dimension. In this case, we can obtain a dimension-free matrix concentration inequality:

$$\Pr_{r\sim\{\pm1\}^n}\left[\left\|\sum_{i=1}^n r_i X_i\right\| > t\right] \leq 2\exp\left(-\Omega(t^2/\sigma^2)\right).$$

  It will beat the general matrix Chernoff bound [Tro15] when σ is not essentially larger than $\|\sum_{i=1}^n X_i^2\|^{1/2}$. Thus, Theorem 19.1 is nearly optimal in this case.

However, Theorem 19.1 is also sub-optimal in this case if we consider the high degree polynomial $h(z) = \prod_{i=1}^{n} z_i$, and $x_i = e_i \in \mathbb{R}^n$. Then, we have $\|x_i\|_h = 1$, and $\|\sum_{i=1}^{n} r_i x_i\|_h = 1$ for any $r \in \{\pm 1\}^n$. Therefore, the probability density function of the hyperbolic spectral norm of the Rademacher sum is a delta function[1] in this case. But our concentration result cannot characterize such a sharp transition.

Theorem 19.1 works for arbitrary vectors in $\mathbb{R}^m$. We also consider the maximum and minimum hyperbolic eigenvalues of the sum of random vectors in the hyperbolic cone, which is a generalization of the positive semi-definite (PSD) cone for matrices. Recall that for independent random PSD matrices $\mathbf{X}_1, \ldots, \mathbf{X}_n$ with spectral norm at most R, let $\mu_{\max} := \lambda_{\max}(\sum_i \mathbb{E}[\mathbf{X}_i])$. Then, matrix Chernoff bound for PSD matrices [Tro15] shows that $\Pr[\lambda_{\max}(\sum_i \mathbf{X}_i) \geq (1+\delta)\mu_{\max}] \leq \; de^{-\Omega(\delta\mu_{\max})}$ for any $\delta \geq 0$. The following theorem gives a hyperbolic version of this result:

Theorem 19.2 (Hyperbolic Chernoff bound for random vectors in hyperbolic cone)**.** *Let h be an m-variate, degree-d hyperbolic polynomial with hyperbolic direction $e \in \mathbb{R}^m$. Let Λ_+ denote the hyperbolic cone[2] of h with respect to e. Suppose $\mathsf{x}_1, \ldots, \mathsf{x}_n$ are n independent random vectors with supports in Λ_+ such that $\lambda_{\max}(\mathsf{x}_i) \leq R$ for all $i \in [n]$. Define the mean of minimum and maximum eigenvalues as $\mu_{\min} := \sum_{i=1}^{n} \mathbb{E}[\lambda_{\min}(\mathsf{x}_i)]$ and $\mu_{\max} := \sum_{i=1}^{n} \mathbb{E}[\lambda_{\max}(\mathsf{x}_i)]$.*

Then, we have

$$\Pr\left[\lambda_{\max}\left(\sum_{i=1}^{n} \mathsf{x}_i\right) \geq (1+\delta)\mu_{\max}\right] \leq \; d \cdot \left(\frac{(1+\delta)^{1+\delta}}{e^{\delta}}\right)^{-\mu_{\max}/R} \quad \forall \delta \geq 0,$$
$$\Pr\left[\lambda_{\min}\left(\sum_{i=1}^{n} \mathsf{x}_i\right) \leq (1-\delta)\mu_{\min}\right] \leq \; d \cdot \left(\frac{(1-\delta)^{1-\delta}}{e^{-\delta}}\right)^{-\mu_{\min}/R} \quad \forall \delta \in [0,1].$$

[1]The delta function is defined as $\delta(x) = \begin{cases} 1 & \text{if } x = 1, \\ 0 & \text{otherwise.} \end{cases}$

[2]The hyperbolic cone is a set containing all vectors with non-negative hyperbolic eigenvalues. See Definition 19.4 for the formal definition.

19.1.2 Hyperbolic anti-concentration

Anti-concentration is an interesting phenomenon in probability theory, which studies the opposite perspective of concentration inequalities. A simple example is the standard Gaussian random variable, which has probability at most $O(\Delta)$ for being in any interval of length Δ. For Rademacher random variables $x \sim \{\pm 1\}^d$, the celebrated Littlewood-Offord theorem [LO43] states that for any degree-1 polynomial $p(x) = \sum_{i=1}^d a_i x_i$ with $|a_i| \geq 1$, the probability of $p(x)$ in any length-1 interval is at most $O(\frac{\log d}{\sqrt{d}})$. Later, the theorem was improved to $O(\frac{1}{\sqrt{d}})$ by Erdös [Erd45], and generalized to higher degree polynomials by [CTV06, RV13, MNV17]. From a geometric prospective, the Littlewood-Offord theorem says that the maximum fraction of hypercube points that lay in the boundary of a halfspace $\mathbf{1}_{\langle a,x\rangle \leq \theta}$ with $|a_i| \geq 1$ for $i \in [d]$ is at most $O(\frac{1}{\sqrt{d}})$. [OST19] extended this result from half-space to polytope and [AY22] further extended to positive spectrahedron.

Following this line of research, we prove the following hyperbolic anti-concentration theorem, which shows that the hyperbolic spectral norm of Rademacher sum of vectors in the hyperbolic cone cannot concentrate within a small interval.

Theorem 19.3 (Hyperbolic anti-concentration theorem, informal)**.** *Let h be an m-variate degree-d hyperbolic polynomial with hyperbolic direction $e \in \mathbb{R}^m$. Let $\{x_i\}_{i\in[n]} \subset \Lambda_+$ be a sequence of vectors in the hyperbolic cone such that $\lambda_{\max}(x_i) \leq \tau$ for all $i \in [n]$ and $\sum_{i=1}^n \lambda_{\min}(x_i)^2 \geq 1$.*

Then, for any $y \in \mathbb{R}^m$ and any $\Delta \geq 20\tau \log d$, we have

$$\Pr_{\epsilon\sim\{-1,1\}^n}\left[\lambda_{\max}\left(\sum_{i=1}^n \epsilon_i x_i - y\right) \in [-\Delta, \Delta]\right] \leq O(\Delta).$$

From the geometric viewpoint, we can define a "positive hyperbolic-spectrahedron" as the space $\{\alpha \in \mathbb{R}^n : \lambda_{\max}(\alpha_1 x_1 + \cdots + \alpha_n x_n - y) \leq 0\}$, where $x_1, \ldots, x_n$ are in the hyperbolic cone. Then, Theorem 19.3 states that the hyperbolic spectral norm of a positive hyperbolic-spectrahedron cannot be concentrated in a small region.

19.1.3 Related work

Chernoff-type bounds There is a long line of work generalizing the classical scalar Chernoff-type bounds to the matrix Chernoff-type bound [Rud99, AW02, RV07, Tro12, MJC$^+$14, GLSS18, KS18, NRR20, ABY20, JLLV20]. [Rud99, RV07] showed a Chernoff-type concentration of spectral norm of matrices which are the outer product of two random vectors. [AW02] first used Laplace transform and Golden-Thompson inequality [Gol65, Tho65] to prove a Chernoff bound for general random matrices. It was improved by [Tro12] and [Oli09] independently. [MJC$^+$14] proved a series of matrix concentration results via Stein's method of exchangeable pairs. Our work further extends this line of research from matrix to hyperbolic polynomials and can fully recover the result of [AW02]. On the other hand, [GLSS18] showed an expander matrix Chernoff bound. [KS18] prove a new matrix Chernoff bound for Strongly Rayleigh distributions.

Hyperbolic polynomials The concept of hyperbolic polynomials was originally studied in the field of partial differential equations [Går51, Hor83, Kry95]. Güler [Gül97] first studied the hyperbolic optimization (hyperbolic programming), which is a generalization of LP and SDP. Later, a few algorithms [Ren06, MT14, RS14, Ren16, NP18, Ren19a] were designed for hyperbolic programming. On the other hand, a lot of recent research focused on the equivalence between hyperbolic programming and SDP, which is closely related to the "Generalized Lax Conjecture" and its variants [HV07, LPR05, Brä14, KPV15, Sau18, Ami19, RRSW19]. In addition to the hyperbolic programming, hyperbolic polynomial is a key component in resolving Kadison-Singer problem [MSS15b, Brä18] and constructing bipartite Ramanujan graphs [MSS18]. Gurvits [Gur06, Gur07] proved some Van der Waerden/Schrijver-Valiant like conjectures for hyperbolic polynomials, giving sharp bounds for the capacity of polynomials. [Son19] gave an approach to certify the non-negativity of polynomials via hyperbolic programming, generalizing the Sum-of-Squares method.

19.1.4 Technique overview

In this section, we provide a proof overview of our results. We first show how prove hyperbolic Chernoff bounds by upper bounding each polynomial moment. After that, we show how to apply our new concentration inequality to prove hyperbolic anti-concentration. Finally, we show how to relax the isotropic condition in [Brä18], and also how to get a more general discrepancy result via hyperbolic concentration.

19.1.4.1 Our technique for hyperbolic Chernoff bound for Rademacher sum

The main idea of our proof of hyperbolic Chernoff bound is to upper bound the polynomial moments.

By definition, the hyperbolic spectral norm of X is the ℓ_∞ norm of the eigenvalues $\lambda(X)$. Inspired by the proof of the matrix Chernoff bound by Tropp [Tro18], we can consider the ℓ_{2q} norm of $\lambda(X)$, for $q \geq 1$. When the hyperbolic polynomial h is the determinant polynomial, this norm is just the Schatten-$2q$ norm of matrices. For general hyperbolic polynomials, we define hyperbolic-$2q$ norm as $\|x\|_{h,2q} := \|\lambda(x)\|_{2q}$. By the result of [BGLS01], hyperbolic-$2q$ norm is actually a norm in $\mathbb{R}^m$. And the following inequality shows the connection between a hyperbolic spectral norm and hyperbolic-$2q$ norm:

$$\mathbb{E}_{r\sim\{\pm 1\}^n}[\|X\|_h] \leq \left(\mathbb{E}_{r\sim\{\pm 1\}^n}\left[\|X\|_{h,2q}^{2q}\right]\right)^{1/(2q)}.$$

In order to compute $\|X\|_{h,2q}^{2q} = \sum_{i=1}^{rank(X)} \lambda_i(X)^{2q}$, we use a deep result about hyperbolic polynomials: the Helton-Vinnikov Theorem [HV07], which proved a famous conjecture by Lax [Lax57], to translate between hyperbolic polynomials and matrices. The theorem is stated as follows.

Theorem 19.4 ([HV07]). *Let $f \in \mathbb{R}[x, y, z]$ be hyperbolic with respect to $e = (e_1, e_2, e_3) \in \mathbb{R}^3$. Then there exist symmetric real matrices $A, B, C \in \mathbb{R}^{d\times d}$ such that $f = \det(xA + yB + zC)$ and $e_1A + e_2B + e_3C \succ 0$.*

Gurvits [Gur04] proved a corollary (Corollary 19.17) that for any m-variate hyperbolic polynomial h, and $x, y \in \mathbb{R}^m$, there exist two symmetric matrices $A, B \in \mathbb{R}^{d\times d}$ such that for any $a, b \in \mathbb{R}$, $\lambda(ax+by) = \lambda(aA+bB)$, where the left-hand side means the hyperbolic eigenvalues of the vector $ax+by$ and the right-hand side means the eigenvalues of the matrix $aA+bB$.

Therefore, we try to separate and consider one random variable r_i at a time. We first consider the expectation over r_1. By conditional expectation, let $X_2 := \sum_{i=2}^n r_i x_i$ and we have

$$\mathbb{E}_{r\sim\{\pm1\}^n}\left[\|X\|_{h,2q}^{2q}\right] = \mathbb{E}_{r_2,\ldots,r_n\sim\{\pm1\}}\left[\mathbb{E}_{r_1\sim\{\pm1\}}\left[\|r_1x_1+X_2\|_{h,2q}^{2q}\right]\right],$$

By Corollary 19.17, there exist two matrices A_1, B_1 such that $\lambda(r_1x_1+X_2) = \lambda(r_1A_1+B_1)$ holds for any r_1. And it follows that

$$\mathbb{E}_{r_1\sim\{\pm1\}}\left[\|r_1x_1+X_2\|_{h,2q}^{2q}\right] = \mathbb{E}_{r_1\sim\{\pm1\}}\left[\|r_1A_1+B_1\|_{2q}^{2q}\right].$$

It becomes much easier to compute the expected Schatten-$2q$ norm of matrices. We can prove that

$$\mathbb{E}_{r\sim\{\pm1\}^n}\left[\|X\|_{h,2q}^{2q}\right] \le \sum_{k_1=0}^{q}\binom{2q}{2k_1}\|x_1\|_h^{2k_1}\cdot\mathbb{E}_{r_2,\ldots,r_n}\left[\|X_2\|_{h,2q-2k_1}^{2q-2k_1}\right].$$

Now, we can iterate this process for the remaining expectation $\mathbb{E}_{r_2,\ldots,r_n}\left[\|X_2\|_{h,2q-2k_1}^{2q-2k_1}\right]$. After $n-1$ iterations, we get that

$$\left(\mathbb{E}_{r\sim\{\pm1\}^n}\left[\|X\|_{h,2q}^{2q}\right]\right)^{1/(2q)} \le \sqrt{2q-1}\cdot s^{1/(2q)}\cdot\sigma, \tag{19.2}$$

where $\sigma^2 = \sum_{i=1}^n \|x_i\|_h^2$ and s is the maximum rank of $x_1, \ldots, x_n$. Then, by taking $q := \log(s)$ and $\|X\|_h \le \|X\|_{h,2q}^{2q}$, we get the desired upper bound for the expectation $\mathbb{E}_{r\sim\{\pm1\}^n}[\|\sum_{i=1}^n r_ix_i\|_h]$ in Theorem 19.1.

To obtain the concentration probability inequality, We can apply the result of Ledoux and Talagrand [LT13] for the concentration of Rademacher sums in a normed linear space, which will imply:

$$\Pr_{r\sim\{\pm1\}^n}[\|X\|_h > t] \leq 2\exp\Big(-t^2\Big/\Big(32\,\mathbb{E}_{r\sim\{\pm1\}^n}[\|X\|_h^2]\Big)\Big). \tag{19.3}$$

However, we need to verify that the hyperbolic spectral norm $\|\cdot\|_h$ is indeed a norm, which follows from the result of Gårding [Går59]. Since by Khinchin-Kahane inequality ([SZ22, Theorem A.16]) the second moment of $\|X\|_h$ can be upper-bounded via the first moment. Hence, we can put our expectation upper bound into Eq. (19.3) and have

$$\Pr_{r\sim\{\pm1\}^n}[\|X\|_h > t] \leq C_1\exp\left(-\frac{C_2 t^2}{\sigma^2\log(s+1)}\right),$$

for constants $C_1, C_2 > 0$, and hence Theorem 19.1 is proved. We defer the formal proof in the full version [SZ22, Section B].

19.1.4.2 Our technique for hyperbolic Chernoff bound for positive vectors

We can use similar techniques in the previous section to prove Theorem 19.2.

For any random vectors $\mathsf{x}_1,\ldots,\mathsf{x}_n \in \Lambda_+$, we may assume $\|\mathsf{x}_i\|_h \leq 1$. Using the Taylor expansion of the mgf, we can show that:

$$\Pr\left[\lambda_{\max}\left(\sum_{i=1}^n \mathsf{x}_i\right) \geq t\right] \leq \inf_{\theta>0}\ e^{-\theta t}\cdot\sum_{q\geq0}\frac{\theta^q}{q!}\,\mathbb{E}\left[\left\|\sum_{i=1}^n \mathsf{x}_i\right\|_{h,q}^q\right]. \tag{19.4}$$

Then, for the q-th moment, we separate x_1 and $\sum_{i=2}^n \mathsf{x}_i$ and have

$$\mathbb{E}_{\geq1}\left[\left\|\sum_{i=1}^n x_i\right\|_{h,q}^q\right] = \mathbb{E}_{\geq2}\mathbb{E}_1\left[tr\left[(A_1+B_1)^q\right]\right],$$

where A_1 and B_1 are two PSD matrices obtained via Gurvits's result (Corollary 19.17) such that A_1 depends on x_1 and B_1 depends on $\mathsf{x}_2,\ldots,\mathsf{x}_n$. The next step is different from the case of Rademacher sum, since we cannot drop half of the terms by the distribution of x_1. Instead, we can fully expand the matrix products in the trace and use Horn's inequality to upper bound the eigenvalue products. We have

$$\mathbb{E}_{\geq 2}\mathbb{E}_1\left[tr\left[(A(x_1)+B)^q\right]\right] \leq \mathbb{E}_1\left[\sum_{k_1=0}^{q}\binom{q}{k_1}\lambda_{\max}(\mathsf{x}_1)^{k_1}\cdot\mathbb{E}_{\geq 2}\left[\left\|\sum_{i=2}^{n}\mathsf{x}_i\right\|_{h,q-k_1}^{q-k_1}\right]\right].$$

By repeating this process, we finally have

$$\mathbb{E}\left[\left\|\sum_{i=1}^{n}\mathsf{x}_i\right\|_{h,q}^{q}\right] \leq \mathbb{E}\left[\sum_{\substack{k_1,\ldots,k_n\geq 0\\ k_1+\cdots+k_n=q}}\binom{q}{k_1,\ldots,k_n}\prod_{i=1}^{n}\lambda_{\max}(\mathsf{x}_i)^{k_i}\cdot d\right] \leq d\cdot\mathbb{E}\left[\left(\sum_{i=1}^{n}\|\mathsf{x}_i\|_h\right)^q\right],$$

where the first step follows from the $\mathbb{E}[\|\mathsf{x}_n\|_{h,k_n}^{k_n}] \leq d\cdot\lambda_{\max}(\mathsf{x}_n)^{k_n}$. Then, we put the above upper bound into Eq. (19.4), which gives:

$$\Pr\left[\lambda_{\max}\left(\sum_{i=1}^{n}\mathsf{x}_i\right)\geq t\right] \leq \inf_{\theta>0}\ e^{-\theta t}\cdot d\cdot\prod_{i=1}^{n}\mathbb{E}\left[e^{\theta\|\mathsf{x}_i\|_h}\right].$$

Now, we use some similar calculations in the matrix case [Tro12] to prove that

$$\Pr\left[\lambda_{\max}\left(\sum_{i=1}^{n}\mathsf{x}_i\right)\geq t\right] \leq \inf_{\theta>0}\ d\cdot\exp\left(-\theta t+(e^{\theta}-1)\mu_{\max}\right).$$

By taking $\theta := \log(t/\mu_{\max})$ and $t := (1+\delta)\mu_{\max}$, we get that

$$\Pr\left[\lambda_{\max}\left(\sum_{i=1}^{n}\mathsf{x}_i\right)\geq(1+\delta)\mu_{\max}\right] \leq d\cdot\left(\frac{(1+\delta)^{1+\delta}}{e^{\delta}}\right)^{-\mu_{\max}} \tag{19.5}$$

For the minimum eigenvalue case, we can define $\mathsf{x}_i' := e - \mathsf{x}_i$ for $i\in[n]$. Then, by the property of hyperbolic eigenvalues (Fact 19.9) and the assumption that $\|\mathsf{x}_i\|_h \leq 1$, we know that x_i' are also in the hyperbolic cone and $\lambda_{\max}(\mathsf{x}_i') = 1-\lambda_{\min}(\mathsf{x}_i')$. Therefore, we can obtain the Chernoff bound for the minimum eigenvalue of x by applying Eq. (19.5) with x_i'. We defer the formal proof in the full version [SZ22, Section C].

19.1.4.3 Our technique for hyperbolic anti-concentration

In this part, we will show how to prove the hyperbolic anti-concentration result (Theorem 19.3) via the hyperbolic Chernoff bound for vectors in the hyperbolic cone (Theorem 19.2).

In [OST19], they studied the unate functions on hypercube $\{-1,1\}^n$, which is defined as the function being increasing or decreasing with respect to any one of the coordinates. Then, they showed that the Rademacher measure of a unate function is determined by the expansion of its indicator set in hypercube. In particular, for the maximum hyperbolic eigenvalue, it is easy to see that the indicator function $\left[\lambda_{\max}\left(\sum_{i=1}^n \epsilon_i x_i^j - y_j\right) \in [-\Delta, \Delta]\right]$ is unate when $x_i \in \Lambda_+$. Hence, we can show the anti-concentration inequality by studying the expansion in the hypercube, which by [AY22], is equivalent to lower-bound the minimum eigenvalue of each vector. However, for the initial input x_i, we only assume that $\sum_{i=1}^n \lambda_{\min}(x_i)^2 \geq 1$, but we need a $\Omega(\frac{1}{\sqrt{\log d}})$ lower bound for each x_i to prove the theorem. To amplify the minimum eigenvalue, we follow the proof in [AY22] that uses a random hash function to randomly assign the input vectors into some buckets and considers the sum of the vectors in each bucket as the new input. They proved that the "bucketing" will not change the distribution. Then, we can use Theorem 19.2 to lower bound the minimum hyperbolic eigenvalue of each bucket, which is a sum of independent random vectors in the hyperbolic cone. Hence, we get that

$$\Pr\left[\lambda_{\min}\left(\sum_{i=1}^n z_{i,j} x_i\right) \leq \Omega(\frac{1}{\sqrt{\log d}})\right] \leq \frac{1}{10},$$

which $z_{i,j} \in \{0,1\}$ is a random variable indicating that x_i is hashed to the j-th bucket. Then, by the standard Chernoff bound for negatively associated random variables, we can prove that most of the buckets have large minimum eigenvalues, which concludes the proof of the hyperbolic anti-concentration theorem. We defer the formal proof in Section 19.5.

19.1.5 Discussion and open problems

In this chapter, we initiate the study of concentration with respect to the hyperbolic spectral norm, and we generalize several classical concentration and anti-concentration results to the hyperbolic polynomial setting. Our results are closely related to the discrepancy theory and pseudorandomness. We provide some open problems in below.

Tighter hyperbolic Chernoff bound? Our current result has a worse dependence on the variance σ^2 than the matrix Chernoff bound [Tro15]. Can we match the results when $h = \det(X)$? We note that there is a limitation for using the techniques like Golden-Thompson inequality and Lieb's theorem, which were used in [Oli09, Tro12] to improve the original matrix Chernoff bound [AW02], to tighten our result. Because for any symmetric matrix X, we can define a mapping such that $\phi(X)$'s eigenvalues are the p-th power of X's eigenvalues for any $p > 0$, where the mapping is just X^p. However, we cannot find such a mapping for vectors with respect to the hyperbolic eigenvalues. Some new techniques may be required to get a hyperbolic Chernoff bound matching the matrix results.

Resolving the hyperbolic Spencer conjecture? Inspired by the matrix Spencer conjecture (due to Meka [Mek14]), we came up with a more general conjecture for hyperbolic discrepancy. Can we prove or disprove this conjecture? It is also interesting to study the connection between hyperbolic Spencer conjecture and the generalized Lax conjecture [HV07, LPR05, Brä14, KPV15, Sau18, Ami19, RRSW19]. If we assume the matrix Spencer conjecture and the generalized Lax conjecture, can we prove the hyperbolic Spencer conjecture? On the other hand, in a very recent work by Reis and Rothvoss [RR20], they conjectured a weaker matrix Spencer by considering the Schatten-p norm of matrices. We can also define such an ℓ_p version of the hyperbolic Spencer conjecture by looking at the ℓ_p-norm of hyperbolic eigenvalues (the hyperbolic-p norm). Any progress towards the ℓ_p-hyperbolic Spencer conjecture will provide more insights in matrix and hyperbolic discrepancy theory.

Fooling hyperbolic cone? One of the results in this chapter is showing an anti-concentration inequality with respect to the hyperbolic spectral norm, which generalizes the results in [OST19, AY22]. They actually combined the anti-concentration results with the Meka-Zuckerman [MZ13] framework to construct PRGs fooling polytopes/positive spectrahedrons. Hence, an open question in complexity theory and pseudorandomness is: can we apply the hyperbolic anti-concentration inequality to construct a PRG fooling positive hyperbolic-spectrahedrons, or even hyperbolic cones?

Concentration of random tensors? Tensor concentration is another natural generalization of matrix concentration. Although there have been a large number of works on this problem [Lat06, Leh11, AL12, AW15, Ver20, ALM21], it is still unclear what is the optimal concentration bound for the Euclidean norm of random tensor $X \in \mathbb{R}^{n^d}$, even in the simple case when $X = x_1 \otimes \cdots \otimes x_d$ for random vectors $x_1, \ldots, x_d \in \mathbb{R}^n$. On the other hand, people also care about whether random tensors are well-conditioned, which is more related to TCS problems including tensor decompositions and learning Gaussian mixtures. The current results [Ver20, Aar15, BCMV14] have a large gap between the matrix case. For these tensor concentration problems, is it possible to study them via hyperbolic polynomials and obtain tighter bounds?

19.2 Preliminaries

19.2.1 Notations

For a vector x, we use $\|x\|_0$ to denote the number of non-zeros, use $\|x\|_1$ to denotes its ℓ_1 norm, and use $\|x\|_p$ to denote its ℓ_p norm for $0 < p \leq \infty$.

We use $r \in \{\pm 1\}^n$ to denote n i.i.d. random variables where each r_i is 1 with probability $1/2$ and -1 otherwise.

The general definition of semi-norm and norm is as follows:

Definition 19.1 (Semi-norm and norm)**.** Let $\|\cdot\| : V \to \mathbb{R}$ be a nonnegative function on vector space V. We say $\|\cdot\|$ is a semi-norm if it satisfies the following properties: For all $a \in \mathbb{R}$ and $x, y \in V$,

- $\|x + y\| \leq \|x\| + \|y\|$;
- $\|ax\| = |a| \cdot \|x\|$.

If furthermore, $\|x\| = 0$ implies $x = 0$ the zero vector of V, then we say $\|\cdot\|$ is a norm.

Definition 19.2 (Normed linear space)**.** A normed linear space is a vector space over $\mathbb{R}$ or $\mathbb{C}$, on which a normed is defined.

19.2.2 Basic definitions of hyperbolic polynomials

We provide the definition of hyperbolic polynomial.

Definition 19.3 (Hyperbolic polynomial)**.** A homogeneous polynomial $h : \mathbb{R} \to \mathbb{R}$ is hyperbolic with respect to a vector $e \in \mathbb{R}^m$ if $h(e) \neq 0$, and for all $x \in \mathbb{R}^m$, the univariate polynomial $t \mapsto h(te - x)$ has only real zeros.

The following fact shows how to factorize a hyperbolic polynomial, which easily follows from the homogeneity of the polynomial:

Fact 19.5 (Hyperbolic polynomial factorization)**.** *For a degree-d polynomial $h \in \mathbb{R}[z_1, \ldots, z_m]$ hyperbolic with respect to $e \in \mathbb{R}^m$, we have*

$$h(te - x) = h(e) \prod_{i=1}^{d} (t - \lambda_i(x))$$

where $\lambda_1(x) \geq \lambda_2(x) \geq \cdots \geq \lambda_d(x)$ are real roots of $h(te - x)$.

All the vectors with nonnegative hyperbolic eigenvalues form a cone, which is proved by Gårding [Går59]. It is a very important object related to the geometry of hyperbolic polynomials. The formal definition is as follows:

Definition 19.4 (Hyperbolic cone)**.** For a degree d hyperbolic polynomial h with respect to $e \in \mathbb{R}^m$, its hyperbolic cone is

$$\Lambda_+(e) := \{x : \lambda_d(x) \geq 0\}.$$

The interior of Λ_+^m is

$$\Lambda_{++}(e) := \{x : \lambda_d(x) > 0\}.$$

Gårding [Går59] showed the following fundamental properties of the hyperbolic cone:

Theorem 19.6 ([Går59])**.** *Suppose $h \in \mathbb{R}[z_1, \ldots, z_m]$ is hyperbolic with respect to $e \in \mathbb{R}^n$. Then,*

1. *$\Lambda_+(e), \Lambda_{++}(e)$ are convex cones.*
2. *$\Lambda_+ + (e)$ is the connected component of $\{x \in \mathbb{R}^m : h(x) \neq 0\}$ which contains e.*
3. *$\lambda_{\min} : \mathbb{R}^m \to \mathbb{R}$ is a concave function, and $\lambda_{\max} : \mathbb{R}^m \to \mathbb{R}$ is convex.*
4. *If $e' \in \Lambda_{++}(e)$, then h is also hyperbolic with respect to e' and $\Lambda_{++}(e') = \Lambda_{++}(e)$.*

For simplicity, we may use Λ_+ and Λ_{++} to denote $\Lambda_+(e), \Lambda_{++}(e)$, when e is clear from context. In this chapter, we always assume that e is any fixed vector in the hyperbolic cone of h.

We define the trace, rank and spectral norm respect to hyperbolic polynomial h.

Definition 19.5 (Hyperbolic trace, rank, spectral norm)**.** Let h be a degree d hyperbolic polynomial with respect to $e \in \mathbb{R}^m$. For any $x \in \mathbb{R}^m$,

$$tr_h[x] := \sum_{i=1}^{d} \lambda_i(x), \quad rank(x) := \#\{i : \lambda_i(x) \neq 0\}, \quad \|x\|_h := \max_{i \in [d]} |\lambda_i(x)| = \max\{\lambda_1(x), -\lambda_d(x)\}.$$

We define the p norm with respect to hyperbolic polynomial h.

Definition 19.6 ($\|\cdot\|_{h,p}$ norm)**.** For any $p \geq 1$, we define the hyperbolic p-norm $\|\cdot\|_{h,p}$ defined as:

$$\|x\|_{h,p} := \|\lambda(x)\|_p = \Big(\sum_{i=1}^{d} |\lambda_i(x)|^p\Big)^{1/p} \quad \forall x \in \mathbb{R}^m.$$

It has been shown that $\|\cdot\|_h$ and $\|\cdot\|_{h,p}$ are indeed norms:

Theorem 19.7 ([Går59, Brä18, Ren19a])**.** *$\|\cdot\|_h$ is a semi-norm.*

Furthermore, if Λ_+ is regular, i.e., $(\Lambda_+ \cap -\Lambda_+) = \{0\}$, then $\|\cdot\|_h$ is a norm on $\mathbb{R}^m$.

Theorem 19.8 ([BGLS01])**.** *For any $p \geq 1$, $\|\cdot\|_{h,p}$ is a semi-norm. Moreover, if the hyperbolic cone Λ_+ is regular, then $\|\cdot\|_{h,p}$ is a norm.*

19.2.3 Basic properties of hyperbolic polynomials

We state a fact for the eigenvalues $\lambda(\cdot)$ of degree-d hyperbolic polynomial h.

Fact 19.9 ([BGLS01])**.** *For all $i \in [d]$,*

$$\lambda_i(s \cdot x + t \cdot e) = \begin{cases} s \cdot \lambda_i(x) + t, & \text{if } s \geq 0; \\ s \cdot \lambda_{d-i}(x) + t, & \text{if } s < 0. \end{cases}$$

Then, we show that the elementary symmetric sum-products of eigenvalues can be computed from the directional derivatives of the polynomial.

Observation 19.10 ([BGLS01])**.** *For a degree-d hyperbolic polynomial h with respect to e, we have*

$$h(te+x) = p(e) \cdot \prod_{i=1}^{d}(t+\lambda_i(x)) = \sum_{i=0}^{d} s_i(\lambda(x)) \cdot t^{d-i},$$

where $\lambda(x) = (\lambda_1(x), \cdots, \lambda_d(x))$ are the hyperbolic eigenvalues of x and $s_i : \mathbb{R}^d \to \mathbb{R}$ is the i-th elementary symmetric polynomial:

$$s_i(y) := \begin{cases} \sum_{S \in \binom{[d]}{i}} \prod_{j \in S} y_j, & \forall i \in [d]; \\ 1 & \text{if } i = 0. \end{cases}$$

Furthermore, for each $i \in \{0, 1, \cdots, d\}$,

$$h(e) \cdot s_i(\lambda(x)) = \frac{1}{(d-i)!} \cdot \nabla^{d-i} h(x) \underbrace{[e, e, \ldots, e]}_{(d-i) \text{ terms}}.$$

If $i \in [d]$, then $s_i \circ \lambda$ is hyperbolic with respect to e of degree i.

Corollary 19.11. *$tr[x]$ is a linear function.*

Proof. By Observation 19.10, we have

$$tr_h[x] = s_1(\lambda(x)) = \frac{1}{h(e) \cdot (d-1)!} \cdot \nabla^{d-1} h(x)[e, e, \ldots, e].$$

Since h is of degree d, $\nabla^{d-1} h$ is a degree-1 polynomial. Hence, $tr_h[x]$ is a linear function. □

19.2.4 Concentration inequalities

In general, for any normed linear space, as mentioned in [LT13], we have the following concentration result:

Theorem 19.12 (Theorem 4.7 in [LT13])**.** *Let $x_1, \ldots, x_n \in \mathcal{B}$ be a fixed finite sequence in normed linear space $\mathcal{B}$. Let $X = \sum_{i=1}^{n} r_i x_i$, where $r_1, \ldots, r_n$ are independent Rademacher random variables. Then, for every $t > 0$,*

$$\Pr_{r \sim \{\pm 1\}^n}[\|X\|_B > t] \leq 2 \exp(-t^2/(32\, \mathbb{E}[\|X\|_B^2])).$$

For matrices with Schatten-p norm, the expectation of Schatten-$2p$ norm of Rademacher sum can be upper-bounded as follows.

Theorem 19.13 (Theorem 3.1 in [TJ74]). *Let $p \geq 1$. For a matrix A, we use $\|A\|_p$ to denote the Shatten-p norm. For any fixed $X_1, X_2, \ldots, X_n \in \mathbb{R}^{d\times d}$, and for independent Rademacher random variables $r_1, r_2, \ldots, r_n$, we have*

$$\left(\mathbb{E}_{r\sim\{\pm1\}^n} \left[\left\| \sum_{i=1}^n r_i X_i \right\|_{2p}^{2p} \right] \right)^{1/(2p)} \leq \sqrt{2p-1} \cdot \left(\sum_{i=1}^n \|X_i\|_{2p}^2 \right)^{1/2}$$

19.2.5 Khinchin-Kahane inequality

In any normed linear space, for any $p, q \geq 1$, the p-th moment and q-th moment of the norm of Rademacher sum are equivalent up to a constant, as shown in [Kah64], which generalized the Khinchin inequality [Khi23].

Theorem 19.14 ([Kah64]; also in [LO94, LT13, KR16]). *For all $p, q \in [1, \infty)$, there exists a universal constant $C_{p,q} > 0$ depending only on p, q, such that for all choices of normed linear space $\mathcal{B}$, finite sets of vectors $x_1, x_2, \cdots, x_n \in \mathcal{B}$, and independent Rademacher variables $r_1, r_2, \cdots, r_n$,*

$$\left(\mathbb{E}_{r\sim\{\pm1\}^n} \left[\left\| \sum_{i=1}^n r_i x_i \right\|^q \right] \right)^{1/q} \leq C_{p,q} \cdot \left(\mathbb{E}_{r\sim\{\pm1\}^n} \left[\left\| \sum_{i=1}^n r_i x_i \right\|^p \right] \right)^{1/p}.$$

If moreover $1 = p \leq q \leq 2$, then $C_{1,q} = 2^{1-1/q}$ is optimal. If $q \in [1, \infty]$, then $C_{1,q} \leq \sqrt{q}$.

19.2.6 Matrix analysis tools

We state a Lemma for singular values of the product of matrices.

Lemma 19.15 (General Horn inequality, Lemma 1.2 in [TJ74]). *Let $A_1, \cdots, A_n \in \mathbb{R}^{d\times d}$ be symmetric matrices. Let $\sigma_1(A), \ldots, \sigma_d(A)$ denote the singular values of A.*

Then, for each $k \in [d]$,

$$\sum_{j=1}^{k} \sigma_j \left(\prod_{i=1}^{n} A_i \right) \leq \sum_{j=1}^{k} \prod_{i=1}^{n} \sigma_j(A_i).$$

We state a Lemma which is implied by Hölder inequality.

Lemma 19.16 (Lyapunov's inequality)**.** *Let $0 < r < s < \infty$ and X be a random variable. Then,*

$$\mathbb{E}\left[|X|^r\right] \leq \left(\mathbb{E}\left[|X|^s\right]\right)^{r/s}.$$

19.2.7 Helton-Vinnikov Theorem

We state a corollary of Helton-Vinnikov Theorem (Theorem 19.4), proved by Gurvits [Gur04]:

Corollary 19.17 (Proposition 1.2 in [Gur04])**.** *Let $h(x)$ be a m-variable degree-d hyperbolic polynomial. Then, for $x, y \in \mathbb{R}^m$, there exists two symmetric real matrices $A, B \in \mathbb{R}^{d \times d}$ such that for any $a, b \in \mathbb{R}$, the ordered eigenvalues $\lambda(ax + by) = \lambda(aA + bB)$.*

This Corollary relates the hyperbolic eigenvalues of a vector $ax + by$ to the eigenvalues of matrix $aA + bB$, which allows us to study some properties of hyperbolic eigenvalues using results in matrix theory.

19.3 Hyperbolic Chernoff bound for Rademacher Sums

In this section, we will prove the Chernoff bound for hyperbolic polynomials (Theorem 19.25). In Section 19.3.1, we provide some basic facts on the concentration of hyperbolic norm. Then, we prove the main result in Section 19.3.2 and Section 19.3.3.

19.3.1 Preliminaries

Recall that the hyperbolic spectral norm $\|\cdot\|_h$ is defined as:

$$\|x\|_h := \|\lambda(x)\|_\infty.$$

We should assume that the hyperbolic cone $\Lambda_{h,+}$ is regular. By Theorem 19.7, we know that $\|\cdot\|_h$ is a norm and $(\mathbb{R}^m, \|\cdot\|_h)$ is a normed linear space. Applying the general concentration on normed linear space (Theorem 19.12) to the $\|\cdot\|_h$ norm, and get the following result:

Corollary 19.18 (Concentration of hyperbolic norm). *Let $X = \sum_{i=1}^n r_i x_i$, where $r_1, r_2, \cdots, r_n$ are independent Rademacher variables and $x_1, x_2, \cdots, x_n \in \mathbb{R}^n$. Then, for every $t > 0$,*

$$\Pr_{r\sim\{\pm 1\}^n}[\|X\|_h > t] \leq 2\exp\Big(-t^2/\Big(32\,\mathbb{E}_{r\sim\{\pm1\}^n}[\|X\|_h^2]\Big)\Big).$$

By Theorem 19.14, we know that any moments of $\|X\|_h$ are equivalent up to a constant factor. In particular,

Claim 19.19 (Equivalence between first- and second-moment). *Given n vectors $x_1, x_2, \cdots, x_n$. Let $r_1, r_2, \cdots, r_n$ denote a sequence of random Rademacher variables. Let $X = \sum_{i=1}^n r_i x_i$. Then,*

$$(\mathbb{E}[\|X\|_h^2])^{1/2} \leq \sqrt{2}\cdot\mathbb{E}[\|X\|_h].$$

We state two useful facts (Fact 19.20 and 19.21) that upper and lower bound the hyperbolic-p norm by hyperbolic spectral norm.

Fact 19.20. *Let h denote a m-variate degree-d hyperbolic polynomial. For any vector x, for any $q > 1$, we have*

$$\|x\|_{h,q} \leq d^{1/q}\cdot\|x\|_h.$$

Proof. We have

$$\|x\|_{h,q} = \|\lambda(x)\|_q \leq d^{1/q} \cdot \|\lambda(x)\|_\infty = d^{1/q} \cdot \|x\|_h.$$

Thus, we complete the proof. $\square$

Fact 19.21. *Let h denote a m-variate degree-d hyperbolic polynomial. For any vector x and for any $q \geq 1$, we have*

$$\|x\|_h \leq \|x\|_{h,q}.$$

Proof. We have

$$\|x\|_h = \|\lambda(x)\|_\infty \leq \|\lambda(x)\|_q = \|x\|_{h,q}.$$

Thus, we complete the proof. $\square$

Fact 19.22. *Let h denote a m-variate degree-d hyperbolic polynomial. For any vector x, if there exists a matrix $A \in \mathbb{R}^{d \times d}$ such that $\lambda(x) = \lambda(A)$, then we have*

$$\|x\|_h = \sigma_1(A).$$

Proof. We have

$$\|x_1\|_h = \|\lambda(x_1)\|_\infty = \|\lambda(A_1)\|_\infty = \sigma_1(A_1).$$

$\square$

We state a useful tool from previous work [TJ74, Zyg02].

Lemma 19.23 ([TJ74, Zyg02])**.** *For $q \geq 2$, we have*

$$\binom{2q}{2k_1, \ldots, 2k_n} \leq M_{2q}^{2q} \cdot \binom{q}{k_1, \ldots, k_n},$$

where $M_{2q} = (\frac{(2q)!}{2^q q!})^{1/(2q)}$.

Using elementary calculations, we can upper bound M_{2q}.

Fact 19.24. *For any $q \geq 1$, we have*

$$\left(\frac{(2q)!}{2^q q!}\right)^{1/(2q)} \leq \sqrt{2q-1}.$$

Proof. We have

$$\begin{aligned}\left(\frac{(2q)!}{2^q q!}\right)^{1/(2q)} &\leq \left(\frac{e \cdot (2q)^{2q} \cdot \sqrt{2q} \cdot e^{-2q}}{2^q \cdot \sqrt{2\pi} \cdot q^q \cdot \sqrt{q} \cdot e^{-q}}\right)^{1/(2q)} \\ &= \left(\frac{e^{1-q}}{\sqrt{\pi}} \cdot 2^q \cdot q^q\right)^{1/(2q)} \\ &\leq \sqrt{2q-1},\end{aligned}$$

where the first step follows from Stirling's formula, and the last step follows from $q \geq 1$. □

19.3.2 Proof of the Chernoff bound for hyperbolic polynomials

The goal of this section is to prove Theorem 19.25.

Theorem 19.25 (Chernoff bound for hyperbolic polynomial)**.** *Let h be an m-variable, degree-s hyperbolic polynomial with respect to e. Given $x_1, x_2, \cdots, x_n \in \mathbb{R}^m$ such that $\mathrm{rank}(x_i) \leq s$ for all $i \in [n]$ and for some $0 < s \leq d$. Let $\sigma = (\sum_{i=1}^n \|x_i\|_h^2)^{1/2}$. Then,*

$$\mathbb{E}_{r \sim \{\pm 1\}^n}\left[\left\|\sum_{i=1}^n r_i x_i\right\|_h\right] \leq \min\{2\sqrt{\log s}, 1\} \cdot \sigma.$$

Furthermore, there exist two constants $C_1, C_2 > 0$ such that for every $t > 0$,

$$\Pr_{r \sim \{\pm 1\}^n}\left[\left\|\sum_{i=1}^n r_i x_i\right\|_h > t\right] \leq C_1 \exp\left(-\frac{C_2 t^2}{\sigma^2 \log(s+1)}\right).$$

Proof. We first upper bound $\mathbb{E}_{r\sim\{\pm1\}^n}[\|\sum_{i=1}^n r_i x_i\|_h]$ by

$$\begin{aligned}
\mathbb{E}_{r\sim\{\pm1\}^n}\left[\left\|\sum_{i=1}^n r_i x_i\right\|_h\right] \le&\ \mathbb{E}_{r\sim\{\pm1\}^n}\left[\left\|\sum_{i=1}^n r_i x_i\right\|_{h,2q}\right] \\
\le&\ \left(\mathbb{E}_{r\sim\{\pm1\}^n}\left[\left\|\sum_{i=1}^n r_i x_i\right\|_{h,2q}^{2q}\right]\right)^{1/(2q)} \\
\le&\ \sqrt{2q-1}\cdot s^{1/(2q)}\cdot\left(\sum_{i=1}^n \|x_i\|_h^2\right)^{1/2},
\end{aligned} \tag{19.6}$$

where the first step follows from $\|x\|_h \le \|x\|_{h,2q}$ when $q \ge 1$ (Fact 19.21), the second step follows from the Lyapunov inequality (Lemma 19.16), and the third step follows from Lemma 19.26.

Let's first assume $s > 1$. By taking $q = \log s$, we have

$$\begin{aligned}
\mathbb{E}_{r\sim\{\pm1\}^n}\left[\left\|\sum_{i=1}^n r_i x_i\right\|_h\right] \le&\ \sqrt{4(\log s)-2}\cdot\left(\sum_{i=1}^n \|x_i\|_h^2\right)^{1/2} \\
=&\ \sqrt{4(\log s)-2}\cdot\sigma \\
\le&\ 2\sqrt{\log s}\cdot\sigma.
\end{aligned}$$

where the second step follows from $\sigma := (\sum_{i=1}^n \|x_i\|_h^2)^{1/2}$. When $s = 1$, by taking $q = 1$, we get that

$$\mathbb{E}_{r\sim\{\pm1\}^n}\left[\left\|\sum_{i=1}^n r_i x_i\right\|_h\right] \le\ \sigma.$$

By Claim 19.19,

$$\mathbb{E}_{r\sim\{\pm1\}^n}\left[\left\|\sum_{i=1}^n r_i x_i\right\|_h^2\right] \le 2\left(\mathbb{E}_{r\sim\{\pm1\}^n}\left[\left\|\sum_{i=1}^n r_i x_i\right\|_h\right]\right)^2 \le \min\{8\log s, 2\}\cdot\sigma^2.$$

Then, by Corollary 19.18,

$$\begin{aligned}
\Pr_{r\sim\{\pm1\}^n}\left[\left\|\sum_{i=1}^n r_i x_i\right\|_h > t\right] \le&\ 2\exp\left(-\frac{t^2}{32\,\mathbb{E}_{r\sim\{\pm1\}^n}[\|\sum_{i=1}^n r_i x_i\|_h^2]}\right) \\
\le&\ 2\exp\left(-\frac{t^2}{64\min\{4\log s, 1\}\cdot\sigma^2}\right).
\end{aligned}$$

Thus, we complete the proof. $\square$

19.3.3 Expected hyperbolic-$2q$ norm bound

The goal of this section is to prove Lemma 19.26.

Lemma 19.26 (Expected hyperbolic-$2q$ norm of Rademacher sum)**.** *Let h be an m-variate, degree-d hyperbolic polynomial. Given n vectors $x_1, \cdots, x_n \in \mathbb{R}^m$ such that rank$(x_i) \le s$ for all $i \in [n]$ and for some $0 < s \le d$. For any $q \ge 1$, we have*

$$\left(\mathbb{E}_{r \sim \{\pm 1\}^n} \left[\left\| \sum_{i=1}^n r_i x_i \right\|_{h,2q}^{2q} \right] \right)^{1/(2q)} \le \sqrt{2q-1} \cdot s^{1/(2q)} \cdot \left(\sum_{i=1}^n \|x_i\|_h^2 \right)^{1/2}.$$

Proof. The main idea is to consider the random variables $r_1, r_2, \cdots, r_n$ one at a time. By the conditional expectation, we have

$$\begin{aligned} \mathbb{E}_{r \sim \{\pm 1\}^n} \left[\left\| \sum_{i=1}^n r_i x_i \right\|_{h,2q}^{2q} \right] &= \mathbb{E}_{r_2, \cdots, r_n \sim \{\pm 1\}} \left[\mathbb{E}_{r_1 \sim \{\pm 1\}} \left[\left\| \sum_{i=1}^n r_i x_i \right\|_{h,2q}^{2q} \right] \right] \\ &= \mathbb{E}_{r_2, \ldots, r_n \sim \{\pm 1\}} \left[\mathbb{E}_{r_1 \sim \{\pm 1\}} \left[\sum_{j=1}^d \lambda_j \left(r_1 x_1 + \sum_{i=2}^n r_i x_i \right)^{2q} \right] \right]. \end{aligned}$$

where the last step follows from the definition of $\| \cdot \|_{h,2q}$ norm.

To apply Corollary 19.17, let $x = x_1, y = \sum_{i=2}^n r_i x_i$. Then, there exists two symmetric matrices $A_1, B_1 \in \mathbb{R}^{d \times d}$ such that

$$\lambda \left(r_1 x_1 + \sum_{i=2}^n r_i x_i \right) = \lambda(r_1 A_1 + B_1), \tag{19.7}$$

where λ is the vector of eigenvalues ordered from large to small. Then, we have

$$\lambda(x_1) = \lambda(A_1), \quad \lambda\left(\sum_{i=2}^n r_i x_i \right) = \lambda(B_1).$$

Hence, by the definition of Schatten-p norm,

$$\begin{aligned}\sum_{j=1}^{d}\left(\lambda_j\Big(r_1x_1+\sum_{i=2}^{n}r_ix_i\Big)\right)^{2q} &= \|r_1A_1+B_1\|_{2q}^{2q}\\ &= tr\left[(r_1A_1+B_1)^{2q}\right]\\ &= \sum_{\beta\in\{0,1\}^{2q}} tr\left[\prod_{i=1}^{2q}A_1^{\beta_i}B_1^{1-\beta_i}\right]\cdot r_1^{\sum_{i=1}^{2q}\beta_i}. \end{aligned} \tag{19.8}$$

where the first step follows from Eq. (19.7) and the definition of matrix Schatten p-norm, the second step follows from $\|A\|_{2q}^{2q}=tr[A^{2q}]$ for symmetric matrix A and $q\ge 1$, and the last step follows from the linearity of trace.

We define a set which will be used later.

$$\mathcal{B}_{\text{even}} := \left\{\beta\in\{0,1\}^{2q} : \sum_{i=1}^{2q}\beta_i \text{ is even}\right\}.$$

By taking expectation for r_1, we have

$$\begin{aligned}\mathbb{E}_{r_1\sim\{\pm1\}}\left[\sum_{j=1}^{d}\lambda_j\left(r_1x_1+\sum_{i=2}^{n}r_ix_i\right)^{2q}\right] &= \sum_{\beta\in\{0,1\}^{2q}} tr\left[\prod_{i=1}^{2q}A_1^{\beta_i}B_1^{1-\beta_i}\right]\cdot\mathbb{E}_{r_1\sim\{\pm1\}}\left[r_1^{\sum_{i=1}^{2q}\beta_i}\right]\\ &= \sum_{\beta\in\mathcal{B}_{\text{even}}} tr\left[\prod_{i=1}^{2q}A_1^{\beta_i}B_1^{1-\beta_i}\right]\end{aligned}$$

where the first step follows from Eq. (19.8) and the linearity of expectation, and the last step follows from

$$\mathbb{E}_{r_1\sim\{\pm1\}}\left[r_1^k\right] = \begin{cases}0 & \text{if } k \text{ is odd,}\\ 1 & \text{if } k \text{ is even.}\end{cases}$$

For each $\beta\in\mathcal{B}_{\text{even}}$, we have

$$tr\left[\prod_{i=1}^{2q}A_1^{\beta_i}B_1^{1-\beta_i}\right] \le \sum_{j=1}^{s}\sigma_j\left(\prod_{i=1}^{2q}A_1^{\beta_i}B_1^{1-\beta_i}\right) \le \sum_{j=1}^{s}\prod_{i=1}^{2q}\sigma_j\left(A_1^{\beta_i}B_1^{1-\beta_i}\right),$$

where $\sigma_j(A)$ is the j-th singular value of A and the first step follows from $tr[A]\le\sum_{i=1}^{rank(A)}\sigma_j(A)$ for any real square matrix A, and the second step follows from general Horn inequality (Lemma 19.15).

Then, it follows that

$$\begin{aligned}\sum_{\beta\in\mathcal{B}_{\text{even}}} tr\left[\prod_{i=1}^{2q} A_1^{\beta_i}B_1^{1-\beta_i}\right] \leq & \sum_{\beta\in\mathcal{B}_{\text{even}}}\sum_{j=1}^{s}\sigma_j(A_1)^{\sum_{i=1}^{2q}\beta_i}\sigma_j(B_1)^{2q-\sum_{i=1}^{2q}\beta_i} \\ = & \sum_{k=0}^{q}\binom{2q}{2k}\sum_{j=1}^{s}\sigma_j(A_1)^{2k}\sigma_j(B_1)^{2q-2k}, \qquad (19.9)\end{aligned}$$

where the first step follows from $rank(A) = rank(x_1) \leq s$. Hence,

$$\begin{aligned}\mathbb{E}_{r_1,\dots,r_n\sim\{\pm1\}}\left[\left\|\sum_{i=1}^{n} r_i x_i\right\|_{h,2q}^{2q}\right] \leq & \sum_{k_1=0}^{q}\binom{2q}{2k_1}\mathbb{E}_{r_2,\dots,r_n}\left[\sum_{j=1}^{s}\sigma_j(A_1)^{2k_1}\sigma_j(B_1)^{2q-2k_1}\right] \\ \leq & \sum_{k_1=0}^{q}\binom{2q}{2k_1}\mathbb{E}_{r_2,\dots,r_n}\left[\sigma_1(A_1)^{2k_1}\sum_{j=1}^{s}\sigma_j(B_1)^{2q-2k_1}\right] \\ = & \sum_{k_1=0}^{q}\binom{2q}{2k_1}\mathbb{E}_{r_2,\dots,r_n}\left[\|x_1\|_h^{2k_1}\sum_{j=1}^{s}\lambda_j(B_1)^{2q-2k_1}\right] \\ = & \sum_{k_1=0}^{q}\binom{2q}{2k_1}\|x_1\|_h^{2k_1}\,\mathbb{E}_{r_2,\dots,r_n}\left[\sum_{j=1}^{s}\lambda_j(B_1)^{2q-2k_1}\right] \\ = & \sum_{k_1=0}^{q}\binom{2q}{2k_1}\|x_1\|_h^{2k_1}\,\mathbb{E}_{r_2,\dots,r_n}\left[\left\|\sum_{i=2}^{n} r_i x_i\right\|_{h,2q-2k_1}^{2q-2k_1}\right], \qquad (19.10)\end{aligned}$$

where the second step follows from $\sigma_1(A) \geq \cdots \geq \sigma_s(A)$, the third step follows from for $\sum_{i=1}^{s}\sigma_i(A)^k = \sum_{i=1}^{s}\lambda_i(A)^k$ for even k, the forth step follows from Fact 19.22, and the last step follows from definition of $\|\cdot\|_{h,q}$.

Now, we can iterate this process for $\mathbb{E}_{r_2,\dots,r_n}\left[\|\sum_{i=2}^{n} r_i x_i\|_{h,2q-2k_1}^{2q-2k_1}\right]$. Consider $r_2x_2 + \sum_{i=3}^{n} r_i x_i$. By Corollary 19.17, there exists two symmetric matrices $A_2, B_2 \in \mathbb{R}^{d\times d}$ such that

$$\lambda\left(r_2x_2 + \sum_{i=3}^{n} r_i x_i\right) = \lambda(r_2A + B)$$

for all $r_2 \in \{-1, 1\}$. By the conditional expectation again, we can get that

$$\begin{aligned}
\mathbb{E}_{r_1,\dots,r_n}\left[\left\|\sum_{i=1}^n r_i x_i\right\|_{h,2q}^{2q}\right] &\le \sum_{k_1=0}^{q}\binom{2q}{2k_1}\|x_1\|_h^{2k_1}\,\mathbb{E}_{r_2,\dots,r_n}\left[\left\|\sum_{i=2}^n r_i x_i\right\|_{h,2q-2k_1}^{2q-2k_1}\right] \\
&\le \sum_{k_1=0}^{q}\binom{2q}{2k_1}\|x_1\|_h^{2k_1}\sum_{k_2=0}^{2q-2k_1}\binom{2q-2k_1}{2k_2}\|x_2\|_h^{2k_2}\,\mathbb{E}_{r_3,\dots,r_n}\left[\left\|\sum_{i=3}^n r_i x_i\right\|_{h,2k_3}^{2k_3}\right],
\end{aligned}$$

where $k_3 = q - k_1 - k_2$ and the second step follows from applying Eq. (19.10) for $\mathbb{E}_{r_2,\dots,r_n}\left[\|\sum_{i=2}^n r_i x_i\|_{h,2q-2k_1}^{2q-2k_1}\right]$.

If we iterate $n-1$ times, we finally get

$$\begin{aligned}
\mathbb{E}\left[\left\|\sum_{i=1}^n r_i x_i\right\|_{h,2q}^{2q}\right] &\le \sum_{\substack{k_1,\dots,k_n\ge 0\\ k_1+\cdots+k_n=q}}\prod_{i=1}^{n-1}\binom{2q-\sum_{j=1}^{i-1}2k_j}{2k_i}\|x_i\|_h^{2k_i}\cdot\mathbb{E}_{r_n}\left[\|r_n x_n\|_{h,2k_n}^{2k_n}\right] \\
&= \sum_{\substack{k_1,\dots,k_n\ge 0\\ k_1+\cdots+k_n=q}}\binom{2q}{2k_1,\dots,2k_n}\prod_{i=1}^{n-1}\|x_i\|_h^{2k_i}\cdot\mathbb{E}_{r_n}\left[\|r_n x_n\|_{h,2k_n}^{2k_n}\right] \\
&= \sum_{\substack{k_1,\dots,k_n\ge 0\\ k_1+\cdots+k_n=q}}\binom{2q}{2k_1,\dots,2k_n}\prod_{i=1}^{n-1}\|x_i\|_h^{2k_i}\cdot\|x_n\|_{h,2k_n}^{2k_n}, \qquad (19.11)
\end{aligned}$$

where the first step follows from iterating the same rule for $n-1$ times, the second step follows from

$$\begin{aligned}
\prod_{i=1}^{n-1}\binom{2q-\sum_{j=1}^{i-1}2k_j}{2k_i} &= \binom{2q}{2k_1}\cdot\binom{2q-2k_1}{2k_2}\cdot\binom{2q-2k_1-2k_2}{2k_3}\cdots\binom{2k_{n-1}+2k_n}{2k_{n-1}} \\
&= \binom{2q}{2k_1}\cdot\binom{2q-2k_1}{2k_2}\cdot\binom{2q-2k_1-2k_2}{2k_3}\cdots\binom{2k_{n-1}+2k_n}{2k_{n-1}}\cdot\binom{2k_n}{2k_n} \\
&= \binom{2q}{2k_1,\dots,2k_n}.
\end{aligned}$$

By Lemma 19.23, we have that

$$\binom{2q}{2k_1,\dots,2k_n} \le M_{2q}^{2q}\cdot\binom{q}{k_1,\dots,k_n},$$

where $M_{2q} = \left(\frac{(2q)!}{2^q q!}\right)^{1/(2q)} \leq \sqrt{2q-1}$ by Fact 19.24.

Hence,

$$\begin{aligned}
\mathbb{E}_{r\sim\{\pm1\}^n}\left[\left\|\sum_{i=1}^n r_i x_i\right\|_{h,2q}^{2q}\right] &\leq M_{2q}^{2q} \sum_{\substack{k_1,\dots,k_n\geq 0\\ k_1+\cdots+k_n=q}} \binom{q}{k_1,\dots,k_n} \prod_{i=1}^{n-1} \|x_i\|_h^{2k_i} \cdot \|x_n\|_{h,2k_n}^{2k_n} \\
&\leq M_{2q}^{2q} \sum_{\substack{k_1,\dots,k_n\geq 0\\ k_1+\cdots+k_n=q}} \binom{q}{k_1,\dots,k_n} \prod_{i=1}^{n-1} \|x_i\|_h^{2k_i} \cdot s \cdot \|x_n\|_h^{2k_n} \\
&= M_{2q}^{2q} \cdot s \cdot \sum_{\substack{k_1,\dots,k_n\geq 0\\ k_1+\cdots+k_n=q}} \binom{q}{k_1,\dots,k_n} \prod_{i=1}^{n} \|x_i\|_h^{2k_i} \\
&= M_{2q}^{2q} \cdot s \cdot \left(\sum_{i=1}^n \|x_i\|_h^2\right)^q,
\end{aligned}$$

where the second step follows from $\|x_n\|_{h,2k_n} \leq s^{1/(2k_n)} \cdot \|x_n\|_h$ for rank-s vector (see Fact 19.20), the third step follows from re-organizing the terms, and the last step follows from expanding $(\sum_{i=1}^n \|x_i\|_h^2)^q$.

Therefore,

$$\left(\mathbb{E}_{r\sim\{\pm1\}^n}\left[\left\|\sum_{i=1}^n r_i x_i\right\|_{h,2q}^{2q}\right]\right)^{1/(2q)} \leq \sqrt{2q-1} \cdot s^{1/(2q)} \cdot \left(\sum_{i=1}^n \|x_i\|_h^2\right)^{1/2},$$

which completes the proof of Lemma 19.26.

Remark 19.1. This upper bound depends essentially on the the maximum hyperbolic rank of all the vectors $x_1,\dots,x_n$, instead of just the last one. It follows since Eq. (19.10) can be expanded as:

$$\begin{aligned}
\mathbb{E}_{r_1,\dots,r_n\sim\{\pm1\}}\left[\left\|\sum_{i=1}^n r_i x_i\right\|_{h,2q}^{2q}\right] \leq{}& \mathbb{E}_{r_2,\dots,r_n\sim\{\pm1\}}\left[\left\|\sum_{i=2}^n r_i x_i\right\|_{h,2q}^{2q}\right] + \|x_1\|_{h,2q}^{2q} \\
&+ \sum_{k_1=1}^{q-1} \binom{2q}{2k_1} \|x_1\|_h^{2k_1} \cdot \mathbb{E}_{r_2,\dots,r_n\sim\{\pm1\}}\left[\left\|\sum_{i=2}^n r_i x_i\right\|_{h,2q-k_1}^{2q-k_1}\right].
\end{aligned}$$

We can see that the first term depends on the rank of $x_2, \dots, x_n$, the second term depends on the $rank(x_1)$. Hence, the whole summation cannot be uniformly bounded by the rank of the last vector x_n. Hence, adding a rank-1 dummy vector cannot improve the bound.

$\square$

19.4 Hyperbolic Chernoff bound for hyperbolic cone vectors

The goal of this section is to prove the following theorem, which generalizes the matrix Chernoff bound for positive semi-definite matrices to the hyperbolic version with respect to random vectors in the hyperbolic cone.

Theorem 19.27. *Let h be an m-variate, degree-d hyperbolic polynomial with hyperbolic direction $e \in \mathbb{R}^m$. Let Λ_+ denote the hyperbolic cone of h with respect to e. Suppose $\mathsf{x}_1, \dots, \mathsf{x}_n$ are n independent random vectors with supports in Λ_+ such that $\lambda_{\max}(\mathsf{x}_i) \le R$ for all $i \in [n]$.*

Define the mean of minimum and maximum eigenvalues as follows:

$$\mu_{\min} := \sum_{i=1}^{n} \mathbb{E}[\lambda_{\min}(\mathsf{x}_i)], \quad \textit{and} \quad \mu_{\max} := \sum_{i=1}^{n} \mathbb{E}[\lambda_{\max}(\mathsf{x}_i)].$$

Then, we have

$$\Pr\left[\lambda_{\max}\left(\sum_{i=1}^{n} \mathsf{x}_i\right) \ge (1+\delta)\mu_{\max}\right] \le d \cdot \left(\frac{(1+\delta)^{1+\delta}}{e^{\delta}}\right)^{-\mu_{\max}/R} \quad \forall \delta \ge 0,$$
$$\Pr\left[\lambda_{\min}\left(\sum_{i=1}^{n} \mathsf{x}_i\right) \le (1-\delta)\mu_{\min}\right] \le d \cdot \left(\frac{(1-\delta)^{1-\delta}}{e^{-\delta}}\right)^{-\mu_{\min}/R} \quad \forall \delta \in [0,1].$$

Proof. Without loss of generality, we may assume that $\lambda_{\max}(\mathsf{x}_i) \le 1$. The general case will follow from scaling.

Maximum eigenvalue: By the Laplace transform method, we have

$$\begin{aligned}\Pr\left[\lambda_{\max}\left(\sum_{i=1}^{n}\mathsf{x}_i\right)\geq t\right] &\leq \inf_{\theta>0}\ e^{-\theta t}\cdot\mathbb{E}\left[\exp\left(\theta\lambda_{\max}\left(\sum_{i=1}^{n}\mathsf{x}_i\right)\right)\right]\\ &= \inf_{\theta>0}\ e^{-\theta t}\cdot\mathbb{E}\left[\sum_{q\geq 0}\frac{\theta^q}{q!}\lambda_{\max}\left(\sum_{i=1}^{n}\mathsf{x}_i\right)^q\right]\\ &\leq \inf_{\theta>0}\ e^{-\theta t}\cdot\sum_{q\geq 0}\frac{\theta^q}{q!}\,\mathbb{E}\left[\sum_{j=1}^{d}\lambda_j\left(\sum_{i=1}^{n}\mathsf{x}_i\right)^q\right],\end{aligned}\tag{19.12}$$

where the second step follows from Taylor expansion, and the third step follows from $\mathsf{x}_i\in\Lambda_+$ and each term in the summation are non-negative.

Then, the remaining task is very similar to the proof of Lemma 19.26. We will upper bound the expectation of trace moments as follows: let $\mathbb{E}_i$ denote the expectation over x_i, and $\mathbb{E}_{\geq i}$ denote the expectation over $\mathsf{x}_i,\ldots,\mathsf{x}_n$. Then, we have

$$\begin{aligned}\mathbb{E}_{\geq 1}\left[\sum_{j=1}^{d}\lambda_j\left(\sum_{i=1}^{n}\mathsf{x}_i\right)^q\right] &= \mathbb{E}_{\geq 2}\mathbb{E}_1\left[\sum_{j=1}^{d}\lambda_j\left(\mathsf{x}_1+\sum_{i=2}^{n}\mathsf{x}_i\right)^q\right]\\ &= \mathbb{E}_{\geq 2}\mathbb{E}_1\left[tr\left[(A_1+B_1)^q\right]\right],\end{aligned}$$

where $A_1, B_1\in\mathbb{R}^{d\times d}$ are two symmetric matrices given by Corollary 19.17 such that A_1 depends on the value of x_1 and B_1 depends on the values of $\mathsf{x}_2,\ldots,\mathsf{x}_n$. Since all eigenvalues of $\mathsf{x}_1,\ldots,\mathsf{x}_n$ are non-negative, A and B are positive semi-definite matrices.

Then, we have

$$\begin{aligned}
\mathbb{E}_{\geq 2}\mathbb{E}_1\left[tr\left[(A_1+B_1)^q\right]\right] &= \mathbb{E}_{\geq 2}\mathbb{E}_1\left[\sum_{\beta\in\{0,1\}^q} tr\left[\prod_{k=1}^{q} A_1^{\beta_i} B_1^{1-\beta_i}\right]\right] \\
&\leq \mathbb{E}_{\geq 2}\mathbb{E}_1\left[\sum_{\beta\in\{0,1\}^q}\sum_{j=1}^{d} \lambda_j(A_1)^{\sum_{i=1}^q \beta_i}\cdot\lambda_j(B_1)^{q-\sum_{i=1}^q \beta_i}\right] \\
&= \mathbb{E}_{\geq 2}\mathbb{E}_1\left[\sum_{k_1=0}^{q}\binom{q}{k_1}\sum_{j=1}^{d}\lambda_j(A_1)^{k_1}\cdot\lambda_j(B_1)^{q-k_1}\right] \\
&\leq \mathbb{E}_{\geq 2}\mathbb{E}_1\left[\sum_{k_1=0}^{q}\binom{q}{k_1}\lambda_{\max}(A_1)^{k_1}\cdot\sum_{j=1}^{d}\lambda_j(B)^{q-k_1}\right] \\
&= \mathbb{E}_1\left[\sum_{k_1=0}^{q}\binom{q}{k_1}\lambda_{\max}(\mathsf{x}_1)^{k_1}\cdot\mathbb{E}_{\geq 2}\left[\sum_{j=1}^{d}\lambda_j\left(\sum_{i=2}^{n}\mathsf{x}_i\right)^{q-k_1}\right]\right].
\end{aligned}$$

where the second step follows from the repeated application of Horn's inequality (Lemma 19.15) and A_1, B are positive semi-matrices, and the last step follows from x_1 is independent with $\mathsf{x}_2, \ldots, \mathsf{x}_n$.

Then, by repeating this process, we finally get that

$$\begin{aligned}
\mathbb{E}\left[\sum_{j=1}^{d}\lambda_j\left(\sum_{i=1}^{n}\mathsf{x}_i\right)^q\right] &\leq \mathbb{E}\left[\sum_{\substack{k_1,\ldots,k_n\geq 0\\ k_1+\cdots+k_n=q}}\binom{q}{k_1,\ldots,k_n}\prod_{i=1}^{n}\lambda_{\max}(\mathsf{x}_i)^{k_i}\cdot d\right] \\
&= d\cdot\mathbb{E}\left[\left(\sum_{i=1}^{n}\lambda_{\max}(\mathsf{x}_i)\right)^q\right].
\end{aligned}$$

Putting it to Eq. (19.12), we have

$$\begin{aligned}
\Pr\left[\lambda_{\max}\left(\sum_{i=1}^n \mathsf{x}_i\right) \geq t\right] &\leq \inf_{\theta>0}\ e^{-\theta t}\cdot\sum_{q\geq 0}\frac{\theta^q}{q!}\,\mathbb{E}\left[\sum_{j=1}^d \lambda_j\left(\sum_{i=1}^n \mathsf{x}_i\right)^q\right] \\
&\leq \inf_{\theta>0}\ e^{-\theta t}\cdot\sum_{q\geq 0}\frac{\theta^q}{q!}\cdot d\cdot\mathbb{E}\left[\left(\sum_{i=1}^n \lambda_{\max}(\mathsf{x}_i)\right)^q\right] \\
&= \inf_{\theta>0}\ e^{-\theta t}\cdot d\cdot\mathbb{E}\left[\exp\left(\theta\cdot\sum_{i=1}^n \lambda_{\max}(\mathsf{x}_i)\right)\right] \\
&= \inf_{\theta>0}\ e^{-\theta t}\cdot d\cdot\prod_{i=1}^n \mathbb{E}\left[e^{\theta\lambda_{\max}(\mathsf{x}_i)}\right],
\end{aligned}$$

where the third step follows from the linearity of expectation, and the last step follows from the independence of $\mathsf{x}_1,\ldots,\mathsf{x}_n$.

For $x \in [0,1]$, we know that $e^{\theta x} \leq 1+(e^\theta-1)x$ holds for $\theta\in\mathbb{R}$. Thus,

$$\begin{aligned}
e^{-\theta t}\cdot d\cdot\prod_{i=1}^n \mathbb{E}\left[e^{\theta\lambda_{\max}(\mathsf{x}_i)}\right] &\leq e^{-\theta t}\cdot d\cdot\prod_{i=1}^n(1+(e^\theta-1)\,\mathbb{E}[\lambda_{\max}(\mathsf{x}_i)]) \\
&= d\cdot\exp\left(-\theta t+\sum_{i=1}^n\log\left(1+(e^\theta-1)\,\mathbb{E}[\lambda_{\max}(\mathsf{x}_i)]\right)\right) \\
&\leq d\cdot\exp\left(-\theta t+\sum_{i=1}^n(e^\theta-1)\,\mathbb{E}[\lambda_{\max}(\mathsf{x}_i)]\right) \\
&= d\cdot\exp\left(-\theta t+(e^\theta-1)\mu_{\max}\right)
\end{aligned}$$

where the second step follows from our assumption that $\lambda_{\max}(x)\in[0,1]$ for all $i\in[n]$, and the third step follows from $\log(1+x)\leq x$ for $x>-1$. Therefore, by taking $\theta := \log(t/\mu_{\max})$, we have

$$\Pr\left[\lambda_{\max}\left(\sum_{i=1}^n \mathsf{x}_i\right)\geq t\right] \leq d\cdot\left(\frac{t}{\mu_{\max}}\right)^{-t}\cdot e^{t-\mu}. \tag{19.13}$$

If we choose $t := (1+\delta)\mu_{\max}$, we get that

$$\Pr\left[\lambda_{\max}\left(\sum_{i=1}^n \mathsf{x}_i\right)\geq (1+\delta)\mu_{\max}\right] \leq d\cdot\left(\frac{(1+\delta)^{1+\delta}}{e^\delta}\right)^{-\mu_{\max}},$$

which completes the proof of the maximum eigenvalue case.

Minimum eigenvalue: We reduce this case to the maximum eigenvalue case by defining $\mathsf{x}_i' := e - \mathsf{x}_i$ for $i \in [n]$. Then, by Fact 19.9,

$$\lambda_{\max}(\mathsf{x}_i') = 1 - \lambda_{\min}(\mathsf{x}_i) \leq 1, \quad \text{and} \quad \lambda_{\min}(\mathsf{x}_i') = 1 - \lambda_{\max}(\mathsf{x}_i) \geq 0.$$

Thus,

$$\begin{aligned}
\Pr\left[\lambda_{\min}\left(\sum_{i=1}^n \mathsf{x}_i\right) \leq (1-\delta)\mu_{\min}\right] =&\ \Pr\left[\lambda_{\max}\left(\sum_{i=1}^n \mathsf{x}_i'\right) \geq n - (1-\delta)\mu_{\min}\right] \\
&\leq d \cdot \left(\frac{n - (1-\delta)\mu_{\min}}{n - \mu_{\min}}\right)^{n-(1-\delta)\mu_{\min}} \cdot e^{\delta\mu_{\min}} \\
&= d \cdot \left(1 + \frac{\delta}{n/\mu_{\min} - 1}\right)^{\left(\frac{n}{(1-\delta)\mu_{\min}} - 1\right)\cdot(1-\delta)\mu_{\min}} \cdot e^{\delta\mu_{\min}} \\
&\leq d \cdot \left(\frac{(1-\delta)^{1-\delta}}{e^{-\delta}}\right)^{-\mu_{min}},
\end{aligned}$$

where the second step follows from taking $t := n - (1-\delta)\mu_{\min}$ in Eq. (19.13) and $\mu'_{\max} = n - \mu_{\min}$, the last step follows from $n/\mu_{\min} > 0$.

Hence, the proof of the theorem is completed. $\square$

19.5 Hyperbolic Anti-Concentration Bound

19.5.1 Our result

In this section, we will prove an anti-concentration bound for random vectors with respect to the hyperbolic norm, which generalizes the result for PSD matrices in [AY22]. In particular, an important tool we use is the hyperbolic Chernoff bound for random vectors in the hyperbolic cone (Theorem 19.27), together with a robust Littlewood-Offord theorem for hyperbolic cone (Theorem 19.29).

Theorem 19.28 (Hyperbolic anti-concentration theorem)**.** *Let h_1, h_2 be an m-variate degree-d hyperbolic polynomial with hyperbolic direction $e_1, e_2 \in \mathbb{R}^m$, respectively. Let $y_1, y_2 \in \mathbb{R}^m$ be two vectors. Let $\{x_i^1\}_{i\in[n]}$ and $\{x_i^2\}_{i\in[n]}$ be two sequences of vectors such that $x_i^1 \in \Lambda_{+,h_1}$ and $x_i^2 \in (-\Lambda_{+,h_2})$, i.e., $\lambda_{\min,h_1}(x_i^1) \geq 0$, $\lambda_{\max,h_2}(x_i^2) \leq 0$ for all $i \in [n]$.*

Let $\tau \leq \frac{1}{\sqrt{\log d}}$. We further assume that $\lambda_{\max,h_1}(x_i^1) \leq \tau$ and $\lambda_{\min,h_2}(x_i^2) \geq -\tau$ for all $i \in [n]$. And for $j \in [2]$, we have $\sum_{i=1}^n \lambda_{\min}(x_i^j)^2 \geq 1$.

Then, for $\Delta \geq 20\tau \log d$, we have

$$\Pr_{\epsilon \sim \{-1,1\}^n} \left[\exists j \in [2] : \left\| \sum_{i=1}^n \epsilon_i x_i^j - y_j \right\|_{h_j} \leq \Delta \right] \leq O(\Delta).$$

Proof. We follow the proof in [AY22] but adapt it to the hyperbolic polynomial.

Let $f_j(\epsilon) := \sum_{i=1}^n \epsilon_i x_i^j$ for $j \in [2]$. And we will first show that

$$\Pr_{\epsilon \sim \{\pm 1\}^n} [\exists j \in [2] : \lambda_{\max,h_j}(f_j(\epsilon) - y_j) \leq \Delta] \leq O(\Delta),$$

which implies the anti-concentration bound for the hyperbolic spectral norm.

Let $p := \frac{1}{20\tau^2 \log d}$ and let $\pi : [n] \to [2p]$ be a random hash function that independently assigns each $i \in [n]$ to uniformly random bucket in $[2p]$. For $i \in [2p]$, let $C_i := \{j \in [n] : \pi[j] = i\}$ be the set of elements in the i-th bucket. Let $\gamma \sim \{\pm 1\}^{2p}$. For $j \in [2]$, define a new function $g_j(\gamma) : \{\pm\}^{2k} \to \mathbb{R}^m$ as follows:

$$g_j(\gamma) := \sum_{i=1}^{2p} \gamma_i \cdot \sum_{j \in C_i} x_i^j.$$

That is, we assign the same sign for vectors hashed into the same bucket.

[AY22] proved that $f_j(\epsilon)$ and $g_j(\gamma)$ have the same distribution using a direct argument about the random hash function. Thus, it is also true in our case and we just need to prove

$$\Pr_{\gamma \sim \{\pm 1\}^{2p}} [\exists j \in [2] : \lambda_{\max,h_j}(g_j(\gamma) - y_j) \leq \Delta] \leq O(\Delta)$$

For $j = 1$, define the good bucket set

$$\mathcal{B}^1_{\text{good}} := \left\{ c \in [2p] : \lambda_{\min,h_1} \left(\sum_{i \in \pi^{-1}(c)} x_i^1 \right) \geq \frac{1}{4\tau p} \right\}.$$

By Lemma 19.30, with probability at least $1 - e^{-p/2}$, we have $|\mathcal{B}^1_{\text{good}}| \geq \frac{8}{5}p$.

For $j = 2$, define the good bucket set

$$\mathcal{B}^2_{\text{good}} := \left\{ c \in [2p] : \lambda_{\max,h_2}\left(\sum_{i\in\pi^{-1}(c)} x_i^2\right) \leq -\frac{1}{4\tau p} \right\}.$$

By considering $-x_i^2$ and applying Lemma 19.30, we get that with with probability at least $1 - e^{-p/2}$, we have $|\mathcal{B}^2_{\text{good}}| \geq \frac{8}{5}p$.

By a pigeonhole principle and union bound, with probability $1-2e^{-p/2}$, $|\mathcal{B}^1_{\text{good}} \cap \mathcal{B}^2_{\text{good}}| \geq \frac{6}{5}p$. That is, at least $\frac{3}{5}$-fraction of $i \in [2p]$ such that

$$\lambda_{\min,h_1}\left(\sum_{j\in\pi^{-1}(c)} x_i^1\right) \geq \frac{1}{4\tau p}, \quad \text{and} \quad \lambda_{\max,h_2}\left(\sum_{j\in\pi^{-1}(c)} x_i^2\right) \leq -\frac{1}{4\tau p}.$$

Thus, we can apply Theorem 19.29 with $\alpha = \frac{3}{5}, \rho = \frac{1}{4\tau p}$ and get that

$$\Pr_{\gamma\sim\{-1,1\}^{2p}}\left[\exists j \in [2]: \ \lambda_{\max,h_j}\left(\sum_{i=1}^{2p} \gamma_i \cdot \sum_{k\in\pi^{-1}} x_k^j - y_j\right) \in \left(-\frac{1}{2\tau p}, 0\right]\right] \leq O\left(\frac{1}{\sqrt{p}}\right) + 2e^{-p/2}.$$

Now, we transform back to the distribution of $f_j(\epsilon)$ and have the following bound:

$$\Pr_{\epsilon\sim\{\pm 1\}^n}\left[\exists j \in [2] : \lambda_{\max,h_j}(f_j(\epsilon) - y_j) \in \left(-\frac{1}{2\tau p}, 0\right]\right] \leq O\left(\frac{1}{\sqrt{p}}\right) + 2e^{-p/2}. \quad (19.14)$$

However, by our choice of parameters, $\Delta \geq \frac{1}{2\tau p}$. We can partition the interval $[-\Delta, 0]$ into $\lceil 2\tau p\Delta \rceil$ sub-intervals each of length $\frac{1}{2\tau p}$. Since Eq. (19.14) holds for any $y_j \in \mathbb{R}^m$, we can use it to bound the probability of the event $\lambda_{\max,h_j}(f_j(\epsilon) - y_j) \in (-\frac{k}{2\tau p}, -\frac{k-1}{2\tau p}]$ by shifting $y_j' := y_j + \frac{k-1}{2\tau p} \cdot e$. Therefore, by the union bound, we have

$$\begin{aligned}
\Pr_{\epsilon\sim\{\pm 1\}^n}\left[\exists j \in [2] : \lambda_{\max,h_j}(f_j(\epsilon) - y_j) \in [\Delta, 0]\right] &\leq \lceil 2\tau p\Delta \rceil \cdot \left(O\left(\frac{1}{\sqrt{p}}\right) + 2e^{-p/2}\right) \\
&= O(\Delta \cdot \tau\sqrt{p}) \\
&= O(\Delta),
\end{aligned}$$

where the last step follows from $\tau\sqrt{p} = O(1)$ by our choice of parameters.

We note that the above upper bound also holds for the interval $[0, \Delta]$. Hence, we complete the proof of the theorem. □

19.5.2 Technical lemmas

To prove Theorem 19.28, we need a robust Littlewood–Offord theorem for hyperbolic cones. This kind of theorems were previously proved by [OST19] for polytopes and [AY22] for positive spectrahedrons.

We first give some definitions in [OST19] about functions on hypercube.

Definition 19.7 (Unateness)**.** A function $F : \{-1,1\}^n \to \{0,1\}$ is unate if for all $i \in [n]$, F is either increasing or decreasing with respect to the ith coordinate, i.e.,

$$F(x_1,\dots,x_{i-1},-1,x_{i+1},\dots,x_n) \le F(x_1,\dots,x_{i-1},1,x_{i+1},\dots,x_n) \quad \forall x \in \{\pm 1\}^n, \quad \text{or}$$
$$F(x_1,\dots,x_{i-1},-1,x_{i+1},\dots,x_n) \ge F(x_1,\dots,x_{i-1},1,x_{i+1},\dots,x_n) \quad \forall x \in \{\pm 1\}^n.$$

Let $H, \overline{H}$ be the indicator set of two unate functions and $H \subset \overline{H}$. The boundary of H is denoted by $\partial H := \overline{H} \backslash H$.

Definition 19.8 (Semi-thin)**.** For $\alpha \in [0,1]$, we say ∂H is α-semi thin if for all $x \in H$, at least α-fraction of its hypercube neighbors (different in one coordinate) are not in ∂H.

Now, we state the main theorem of this section:

Theorem 19.29 (Robust Littlewood-Offord theorem for hyperbolic cones)**.** *Let $\alpha \in [0,1], \rho > 0$. Let $\{x_i^j\}_{i\in[n],j\in[2]}$ be $2n$ vectors in $\mathbb{R}^m$ such that $x_i^1 \in \Lambda_{+,h_1}, x_i^2 \in (-\Lambda_{+,h_2})$ for all $i \in [n]$. If there are at least α-fraction of $i \in [n]$ such that $\lambda_{\min,h_1}(x_i^1) \ge \rho$ and $\lambda_{\max,h_2}(x_i^2) \le -\rho$, then we have*

$$\Pr_{\epsilon\sim\{-1,1\}^n}\left[\exists j \in [2] : \ \lambda_{\max,h_j}\left(\sum_{i=1}^n \epsilon_i x_i^j - y_j\right) \in (-2\rho, 0]\right] \le O\left(\frac{1}{\alpha\sqrt{n}}\right).$$

Proof. For each $j \in [2]$, define two sets:

$$H_j := \left\{\epsilon \in \{-1,1\}^n : \lambda_{\max,h_1}\left(\sum_{i=1}^n \epsilon_i x_i^j\right) \le -2\rho\right\},$$
$$\overline{H_j} := \left\{\epsilon \in \{-1,1\}^n : \lambda_{\max,h_2}\left(\sum_{i=1}^n \epsilon_i x_i^j\right) \le 0\right\}.$$

Then, we have

$$\partial H_j := \overline{H_j} \backslash H_j = \left\{ \epsilon \in \{-1,1\}^n : \lambda_{\max,h_j}\left(\sum_{i=1}^n \epsilon_i x_i^j\right) \in (-2\rho, 0] \right\}.$$

Define $F := H_1 \cap H_2$ and $\partial F := (\overline{H_1} \cap \overline{H_2}) \backslash F$. Hence,

$$\partial F = \left\{ \epsilon \in \{-1,1\}^n : \exists j \in [2] \text{ s.t. } \lambda_{\max,h_j}\left(\sum_{i=1}^n \epsilon_i x_i^j\right) \in (-2\rho, 0] \right\}.$$

For any $\epsilon \in H_1$, consider its hypercube-neighbour ϵ' which flip the k-th coordinate of ϵ. If $\epsilon' \in \partial H_1$, then we have

$$\lambda_{\max,h_1}\left(\sum_{i=1}^n \epsilon_i x_i^1 - 2\epsilon_k x_k^1\right) \in (-2\rho, 0], \quad \lambda_{\max,h_1}\left(\sum_{i=1}^n \epsilon_i x_i^1\right) \le -2\rho.$$

It implies that $\epsilon_k = -1$. By the fact that $\lambda_{\max}(x+y) \ge \lambda_{\max}(x) + \lambda_{\min}(y)$, we have

$$\lambda_{\max,h_1}\left(\sum_{i=1}^n \epsilon_i x_i^1\right) + 2\lambda_{\min,h_1}(x_k^1) \le \lambda_{\max}\left(\sum_{i=1}^n \epsilon_i x_i^1 + 2x_k^1\right) \le 0,$$

which means $\lambda_{\min,h_1}(x_k^1) \le \rho$. However, we assume that there are α-fraction of $k \in [n]$ such that $\lambda_{\min,h_1}(x_k^1) \ge \rho$. Hence, H_1 is α-semi thin.

Similarly, for $\epsilon \in H_2$ and its hypercube-neighbor ϵ' with the k-th coordinate flipped, if $\epsilon' \in \partial H_2$, we have

$$\lambda_{\max,h_2}\left(\sum_{i=1}^n \epsilon_i x_i^2 - 2x_k^2\right) \in (-2\rho, 0], \quad \lambda_{\max,h_2}\left(\sum_{i=1}^n \epsilon_i x_i^2\right) \le -2\rho.$$

Hence,

$$\lambda_{\max,h_2}\left(\sum_{i=1}^n \epsilon_i x_i^2\right) + 2\lambda_{\min,h_2}(-x_k^2) = \lambda_{\max,h_2}\left(\sum_{i=1}^n \epsilon_i x_i^2\right) - 2\lambda_{\max,h_2}(x_k^2) \le 0,$$

which implies $\lambda_{\max,h_2}(x_k^2) \ge -\rho$. Then, by our assumption, H_2 is also α-semi thin.

Thus, by Theorem 7.18 in [OST19], we have

$$\mathrm{vol}(\partial F) \le O(1/(\alpha\sqrt{n})),$$

which implies the probability upper bound in the lemma. □

In order to satisfy the α-semi thin condition in Theorem 19.29, we use the following lemma using random hash function to bucket the vectors such that the resulting distribution will make the condition hold.

Lemma 19.30 (Lemma 46 in [AY22])**.** *Let $\tau \in (0, \frac{1}{100\sqrt{\log d}}]$. Let $\{x_i\}_{i\in[n]} \subset \mathbb{R}^m$ be a sequence of vectors in the hyperbolic cone Λ_+ of h such that*

$$\lambda_{\max}(x_i) \leq \tau, \quad \sum_{i=1}^n \lambda_{\min}(x_i)^2 \geq 1 \quad \forall i \in [n].$$

Let $p \geq \frac{1}{10\tau^2 \log d}$ and $\pi : [n] \to [p]$ be a random hash function that independently assigns each $i \in [n]$ to a uniformly random bucket in $[p]$. For each $c \in [p]$, define $\sigma_c := \sum_{i\in\pi^{-1}(c)} x_i$. And we say $c \in [p]$ is good if $\lambda_{\min}(\sigma_c) \geq \frac{1}{2\tau p}$.

Then, we have

$$\Pr\left[|\{c \in [p] : c \textit{ is good}\}| \leq \frac{4}{5}p\right] \leq \exp(-p/4).$$

Proof. Fix $c \in [p]$. Define indicator random variables $z_i \in \{0,1\}$ for $i \in [n]$ such that $z_i = 1$ if $\pi(i) = c$. Since π is a random hash function, we have $\Pr[z_i = 1] = \frac{1}{p}$. Then, consider the random vectors $\{z_i x_i\}_{i\in[n]}$. For each x_i, $\mathrm{supp}(z_i x_i) \in \Lambda_+$ and $\lambda_{\max}(z_i x_i) \leq \tau$. We note that $\sigma_c = \sum_{i=1}^n z_i x_i$, and

$$\begin{aligned}
\mu_{\min} = \sum_{i=1}^n \mathbb{E}[\lambda_{\min}(z_i x_i)] &= \frac{1}{m}\sum_{i=1}^n \lambda_{\min}(x_i) \\
&\geq \frac{1}{m}\sum_{i=1}^n \lambda_{\min}(x_i)^2 \cdot \frac{1}{\lambda_{\max}(x_i)} \\
&\geq \frac{1}{\tau m}.
\end{aligned}$$

Then, by Theorem 19.27 with $\delta = 1/2, \mu_{\min} = 1/(\tau p), R = \tau$, we have

$$\Pr\left[\lambda_{\min}\left(\sum_{i=1}^n z_i x_i\right) \leq \frac{1}{2\tau p}\right] \leq d \cdot (2/e)^{\frac{1}{2\tau^2 p}} \leq \frac{1}{10},$$

where the last step follows from $p \geq \frac{1}{10\tau^2 \log d}$. That is,

$$\Pr\left[\lambda_{\min}(\sigma_c) \geq \frac{1}{2\tau p}\right] \geq 1 - \frac{1}{10} = \frac{9}{10}. \tag{19.15}$$

Then, for all $c \in [p]$ and $i \in [n]$, define the indicator variables $B_{c,i} := \mathbf{1}[\pi(i) = c]$. Then, $\{B_{c,i}\}_{c\in[p],i\in[n]}$ are negatively associated by a balls and bins argument (see [MR95] for details). Now, the even that c is good can be represented by the indicator variables $G_c := \mathbf{1}[\lambda_{\min}(\sum_{i=1}^{n} B_{c,i}x_i) \geq \frac{1}{2\tau p}]$ for $c \in [p]$, which is constructed by applying a monotone non-decreasing function to $\{B_{c,i}\}_{i\in[n]}$. Hence, we know that $\{G_c\}_{c\in[p]}$ are also negatively associated. By Eq. (19.15), we have $\mathbb{E}[G_c] \geq \frac{9}{10}$. Thus, by the Chernoff bound for negatively associated random variables, we have

$$\Pr\left[\sum_{i=1}^{p} G_c \leq \frac{4}{5}p\right] \leq \exp(-p/4),$$

which completes the proof of the lemma. $\square$

Chapter 20: Hyperbolic Polynomials II: Discrepancy and Kadison-Singer-Type Results

20.1 Introduction

Introduced by [KS59], the Kadison-Singer problem was a long-standing open problem in mathematics. It was resolved by Marcus, Spielman, and Srivastrava in their seminal work [MSS15b]: For any set of independent random vectors $u_1, \cdots, u_n$ such that each u_i has finite support, and $u_1, \cdots, u_n$ are in isotropic positions in expectation, there is positive probability that $\sum_{i=1}^n u_i u_i^*$ has spectral norm bounded by $1 + O(\max_{i\in[n]} \|u_i\|)$. The main result of [MSS15b] is as follows:

Theorem 20.1 (Main result of [MSS15b])**.** *Let $\epsilon > 0$ and let $v_1, \cdots, v_n \in \mathbb{C}^m$ be n independent random vectors with finite support, such that $\mathbb{E}[\sum_{i=1}^n v_i v_i^*] = I$, and $\mathbb{E}[\|v_i\|^2] \le \epsilon$, $\forall i \in [n]$. Then*

$$\Pr\left[\Big\| \sum_{i\in[n]} v_i v_i^* \Big\| \le (1+\sqrt{\epsilon})^2\right] > 0.$$

The Kadison-Singer problem is closely related to discrepancy theory, which is an essential area in mathematics and theoretical computer science. A classical discrepancy problem is as follows: given n sets over n elements, can we color each element in red or blue such that each set has roughly the same number of elements in each color? More formally, for vectors $a_1, \ldots, a_n \in \mathbb{R}^n$ with $\|a_i\|_\infty \le 1$ and a coloring $s \in \{\pm 1\}^n$, the discrepancy is defined by $\mathrm{Disc}(a_1, \ldots, a_n; s) := \|\sum_{i\in[n]} s_i a_i\|_\infty$. The famous Spencer's Six Standard Deviations Suffice Theorem [Spe85] shows that there exists a coloring with discrepancy at most $6\sqrt{n}$, which beats the standard Chernoff bound showing that a random coloring has discrepancy $\sqrt{n \log n}$.

More generally, we can consider the "matrix version" of discrepancy: for matrices $A_1, \ldots, A_n \in \mathbb{R}^{d\times d}$ and a coloring $s \in \{\pm 1\}^n$,

$$\mathrm{Disc}(A_1, \ldots, A_n; s) := \Big\| \sum_{i\in[n]} s_i A_i \Big\|.$$

By the matrix Chernoff bound, it follows that for any symmetric matrix $A_1, \ldots, A_n \in \mathbb{R}^{d\times d}$ with $\|A_i\| \le 1$, for a uniformly random $s \sim \{-1, 1\}^n$,

$$\mathrm{Disc}(A_1, \ldots, A_n; s) \le O(\sqrt{n \cdot \log d})$$

with high probability. An important open question is, can we shave the $\log(d)$ factor for *some* choice of the signs?

Conjecture 20.2 (Matrix Spencer Conjecture, [Mek14])**.** For any symmetric matrices $X_1, \ldots, X_n \in \mathbb{R}^{d\times d}$ with $\|X_i\| \le 1$, there exist signs $r \in \{-1, 1\}^n$ such that $\|\sum_{i=1}^n r_i X_i\| = O(\sqrt{n})$.

We note that Theorem 20.1 is equivalent to the following discrepancy result for rank-1 matrices:

Theorem 20.3 ([MSS15b])**.** *Let* $u_1, \ldots, u_n \in \mathbb{C}^m$ *and suppose* $\max_{i\in[n]} \|u_i u_i^*\| \le \epsilon$ *and* $\sum_{i=1}^n u_i u_i^* = I$*. Then,*

$$\min_{s\in\{\pm 1\}^n} \mathrm{Disc}(u_1 u_1^*, \ldots, u_n u_n^*; s) \le O(\sqrt{\epsilon}).$$

In other words, the minimum discrepancy of rank-1 isotropic matrices is bounded by $O(\sqrt{\epsilon})$, where ϵ is the maximum spectral norm. This result also beats the matrix Chernoff bound [Tro15], which shows a discrepancy bound $O(\sqrt{\epsilon \log d})$. The main techniques in [MSS15b] are the method of interlacing polynomials and the barrier methods developed in [MSS15a].

Several generalizations of the Kadison-Singer-type results, which have interesting applications in theoretical computer science, have been established using the same technical framework as described in [MSS15b]. In particular, Kyng, Luh, and

Song [KLS20] provided a "four derivations suffice" version of Kadison-Singer conjecture: Instead of assuming every independent random vector has a bounded norm, the main result in [KLS20] only requires that the sum of the squared spectral norm is bounded by σ^2, and showed a discrepancy bound of 4σ:

Theorem 20.4 ([KLS20])**.** *Let $u_1, \ldots, u_n \in \mathbb{C}^m$ and $\sigma^2 = \| \sum_{i=1}^n (u_i u_i^*)^2 \|$. Then, we have*

$$\Pr_{\xi \sim \{\pm 1\}^n} \left[\Big\| \sum_{i=1}^n \xi_i u_i u_i^* \Big\| \leq 4\sigma \right] > 0.$$

This result was recently applied by [LZ20] to approximate solutions of generalized network design problems.

Moreover, Anari and Oveis-Gharan [AO14] generalized the Kadison-Singer conjecture into the setting of real-stable polynomials. Instead of assuming the random vectors are independent, [AO14] assumes that the vectors are sampled from any homogeneous strongly Rayleigh distribution with bounded marginal probability, have bounded norm, and are in an isotropic position:

Theorem 20.5 ([AO14])**.** *Let μ be a homogeneous strongly Rayleigh probability distribution on $[n]$ such that the marginal probability of each element is at most ϵ_1, and let $u_1, \cdots, u_n \in \mathbb{R}^m$ be vectors in an isotropic position, $\sum_{i=1}^n u_i u_i^* = I$, such that* $\max_{i \in [n]} \|u_i\|^2 \leq \epsilon_2$*. Then*

$$\Pr_{S \sim \mu} \left[\Big\| \sum_{i \in S} u_i u_i^* \Big\| \leq 4(\epsilon_1 + \epsilon_2) + 2(\epsilon_1 + \epsilon_2)^2 \right] > 0.$$

Theorem 20.5 has a direct analog in spectral graph theory: Given any (weighted) connected graph $G = (V, E)$ with Laplacian L_G. For any edge $e = (u, v) \in E$, define the vector corresponding to e as $v_e = L_G^{\dagger/2}(\mathbf{1}_u - \mathbf{1}_v)$ (here $L_G^{\dagger}$ is the Moore-Penrose inverse). Then the set of $\{v_e : e \in E\}$ are in isotropic position, and $\|v_e\|^2$ equals to the graph effective resistance with respect to e. Also, any spanning tree distribution of the edges in E is homogeneous strongly Rayleigh. It follows from Theorem 20.5 that any graph with bounded maximum effective resistance has a spectrally-thin spanning tree [AO14]. Moreover, [AO15] provided an exciting application to the asymmetric traveling salesman problem and obtained an $O(\log \log n)$-approximation.

Another perspective of generalizing the Kadison-Singer theorem is to study the discrepancy with respect to a more general norm than the spectral norm, which is the largest root of a determinant polynomial. A recent work by Bränden [Brä18] proved a high-rank version of Theorem 20.3 for *hyperbolic* polynomial, which is a larger class of polynomials including the determinant polynomial. Moreover, the hyperbolic norm on vectors is a natural generalization of the matrix spectral norm. (We will introduce hyperbolic polynomials in Section 20.1.1.) From this perspective, it is very natural to ask:

Can we extend more matrix discrepancy results (e.g., Theorem 20.4 and Theorem 20.5) to the hyperbolic polynomial setting?

20.1.1 Our results

Our results in this chapter falls into three classes:

1. In the Kadison-Singer (KS) setting, we show that it holds with respect to the hyperbolic norm for high-rank, sub-isotropic vectors in the hyperbolicity cone (Theorem 20.6), which slightly improves Brändén [Brä18]'s result.

2. In the Spencer setting, we propose a hyperbolic Spencer conjecture (Conjecture 20.9), and prove the constant-rank case (Theorem 20.8).

3. In the KS setting, we also generalize Theorem 20.4 and Theorem 20.5 to the hyperbolic norm. Theorem 20.10 removes the (sub-)isotropic condition in the rank-1 hyperbolic KS theorem and gives a tighter control for the hyperbolic deviation. Theorem 20.11 extends the rank-1 hyperbolic KS theorem to the Strongly-Rayleigh distributions.

Before stating our main results, we recall some basic notation of hyperbolic polynomials below (which was already introduced in Chapter 19).

Hyperbolic polynomials form a broader class of polynomials that encompasses determinant polynomials and homogeneous real-stable polynomials. An m-variate, degree-d homogeneous polynomial $h \in \mathbb{R}[x_1, \cdots, x_m]$ is *hyperbolic* with respect to a direction $e \in \mathbb{R}^m$ if the univariate polynomial $t \mapsto h(te - x)$ has only real roots for all $x \in \mathbb{R}^m$. (See Appendix 20.10 for some examples of hyperbolic/real-stable polynomials.) The set of $x \in \mathbb{R}^m$ such that all roots of $h(te - x)$ are non-negative (or strictly positive) is referred to as the hyperbolicity cone $\Gamma_+^h(e)$ (or $\Gamma_{++}^h(e)$). It is a widely recognized result [Brä10] that any vector x in the open hyperbolicity cone $\Gamma_{++}^h(e)$ is itself hyperbolic with respect to the polynomial h and have the same hyperbolicity cone as e, meaning that $\Gamma_{++}^h(e) = \Gamma_{++}^h(x)$. Therefore, the unique hyperbolicity cone of h can simply be expressed as Γ_+^h.

The hyperbolic polynomials have similarities to determinant polynomials of matrices, as they both can be used to define trace, norm, and eigenvalues. Given a hyperbolic polynomial $h \in \mathbb{R}[x_1, \cdots, x_m]$ and any vector $e \in \Gamma_{++}^h$, we can define a norm with respect to $h(x)$ and e as follows: for any $x \in \mathbb{R}^m$, its *hyperbolic norm* $\|x\|_h$ is equal to the largest root (in absolute value) of the linear restriction polynomial $h(te - x) \in \mathbb{R}[t]$. Similar to the eigenvalues of matrices, we define the *hyperbolic eigenvalues* of x to be the d roots of $h(te - x)$, denoted by $\lambda_1(x) \geq \cdots \geq \lambda_d(x)$. We can also define the *hyperbolic trace* and the *hyperbolic rank*:

$$tr_h[x] := \sum_{i=1}^{d} \lambda_i(x), \quad \text{and} \quad rank(x)_h := |\{i \in [d] : \lambda_i(x) \neq 0\}|.$$

High-rank hyperbolic KS Brändén [Brä18] proved a hyperbolic Kadison-Singer theorem, which generalizes Theorem 20.1 to the hyperbolic spectral norm and vectors with arbitrary rank. However, his result requires the vectors in the isotropic condition. Our result relaxes the isotropic condition to sub-isotropic:

Theorem 20.6 (Hyperbolic Kadison-Singer with sub-isotropic condition, informal)**.** *Let $k \geq 2$ be an integer and $\epsilon, \sigma > 0$. Suppose h is hyperbolic with respect to $e \in \mathbb{R}^m$, and let $x_1, \ldots, x_n$ be n vectors in the hyperbolic cone such that*

$$tr_h[x_i] \leq \epsilon \quad \forall i \in [n], \text{ and } \quad \Big\| \sum_{i=1}^{n} x_i \Big\|_h \leq \sigma. \tag{20.1}$$

where $tr_h[x] := \sum_{i=1}^{d} \lambda_i(x)$. Then, there exists a partition $S_1 \cup S_2 \cup \cdots \cup S_k = [n]$ such that for all $j \in [k]$,

$$\Big\| \sum_{i \in S_j} x_i \Big\|_h \leq \left(\sqrt{\epsilon} + \sqrt{\sigma/k}\right)^2.$$

Theorem 20.6 implies the high-rank case of [MSS15b] result (Theorem 20.1) without the isotropic condition. We note that there is a naive approach to relax the isotropic condition in [MSS15b, Brä18]'s results by adding several small dummy vectors to make the whole set in an isotropic position. (See [Gha15] for more details.) However, Theorem 20.6 is slightly better than this approach, since the naive approach will increase the number of vectors which results in a worse bound. (See Remark 20.4 for more details.)

Theorem 20.6 also implies the following hyperbolic discrepancy result:

Corollary 20.7 (Hyperbolic discrepancy for sub-isotropic vectors)**.** *Let $0 < \epsilon \leq \frac{1}{2}$. Suppose $h \in \mathbb{R}[z_1, \ldots, z_m]$ is hyperbolic with respect to $e \in \mathbb{R}^m$, and let $x_1, \ldots, x_n \in \Lambda_+(h, e)$ that satisfy Eq. (20.1). Then, there exist signs $r \in \{-1, 1\}^n$ such that*

$$\Big\| \sum_{i=1}^{n} r_i x_i \Big\|_h \leq 2\sqrt{\epsilon(2\sigma - \epsilon)}.$$

We note that this result is incomparable with [KLS20] due to the following reasons: 1) [KLS20] only works for rank-1 matrices while our result holds for arbitrary rank vectors in the hyperbolic cone; 2) the upper bound of [KLS20] depends on $\| \sum_{i=1}^{n} X_i^2 \|^{1/2}$ while our result depends on the hyperbolic trace and spectral norm of the sum of vectors.

Hyperbolic Spencer To obtain a hyperbolic discrepancy upper bound for arbitrary vectors (as in the case of Conjecture 20.2), we can apply hyperbolic Chernoff bound (Theorem 19.1) and get the following discrepancy result which holds with high probability:

Theorem 20.8. *Let h be a degree-d hyperbolic polynomial with respect to $e \in \mathbb{R}^m$. We are given vectors $x_1, x_2, \cdots, x_n \in \mathbb{R}^m$ such that $\|x_i\|_h \le 1$ and rank$(x_i) \le s$ for all $i \in [n]$ and some $s \in \mathbb{N}_+$. Then for uniformly random signs $r \sim \{-1, 1\}^n$,*

$$\left\| \sum_{i=1}^{n} r_i x_i \right\|_h \le O(\sqrt{n \log(s+1)})$$

holds with probability at least 0.99.

This result may not be tight when the ranks of the input vectors are large. It is thus interesting to study whether we can do better to improve the $\sqrt{\log d}$ factor in the non-constructive case. We thus conjecture the following hyperbolic discrepancy bound:

Conjecture 20.9 (Hyperbolic Spencer Conjecture)**.** We are given vectors $x_1, x_2, \cdots, x_n \in \mathbb{R}^m$ and a degree d hyperbolic polynomial $h \in \mathbb{R}[z_1, \ldots, z_m]$ with respect to $e \in \mathbb{R}^m$, where $\|x_i\|_h \le 1$ for all $i \in [n]$. Then, there exist signs $r \in \{-1, 1\}^n$, such that

$$\left\| \sum_{i=1}^{n} r_i x_i \right\|_h \le O(\sqrt{n}).$$

Note that Conjecture 20.9 is more general than the matrix Spencer conjecture (Conjecture 20.2). And for constant degree d or constant maximum rank s, this conjecture is true by Theorem 20.8.

Rank-1 hyperbolic KS generalizations We first generalize the result of [KLS20] to the hyperbolic setting:

Theorem 20.10 (Hyperbolic version of [KLS20] (Theorem 1.4), informal statement of Theorem 20.43)**.** *Let $h \in \mathbb{R}[x_1, \dots, x_m]$ denote a hyperbolic polynomial in direction $e \in \mathbb{R}^m$. Let $v_1, \dots, v_n \in \Gamma_+^h$ be n vectors in the closed hyperbolicity cone. Let $\xi_1, \dots, \xi_n$ be n independent random variables with finite supports and $\mathbb{E}[\xi_i] = \mu_i$ and* $\mathbf{Var}[\xi_i] = \tau_i^2$*. Suppose $\sigma := \| \sum_{i=1}^n \tau_i^2 tr_h[v_i] v_i \|_h$. Then there exists an assignment $(s_1, \dots, s_n)$ with s_i in the support of ξ_i for all $i \in [n]$, such that*

$$\Big\| \sum_{i=1}^n (s_i - \mu_i) v_i \Big\|_h \leq 4\sigma.$$

We remark that Theorem 20.10 does not require the isotropic position condition of $v_1, \cdots, v_n$ as in [Brä18]. In addition, we only need the sum of $tr_h[v_i]v_i$'s hyperbolic norm to be bounded, while [Brä18]'s result requires each vector's trace to be bounded individually.

We would also like to note that the class of hyperbolic polynomials is much broader than that of determinant polynomials, which were used in the original Kadison-Singer-type theorems. Lax conjectured in [Lax57] that every 3-variate hyperbolic/real-stable polynomial could be represented as a determinant polynomial, this was later resolved in [HV07, LPR05]. However, the Lax conjecture is false when the number of variables exceeds 3, as demonstrated in [Brä11, BVY14] with counterexamples of hyperbolic/real-stable polynomials $h(x)$ for which even $(h(x))^k$ cannot be represented by determinant polynomials for any $k > 0$.

Our second result considers the setting where the random vectors are not independent, but instead, sampled from a strongly Rayleigh distribution. We say a distribution μ over the subsets of $[n]$ is *strongly Rayleigh* if its generating polynomial $g_\mu(z) := \sum_{S \subseteq [n]} \mu(S) z^S \in \mathbb{R}[z_1, \dots, z_n]$ is a *real-stable polynomial*, which means $g_\mu(z)$ does not have any root in the upper-half of the complex plane, i.e., $g_\mu(z) \neq 0$ for any $z \in \mathbb{C}^n$ with $\Re(z) \succ 0$.

Theorem 20.11 (Hyperbolic version of [AO14] (Theorem 1.2), informal statement of Theorem 20.54)**.** *Let $h \in \mathbb{R}[x_1, \dots, x_m]$ denote hyperbolic polynomial in direction*

$e \in \mathbb{R}^m$. Let μ be a homogeneous strongly Rayleigh probability distribution on $[n]$ such that the marginal probability of each element is at most ϵ_1.

Suppose $v_1, \cdots, v_n \in \Gamma_+^h$ are in the hyperbolicity cone of h such that $\sum_{i=1}^n v_i = e$, and for all $i \in [n]$, $\|v_i\|_h \leq \epsilon_2$. Then there exists $S \subseteq [n]$ in the support of μ, such that

$$\Big\| \sum_{i \in S} v_i \Big\|_h \leq 4(\epsilon_1 + \epsilon_2) + 2(\epsilon_1 + \epsilon_2)^2.$$

It is worth mentioning that the previous paper [KLS20, AO14] focused on the determinant polynomial, leaving the question of whether their techniques could be extended to the hyperbolic/real-stable setting unresolved. In our paper, we address this gap by developing new techniques specifically tailored to hyperbolic polynomials.

In addition, we follow the results from [AOSS18] and give an algorithm that can find the approximate solutions of both Theorem 20.10 and Theorem 20.11 in time sub-exponential to m:

Proposition 20.12 (Sub-exponential algorithm for Theorem 20.10, informal statement of Corollary 20.64)**.** *Let $h \in \mathbb{R}[x_1, \ldots, x_m]$ denote a hyperbolic polynomial with direction $e \in \mathbb{R}^m$. Let $v_1, \ldots, v_n \in \Gamma_+^h$ be n vectors in the hyperbolicity cone Γ_+^h of h. Suppose $\sigma = \| \sum_{i=1}^n tr_h[v_i] v_i \|_h$.*

Let $\mathcal{P}$ be the interlacing family used in the proof of Theorem 20.11. Then there exists an sub-exponential time algorithm KadisonSinger$(\delta, \mathcal{P})$*, such that for any $\delta > 0$, it returns a sign assignment $(s_1, \cdots, s_n) \in \{\pm 1\}^n$ satisfying*

$$\Big\| \sum_{i=1}^n s_i u_i \Big\|_h \leq 4(1+\delta)\sigma.$$

Proposition 20.13 (Sub-Exponential algorithm for Theorem 20.11, informal statement of Corollary 20.68)**.** *Let $h \in \mathbb{R}[x_1, \ldots, x_m]$ denote a hyperbolic polynomial in direction $e \in \mathbb{R}^m$. Let μ be a homogeneous strongly Rayleigh probability distribution on $[n]$ such that the marginal probability of each element is at most ϵ_1, and let $v_1, \cdots, v_n \in \Gamma_+^h$ be n vectors such that $\sum_{i=1}^n v_i = e$, and for all $i \in [n]$, $\|v_i\|_h \leq \epsilon_2$.*

Let $\mathcal{Q}$ be the interlacing family used in the proof of Theorem 20.11. Then there exists an sub-exponential time algorithm KadisonSinger$(\delta, \mathcal{Q})$*, such that for any $\delta > 0$, it returns a set S in the support of μ satisfying*

$$\Big\| \sum_{i \in S} u_i \Big\|_h \leq (1+\delta) \cdot \big(4(\epsilon_1 + \epsilon_2) + 2(\epsilon_1 + \epsilon_2)^2\big) .$$

20.2 Related work

Real-Stable Polynomials Real-stability is an important property for multivariate polynomials. In [BB09], the authors used the real-stability to give a unified framework for Lee-Yang type problems in statistical mechanics and combinatorics. Real-stable polynomials are also related to the permanent. Gurvits [Gur07] proved the Van der Waerden conjecture, which conjectures that the permanent of n-by-n doubly stochastic matrices are lower-bounded by $n!/n^n$, via the capacity of real-stable polynomials. Recently, [GL21] improved the capacity lower bound for real-stable polynomials, which has applications in matrix scaling and metric TSP. In addition, real-stable polynomials are an important tool in solving many counting and sampling problems [NS16, AOR16, AO17, SV17, AOSS16, AMOV18, AOV18, ALOV19, ALO$^+$21].

Hyperbolic Polynomials Hyperbolic polynomial was originally defined to study the stability of partial differential equations [Går51, Hor83, Kry95]. In theoretical computer science, Güler [Gül97] first introduced hyperbolic polynomial for optimization (hyperbolic programming), which is a generalization of LP and SDP. Later, a few algorithms [Ren06, MT14, RS14, Ren16, NP18, Ren19a] were designed for hyperbolic programming. On the other hand, a significant effort has been put into the equivalence between hyperbolic programming and SDP, which is closely related to the "Generalized Lax Conjecture" (which conjectures that every hyperbolicity cone is spectrahedral) and its variants [HV07, LPR05, Brä14, KPV15, Sau18, Ami19, RRSW19].

Strongly Rayleigh Distribution The strongly Rayleigh distribution was introduced by [BBL09]. The authors also proved numerous basic properties of strongly Rayleigh distributions, including negative association, and closure property under operations such as conditioning, product, and restriction to a subset. [PP14] proved a concentration result for Lipschitz functions of strongly Rayleigh variables. [KS18] showed a matrix concentration for strongly Rayleigh random variables, which implies that adding a small number of uniformly random spanning trees gives a graph spectral sparsifier.

Strongly Rayleigh distribution also has many algorithmic applications. [AOR16] exploited the negative dependence property of homogeneous strongly Rayleigh distributions, and designed efficient algorithms for generating approximate samples from Determinantal Point Process using Monte Carlo Markov Chain. The strongly Rayleigh property of spanning tree distribution is a key component for improving the approximation ratios of TSP [KKO20, KKO21] and k-edge connected graph problem [KKOZ21].

Other generalizations of the Kadison-Singer-type results The upper bound of the rank-one Kadison-Singer theorem was improved by [BCMS19, RL20]. [AB20] further extended [RL20]'s result to prove a real-stable version of Anderson's paving conjecture. However, they used a different norm for real-stable polynomials, and hence their results and ours are incomparable. In the high-rank case, [Coh16a] also proved a Kadison-Singer result for high-rank matrices.

Another direction of generalizing the Kadison-Singer-type result is to relax the $\{+1,-1\}$-coloring to $\{0,1\}$-coloring, which is called the one-sided version of Kadison-Singer problem in [Wea04]. More specifically, given n isotropic vectors $v_1,\dots,v_n \in \mathbb{R}^m$ with norm $\frac{1}{\sqrt{N}}$, the goal is to find a subset $S \subset [n]$ of size k such that $\|\sum_{i\in S} v_i v_i^\top\| \le \frac{k}{n} + O(1/\sqrt{N})$. Unlike the original Kadison-Singer problem, Weaver [Wea04] showed that this problem can be solved in polynomial time. Very recently, Song, Xu and

Zhang [SXZ22] improved the time complexity of the algorithm via an efficient inner product search data structure.

Applications of Kadison-Singer Problem There are many interesting results developed from the Kadison-Singer theorem. In spectral graph theory, [HO14] exploited the same proof technique of interlacing families to show a sufficient condition of the spectrally thin tree conjecture. [AO14] used the strongly-Rayleigh extension of Kadison-Singer theorem to show a weaker sufficient condition. Based on this result, [AO15] showed that any k-edge-connected graph has an $O(\frac{\log\log(n)}{k})$-thin tree, and gave a $poly(\log\log(n))$-integrality gap of the asymmetric TSP. [MSS18, Coh16b] used the Kadison-Singer theorem to construct bipartite Ramanujan graphs of all sizes and degrees. In the network design problem, [LZ20] exploited the result in [KLS20], and built a spectral rounding algorithm for the general network design convex program, which has applications in weighted experimental design, spectral network design, and additive spectral sparsifier.

Discrepancy theory For theoretical results, very recently, [HRS21, DJR21] proved some special cases of the matrix Spencer conjecture. For algorithmic results, Bansal [Ban10] proposed the first constructive version of partial coloring for discrepancy minimization. Based on this work, more approaches [LM15, Raz17, LRR17, ES18, BDGL18, DNTTJ18] were discovered in recent years. For applications of the discrepancy theory, we refer to the excellent book by Matousek [Mat09].

20.3 Proof Overview

20.3.1 Hyperbolic discrepancy for high-rank vectors

In this section, we sketch our techniques for improving the result of [Brä18] and proving the hyperbolic Spencer conjecture for constant-rank case.

To relax the isotropic condition in [Brä18], we basically follow their proof.

The high-level idea is to construct a compatible family of polynomials[1] such that the probability in the hyperbolic Kadison-Singer problem (Theorem 20.6) can be upper-bounded by the largest root of the expected polynomial of the family, which can be further upper-bounded by the largest root of the mixed hyperbolic polynomial $h[v_1, \ldots, v_n] \in \mathbb{R}[x_1, \ldots, x_m, y_1, \ldots, y_n]$, defined as $h[v_1, \ldots, v_n] := \prod_{i=1}^m (1 - y_i D_{v_i}) h(x)$, where D_{v_i} is the directional derivative with respect to v_i. In particular, we can consider the roots of the linear restriction $h[v_1, \ldots, v_n](te + \mathbf{1}) \in \mathbb{R}[t]$. Then, using Gårding's result [Går59] on hyperbolic cone, we know that the largest root equals the minimum $\rho > 0$ such that the vector $\rho e + \mathbf{1}$ is in the hyperbolic cone Γ_+ of $h[v_1, \ldots, v_n]$, which can be upper-bounded via similar techniques in [MSS15b, KLS20] to iteratively add each vector v_i while keeping the sum in the hyperbolic cone. Our key observation is that the proof in [Brä18] essentially proved that

$$\frac{\epsilon \mu e + \left(1 - \frac{1}{n}\right) \delta \sum_{i=1}^n v_i}{1 + \frac{\mu - 1}{n}} + \mathbf{1} \in \Gamma_+$$

holds for any vectors $v_i \in \Lambda_+$. Hence, once we assume that $\| \sum_{i=1}^n v_i \|_h \leq \sigma$, then by the convexity of the hyperbolic cone, we get that $\rho \leq \frac{\left(\epsilon\mu + \left(1 - \frac{1}{n}\right)\delta\sigma\right)}{1 + \frac{\mu - 1}{n}}$, which will imply the upper bound in Theorem 20.6. We defer the formal proof in Section 20.5.2.

To obtain the discrepancy result for arbitrary vectors (Theorem 20.8), we can use the hyperbolic Chernoff bound for Rademacher sum (Theorem 19.1) to derive the discrepancy upper bound. For any vectors $x_1, \ldots, x_n$ with maximum rank s, by setting $t = O(\sigma\sqrt{\log s})$ in Theorem 19.1, we get that $\|\sum_{i=1}^n r_i x_i\|_h \leq O(\sigma\sqrt{\log s})$ holds with high probability for uniformly random signs $r \sim \{\pm 1\}^n$.

20.3.2 Hyperbolic deviations

In this section, we will sketch the proof of our hyperbolic generalization of the Kadison-Singer theorem (Theorem 20.10). We will use the same strategy as

[1]The compatible family of polynomials is closely related to the interlacing family in [MSS15b, MSS18]. See Definition 20.12.

the original Kadison-Singer theorem (Theorem 20.1) in [MSS15a, MSS15b], following three main technical steps.

For simplicity, we assume that the random variables $\xi_1, \ldots, \xi_n \in \{\pm 1\}$ are independent Rademacher random variables, i.e., $\Pr[\xi_i = 1] = \frac{1}{2}$ and $\Pr[\xi_i = -1] = \frac{1}{2}$ for all $i \in [n]$.

To generalize the Kadison-Singer statement into the hyperbolic norm, one main obstacle is to define the variance of the hyperbolic norm of the sum of random vectors $\sum_{i=1}^n \xi_i v_i$. In the determinant polynomial case, each v_i corresponds to a rank-1 matrix $u_i u_i^*$, and it is easy to see that the variance of the spectral norm is $\| \sum_{i=1}^n (u_i u_i^*)^2 \|$. However, there is no analog of "matrix square" in the setting of hyperbolic/real-stable polynomials. Instead, we define the *hyperbolic variance*:

$$\left\| \sum_{i=1}^n tr_h[v_i] v_i \right\|_h$$

in terms of the hyperbolic trace, and show that *four hyperbolic deviations suffice.*

Defining interlacing family of characteristic polynomials. In the first step, we construct a family of *characteristic polynomials* $\{p_s : s \in \{\pm 1\}^t, t \in \{0, \cdots, n\}\}$ as follows: For each $\mathbf{s} \in \{\pm 1\}^n$, define the leaf-node-polynomial:

$$p_{\mathbf{s}}(x) := \left(\prod_{i=1}^n p_{i,s_i} \right) \cdot h \left(xe + \sum_{i=1}^n s_i v_i \right) \cdot h \left(xe - \sum_{i=1}^n s_i v_i \right),$$

and for all $\ell \in \{0, \ldots, n-1\}$, $\mathbf{s}' \in \{\pm 1\}^\ell$, we construct an inner node with a polynomial that corresponds to the bit-string $\mathbf{s}'$:

$$p_{\mathbf{s}'}(x) := \sum_{\mathbf{t} \in \{\pm 1\}^{n-\ell}} p_{(\mathbf{s}', \mathbf{t})}(x).$$

where $(\mathbf{s}', \mathbf{t}) \in \{\pm 1\}^n$ is the bit-string concatenated by $\mathbf{s}'$ and $\mathbf{t}$.

We will then show that the above family of characteristic polynomials forms an *interlacing family* (see Lemma 20.47 for detail) . By basic properties of interlacing family, we can always find a leaf-root-polynomial p_s (where $s \in \{\pm 1\}^n$) whose largest root is upper bounded by the largest root of the top-most polynomial.

$$p_\emptyset(x) = \mathbb{E}_{\xi_1,\cdots,\xi_n}\left[h\Big(xe + \sum_{i=1}^n \xi_i v_i\Big) \cdot h\Big(xe - \sum_{i=1}^n \xi_i v_i\Big)\right].$$

(we call $p_\emptyset$ to be the *mixed characteristic polynomial*). Notice that by rewriting the largest root of p_s to be the expected hyperbolic norm of $\sum_{i=1}^n s_i v_i$, we get that

$$\lambda_{\max}(p_\emptyset) = \left\|\sum_{i=1}^n s_i v_i\right\|_h . \tag{20.2}$$

(See Corollary 20.48 for a formal statement).

Also, we will take $s \in \{\pm 1\}^n$ as the corresponding sign assignment in the main theorem (Theorem 20.10) It then suffices to upper-bound the largest root of the mixed characteristic polynomial.

From mixed characteristic polynomial to multivariate polynomial. In the second step, we will show that the mixed characteristic polynomial that takes the average on n random variables

$$p_\emptyset(x) = \mathbb{E}_{\xi_1,\ldots,\xi_n}\left[h\Big(xe + \sum_{i=1}^n \xi_i v_i\Big) \cdot h\Big(xe - \sum_{i=1}^n \xi_i v_i\Big)\right]$$

is equivalent to a polynomial with n extra variables $z_1, \cdots, z_n$:

$$\prod_{i=1}^n \Big(1 - \frac{1}{2}\frac{\partial^2}{\partial z_i^2}\Big)\bigg|_{z=0} \left(h\Big(xe + \sum_{i=1}^n z_i v_i\Big)\right)^2 . \tag{20.3}$$

(See Lemma 20.49 for more detail.) Thus, we can reduce the upper bound of $\chi_{\max}(p_\emptyset)$ to an upper bound of the largest root in (20.3). The latter turns out to be easier to estimate with the help of a barrier argument [MSS15b].

To show such equivalence holds, we use induction on the random variables $\xi_1, \ldots, \xi_n$. More specifically, we start from ξ_1 and are conditioned on any fixed choice of $\xi_2, \ldots, \xi_n$. We prove that taking expectation over ξ_1 is equivalent to applying the operator $(1 - \frac{\partial^2}{\partial z_1^2})$ to the polynomial

$$\left(h(xe + z_1 v_1 + \sum_{i=2}^{n} \xi_i v_i)\right)^2$$

and setting $z_1 = 0$. Here we use the relation between expectation and the second derivatives: for any Rademacher random variable ξ,

$$\mathbb{E}_\xi[h(x_1 - \xi v) \cdot h(x_2 + \xi v)] = \left(1 - \frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}t^2}\right)\Bigg|_{t=0} h(x_1 + tv)h(x_2 + tv).$$

Repeating this process and removing one random variable at a time. After n iterations, we obtain the desired multivariate polynomial.

We also need to prove the real-rootedness of the multivariate polynomial (Eqn. (20.3)). We first consider an easy case where h itself is a real-stable polynomial, as in the determinant polynomial case. Then the real-rootedness easily follows from the closure properties of the real-stable polynomial (see Fact 20.17) . More specifically, we can show that $(h(xe + \sum_{i=1}^n z_i v_i))^2$ is also a real-stable polynomial. Furthermore, applying the operators $(1 - \frac{1}{2}\frac{\partial^2}{\partial z_i^2})$ and restricting $z = 0$ preserve the real-stability. Therefore, the multivariate polynomial is a univariate real-stable polynomial, which is equivalent to being real-rooted.

Next, we show that when h is a hyperbolic polynomial, the multivariate polynomial (Eqn. (20.3)) is also real-rooted. our approach is to show that the linear restriction of h: $h(xe + \sum_{i=1}^n z_i v_i)$ is a real-stable polynomial in $\mathbb{R}[x, z_1, \ldots, z_n]$. A well-known test for real-stability is that if for any $a \in \mathbb{R}^{n+1}_{>0}, b \in \mathbb{R}^{n+1}$, the one-dimensional restriction $p(at + b) \in \mathbb{R}[t]$ is non-zero and real-rooted, then $p(x)$ is real-stable. We test $h(xe + \sum_{i=1}^n z_i v_i)$ by restricting to $at + b$, and get the following polynomial:

$$h\Big((a_1 e + \sum_{i=1}^{n} a_{i+1} v_i)t + y\Big) \in \mathbb{R}[t],$$

where y is a fixed vector depending on b. Since $a_i > 0$ for all $i \in [n+1]$ and $e, v_1, \dots, v_n$ are vectors in the hyperbolicity cone, it implies that the vector $a_1 e + \sum_{i=1}^n a_{i+1} v_i$ is also in the hyperbolicity cone. Then, by the definition of hyperbolic polynomial, we immediately see that $h((a_1 e + \sum_{i=1}^n a_{i+1} v_i)t + y)$ is real-rooted for any $a \in \mathbb{R}_{>0}^{n+1}$ and $b \in \mathbb{R}^{n+1}$. Hence, we can conclude that the restricted hyperbolic polynomial $h(xe + \sum_{i=1}^n z_i v_i)$ is real-stable and the remaining proof is the same as the real-stable case.

Applying barrier argument. Finally, we use *barrier argument* to find an "upper barrier vector" whose components lie above any roots of multivariate polynomial can take. In particular, we consider the multivariate polynomial $P(x, z) = (h(xe + \sum_{i=1}^n z_i v_i))^2$. Define the *barrier function* of any variable $i \in [n]$ as the following:

$$\Phi_P^i(\alpha(t), -\delta) = \frac{\partial_{z_i} P(x, z)}{P(x, z)}\Big|_{x=\alpha(t), z=-\delta},$$

where $\delta \in \mathbb{R}^n$ where $\delta_i = t tr_h[v_i]$ for $i \in [n]$ and $\alpha(t) > t$ is a parameter that depends on t.

As a warm-up, consider the case when $\sigma = 1$ and assuming $\| \sum_{i=1}^n tr_h[v_i] v_i \|_h \leq 1$. It is easy to show that $(\alpha(t), -\delta)$ is an upper barrier of P, from the linearity of the hyperbolic eigenvalues and the assumption. Next, we upper-bound the barrier function's value at $(\alpha(t), -\delta)$. When h is a determinant polynomial, this step is easy because the derivative of $\log\det$ is the trace of the matrix. For a general hyperbolic polynomial, we will rewrite the partial derivative ∂_{z_i} as a directional derivative D_{v_i} and get

$$\Phi_P^i(\alpha(t), -\delta) = 2 \cdot \frac{(D_{v_i} h)(\alpha e - te + t(e - \sum_{j=1}^n tr_h[v_j] v_j))}{h(\alpha e - te + t(e - \sum_{j=1}^n tr_h[v_j] v_j))}.$$

We observe that our assumption $\| \sum_{i=1}^n tr_h[v_i] v_i \|_h \leq 1$ implies that $e - \sum_{j=1}^n tr_h[v_j] v_j \in \Gamma_+^h$. By the concavity of the function $\frac{h(x)}{D_{v_i} h(x)}$ in the hyperbolicity cone, we can prove that

$$\Phi_P^i(\alpha(t), -\delta) \leq \frac{2 tr_h[v_i]}{\alpha(t) - t}.$$

Now, we can apply the barrier update lemma in [KLS20] (Lemma 20.26) with $\alpha(t) = 2t = 4$ to show that

$$\Phi^j_{(1-\frac{1}{2}\partial^2_{z_i})P}(4, -\delta + \delta_i \mathbf{1}_i) \leq \Phi^j_P(4, -\delta).$$

In other words, the partial differential operator $(1 - \frac{1}{2}\partial^2_{z_i})$ shifts the upper-barrier by $(0, \cdots, 0, \delta_i, 0, \cdots, 0)$. Using induction for the variables $\delta_1, \cdots, \delta_n$, we can finally finally get an upper-barrier of

$$(4, -\delta + \sum_{i=1}^n \delta_i \mathbf{1}_i) = (4, 0, \ldots, 0),$$

which implies that $(4, 0, \ldots, 0)$ is above the roots of

$$\prod_{i=1}^n \Big(1 - \frac{1}{2}\frac{\partial^2}{\partial z_i^2}\Big)\left(h\Big(xe + \sum_{i=1}^n z_i \tau_i v_i\Big)\right)^2 \tag{20.4}$$

(See Lemma 20.53 for detail).

A challenge in this process is ensuring that the barrier function remains non-negative. To achieve this, we use the multidimensional convexity of the hyperbolic barrier function as established in [Tao13] (Lemma 20.27) . For cases where $\sigma \neq 1$, this requirement is satisfied through a simple scaling argument.

Combining the above three steps together, we can prove that $\Pr_{\xi_1, \cdots, \xi_n}[\| \sum_{i=1}^n \xi_i v_i \|_h \leq 4\sigma] > 0$ for vectors $v_1, \ldots, v_n$ in the hyperbolicity cone with $\| \sum_{i=1}^n tr_h[v_i] v_i \|_h = \sigma^2$. The complete proof can be found in Section 20.7.

20.3.3 Generalization to strongly Rayleigh distributions

Our main technical contribution to Theorem 20.11 is a more universal and structured method to characterize the mixed characteristic polynomial. Define the mixed characteristic polynomial as

$$q_S(x) = \mu(S) \cdot h\left(xe - \sum_{i \in S} v_i\right). \tag{20.5}$$

we want to show that it is equivalent to the restricted multivariate polynomial:

$$\prod_{i=1}^{n}(1-\frac{1}{2}\frac{\partial^2}{\partial z_i^2})\Big(h(xe+\sum_{i=1}^{n}z_iv_i)g_\mu(x\mathbf{1}+z)\Big)\Big|_{z=0}\in\mathbb{R}[x,z_1,\cdots,z_n]. \tag{20.6}$$

Although Eqn. (20.5) and Eqn. (20.6) are the hyperbolic generalization of [AO14], we are unable to apply the previous techniques. This is because [AO14] computes the mixed characteristic polynomial explicitly, which heavily relies on the fact that the characteristic polynomial is a determinant. It is unclear how to generalize this method to hyperbolic/real-stable characteristic polynomials.

The key step in [AO14] is to show the following equality between mixed characteristic polynomial and multivariate polynomial:

$$\begin{aligned}&x^{d_\mu-d}\cdot\mathbb{E}_{S\sim\mu}\left[\det\left(x^2I-\sum_{i\in S}2v_iv_i^\top\right)\right]\\&=\prod_{i=1}^{n}(1-\partial_{z_i}^2)\left(g_\mu(x\mathbf{1}+z)\cdot\det(xI+\sum_{i=1}^{n}z_iv_iv_i^\top)\right)\Bigg|_{z=\mathbf{0}}\end{aligned}$$

where d_μ is the degree of the homogeneous strongly-Rayleigh distribution μ (i.e. the degree of g_μ), and m is the dimension of v_i.

Then they expand the right-hand side to get:

$$\begin{aligned}\text{RHS}=&\sum_{k=0}^{m}(-1)^kx^{d_\mu+m-2k}\sum_{S\in\binom{[n]}{k}}\Pr_{T\sim\mu}[S\subseteq T]\cdot\sigma_k(\sum_{i\in S}2v_iv_i^\top)\\=&\,x^{d_\mu-m}\cdot\mathbb{E}_{S\sim\mu}\left[\det\left(x^2I-\sum_{i\in S}2v_iv_i^\top\right)\right]=\text{LHS},\end{aligned}$$

where $\sigma_k(M)$ equals to the sum of all $k\times k$ principal minors of $M\in\mathbb{R}^{m\times m}$. The first step comes from expanding the product $\prod_{i=1}^{n}(1-\partial^2z_i)$, and the second step comes from that

$$\det(x^2I-\sum_{i=1}^{n}v_iv_i^\top)=\sum_{k=0}^{m}(-1)^{2k}x^{2m-2k}\sum_{S\in\binom{[n]}{k}}\sigma_k(\sum_{i\in S}v_iv_i^\top).$$

The naive generalization of a technique to hyperbolic/real-stable polynomial h faces challenges. One such challenge is the absence of an explicit form for h, unlike in the case of $h = \det$ where the determinant can be expressed as a combination of minors. This lack of a well-defined minor presents difficulty in rewriting the hyperbolic/real-stable polynomial. To tackle this issue, we devised a new and structured proof that relies on induction, offering a novel solution to this problem.

Inductive step. We first rewrite the expectation over the Strongly-Rayleigh distribution $T \sim \mu$ as follows:

$$x^{d_\mu} \cdot 2^{-n} \cdot \mathbb{E}_{T\sim\mu}[h(xe - \sum_{i\in T} v_i)] = \frac{1}{2}\mathbb{E}_{\xi_2,\cdots,\xi_n\sim\{0,1\}^{n-1}}\Big[(1-\partial_{z_1})h(x_2+z_1v_1)x\partial_{z_1}g_2(x+z_1) + h(x_2)(1-x\partial_{z_1})g_2(x+z_1)\Big|_{z_1=0}\Big]$$

where g_2 is defined as

$$g_2(t) := x^{\sum_{i=2}^n \xi_i} \cdot \prod_{i=2}^n \Big(\xi_i\partial_{z_i} + (1-\xi_i)(1-x\partial_{z_i})\Big) g_\mu(t, x+z_2, x+z_3, \cdots, x+z_n)\Big|_{z_2,\ldots,z_n=0}$$

and $x_2 = x^2e - \sum_{i=2}^n \xi_i v_i$. The main observation is that the marginals of a homogeneous Strongly-Rayleigh distribution can be computed from the derivatives of its generating polynomial (Fact 20.55) .

Then, we can expand the term inside the expectation as

$$(1-\frac{x}{2}\partial_{z_1}^2)\Big(h(x_2+z_1v_1)g_2(x+z_1)\Big)\Big|_{z_1=0},$$

using the fact that $rank(v_1)_h \le 1$ and the degree of $g_2(t)$ is at most 1.

Hence, we obtain our inductive step as

$$\begin{aligned} & x^{d_\mu} \cdot 2^{-n} \cdot \mathbb{E}_{\xi\sim\mu}\left[h(xe - \sum_{i=1}^n \xi_i v_i)\right] \\ = \; & \frac{1}{2}(1-\frac{x}{2}\partial_{z_1}^2)\left(\mathbb{E}_{\xi_2,\cdots,\xi_n}\left[h(xe - \sum_{i=2}^n \xi_i v_i + z_1 v_1)\cdot g_2(x+z_1)\right]\right)\Bigg|_{z_1=0}. \end{aligned}$$

Applying the step inductively. Repeating the above process for n times, we finally get

$$x^{d_\mu} \cdot \mathbb{E}_{\xi\sim\mu}\left[h(x^2 e - (\sum_{i=1}^n \xi_i v_i))\right] = \sum_{T\subseteq[n]} (-\frac{x}{2})^{|T|}\partial_{z^T}^2 \Big(h(x^2 e + \sum_{i=1}^n z_i v_i) g_\mu(x\mathbf{1}+z)\Big)\Big|_{z=0}.$$

Then, we rewrite the partial derivatives as directional derivatives (see Definition 20.20 for detail) . For any subset $T \subseteq [n]$ of size k, we have

$$\begin{aligned}&(-\frac{x}{2})^k \partial_{z^T}^2 \Big(h(x^2 e + \sum_{i=1}^n z_i v_i) g_\mu(x\mathbf{1}+z)\Big)\Big|_{z=0}\\ =\ &(-\frac{x}{2})^k \cdot 2^k \cdot \left(\prod_{i\in T} D_{v_i}\right) h(x^2 e)\cdot g_\mu^{(T)}(x\mathbf{1}),\end{aligned}$$

where $g_\mu^{(T)}(x\mathbf{1}) = \prod_{i\in T}\partial_{z_i} g_\mu(x\mathbf{1}+z)\Big|_{z=0}$. And by the homogeneity of h, it further equals to

$$x^d \cdot (-\frac{1}{2})^k \partial_{z^T}^2 \left(h(xe + \sum_{i=1}^n z_i v_i) g_\mu(x\mathbf{1}+z)\right)\Bigg|_{z=0}.$$

Therefore, we prove the following formula that relates the characteristic polynomial under SR distribution to the multivariate polynomial:

$$x^{d_\mu} \cdot \mathbb{E}_{\xi\sim\mu}\left[h(x^2 e - (\sum_{i=1}^n \xi_i v_i))\right] = x^d \cdot \prod_{i=1}^n (1 - \frac{1}{2}\partial_{z_i}^2)\Big(h(xe + \sum_{i=1}^n z_i v_i) g_\mu(x\mathbf{1}+z)\Big)\Big|_{z=0}.$$

The complete proof can be found in Section 20.8.

Roadmap. We provide preliminary definitions and facts in Section 20.4. We prove a relaxed version of [Brä18]'s result in Section 20.5. We prove a special case of the hyperbolic Spencer conjecture in Section 20.6. In Section 20.7, we prove our first main result (Theorem 20.10), which is a hyperbolic generalization of Kadison-Singer result for a weaker condition (sum of squares of vectors is bounded). In Section 20.8, we prove our second main result (Theorem 20.11), which is a hyperbolic extension of the Kadison-Singer result for strongly-Rayleigh distributions. We put the sub-exponential algorithm for our main results in Section 20.9. In Section 20.10, we provide some examples of real-stable and hyperbolic polynomials.

20.4 Preliminaries

We gather several basic linear algebraic and analytic facts in the following subsections.

Notations. For any positive integer n, we use $[n]$ to denote set $\{1, 2, \cdots, n\}$. We use $\mathbf{1}$ to denote the all-one vector and $\mathbf{1}_i$ to denote the vector with one in the i-th coordinate and zero in other coordinates.

20.4.1 Real-stable polynomials

Definition 20.1. A multivariate polynomial $p \in \mathbb{C}[x_1, \cdots, x_m]$ is *stable* if it has no zeros in the region $\{(x_1, \ldots, x_m) : \mathrm{Im}(x_i) > 0 \text{ for all } i \in [m]\}$. p is *real stable* if p is stable and has real coefficients.

In the rest of this chapter, we restrict our discussion into polynomials with real coefficients.

Fact 20.14. *We say a univariate polynomial $p \in \mathbb{R}[t]$ is real-rooted iff it is real-stable.*

Fact 20.15 (Equivalent definition of real-stable polynomial)**.** *A multivariate polynomial $p \in \mathbb{R}[x_1, \cdots, x_m]$ is real stable iff for any $a \in \mathbb{R}^m_{>0}$ and $b \in \mathbb{R}^m$, the univariate polynomial $p(at + b)$ with respect to t is not identically zero and is real-rooted.*

Lemma 20.16 (Proposition 2.4, [BB08])**.** *If $A_1, \ldots, A_n$ are positive semidefinite symmetric matrices, then the polynomial*

$$\det\left(\sum_{i=1}^{n} z_i A_i\right)$$

is real stable.

We also need that real stability is preserved under product (see [BBL09]), restricting variables to real values (see [Wag11, Lemma 2.4(d)]), and taking $(1-\partial_{x_i}^2)$ (see [AO14, Corollary 2.8]).

Fact 20.17 (Closure operations of real-stable polynomials)**.** *Let $p, q \in \mathbb{R}[x_1, \dots, x_m]$ be two real stable polynomials. Then the following operations preserve real-stability:*

- **(Product)** $p \cdot q$.
- **(Restriction to real values)** *For any $a \in \mathbb{R}$, $p|_{x_1=a} = p(a, x_2, \dots, x_m) \in \mathbb{R}[x_2, \dots, x_m]$.*
- **(One minus second partial derivative)** *For any $c \in \mathbb{R}_+, i \in [n]$, $(1 - c \cdot \partial_{x_i}^2)p(x_1, \dots, x_n)$.*

20.4.2 Hyperbolic polynomials

Hyperbolic polynomial and its hyperbolicity cone have been defined in Chapter 19. For convenience, we recall their definitions here.

Definition 20.2 (Hyperbolic polynomials)**.** A homogeneous polynomial $h \in \mathbb{R}[x_1, \cdots, x_m]$ is hyperbolic with respect to vector $e \in \mathbb{R}^m$ with $h(e) > 0$, if for all $x \in \mathbb{R}^m$, the univariate polynomial $t \mapsto h(te - x)$ only has real roots.

Furthermore, if h has degree d, fix any $x \in \mathbb{R}^m$ we can write

$$h(te - x) = h(e) \prod_{i=1}^{d} (t - \lambda_i(x)),$$

where $\lambda_1(x) \geq \lambda_2(x) \geq \cdots \geq \lambda_d(x)$ are the real roots of the univariate polynomial $h(te - x)$. In particular,

$$h(x) = h(e) \prod_{i=1}^{d} \lambda_i(x).$$

We denote $\lambda_i(x)$ as the i-th eigenvalue of x.

Definition 20.3 (hyperbolicity cone)**.** Let $h \in \mathbb{R}[x_1, \cdots, x_m]$ be a degree d hyperbolic polynomial with respect to $e \in \mathbb{R}^m$. For any $x \in \mathbb{R}^m$, let $\lambda_1(x) \geq \cdots \geq \lambda_d(x) \in \mathbb{R}^d$ be the real roots of $h(te - x)$. Define the hyperbolicity cone of h as

$$\Gamma_{++}^h := \{x \ : \ \lambda_d(x) > 0\}.$$

Furthermore, define the closure of Γ_+^h as

$$\Gamma_+^h := \{x \ : \ \lambda_d(x) \geq 0\}.$$

Fact 20.18. *For any $e \in \mathbb{R}^m_{>0}$ and any homogeneous real-stable polynomial $h \in \mathbb{R}[x_1, \cdots, x_m]$, we have h is hyperbolic with respect to e. In other words, $\mathbb{R}^m_{>0} \subseteq \Gamma_+^h$.*

Proof. For any $e \in \mathbb{R}^m_{>0}$ and for any $x \in \mathbb{R}^m$, by Fact 20.15 (with replacing a by e and b by x), the uni-variate polynomial $h(te - x)$ is real rooted. By Definition 20.2 h is hyperbolic with respect to direction e. $\square$

The hyperbolic trace, rank, and spectral norm are defined in Definition 19.5.

Fact 20.19 (Hyperbolic norm in terms of largest root of characteristic polynomial)**.** *Let $h \in \mathbb{R}[x_1, \cdots, x_m]$ be a degree d hyperbolic polynomial with respect to $e \in \mathbb{R}^m$. For any $v_1, \cdots v_m \in \Gamma_+^h$ and any $s_1, \cdots, s_m \in \mathbb{R}$, we have*

$$\left\| \sum_{i=1}^m s_i v_i \right\|_h = \lambda_{\max} \left(h \left(xe - \sum_{i=1}^n s_i v_i \right) \cdot h \left(xe + \sum_{i=1}^n s_i v_i \right) \right)$$

where $\lambda_{\max}(f(x))$ is the maximum root of $f(x)$.

Proof. For any vector $v \in \mathbb{R}^m$, it is easy to see that the i-th largest eigenvalue $\lambda_i(v) = -\lambda_{d-i+1}(-v)$. Then, we have

$$\begin{aligned} \lambda_{\max} \left(h \left(xe - \sum_{i=1}^n s_i v_i \right) \cdot h \left(xe + \sum_{i=1}^n s_i v_i \right) \right) &= \max \left\{ \lambda_1 \left(\sum_{i=1}^m s_i v_i \right), \lambda_1 \left(- \sum_{i=1}^m s_i v_i \right) \right\} \\ &= \max \left\{ \lambda_1 \left(\sum_{i=1}^m s_i v_i \right), -\lambda_d \left(\sum_{i=1}^m s_i v_i \right) \right\} \\ &= \left\| \sum_{i=1}^m s_i v_i \right\|_h . \end{aligned}$$

$\square$

Remark 20.1. It is useful to think of hyperbolic polynomials as generalizations of the determinant polynomials. Let $X \in S^n(\mathbb{R})$ be a symmetric matrix. Define $h : S^n(\mathbb{R}) \mapsto \mathbb{R}$ as

$$h(X) = \det(X).$$

Then, h is hyperbolic with respect to the identity matrix $I_n \in S^n(\mathbb{R})$, since for all $X \in S^n(\mathbb{R})$, the roots of $h(tI - X)$ are the eigenvalues of X, thus $h(tI - X)$ is real-rooted.

The basic concepts for hyperbolic polynomials in Definition 20.3 and Definition 19.5 also have analogues in linear algebra. To illustrate them, let h be the determinant polynomial:

- The hyperbolicity cone of h is

$$\Gamma_+^h = \{X \in \mathbb{R}^{n\times n} \ : \ \lambda_n(X) > 0\} = \{X \in \mathbb{R}^{n\times n} \ : \ X \succ 0\}.$$

- For all $X \in S^n(\mathbb{R})$, the hyperbolic trace of X is

$$tr_h[X] = \sum_{i=1}^n \lambda_i(X) = tr[X].$$

- For all $X \in S^n(\mathbb{R})$, the hyperbolic rank of X is

$$rank_h(X) = |\{i \ : \ \lambda_i(X) \neq 0\}| = rank(X).$$

- For all $X \in S^n(\mathbb{R})$, the hyperbolic spectral norm of X is

$$\|X\|_h = \max_{i\in[n]} |\lambda_i(X)| = \|X\|.$$

  here $\|X\|$ denotes the spectral norm of X.

Definition 20.4 (Directional derivative)**.** Given a polynomial $h \in \mathbb{R}[x_1, \cdots, x_m]$ and a vector $v \in \mathbb{R}^m$, define the directional derivative of h in direction v as

$$D_v h := \sum_{i=1}^{n} v_i \frac{\partial h}{\partial x_i}.$$

Fact 20.20 (Equivalent definition of directional derivative)**.** *Given polynomial $h \in \mathbb{R}[x_1, \cdots, x_m]$ and vectors $x, v \in \mathbb{R}^m$,*

$$(D_v h)(x) = \frac{\mathrm{d}}{\mathrm{d}t} h(x + tv), \tag{20.7}$$

$$(D_v^2 h)(x) = \frac{\mathrm{d}^2}{\mathrm{d}t^2} h(x + tv). \tag{20.8}$$

Fact 20.21 (Directional derivative of hyperbolic polynomials)**.** *Let $h \in \mathbb{R}[x_1, \cdots, x_m]$ denote a hyperbolic polynomial with respect to $e \in \mathbb{R}^m$. Let $v_1, \ldots, v_n \in \mathbb{R}^m$ be n vectors such that $\forall i \in [n]$, $\mathrm{rank}_h(v_i) \leq 1$. Then, for any $t \in \mathbb{R}$,*

$$(\prod_{i=1}^{m} D_{v_i} h)(te) = (\prod_{i=1}^{n} \partial_{z_i} h)(te + \sum_{i=1}^{n} z_i v_i).$$

Moreover, for any $T \subseteq [m]$, we have

$$(\prod_{i \in T} D_{v_i} h)(te) = (\prod_{i \in T} \partial_{z_i} h)(te + \sum_{i=1}^{n} z_i v_i)\Big|_{z=0}.$$

Fact 20.22 (First-order expansion of hyperbolic polynomial)**.** *Let $h \in \mathbb{R}[x_1, \cdots, x_m]$ be any hyperbolic polynomial. For any vectors $v_1, \ldots, v_n \in \mathbb{R}^m$ such that $\forall i \in [n]$, $\mathrm{rank}_h(v_i) \leq 1$, and for any real vector $x \in \mathbb{R}^m$,*

$$h(x \pm \sum_{i=1}^{n} v_i) = \prod_{i=1}^{n} (1 \pm \partial_{z_i}) h(x + \sum_{i=1}^{n} z_i v_i)\Bigg|_{z=0}.$$

Furthermore, for any $S \subseteq [n]$,

$$h(x \pm \sum_{i \in S} v_i) = \prod_{i \in S} (1 \pm \partial_{z_i}) h(x + \sum_{i=1}^{n} z_i v_i)\Bigg|_{z=0}.$$

Proof. Prove by induction on n.

If $n = 1$, we have

$$h(x \pm v_1) = (1 \pm D_{v_1})h(x) = (1 \pm \partial_{z_1})h(x + z_1 v_1)\Big|_{z_1=0},$$

which follows from $rank_h(v_1) \leq 1$.

Assume it holds for $n = k$.

When $n = k + 1$, let $x' = x \pm \sum_{i=1}^{k} v_i$. We have

$$\begin{aligned} h(x \pm \sum_{i=1}^{k+1} v_i) &= h(x' \pm v_{k+1}) \\ &= (1 \pm \partial_{z_{k+1}})h(x' + z_{k+1}v_{k+1})\Big|_{z_{k+1}=0} \\ &= (1 \pm \partial_{z_{k+1}})h(x + z_{k+1}v_{k+1} \pm \sum_{i=1}^{k} v_i)\Big|_{z_{k+1}=0} \\ &= (1 \pm \partial_{z_{k+1}}) \prod_{i=1}^{k}(1 \pm \partial_{z_i})h(x + \sum_{i=1}^{k+1} z_i v_i)\Big|_{z=0} \\ &= \prod_{i=1}^{k+1}(1 \pm \partial_{z_i})h(x + \sum_{i=1}^{k+1} z_i v_i)\Big|_{z=0}, \end{aligned}$$

where the second step follows from $rank_h(v_{k+1}) \leq 1$, the forth step follows from the induction hypothesis, and the last step follows from the operators $(1 \pm \partial_{z_i})$ and $(1 \pm \partial_{z_j})$ commute for $i \neq j \in [k + 1]$.

Hence, the fact is proved. And for the furthermore part, it follows from the remaining variables z_i for $i \notin S$ will disappear when we set $z = 0$. $\square$

20.4.3 Interlacing families

We recall the definition and properties of interlacing families from [MSS15a].

Definition 20.5 (Interlacing polynomials and common interlacing)**.** We say a real rooted polynomial $g(x) = C\prod_{i=1}^{n-1}(x-\alpha_i)$ *interlaces* the real rooted polynomial $f(x) =$

$C' \prod_{i=1}^{n}(x - \beta_i)$ if

$$\beta_1 \leq \alpha_1 \leq \cdots \leq \alpha_{n-1} \leq \beta_n.$$

We say the polynomials $f_1, \ldots, f_k$ have a *common interlacing* if there is a polynomial g that interlaces each of the f_i.

The following lemma relates the roots of a sum of polynomials to those of a common interlacing.

Lemma 20.23 (Lemma 4.2, [MSS15a]). *Let $f_1, \ldots, f_k$ be degree d real rooted polynomials with positive leading coefficients. Define*

$$f_\emptyset := \sum_{i=1}^{k} f_i.$$

If $f_1, \ldots, f_k$ have a common interlacing, then there exists an $i \in [k]$ such that the largest root of f_i is at most the largest root of $f_\emptyset$.

Definition 20.6 (Definition 4.3, [MSS15a]). Let $S_1, \ldots, S_n$ be finite sets. For each choice of assignment $(s_1, \ldots, s_n) \in S_1 \times \cdots \times S_n$, let $f_{s_1,\ldots,s_n}(x)$ be a real rooted degree d polynomial with positive leading coefficient. For a partial assignment $s_1, \ldots, s_k \in S_1 \times \cdots \times S_k$ for $k < n$, we define

$$f_{s_1,\ldots,s_k} := \sum_{s_{k+1} \in S_{k+1}, \ldots, s_n \in S_n} f_{s_1,\ldots,s_k,s_{k+1},\ldots,s_n}. \tag{20.9}$$

Note that this is compatible with our definition of $f_\emptyset$ from Lemma 20.23. We say that the polynomials $\{f_{s_1,\ldots,s_n}\}$ form an *interlacing family* if for all $k = 0, \ldots, n-1$ and all $(s_1, \ldots, s_k) \in S_1 \times \cdots \times S_k$, the polynomials

$$\left\{ f_{s_1,\cdots,s_k,t} : \ t \in S_{k+1} \right\}$$

have a common interlacing.

The following lemma relates the roots of the interlacing family to those of $f_\emptyset$.

Lemma 20.24 (Theorem 4.4, [MSS15a])**.** *Let $S_1, \dots, S_n$ be finite sets and let $\{f_{s_1,\dots,s_n}\}$ be an interlacing family. Then there exists some $(s_1, \dots, s_n) \in S_1 \times \dots \times S_n$ so that the largest root of $f_{s_1,\dots,s_n}$ is upper bounded by the largest root of $f_\emptyset$.*

Finally, we recall a relationship between real-rootedness and common interlacings which has been discovered independently several times [DG94, Fel80, CS07].

Lemma 20.25 ([DG94, Fel80, CS07])**.** *Let $f_1, \dots, f_k$ be univariate polynomials of the same degree with positive leading coefficient. Then $f_1, \dots, f_k$ have a common interlacing if and only if $\sum_{i=1}^k \alpha_i f_i$ is real rooted for all convex combinations α_i, $\sum_{i=1}^k \alpha_i = 1$.*

20.4.4 Barrier method

Definition 20.7 (Upper barrier of the roots of a polynomial)**.** For a multivariate polynomial $p(z_1, \dots, z_n)$, we say $z \in \mathbb{R}^n$ is above all roots of p if for all $t \in \mathbb{R}_+^n$,

$$p(z+t) > 0.$$

We use $\mathbf{Ab}_p$ to denote the set of points which are above all roots of p.

We use the barrier function as in [MSS15b] and [KLS20].

Definition 20.8 (Barrier function)**.** For a multivariate polynomial p and $z \in \mathbf{Ab}_p$, the barrier function of p in direction i at z is

$$\Phi_p^i := \frac{\partial_{z_i} p(z)}{p(z)}.$$

We will make use of the following lemma that controls the deviation of the roots after applying a second order differential operator. This lemma is a slight variation of Lemma 5.3 in [KLS20].

Lemma 20.26 (Lemma 5.3 of [KLS20])**.** *Suppose that $p(z_1, \cdots, z_m)$ is real stable and $z \in \mathbf{Ab}_p$. For any $c \in [0,1]$ and $i \in [m]$, if*

$$\Phi_p^i(z) < \sqrt{1/c}, \tag{20.10}$$

then $z \in \mathbf{Ab}_{(1-c\cdot\partial_{z_i}^2)p}$. If additionally for $\delta > 0$,

$$c \cdot \left(\frac{2}{\delta}\Phi_p^i(z) + (\Phi_p^i(z))^2\right) \leq 1, \tag{20.11}$$

then, for all $j \in [m]$,

$$\Phi^j_{(1-c\cdot\partial_i^2)p}(z + \delta \mathbf{1}_i) \leq \Phi_p^j(z).$$

Remark 20.2. We choose $c = 1/2$ when we use the above lemma in Section 20.7.4 and Section 20.8.4.

Lemma 20.27 (Multi-dimensional convexity, [Tao13])**.** *Let $p(z_1, \ldots, z_m)$ be a real stable polynomial of m variables. For any $i \in [m]$,*

$$(-1)^k \frac{\partial^k}{\partial z_j^k} \Phi_p^i(x) \geq 0$$

for all $k = 0, 1, 2, \ldots$ and $x \in \mathbf{Ab}_p$.

20.5 High-Rank Hyperbolic Kadison-Singer with Sub-Isotropic Condition

In this section, we will show how to relax the isotropic condition in the hyperbolic Kadison-Singer theorem [Brä18].

20.5.1 Formal Statements of Previous Matrix Discrepancy Results

In this section, we formally state some matrix discrepancy results. We first recall the discrepancy theorem implied by the Kadison-Singer theorem.

Theorem 20.28 (A restatement of Theorem 20.3, [MSS15b])**.** *Let $x_1, \ldots, x_n \in \mathbb{C}^m$ and suppose $\|x_i x_i^*\| \leq \epsilon$ for all $i \in [n]$ and $\sum_{i=1}^n x_i x_i^* = I$. Then, there exist signs $r \in \{-1, 1\}^n$ such that*

$$\left\| \sum_{i=1}^n r_i x_i x_i^* \right\| \leq O(\sqrt{\epsilon}).$$

This theorem also holds for high rank matrices as long as the isotropic condition holds:

Theorem 20.29 (High rank Kadison-Singer [Coh16a, Brä18])**.** *Let $X_1, \ldots, X_n \in \mathbb{C}^{d \times d}$ be positive semi-definite symmetric matrices such that $tr[X_i] \leq \epsilon$ for all $i \in [n]$ and $\sum_{i=1}^n X_i = I$. Then, there exist signs $r \in \{-1, 1\}^n$ such that*

$$\left\| \sum_{i=1}^n r_i X_i \right\| \leq O(\sqrt{\epsilon}).$$

20.5.2 Hyperbolic Kadison-Singer with relaxed condition

The goal of this section is to prove Theorem 20.31, which relaxes the isotropic condition in Corollary 20.7 to the bounded hyperbolic norm.

We first formally state the upper bound in [Brä18]:

Definition 20.9. For $r \in \mathbb{N}_+$, let U_r be the set of all pairs $(\delta, \mu) \in \mathbb{R}_+ \times \mathbb{R}_+$ such that

$$\delta - 1 \geq \frac{\delta}{\mu} \cdot \frac{\left(1 + \frac{\delta}{r\mu}\right)^{r-1} - \left(\frac{\delta}{r\mu}\right)^{r-1}}{\left(1 + \frac{\delta}{r\mu}\right)^{r} - \left(\frac{\delta}{r\mu}\right)^{r}},$$

and either $\mu > 1$, or $\delta \in [1, 2], \mu > 1 - \delta/r$.

Then, the upper bound in [Brä18] is:

$$\delta(\epsilon, n, r) := \inf_{(\delta, \mu) \in U_r} \frac{\epsilon\mu + (1 - \frac{1}{n})\delta}{1 + \frac{\mu - 1}{n}}.$$

In particular, $\delta(\epsilon, \infty, r) := \inf_{(\delta, \mu) \in U_r} \epsilon\mu + \delta$.

Theorem 20.30 ([Brä18]). *Let $k \geq 2$ be an integer and ϵ a positive real number. Suppose h is hyperbolic with respect to $e \in \mathbb{R}^m$, and let $x_1, \dots, x_n \in \Lambda_+(h, e)$ be such that*

$$tr_h[x_i] \leq \epsilon, \quad rank(x_i) \leq r \quad \forall i \in [n], \text{ and } \quad \sum_{i=1}^{n} x_i = e.$$

Then there is a partition $S_1 \cup S_2 \cup \cdots \cup S_k = [n]$ such that for all $j \in [k]$,

$$\left\| \sum_{i \in S_j} x_i \right\|_h \leq \frac{1}{k} \cdot \delta \left(k\epsilon, n, rk \right).$$

The high-level idea of proving Theorem 20.30 is similar to [MSS15b]. We can show that this discrepancy upper-bound can be obtained by rounding a compatible family of polynomials, which is a generalization of the interlacing family defined in [MSS15b]. Then, this rounding problem is further equivalent to upper-bound the largest root of a mixed hyperbolic polynomial (Definition 20.11), which is achieved by proving a structural result about the hyperbolic cone of the mixed hyperbolic polynomial.

Following this approach, we slightly generalize Theorem 20.30 by relaxing the isotropic condition:

Theorem 20.31. *Let $k \geq 2$ be an integer and $\epsilon, \sigma > 0$. Suppose $h \in \mathbb{R}[z_1, \dots, z_m]$ is hyperbolic with respect to $e \in \mathbb{R}^m$, and let $x_1, \dots, x_n$ be n vectors in the hyperbolic cone $\Lambda_+(h, e)$ (see Definition 19.4) such that*

$$tr_h[x_i] \leq \epsilon, \quad rank(x_i) \leq r \quad \forall i \in [n], \text{ and } \quad \left\| \sum_{i=1}^{n} x_i \right\|_h \leq \sigma.$$

Then, there exists a partition $S_1 \cup S_2 \cup \cdots \cup S_k = [n]$ such that for all $j \in [k]$,

$$\left\| \sum_{i \in S_j} x_i \right\|_h \leq \frac{\sigma}{k} \cdot \delta \left(\frac{k\epsilon}{\sigma}, n, rk \right).$$

Remark 20.3. By Eq. (1.7) in [Brä18], the above bound is at most

$$\frac{\sigma}{k} \cdot \delta(k\epsilon/\sigma, \infty, \infty) = \frac{\sigma}{k} \cdot (1 + \sqrt{k\epsilon/\sigma})^2 = \left(\sqrt{\epsilon} + \sqrt{\sigma/k}\right)^2,$$

which also generalizes the result of [MSS15b] (Theorem 20.1) to hyperbolic polynomials with sub-isotropic condition.

Remark 20.4. We note that a naive approach to relax the isotropic condition is to add some dummy vectors and then apply Theorem 20.30. However, to satisfy the condition that each vector has trace at most ϵ, the number of dummy vector can be $O(n/\epsilon)$ in the worst case. Then, this approach results in an upper bound of $\frac{\sigma}{k} \cdot \delta\left(\frac{k\epsilon}{\sigma}, O(n/\epsilon), rk\right)$. By the property of the δ function, we know that this bound is worse than ours in Theorem 20.31.

The proof of Theorem 20.31 is almost the same as the proof of Theorem 1.3 in [Brä18], but relies on the sub-isotropic version of the following theorem. Therefore, we will only prove Theorem 20.32.

Theorem 20.32 (Sub-isotropic version of Theorem 6.1 in [Brä18])**.** *Suppose $h \in \mathbb{R}[z_1, \ldots, z_m]$ is a hyperbolic polynomial with respect to $e \in \mathbb{R}^m$. Let $\mathsf{x}_1, \ldots, \mathsf{x}_m$ be independent random vectors in $\Lambda_+(e)$ with finite supports such that*

$$tr_h[\mathbb{E}[\mathsf{x}_i]] \le \epsilon, \quad rank(\mathbb{E}[\mathsf{x}_i]) \le r \quad \forall i \in [n], \text{ and } \quad \left\|\sum_{i=1}^n \mathbb{E}[\mathsf{x}_i]\right\|_h \le \sigma.$$

Then, we have

$$\Pr\left[\lambda_{\max}\left(\sum_{i=1}^n \mathsf{x}_\mathsf{i}\right) \le \sigma \cdot \delta(\epsilon/\sigma, n, r)\right] > 0.$$

Proof. Let V_i be the support of x_i for $i \in [n]$. By Theorem 20.38, the family $\{h[v_1, \ldots, v_m](t\overline{e} + \underline{\mathbf{1}})\}_{v_i \in V_i}$ is compatible, where $t\overline{e} + \underline{\mathbf{1}} = \begin{bmatrix} te \\ \mathbf{1} \end{bmatrix} \in \mathbb{R}^{n+m}$

By Theorem 20.37, there exists $(v_1^*, \ldots, v_n^*) \in V_1 \times \cdots \times V_n$ with nonzero probability, such that the largest root of $h[v_1^*, \ldots, v_n^*](t\overline{e} + \underline{\mathbf{1}})$ is at most the largest root of $\mathbb{E}[h[\mathsf{x}_1, \ldots, \mathsf{x}_n]]$.

By Fact 20.36, $\mathbb{E}[h[\mathsf{x}_1,\ldots,\mathsf{x}_n]] = h[\mathbb{E}[\mathsf{x}_1],\ldots,\mathbb{E}[\mathsf{x}_n]]$. Let $\lambda_{\max}(v_1,\ldots,v_n)$ denote the largest root of $h[v_1,\ldots,v_m](t\overline{e}+\underline{\mathbf{1}})$. Then, we have

$$\lambda_{\max}(\mathbb{E}[\mathsf{x}_1],\ldots,\mathbb{E}[\mathsf{x}_n]) \geq \ \lambda_{\max}(v_1^*,\ldots,v_n^*) \geq \ \lambda_{\max}(v_1^*+\cdots+v_n^*),$$

where the second step follows from Theorem 20.39.

It is easy to verify that $\mathbb{E}[\mathsf{x}_1],\ldots,\mathbb{E}[\mathsf{x}_n]$ satisfy the conditions in Theorem 20.41. Thus, by Theorem 20.41, we get that

$$\lambda_{\max}(v_1^*+\cdots+v_n^*) \leq \ \lambda_{\max}(\mathbb{E}[\mathsf{x}_1],\ldots,\mathbb{E}[\mathsf{x}_n]) \leq \ \sigma\cdot\delta(\epsilon/\sigma,n,r),$$

which completes the proof. □

Similar to Corollary 20.7, Theorem 20.31 also implies the following discrepancy result for vectors in sub-isotropic position.

Corollary 20.33. *Let $0<\epsilon\leq\frac{1}{2}$. Suppose $h\in\mathbb{R}[z_1,\ldots,z_m]$ is hyperbolic with respect to $e\in\mathbb{R}^m$, and let $x_1,\ldots,x_n\in\Lambda_+(h,e)$ that satisfy*

$$tr_h[x_i]\leq\epsilon, \ \textit{and} \quad \left\|\sum_{i=1}^n x_i\right\|_h \leq\sigma.$$

Then, there exist signs $r\in\{-1,1\}^n$ such that

$$\left\|\sum_{i=1}^n r_i x_i\right\|_h \leq 2\sqrt{\epsilon(2\sigma-\epsilon)}.$$

Proof. By Theorem 20.31 with $k=2$ and the upper bound in Remark 20.3, there exists a set $S\subseteq[n]$ such that

$$\Big\|\sum_{i\in S} x_i\Big\|_h \leq (\sqrt{\epsilon}+\sqrt{\sigma/2})^2, \ \text{ and } \ \Big\|\sum_{i\notin S} x_i\Big\|_h \leq (\sqrt{\epsilon}+\sqrt{\sigma/2})^2.$$

Since we know that $\|\sum_{i=1}^n x_i\|_h\leq\sigma$, we get that

$$\Big\|\sum_{i\in S}-\sum_{i\notin S} x_i\Big\|_h \leq \sigma-2(\sqrt{\epsilon}+\sqrt{\sigma/2})^2 = 2\sqrt{\epsilon(2\sigma-\epsilon)}.$$

By assigning $r_i=1$ for $i\in S$ and $r_i=-1$ for $i\notin S$, we complete the proof of the corollary. □

20.5.3 Technical tools in previous work

In this section, we provide some necessary definitions and technical tools we used in [Brä18].

Definition 20.10 (Directional derivative)**.** Let $h \in \mathbb{R}[x_1, \ldots, x_m]$. The directional derivative of $h(x)$ with respect to $v \in \mathbb{R}^m$ is defined as

$$D_v h(x) := \sum_{i=1}^{m} v_i \cdot \frac{\partial h}{\partial x_i}(x).$$

The following fact shows the relation between directional derivative and the usual derivative.

Fact 20.34. *For any polynomial $h(x)$ and any vector $v \in \mathbb{R}^m$, we have*

$$D_v h(x + tv) = \frac{\mathrm{d}}{\mathrm{d}t} h(x + tv).$$

If h is a hyperbolic polynomial, then the directional derivative is related to the hyperbolic trace:

Fact 20.35. *If h is hyperbolic with respect to $e \in \mathbb{R}^m$, then for any $v \in \mathbb{R}^m$, we have*

$$tr_h[v] = \frac{D_v h(e)}{h(e)}.$$

Definition 20.11 (Mixed hyperbolic polynomial)**.** If $h(x) \in \mathbb{R}[x_1, \ldots, x_m]$ is a hyperbolic polynomial with respect to $e \in \mathbb{R}^m$, and $v_1, \ldots, v_n \in \Lambda_+$, then the mixed hyperbolic polynomial $h[v_1, \ldots, v_m] \in \mathbb{R}[x_1, \ldots, x_m, y_1, \ldots, y_n]$ is defined as

$$h[v_1, \ldots, v_n] := \prod_{i=1}^{m} (1 - y_i D_{v_i}) h(x).$$

Brändén [Brä18] proved that $h[v_1, \ldots, v_n]$ is also hyperbolic with the hyperbolic cone containing $\Lambda_{++} \times \mathbb{R}^n_{\leq 0}$. In our proof, we will also use the following fact, which can be easily proved by showing that $h[v_1, \ldots, v_n]$ is affine linear in each coordinate.

Fact 20.36. *Let $\mathsf{x}_1, \ldots, \mathsf{x}_n$ be independent random variables in $\mathbb{R}^m$. Then,*

$$\mathbb{E}[h[\mathsf{x}_1, \ldots, \mathsf{x}_n]] = h[\mathbb{E}[\mathsf{x}_1], \ldots, \mathbb{E}[\mathsf{x}_n]].$$

Brändén [Brä18] also defined the compatible family of polynomials, which is a sub-class of interlacing family of polynomials in [MSS15b, MSS18].

Definition 20.12 (Compatible family of polynomials)**.** Let $S_1, \ldots, S_n$ be finite sets. A family of polynomials

$$\mathcal{F} = \{f(S;t)\}_{S \in S_1 \times \cdots \times S_n} \subset \mathbb{R}[t]$$

is called compatible if the following properties hold:

- all the nonzero members of F have the same degree and the same signs of their leading coefficients, and
- for all choices of independent random variables $\mathsf{x}_1 \in S_1, \ldots, \mathsf{x}_n \in S_n$, the polynomial

  $$\mathbb{E}[f(\mathsf{x}_1, \ldots, \mathsf{x}_n; t)]$$

  is real-rooted.

The following theorem characterizes the largest root of the expectation polynomial in the compatible family, which is very similar to the result for interlacing family [MSS15b].

Theorem 20.37 (Theorem 2.3 in [Brä18])**.** *Let $\{f(S;t)\}_{S \in S_1 \times \cdots \times S_n}$ be a compatible family, and let $\mathsf{x}_1 \in S_1, \ldots, \mathsf{x}_n \in S_n$ be independent random variables such that $\mathbb{E}[f(\mathsf{x}_1, \ldots, \mathsf{x}_n)] \not\equiv 0$.*

Then there is a tuple $S = (s_1, \ldots, s_n) \in S_1 \times \cdots \times S_n$, with $\Pr[\mathsf{x}_i = s_i] > 0$ for all $i \in [n]$, such that the largest root of $f(s_1, \cdots, s_n; t)$ is smaller or equal to the largest root of $\mathbb{E}[f(\mathsf{x}_1, \cdots, \mathsf{x}_n; t)]$.

The theorem below shows that mixed hyperbolic polynomials form a compatible family.

Theorem 20.38 (Theorem 3.5 in [Brä18]). *Let $h(x)$ be hyperbolic with respect to $e \in \mathbb{R}^m$, and let $V_1, \dots, V_n$ be finite sets of vectors in Λ_+. Let $w \in \mathbb{R}^{m+n}$. For $V = (v_1, \dots, v_n) \in V_1 \times \cdots \times V_n$, define*

$$f(V;t) := h[v_1, \dots, v_n](t\overline{e} + w),$$

where $\overline{e} := \begin{bmatrix} e \\ 0 \end{bmatrix} \in \mathbb{R}^{n+m}$. Then, $\{f(V;t)\}_{V \in V_1 \times \cdots \times V_n}$ is a compatible family.

Let $\lambda_{\max}(v_1, \dots, v_n)$ denote the largest root of the mixed hyperbolic polynomial $h[v_1, \dots, v_m](t\overline{e} + \underline{\mathbf{1}}) \in \mathbb{R}[t]$, i.e.,

$$\lambda_{\max}(v_1, \dots, v_n) := \lambda_{\max}(h[v_1, \dots, v_m](t\overline{e} + \underline{\mathbf{1}})) \tag{20.12}$$

The following theorem shows that $\lambda_{\max}(v_1, \dots, v_n)$ can upper-bounds the largest hyperbolic eigenvalue of the vector $v_1 + \cdots + v_n$.

Theorem 20.39 (Theorem 5.2 in [Brä18]). *If h is hyperbolic with respect to e and $v_1, \dots, v_n \in \Lambda_+(e)$, then*

$$\lambda_{\max}(v_1 + \cdots + v_n) \le \lambda_{\max}(v_1, \dots, v_n).$$

The following theorem shows a connection between the hyperbolic cone of h and the hyperbolic cone of the mixed hyperbolic polynomial $h[v_1, \dots, v_n]$.

Theorem 20.40 (Corollary 5.5 in [Brä18]). *Suppose h is hyperbolic with respect to $e \in \mathbb{R}^m$, and let Γ_+ be the hyperbolic cone of $h[v_1, \dots, v_n]$, where $v_i \in \Lambda_+(e)$ and $1 \le rank(v_i) \le r_i$ for $i \in [m]$. Suppose $x \in \Lambda_{++}(e)$ be such that for $i \in [m]$, $\overline{x} + \mu_i \underline{e_i} \in \Gamma_+$ for any $\mu_i > 0$.*

Then, for any $(\delta_i, \mu_i) \in U_{r_i}$ for $i \in [m]$,

$$\overline{x} + \left(1 - \frac{1}{m}\right) \sum_{i=1}^{n} \delta_i \overline{v_i} + \left(1 - \frac{1}{m}\right) \underline{\mathbf{1}} + \frac{1}{m} \sum_{i=1}^{n} \mu_i \underline{e_i} \in \Gamma_+.$$

20.5.4 Upper bound for the largest root of the mixed hyperbolic polynomial

The goal of this section is to prove Theorem 20.41, which gives an upper bound for the mixed hyperbolic polynomial with vectors in sub-isotropic position.

Theorem 20.41 (Sub-isotropic version of Theorem 5.6 in [Brä18])**.** *Suppose $h \in \mathbb{R}[z_1, \dots, z_m]$ is hyperbolic with respect to $e \in \mathbb{R}^m$, and let $v_1, \dots, v_n \in \Lambda_+(h, e)$ that satisfy*

$$tr_h[v_i] \leq \epsilon, \quad rank(v_i) \leq r \quad \forall i \in [n], \text{ and } \quad \Big\| \sum_{i=1}^n v_i \Big\|_h \leq \sigma.$$

Then,

$$\lambda_{\max}(v_1, \dots, v_n) \leq \sigma \cdot \delta(\epsilon/\sigma, n, r),$$

where $\lambda_{\max}(v_1, \dots, v_n)$ is defined in Eq. (20.12).

Proof. For $\mu > 0$, let $x := \epsilon\mu \cdot e$ and $\mu_i := \mu$ for $i \in [n]$. Let $e_i \in \mathbb{R}^n$ be the i-th standard basis vector.

Then, we have

$$\begin{aligned} h[v_1, \dots, v_n](\overline{x} + \mu_i \underline{e_i}) &= (1 - \mu D_{v_i}) h(\epsilon\mu e) \\ &= \epsilon_d \mu^d h(e) + \mu^d \epsilon^{d-1} D_{v_i} h(e) \\ &= \mu^d \epsilon^{d-1} h(e)(\epsilon - tr_h[v_i]) \\ &> 0, \end{aligned}$$

where the first step follows from Fact 20.34, the second step follows from the homogeneity of hyperbolic polynomials, and the third step follows from Fact 20.35.

By part (2) of Theorem 19.6, we get that $x + \mu_i \underline{e_i} \in \Gamma_+$, the hyperbolic cone of $h[v_1, \dots, v_n]$, for all $i \in [n]$.

Then, by Theorem 20.40, for any $(\delta, \mu) \in U_r$,

$$\epsilon\mu\overline{e} + \left(1 - \frac{1}{n}\right)\delta \sum_{i=1}^{n} v_i + (1 + \frac{\mu - 1}{n})\underline{\mathbf{1}} \in \Gamma_+,$$

which implies

$$\frac{\epsilon\mu\overline{e} + \left(1 - \frac{1}{n}\right)\delta \sum_{i=1}^{n} v_i}{1 + \frac{\mu-1}{n}} + \underline{\mathbf{1}} \in \Gamma_+,$$

by the homogeneity of Γ_+. Since $\overline{e} \in \Gamma_{++}$, $\lambda_{\max}(\sum_{i=1}^{n} v_i) \leq \sigma$, and Γ_+ is a convex cone, we have

$$\frac{\left(\epsilon\mu + \left(1 - \frac{1}{n}\right)\delta\sigma\right)}{1 + \frac{\mu-1}{n}}\overline{e} + \underline{\mathbf{1}} \in \Gamma_+.$$

Hence, by Remark 5.1 in [Brä18],

$$\lambda_{\max}(v_1, \ldots, v_n) = \inf_{\rho > 0} \quad \rho\overline{e} + \underline{\mathbf{1}} \in \Gamma_+.$$

Hence, we conclude that

$$\begin{aligned}\lambda_{\max}(v_1, \ldots, v_n) \leq & \inf_{(\delta,\mu)\in U_r} \frac{\left(\epsilon\mu + \left(1 - \frac{1}{n}\right)\delta\sigma\right)}{1 + \frac{\mu-1}{n}} \\ & = \sigma \cdot \delta(\epsilon/\sigma, n, r).\end{aligned}$$

□

20.6 Hyperbolic Spencer Result

The goal of this section is to prove Theorem 20.42, which proves the rank-1 case of the hyperbolic Spencer conjecture (Conjecture 20.9).

Theorem 20.42 (Eight deviations suffice)**.** *Given $x_1, x_2, \cdots, x_n \in \mathbb{R}^m$ such that $\mathrm{rank}(x_i) \leq 1$ for all $i \in [n]$. Let h be an m-variable, degree-d hyperbolic polynomial with respect to e. Let $\sigma = (\sum_{i=1}^{n} \|x_i\|_h^2)^{1/2}$. Then, there exists a sign vector $r \sim \{-1, 1\}^n$ such that*

$$\left\|\sum_{i=1}^{n} r_i x_i\right\|_h \leq 8\sigma$$

holds.

Proof. Similar to the proof of Theorem 19.25, we first have

$$\begin{aligned}
\mathbb{E}_{r\sim\{\pm1\}^n}\left[\left\|\sum_{i=1}^n r_i x_i\right\|_h\right] &\leq \left(\mathbb{E}_{r\sim\{\pm1\}^n}\left[\left\|\sum_{i=1}^n r_i x_i\right\|_{h,2q}^{2q}\right]\right)^{1/(2q)} \\
&\leq \sqrt{2q-1}\cdot\left(\sum_{i=1}^n \|x_i\|_h^2\right)^{1/2} \\
&= \sqrt{2q-1}\cdot\sigma,
\end{aligned}$$

where the first step follows from Eq. (19.6).

By setting $q=1$, we have

$$\mathbb{E}_{r\sim\{\pm1\}^n}\left[\left\|\sum_{i=1}^n r_i x_i\right\|_h\right] \leq \sigma.$$

By Claim 19.19,

$$\mathbb{E}_{r\sim\{\pm1\}^n}\left[\left\|\sum_{i=1}^n r_i x_i\right\|_h^2\right] \leq 2\left(\mathbb{E}_{r\sim\{\pm1\}^n}\left[\left\|\sum_{i=1}^n r_i x_i\right\|_h\right]\right)^2 \leq 2\sigma^2.$$

Then, by Corollary 19.18,

$$\begin{aligned}
\Pr_{r\sim\{\pm1\}^n}\left[\left\|\sum_{i=1}^n r_i x_i\right\|_h > t\right] &\leq 2\exp\left(-\frac{t^2}{32\,\mathbb{E}_{r\sim\{\pm1\}^n}[\|\sum_{i=1}^n r_i x_i\|_h^2]}\right) \\
&\leq 2\exp\left(-\frac{t^2}{64\sigma^2}\right).
\end{aligned}$$

By choosing $t=8\sigma$, we have

$$\Pr_{r\sim\{\pm1\}^n}\left[\left\|\sum_{i=1}^n r_i x_i\right\|_h > 8\sigma\right] \leq 2/e.$$

Therefore, with probability $1-2/e$, we have

$$\left\|\sum_{i=1}^n r_i x_i\right\|_h \leq 8\sigma,$$

which proves the theorem. $\square$

Remark 20.5. It is interesting to apply Theorem 20.42 to determinant polynomial $h(x) = \det(X)$. It implies that for rank-1 matrices $X_1, \ldots, X_n \in \mathbb{R}^{d \times d}$,

$$\Pr_{r \sim \{\pm 1\}^n} \left[\left\| \sum_{i=1}^n r_i X_i \right\| > t \right] \leq 2 \exp\left(-\frac{t^2}{64\sigma^2} \right),$$

for $\sigma^2 = \sum_{i=1}^n \|X_i\|^2$.

This result is in fact incomparable to the matrix Chernoff bound [Tro15], which shows that

$$\Pr_{r \sim \{\pm 1\}^n} \left[\left\| \sum_{i=1}^n r_i X_i \right\| > t \right] \leq 2d \cdot \exp\left(-\frac{t^2}{2\widetilde{\sigma}^2} \right),$$

where $\widetilde{\sigma}^2 = \| \sum_{i=1}^n X_i^2 \|$. Because we only know the following relation between σ and $\widetilde{\sigma}$ [Tro15]:

$$\widetilde{\sigma}^2 \leq \sigma^2 \leq d \cdot \widetilde{\sigma}^2.$$

20.7 Hyperbolic Extension of Kadison-Singer for Standard Deviations

The goal of this section is to prove Theorem 20.43:

Theorem 20.43 (Formal statement of Theorem 20.10)**.** *Let $h \in \mathbb{R}[x_1, \ldots, x_m]$ denote a hyperbolic polynomial with respect to a hyperbolic direction $e \in \Gamma^h_{++}$, where $\Gamma^h_{++} \subseteq \mathbb{R}^m$ is the hyperbolicity cone of h. Let $\xi_1, \cdots, \xi_n$ denote n independent random variables with $\mathbb{E}[\xi_i] = \mu_i$ and $\mathbf{Var}[\xi] = \tau_i^2$. Let $v_1, \ldots, v_n \in \Gamma^h_+$ be n vectors such that $\forall i \in [n]$, $rank_h(v_i) \leq 1$. Suppose*

$$\sigma^2 := \left\| \sum_{i=1}^n \tau_i^2 tr_h[v_i] v_i \right\|_h.$$

Then,

$$\Pr_{\xi_1, \cdots, \xi_n} \left[\left\| \sum_{i=1}^n (\xi_i - \mu_i) v_i \right\|_h \leq 4\sigma \right] > 0.$$

We will introduce the preliminary facts in Section 20.7.1. In Section 20.7.2, we define the family of hyperbolic characteristic polynomials and show that it forms an interlacing family. Therefore, it remains to upper-bound the largest root of the average of the interlacing polynomial family, i.e. the mixed characteristic polynomial. We remark that we require a lemma that will be proved later in Section 20.7.3.

In Section 20.7.3, we reduce the problem of upper-bounding the largest root of the mixed characteristic polynomial to an easier task of upper-bounding the largest root of a multivariate polynomial. In Section 20.7.4, we upper bound the largest root of the multivariate polynomial using multivariate barrier method. Finally, in Section 20.7.5, we prove Theorem 20.43.

20.7.1 Preliminaries

In this section, we state several useful facts about hyperbolic polynomials. We first introduce some important properties of the derivatives of hyperbolic polynomial:

Theorem 20.44 (Theorem 3.1 in [Brä18] and known in [Går59, BGLS01, Ren06])**.** *Let $h \in \mathbb{R}[x_1, \cdots, x_m]$ be a hyperbolic polynomial and let $v \in \Gamma_+$ be a vector such that $D_v h \not\equiv 0$. Then*
1) $D_v h$ is hyperbolic with hyperbolicity cone containing Γ_{++}.
2) The polynomial $h(x) - y \cdot D_v h(x) \in \mathbb{R}[x, y]$ is hyperbolic with hyperbolicity cone containing $\Gamma_{++} \times \{y : y \leq 0\}$. Further, we have $(h(x)+y \cdot D_v h(x)) \cdot (h(x)-y \cdot D_v h(x)) \in \mathbb{R}[x, y]$ is hyperbolic with hyperbolicity cone containing $\Gamma_{++} \times \{y : y \leq 0\}$.
3) The rational function $x \to \frac{h(x)}{D_v h(x)}$ is concave on Γ_{++}.

The following lemma correlates hyperbolic trace to directional derivative:

Fact 20.45 (Correlation between hyperbolic trace and derivative)**.** *Let $h \in \mathbb{R}[x_1, \cdots, x_m]$ denote a hyperbolic polynomial with respect to $e \in \mathbb{R}^m$. For any $v \in \mathbb{R}^m$ and $\alpha \in \mathbb{R}$, we have*

$$tr_h[v] = \alpha \cdot \frac{D_v h(\alpha e)}{h(\alpha e)}.$$

Proof. By Theorem 20.44, $D_v h$ is hyperbolic, and thus is homogeneous. By Vieta's formula for the sum of roots of a polynomial, we have

$$tr_h[v] = \frac{D_v h(e)}{h(e)}$$

It follows that h and $D_v h$ are homogeneous, and the degree of h is one larger than that of $D_v h$. Therefore,

$$\alpha \cdot \frac{D_v h(\alpha e)}{h(\alpha e)} = \frac{D_v h(e)}{h(e)} = tr_h[v].$$

□

Fact 20.46. *Let $h \in \mathbb{R}[x_1, \cdots, x_m]$ denote a hyperbolic polynomial with respect to the direction $e \in \Gamma_{++}$ and let $\alpha > 0$. For any $M, v \in \Gamma_+$, we have*

$$\frac{D_v h(\alpha e + M)}{h(\alpha e + M)} \leq \frac{D_v h(\alpha e)}{h(\alpha e)}.$$

Proof. It suffices to show

$$\frac{h(\alpha e + M)}{D_v(\alpha e + M)} \geq \frac{h(\alpha e)}{D_v h(\alpha e)}.$$

By Theorem 20.44, we know that rational function $x \mapsto \frac{h(x)}{D_v h(x)}$ is concave on Γ_{++}. Then we know that

$$\begin{aligned} \frac{h(\alpha e + M)}{D_v(\alpha e + M)} &\geq \frac{1}{2}\frac{h(2\alpha e)}{D_v h(2\alpha e)} + \frac{1}{2}\frac{h(2M)}{D_v h(2M)} \\ &= \frac{h(\alpha e)}{D_v h(\alpha e)} + \frac{h(M)}{D_v h(M)} \\ &\geq \frac{h(\alpha e)}{D_v h(\alpha e)}. \end{aligned}$$

where the last step follows from $\frac{h(M)}{D_v h(M)} \geq 0$ (By [Ren06]). □

20.7.2 Defining interlacing family of characteristic polynomials

In this section, we consider the following interlacing family we will crucially use to prove Theorem 20.43.

Definition 20.13 (Interlacing Family of Theorem 20.43)**.** Let $h \in \mathbb{R}[x_1, \dots, x_m]$ denote a hyperbolic polynomial with respect to hyperbolic direction $e \in \Gamma^h_{++}$. Let $\xi_1, \dots, \xi_n$ denote n independent random variables with finite supports and $\mathbb{E}[\xi_i] = \mu_i$ for $i \in [n]$. Let $v_1, \dots, v_n \in \Gamma^h_+$ be n vectors such that $\mathrm{rank}_h(v_i) \le 1$ for all $i \in [n]$. For each $s = (s_1, \dots, s_n)$ where $s_i \in \mathrm{supp}(\xi_i)$, let $p_s \in \mathbb{R}[x]$ define the following polynomial:

$$p_{\mathbf{s}}(x) := \left(\prod_{i=1}^n p_{i,s_i}\right) \cdot h\left(xe + \sum_{i=1}^n (s_i - \mu_i)v_i\right) \cdot h\left(xe - \sum_{i=1}^n (s_i - \mu_i)v_i\right)$$

where $p_{i,s_i} := \Pr_{\xi_i}[\xi_i = s_i]$. Let $\mathcal{P}$ denote the following family of polynomials:

$$\mathcal{P} := \left\{ p_{(s_1,\cdots,s_\ell)} = \sum_{\substack{t_{\ell+1},\cdots,t_n: \\ t_j \in \mathrm{supp}(\xi_j)\ \forall j \in \{\ell+1,\dots,n\}}} p_{(s_1,\cdots,s_\ell,t_{\ell+1},\cdots,t_n)} :\ \ell \in [n], s_i \in \mathrm{supp}(\xi_i)\ \forall i \in [\ell] \right\}.$$

Lemma 20.47 (Interlacing Family of Theorem 20.43)**.** *The polynomial family $\mathcal{P}$ defined in Definition 20.13 is an interlacing family.*

Proof. Fix any $\ell \in [n-1]$, and fix $s_1, \cdots, s_\ell$ as any partial assignment of $\xi_1, \cdots, \xi_\ell$, i.e. $s_i \in \{\pm 1\}$ for all $i \in \mathrm{supp}(\xi_i)$.

It is easy to see that the polynomial $p_{(s_1,\dots,s_\ell)}$ can be written as:

$$\begin{aligned} \left(\prod_{i=1}^\ell p_{i,s_i}\right) \mathbb{E}_{\xi_{\ell+1},\dots,\xi_n} \Bigg[& h\left(xe + \left(\sum_{i=1}^\ell (s_i - \mu_i)v_i + \sum_{j=\ell+1}^n (\xi_j - \mu_j)v_j\right)\right) \cdot \\ & h\left(xe - \left(\sum_{i=1}^\ell (s_i - \mu_i)v_i + \sum_{j=\ell+1}^n (\xi_j - \mu_j)v_j\right)\right)\Bigg] \end{aligned}$$

Let $\{c_1, \ldots, c_k\}$ be the support of $\xi_{\ell+1}$. Then, by Lemma 20.25, it suffices to show that for any $\alpha \in \mathbb{R}^k_{\geq 0}$ with $\sum_{i=1}^k \alpha_i = 1$, the polynomial

$$\alpha_1 p_{(s_1,\cdots,s_\ell,c_1)}(x) + \cdots + \alpha_k p_{(s_1,\cdots,s_\ell,c_k)}(x)$$

is real-rooted. We interpret α as the probability density of the $(\ell+1)$-th random variable, i.e. let $p_{\ell+1,c_t} = \alpha_t$ for all $t \in [k]$. Then we can define a new random variable $\widetilde{\xi}_{\ell+1}$ with the same support as $\xi_{\ell+1}$ and $\Pr[\widetilde{\xi}_{\ell+1} = c_t] = p_{\ell+1,c_t}$ for all $t \in [k]$.

Notice that

$$\begin{aligned}
\widetilde{p_{\mathbf{s}}}(x) :=& \left(\prod_{i=1}^{\ell} p_{i,s_i}\right) \\
&\times \mathbb{E}_{\widetilde{\xi}_{\ell+1},\xi_{\ell+2}\cdots,\xi_n}\left[h\left(xe - \left(\sum_{i=1}^{\ell}(s_i-\mu_i)v_i + (\widetilde{\xi}_{\ell+1}-\mu_{\ell+1})v_{\ell+1} + \sum_{j=\ell+2}^{n}(\xi_j-\mu_j)v_j\right)\right)\right. \\
&\left.\cdot h\left(xe + \left(\sum_{i=1}^{\ell}(s_i-\mu_i)v_i + (\widetilde{\xi}_{\ell+1}-\mu_{\ell+1})v_{\ell+1} + \sum_{j=\ell+2}^{n}(\xi_j-\mu_j)v_j\right)\right)\right] \\
=& \sum_{t=1}^{k}\left(\prod_{i=1}^{\ell} p_{i,s_i}\right) p_{\ell+1,c_t} \\
&\times \mathbb{E}_{\xi_{\ell+2},\ldots,\xi_n}\left[h\left(xe - \left(\sum_{i=1}^{\ell}(s_i-\mu_i)v_i + c_t v_{\ell+1} + \sum_{j=\ell+2}^{n}(\xi_j-\mu_j)v_j\right)\right)\right. \\
&\left.\cdot h\left(xe + \left(\sum_{i=1}^{\ell}(s_i-\mu_i)v_i + c_t v_{\ell+1} + \sum_{j=\ell+2}^{n}(\xi_j-\mu_j)v_j\right)\right)\right] \\
=& \sum_{i=1}^{k} \alpha_i p_{\mathbf{s},c_i}(x).
\end{aligned}$$

By Lemma 20.49 (we remark that it does not require any result in the current section. We will prove Lemma 20.49 in Section 20.7.3), $\widetilde{p_{\mathbf{s}}}$ is real-rooted, which completes the proof that $\mathcal{P}$ is an interlacing family. $\square$

Lemma 20.47 implies the following corollary, which is a hyperbolic version of Proposition 4.1 in [KLS20]:

Corollary 20.48 (Upper Bound of the Largest Root of Interlacing Family)**.** *Let $h \in \mathbb{R}[x_1, \cdots, x_m]$ denote a hyperbolic polynomial with corresponding hyperbolic direction $e \in \Gamma^h_{++}$. Let $\xi_1, \ldots, \xi_n$ denote n independent random variables with finite supports and $\mathbb{E}[\xi_i] = \mu_i$ for $i \in [n]$. For any $v_1, \ldots v_n \in \Gamma^h_{++}$ such that $\mathrm{rank}_h(v_i) \leq 1$ for all $i \in [n]$, there exists an sign assignment $s = (s_1, \ldots, s_n) \in \mathrm{supp}(\xi_i) \times \cdots \mathrm{supp}(\xi_n)$, such that*

$$\left\| \sum_{i=1}^{m} (s_i - \mu_i) v_i \right\|_h$$

is at most the largest root of

$$p_\emptyset(x) = \mathbb{E}_{\xi_1, \ldots, \xi_m} \left[h\Big(xe + \sum_{i=1}^{m} (\xi_i - \mu_i) v_i\Big) \cdot h\Big(xe - \sum_{i=1}^{m} (\xi_i - \mu_i) v_i\Big) \right].$$

Proof. Let $\mathcal{P}$ be the interlacing family defined in Definition 20.13. Then by Lemma 20.47, $\mathcal{P}$ is an interlacing family.

For any fixed $\ell \in [n]$ and the first ℓ assignments $(s_1, \cdots, s_\ell)$ such that $s_i \in \mathrm{supp}(\xi_i)$ for $i \in [\ell]$, notice that

$$\begin{aligned} p_{s_1, \cdots, s_\ell}(x) = \left(\prod_{i=1}^{\ell} p_{i, s_i} \right) \mathbb{E}_{\xi_{\ell+1}, \ldots, \xi_n} \Bigg[& h \left(xe - \left(\sum_{i=1}^{\ell} (s_i - \mu_i) v_i + \sum_{j=\ell+1}^{n} (\xi_j - \mu_j) v_j \right) \right) \cdot \\ & h \left(xe + \left(\sum_{i=1}^{\ell} (s_i - \mu_i) v_i + \sum_{j=\ell+1}^{n} (\xi_j - \mu_j) v_j \right) \right) \Bigg], \end{aligned}$$

and

$$p_\emptyset = \mathbb{E}_{\xi_1, \ldots, \xi_m} \left[h\Big(xe + \sum_{i=1}^{m} (\xi_i - \mu_i) v_i\Big) \cdot h\Big(xe - \sum_{i=1}^{m} (\xi_i - \mu_i) v_i\Big) \right].$$

Therefore, by Lemma 20.24, there exists a sign assignment

$$(s_1, \ldots, s_n) \in \mathrm{supp}(\xi_i) \times \cdots \mathrm{supp}(\xi_n)$$

such that the largest root of p_s is upper bounded by the largest root of

$$\mathbb{E}_{\xi_1, \ldots, \xi_m} \left[h\Big(xe + \sum_{i=1}^{m} (\xi_i - \mu_i) v_i\Big) \cdot h\Big(xe - \sum_{i=1}^{m} (\xi_i - \mu_i) v_i\Big) \right].$$

Then by Fact 20.19, we have $\|\sum_{i=1}^m (s_i - \mu_i)v_i\|_h = \lambda_{\max}(p_{\mathbf{s}})$, which is upper bounded by the maximum root of $p_\emptyset$.

$\square$

20.7.3 From mixed characteristic polynomial to multivariate polynomial

In this section, we want to show that the mixed characteristic polynomial, i.e. the average of the interlacing family defined in Definition 20.13:

$$p_\emptyset = \mathbb{E}_{\xi_1,\dots,\xi_m}\left[h\Big(xe + \sum_{i=1}^m (\xi_i - \mu_i)v_i\Big) \cdot h\Big(xe - \sum_{i=1}^m (\xi_i - \mu_i)v_i\Big)\right]$$

equals to the following multivariate polynomial after taking $z_1 = \dots = z_n = 0$.

$$\prod_{i=1}^n \Big(1 - \frac{1}{2}\frac{\partial^2}{\partial z_i^2}\Big) h\Big(xe - Q + \sum_{i=1}^n z_i\tau_i v_i\Big) \cdot h\Big(xe + Q + \sum_{i=1}^n z_i \tau_i v_i\Big) \in \mathbb{R}[x, z_1, \cdots, z_n]$$

The largest root of this multivariate polynomial is relatively easy to upper-bound using barrier argument. We will describe the details in Section 20.7.4.

The main lemma of this section is as follows:

Lemma 20.49 (Hyperbolic version of Proposition 3.3 in [KLS20]). *Let $h \in \mathbb{R}[x_1, \cdots, x_m]$ denote a hyperbolic polynomial with respect to a hyperbolic direction $e \in \Gamma_{++}^h$. Let $v_1, \dots v_n \in \Gamma_+^h$ such that $\forall i \in [n]$, $\mathrm{rank}_h(v_i) \le 1$. Let $\xi_1, \dots, \xi_n$ denote n independent random variables such that $\mathbb{E}[\xi_i] = \mu_i$ and $\mathbf{Var}[\xi_i] = \tau_i^2$. For any $Q \in \mathbb{R}^m$, we have*

$$\begin{aligned} &\mathbb{E}_{\xi_1,\dots,\xi_n}\left[h\Big(xe - (Q + \sum_{i=1}^n (\xi_i - \mu_i)v_i)\Big) h\Big(xe + (Q + \sum_{i=1}^n (\xi_i - \mu_i)v_i)\Big)\right] \\ = &\prod_{i=1}^n \Big(1 - \frac{1}{2}\frac{\partial^2}{\partial z_i^2}\Big)\Big|_{z_i=0} h\Big(xe - Q + \sum_{i=1}^n z_i\tau_i v_i\Big) \cdot h\Big(xe + Q + \sum_{i=1}^n z_i\tau_i v_i\Big). \end{aligned} \quad (20.13)$$

Moreover, this is a real-rooted polynomial in x.

Proof. We first show Eqn. (20.13) by induction. Our induction hypothesis will be that for any $0 \leq k \leq n$,

$$\begin{aligned}
&\mathbb{E}_{\xi_1,\ldots,\xi_n}\left[h\Big(xe-(Q+\sum_{i=1}^{n}(\xi_i-\mu_i)v_i)\Big)h\Big(xe+(Q+\sum_{i=1}^{n}(\xi_i-\mu_i)v_i)\Big)\right] \\
=\ &\mathbb{E}_{\xi_{k+1},\ldots,\xi_n}\prod_{i=1}^{k}\Big(1-\frac{1}{2}\frac{\partial^2}{\partial z_i^2}\Big)\Big|_{z_i=0}h\Big(xe-Q-\sum_{i=k+1}^{n}(\xi_i-\mu_i)v_i+\sum_{j=1}^{k}z_j\tau_jv_j\Big) \\
&\times h\Big(xe+Q+\sum_{i=k+1}^{n}(\xi_i-\mu_i)v_i+\sum_{j=1}^{k}z_j\tau_jv_j\Big) \qquad (20.14)
\end{aligned}$$

The base case, $k = 0$ trivially holds as we get the same formula on both sides.

For the inductive step, suppose the induction hypothesis holds for any $k \leq \ell$ where $0 \leq \ell < n$. Applying Claim 20.50 to the right-hand-side of Eqn. (20.14) when letting $k = \ell$ yields

$$\begin{aligned}
&\mathbb{E}_{\xi_1,\ldots,\xi_m}\left[h\Big(xe-(Q+\sum_{i=1}^{n}(\xi_i-\mu_i)v_i)\Big)h\Big(xe+(Q+\sum_{i=1}^{n}(\xi_i-\mu_i)v_i)\Big)\right] \\
=\ &\mathbb{E}_{\xi_{\ell+2},\ldots,\xi_m}\prod_{i=1}^{\ell+1}\Big(1-\frac{1}{2}\frac{\partial^2}{\partial z_i^2}\Big)\Big|_{z_i=0}h\Big(xe-Q-\sum_{i=\ell+2}^{n}(\xi_i-\mu_i)v_i+\sum_{j=1}^{\ell+1}z_j\tau_jv_j\Big) \\
&\times h\Big(xe+Q+\sum_{i=\ell+2}^{n}(\xi_i-\mu_i)v_i+\sum_{j=1}^{\ell+1}z_j\tau_jv_j\Big) \qquad (20.15)
\end{aligned}$$

This completes the proof of Eqn. (20.13).

We now show that Eqn. (20.13) is real-rooted in x. By Claim 20.51, we know that

$$h\Big(xe-Q+\sum_{i=1}^{n}z_i\tau_iv_i\Big) \ \text{ and } \ h\Big(xe+Q+\sum_{i=1}^{n}z_i\tau_iv_i\Big) \in \mathbb{R}[x,z_1,\ldots,z_n],$$

are real-stable polynomials. Then, by Fact 20.17, we get that

$$h\Big(xe-Q+\sum_{i=1}^{n}z_i\tau_iv_i\Big)\cdot h\Big(xe+Q+\sum_{i=1}^{n}z_i\tau_iv_i\Big)$$

is also real-stable. And by Fact 20.17 again, the operator

$$\prod_{i=1}^{n}\Big(1-\frac{1}{2}\frac{\partial^2}{\partial z_i^2}\Big)\Big|_{z_i=0}$$

preserves the real-stability. Hence, by Fact 20.14, the RHS of Eqn. (20.13) is real-rooted. $\square$

The following statements are crucially used in the proof of Lemma 20.49:

Claim 20.50 (Hyperbolic version of Lemma 3.1 in [KLS20])**.** *Let h denote a hyperbolic polynomial. Let ξ denote a random variables with $\mathbb{E}[\xi] = 0$ and $\mathbf{Var}[\xi] = \tau^2$. For any $v \in \mathbb{R}^m$ such that $rank_h(v) \le 1$, and any $x_1, x_2 \in \mathbb{R}^m$, we have*

$$\mathbb{E}_\xi[h(x_1 - \xi v) \cdot h(x_2 + \xi v)] = \left(1 - \frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}t^2}\right)\Bigg|_{t=0} h(x_1 + t\tau v)h(x_2 + t\tau v).$$

Remark 20.6. Claim 20.50 can be easily generalized for non-centered random variable ξ with $\mathbb{E}[\xi] = \mu$:

$$\mathbb{E}_\xi[h(x_1 - (\xi - \mu)v) \cdot h(x_2 + (\xi - \mu)v)] = \left(1 - \frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}t^2}\right)\Bigg|_{t=0} h(x_1 + t\tau v)h(x_2 + t\tau v).$$

Proof. Since $rank_h(v) \le 1$, for all $k \ge 2$, $D_v^k h \equiv 0$. Thus, for all $x_1 \in \mathbb{R}^m$,

$$h(x_1 - \xi v) = \left(\sum_{k=0}^{\infty} \frac{(-\xi)^k D_v^k}{k!}\right) h(x_1) = (1 - \xi D_v)h(x_1).$$

where the first step follows from Taylor expansion of $h(x_1 - \xi v)$ on x_1.

Similarly, for all $x_2 \in \mathbb{R}^m$,

$$h(x_2 + \xi v) = (1 + \xi D_v)h(x_2).$$

Therefore,

$$\begin{aligned} h(x_1 - \xi v) \cdot h(x_2 + \xi v) &= (1 - \xi D_v)h(x_1) \cdot (1 + \xi D_v)h(x_2) \\ &= h(x_1)h(x_2) - \xi h(x_2)D_v h(x_1) + \xi h(x_1)D_v h(x_2) - \xi^2 D_v h(x_1)D_v h(x_2) \end{aligned}$$

Since $\mathbb{E}[\xi] = 0$ and $\mathbf{Var}[\xi] = \tau^2$, we have

$$\begin{aligned}\mathbb{E}_\xi[h(x_1 - \xi v) \cdot h(x_2 + \xi v)] &= \left(1 - \frac{\mathbb{E}[\xi^2]}{2} D_v^2\right) h(x_1)h(x_2) + \mathbb{E}_\xi[\xi] \cdot (h(x_1)D_v h(x_2) - h(x_2)D_v h(x_1)) \\ &= \left(1 - \frac{\tau^2}{2} D_v^2\right) h(x_1)h(x_2) \\ &= \left(1 - \frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}t^2}\right)\bigg|_{t=0} h(x_1 + t\tau v)h(x_2 + t\tau v).\end{aligned}$$

□

Claim 20.51 (Linear restriction of hyperbolic polynomial is real-stable)**.** *Let $h \in \mathbb{R}[x_1, \ldots, x_m]$ be a hyperbolic polynomial with respect to $e \in \mathbb{R}^m$. Let $v_1, \ldots, v_n \in \Gamma_+$ and $Q \in \mathbb{R}^m$. Define*

$$p(x, z) := h\Big(xe + \sum_{i=1}^n z_i v_i - Q\Big) \in \mathbb{R}[x, z_1, \ldots, z_n].$$

Then, $p(x, z)$ is a real-stable polynomial.

Proof. For any $a \in \mathbb{R}_{>0}^{n+1}$, $b \in \mathbb{R}^{n+1}$, we have

$$\begin{aligned}p(at + b) &= h\Big((a_1 t + b_1)e - Q + \sum_{i=1}^n (a_{i+1}t + b_{i+1})v_i\Big) \\ &= h\Big((a_1 e + \sum_{i=1}^n a_{i+1}v_i)t + b_1 e - Q + \sum_{i=1}^n b_{i+1}v_i\Big).\end{aligned}$$

Since $e \in \Gamma_{++}$, $v_1, \ldots, v_n \in \Gamma_+$ and $a_i > 0$ for all $i \in [n+1]$, we have

$$e' := a_1 e + \sum_{i=1}^n a_{i+1} v_i \in \Gamma_{++},$$

which follows from Γ_{++} is a cone.

Since every vector in Γ_{++} is a hyperbolic direction of h (see e.g., [Brä18, Theorem 1.2, item 4]), we know that h is also hyperbolic with respect to the direction e', which implies that $p(at+b)$ is real-rooted and not identical to the zero polynomial.

Hence, by Fact 20.15, $p(x, z)$ is a real-stable polynomial. □

20.7.4 Applying barrier argument to bound the largest root of multivariate polynomial

In this section, we upper bound the largest root of the following multivariate polynomial:

$$\prod_{i=1}^{n}\Big(1-\frac{1}{2}\frac{\partial^2}{\partial z_i^2}\Big)h\Big(xe-Q+\sum_{i=1}^{n} z_i\tau_i v_i\Big)\cdot h\Big(xe+Q+\sum_{i=1}^{n} z_i\tau_i v_i\Big)$$

using the real-stable version of the barrier method in [KLS20].

Definition 20.14. Let $h \in \mathbb{R}[x_1,\cdots,x_m]$ be a hyperbolic polynomial of degree d with respect to hyperbolic direction $e \in \mathbb{R}^m$. For any vectors $u, v \in \mathbb{R}^m$, we say $u \preceq v$ if

$$\lambda_i(u) \leq \lambda_i(v) \quad \forall i \in [d],$$

where $\lambda(u), \lambda(v)$ are the ordered eigenvalues of u and v, respectively.

Claim 20.52. *Let h be a hyperbolic polynomial of degree d with respect to hyperbolic direction $e \in \mathbb{R}^m$. Let $u \in \mathbb{R}^m$ be any vector such that $u \preceq e$. Then we have $e-u \in \Gamma_+^h$.*

Proof. For any $i \in [d]$, we have

$$\lambda_i(e-u) = 1-\lambda_{d-i}(u) \geq 0,$$

where the last step follows from $\lambda_i(u) \leq \lambda_i(e) \leq 1$ for all $i \in [d]$.

Hence, $e-u \in \Gamma_+^h$. □

The goal of this section is to prove the following lemma:

Lemma 20.53. *Let $h \in \mathbb{R}[x_1,\cdots,x_m]$ denote a hyperbolic polynomial with corresponding hyperbolic direction $e \in \Gamma_+^h$. Let $\xi_1,\ldots,\xi_n$ denote n independent random variables with finite supports and $\mathbb{E}[\xi_i] = \mu_i$ and $\mathbf{Var}[\xi_i] = \tau_i^2$ for $i \in [n]$. Let $v_1,\ldots,v_n \in \Gamma_+^h$ such that $\sum_{i=1}^n \tau_i^2 tr_h[v_i]v_i \preceq e$ and $\forall i \in [n]$, $rank_h(v_i) \leq 1$.*

Then all the roots of the following $(n+1)$-variate polynomial

$$\prod_{i=1}^{n}\Big(1-\frac{1}{2}\frac{\partial^2}{\partial z_i^2}\Big)\left(h\Big(xe+\sum_{i=1}^{n} z_i\tau_i v_i\Big)\right)^2 \in \mathbb{R}[x, z_1, \ldots, z_n]$$

lie below $(4, 0, \cdots, 0) \in \mathbb{R}^{n+1}$.

Proof. Define $(n+1)$-variate polynomial $P(x,z) \in \mathbb{R}[x, z_1, \ldots, z_n]$ as

$$P(x,z) := \left(h\Big(xe+\sum_{i=1}^{n} z_i\tau_i v_i\Big)\right)^2 .$$

By Claim 20.51 and Fact 20.17, we can show that $P(x,z)$ is real-stable. Thus, we can apply the multivariate barrier method in [KLS20] with the barrier functions $\Phi_P^i(x,z) = \frac{\partial_{z_i} P(x,z)}{P(x,z)}$ for $i \in [n]$.

For $t > 0$, let $\delta_i = t\tau_i tr_h[v_i]$ and let

$$\delta = (\delta_1, \ldots, \delta_n).$$

For some $\alpha(t) > t$ where $\alpha(t)$ is a parameter to be chosen later, we evaluate P at

$$(\alpha, -\delta) = (\alpha(t), -\delta_1, \ldots, -\delta_n)$$

to find that

$$\begin{aligned} P(\alpha(t), -\delta_1, \ldots, -\delta_n) &= \left(h\Big(\alpha(t)e-\sum_{i=1}^{n} \delta_i\tau_i v_i\Big)\right)^2 \\ &= \left(h\Big(\alpha(t)e-t\sum_{i=1}^{n} \tau_i^2 tr_h[v_i] v_i\Big)\right)^2 \\ &= \left(h(e)\prod_{j=1}^{d}\Big(\alpha(t)-t\lambda_j\Big(\sum_{i=1}^{n}\tau_i^2 tr_h[v_i]v_i\Big)\Big)\right)^2 \end{aligned}$$

where d is the degree of h. Here the last step follows from the hyperbolicity of h, and the fact that the set of roots of

$$h\Big(\alpha(t)e-t\sum_{i=1}^{n}\tau_i^2 tr_h[v_i]v_i\Big)$$

are

$$\left\{ \left(\frac{1}{\alpha(t)} \cdot \lambda_j \Big(\sum_{i=1}^n \tau_i^2 tr_h[v_i] v_i \Big) \right)^{-1} \right\}_{j \in [d]}$$

Since $\sum_{i=1}^n \tau_i^2 tr_h[v_i] v_i \preceq e$, we have for all $j \in [d]$, $\lambda_j(\sum_{i=1}^n \tau_i^2 tr_h[v_i] v_i) \leq \lambda_j(e) = 1$.

Hence, by the assumption of $t < \alpha(t)$, we get that

$$P(\alpha(t), -\delta_1, \ldots, -\delta_n) \geq h(e)^2 (\alpha(t) - t)^{2d} > 0.$$

This implies that $(\alpha, -\delta) \in \mathbb{R}^{m+1}$ is above the roots of $P(x, z)$, i.e. $(\alpha, -\delta) \in \mathbf{Ab}_P$.

Moreover, we can upper bound $\Phi_P^i(\alpha, -\delta)$ as follows:

$$\begin{aligned}
\Phi_P^i(\alpha(t), -\delta) &= \frac{\partial_{z_i} P}{P} \Big|_{x=\alpha(t), z=-\delta} \\
&= \frac{2h(xe + \sum_{i=1}^n z_i \tau_i v_i) \partial_{z_i} h(xe + \sum_{i=1}^n z_i \tau_i v_i)}{h(xe + \sum_{i=1}^n z_i \tau_i v_i)^2} \Big|_{x=\alpha(t), z=-\delta} \\
&= 2 \cdot \frac{\partial_{z_i} h(xe + \sum_{i=1}^n z_i \tau_i v_i)}{h(xe + \sum_{i=1}^n z_i \tau_i v_i)} \Big|_{x=\alpha(t), z=-\delta} \\
&= 2 \cdot \frac{(D_{\tau_i v_i} h)(\alpha e + \sum_{j=1}^n \delta_j \tau_j v_j)}{h(\alpha e + \sum_{j=1}^n \delta_j \tau_j v_j)} \\
&= 2 \cdot \frac{(D_{\tau_i v_i} h)(\alpha e - t \sum_{j=1}^n \tau_j^2 tr_h[v_j] v_j)}{h(\alpha e - t \sum_{j=1}^n \tau_j^2 tr_h[v_j] v_j)} \\
&= 2 \cdot \frac{(D_{\tau_i v_i} h)(\alpha e - te + t(e - \sum_{j=1}^n \tau_j^2 tr_h[v_j] v_j))}{h(\alpha e - te + t(e - \sum_{j=1}^n \tau_j^2 tr_h[v_j] v_j))} \\
&\leq \frac{2(D_{\tau_i v_i} h)(\alpha e - te)}{h(\alpha e - te)} \\
&\leq \frac{2 tr_h[\tau_i v_i]}{\alpha - t}.
\end{aligned}$$

where the second last step follows from $\sum_{i=1}^n \tau_j^2 tr_h[v_i] v_i \preceq e$, Claim 20.52 and Fact 20.46. The last step follows from $\frac{(D_v h)(\beta e)}{h(\beta e)} = \frac{tr_h[v]}{\beta}$ by Fact 20.45.

Since $rank_h(v_i) \leq 1$, we have $tr_h[v_i] = \|v_i\|_h$ for all $i \in [n]$. Since $\sum_{j \neq i} \tau_j^2 tr_h[v_j]v_j \in \Gamma_+^h$, by the monotonicity of the hyperbolic norm (Theorem 2.15 in [HLJ09]), we have

$$(\tau_i tr_h[v_i])^2 = \|\tau_i^2 tr_h[v_i]v_i\|_h \leq \left\| \sum_{j=1}^n \tau_j^2 tr_h[v_j]v_j \right\|_h \leq \|e\|_h = 1$$

for all $i \in [n]$.

Thus, we have

$$\max_{i \in [n]} \quad (\tau_i tr_h[v_i])^2 \leq 1.$$

Choosing $\alpha(t) = 2t = 4$. We get

$$\Phi_P^i(\alpha, -\delta) \leq \frac{2tr_h[\tau_i v_i]}{\alpha - t} = \frac{2\tau_i tr_h[v_i]}{\alpha - t} = \frac{2\tau_i tr_h[v_i]}{2} \leq 1 < \sqrt{2}. \tag{20.16}$$

This coincides with Eqn. (20.10) of Lemma 20.26. Also from Fact 20.17 we know that $(1 - \frac{1}{2}\partial_{z_i}^2)P$ is real stable.

Thus by Lemma 20.26, for all $i \in [n]$,

$$(4, -\delta) \in \mathbf{Ab}_{(1-\frac{1}{2}\partial_{z_i}^2)P}.$$

In addition, $\forall i \in [n]$, since $\delta_i = t\tau_i tr_h[v_i] = 2\tau_i tr_h[v_i] > 0$,

$$\begin{aligned} \frac{1}{\delta_i}\Phi_P^i(4, -\delta) + \frac{1}{2}\Phi_P^i(4, -\delta)^2 &\leq \frac{1}{2\tau_i tr_h[v_i]}\tau_i tr_h[v_i] + \frac{1}{2}(\tau_i tr_h[v_i])^2 \\ &\leq \frac{1}{2} + \frac{1}{2} = 1. \end{aligned} \tag{20.17}$$

This coincides with Eqn. (20.11) of Lemma 20.26. Therefore for all $j \in [n]$,

$$\Phi^j_{(1-\frac{1}{2}\partial_{z_i}^2)P}(4, -\delta + \delta_i \mathbf{1}_i) \leq \Phi_P^j(4, -\delta). \tag{20.18}$$

In particular, we have

$$\Phi^2_{(1-\frac{1}{2}\partial_{z_1}^2)P}(4, -\delta + \delta_1 \mathbf{1}_1) \leq \Phi_P^2(4, -\delta) < \sqrt{2}.$$

We also have $(4, -\delta + \delta_1 \mathbf{1}_1) \in \mathbf{Ab}_{(1-\frac{1}{2}\partial_{z_1}^2)P}$, which follows from $(4, -\delta) \in \mathbf{Ab}_{(1-\frac{1}{2}\partial_{z_1}^2)P}$. By Lemma 20.27 with $k = 0$, we get that

$$\Phi^2_{(1-\frac{1}{2}\partial_{z_1}^2)P}(4, -\delta + \delta_1 \mathbf{1}_1) \geq 0.$$

Hence, we have

$$\begin{aligned}
&\frac{1}{\delta_2}\Phi^2_{(1-\frac{1}{2}\partial_{z_1}^2)P}(4, -\delta + \delta_1 \mathbf{1}_1) + \frac{1}{2}\Phi^2_{(1-\frac{1}{2}\partial_{z_1}^2)P}(4, -\delta + \delta_1 \mathbf{1}_1)^2 \\
\leq\ & \frac{1}{\delta_2}\Phi^2_P(4, -\delta) + \frac{1}{2}\Phi^2_P(4, -\delta)^2 \\
\leq\ & 1,
\end{aligned}$$

where the last step follows from Eqn. (20.17).

Therefore, by Lemma 20.26 again, we have

$$\Phi^i_{(1-\frac{1}{2}\partial_{z_2}^2)(1-\frac{1}{2}\partial_{z_1}^2)P}(4, -\delta + \delta_1 \mathbf{1}_1 + \delta_2 \mathbf{1}_2) \leq \Phi^i_{(1-\frac{1}{2}\partial_{z_1}^2)P}(4, -\delta + \delta_1 \mathbf{1}_1).$$

Repeating this argument for each $i \in [n]$ demonstrates that

$$\begin{aligned}
(4, -\delta + \sum_{i=1}^n \delta_i \mathbf{1}_i) &= (4, 0, 0, \ldots, 0) \\
&\in \mathbf{Ab}_{\prod_{i=1}^n (1-\partial_{z_i}^2/2)P}
\end{aligned}$$

i.e. $(4, 0, 0, \ldots, 0)$ lies above the roots of

$$\prod_{i=1}^n \Big(1 - \frac{1}{2}\frac{\partial^2}{\partial z_i^2}\Big)\Bigg(h\Big(xe + \sum_{i=1}^n z_i \tau_i v_i\Big)\Bigg)^2.$$

□

20.7.5 Combining together: proof of Theorem 20.43

In this section, we will combine the results from the previous section and prove Theorem 20.43:

Proof of Theorem 20.43. Define $u_i := \frac{v_i}{\sigma}$. Note that $\sigma > 0$ since $v_1, \dots, v_n$ are in the hyperbolicity cone of h.

Then, we have

$$\left\| \sum_{i=1}^n \tau_i^2 tr_h[u_i] u_i \right\|_h = \left\| \sum_{i=1}^n \frac{\tau_i^2 tr_h[v_i] v_i}{\sigma^2} \right\|_h = 1,$$

where the first step follows from the linearity of the hyperbolic trace tr_h, and the second step follows from $\|\cdot\|_h$ is a norm.

By Lemma 20.53 and restricting to $z_i = 0$ for all $i \in [n]$, we have that 4 lies above the largest root of the univariate polynomial

$$\prod_{i=1}^n \Big(1 - \frac{1}{2}\frac{\partial^2}{\partial z_i^2}\Big)\Big|_{z_i=0} \left(h\Big(xe + \sum_{i=1}^n z_i \tau_i u_i\Big)\right)^2$$

We then conclude by Lemma 20.49 that 4 upper bounds the largest root of

$$p_\emptyset = \mathbb{E}_{\xi_1,\dots,\xi_n}\left[h\Big(xe + \sum_{i=1}^n (\xi_i - \mu_i) u_i\Big) \cdot h\Big(xe - \sum_{i=1}^n (\xi_i - \mu_i) u_i\Big)\right].$$

where $p_\emptyset$ is the average of the polynomials in the interlacing family $\mathcal{P}$ in Definition 20.13.

Finally, by Corollary 20.48, we conclude that there exists $s_1, \dots, s_n \in \mathrm{supp}(\xi_1) \times \cdots \times \mathrm{supp}(\xi_n)$ such that

$$\left\| \sum_{i=1}^n (s_i - \mu_i) u_i \right\|_h \le 4.$$

Hence we have

$$\left\| \sum_{i=1}^n (s_i - \mu_i) v_i \right\|_h \le 4\sigma.$$

□

20.8 Hyperbolic Extension of Kadision-Singer for Strongly Rayleigh

In this section, we prove Theorem 20.11. We restate the theorem as follows:

Theorem 20.54 (Formal statement of Theorem 20.11)**.** *Let $h \in \mathbb{R}[x_1, \dots, x_m]$ denote a hyperbolic polynomial with respect to hyperbolic direction $e \in \Gamma^h_{++}$. Let μ be a homogeneous strongly Rayleigh probability distribution on $[n]$ such that the marginal probability of each element is at most ϵ_1, and let $v_1, \cdots, v_n \in \Gamma^h_+$ be n vectors in isotropic positions,*

$$\sum_{i=1}^n v_i = e,$$

such that for all $i \in [n]$,

$$\mathrm{rank}_h(v_i) \leq 1, \text{ and } \|v_i\|_h \leq \epsilon_2.$$

Then

$$\Pr_{S \sim \mu}\left[\Big\| \sum_{i \in S} v_i \Big\|_h \leq 4(\epsilon_1 + \epsilon_2) + 2(\epsilon_1 + \epsilon_2)^2\right] > 0.$$

We will introduce the preliminary facts in Section 20.8.1. In Section 20.8.2, we define the family of hyperbolic characteristic polynomials and show that it forms an interlacing family. Therefore, it remains to upper-bound the largest root of the average of the interlacing polynomial family, i.e. the mixed hyperbolic characteristic polynomial. We remark that we require a lemma that will be proved later in Section 20.8.3.

In Section 20.8.3, we reduce the problem of upper-bounding the largest root of the mixed hyperbolic characteristic polynomial to an easier task of upper-bounding the largest root of a multivariate polynomial. In Section 20.8.4, we upper bound the largest root of the multivariate polynomial using the multivariate barrier method. Finally, in Section 20.8.5, we prove Theorem 20.54.

20.8.1 Preliminaries

In this section, we present some preliminary results on strongly Rayleigh distributions.

Definition 20.15 (Generating polynomial of probability distribution)**.** Let $\mu : 2^{[n]} \to \mathbb{R}_{\geq 0}$ be a probability distribution. For a random variable $X \sim \mu$, the generating polynomial of μ is defined as follows:

$$g_\mu(z_1, \ldots, z_n) = \mathbb{E}\left[z^X\right] = \sum_{S \subseteq [n]} \Pr[X = S] z^S.$$

Definition 20.16 (Strongly Rayleigh distribution)**.** Let $\mu : 2^{[n]} \to \mathbb{R}_{\geq 0}$ be a probability distribution and g_μ be its generating polynomial. We say μ is strongly Rayleigh (SR) if g_μ is a real stable polynomial.

Moreover, we say μ is d_μ-homogeneous strongly Rayleigh if g_μ is d_μ-homogeneous real stable.

We provide two facts about the generating polynomials of Strongly Rayleigh distributions.

Fact 20.55 (Marginals of homogeneous SR distributions)**.** *Let $\mu : 2^{[n]} \to \mathbb{R}_{\geq 0}$ be a d_μ-homogeneous SR distribution with generating polynomial g_μ. For $1 \leq k \leq n$, consider the marginal distribution on the first k elements $\mu_k : 2^{[k]} \to \mathbb{R}_{\geq 0}$ such that*

$$\mu_k(S) = \Pr_{T \sim \mu}[T \cap [k] = S] \quad \forall S \subseteq [k].$$

Then, for all $S \subseteq [k]$,

$$\mu_k(S) = x^{|S| - d_\mu} \cdot \prod_{i \in S} \partial_{z_i} \prod_{i \in [k] \setminus S} (1 - x \partial_{z_i}) g_\mu(x\mathbf{1} + z) \Big|_{z=0}. \tag{20.19}$$

In particular,

$$\mu(S) = x^{|S| - d_\mu} \cdot \prod_{i \in S} \partial_{z_i} \prod_{i \in [n] \setminus S} (1 - x \partial_{z_i}) g_\mu(x\mathbf{1} + z) \Big|_{z=0}.$$

Remark 20.7. We note that the dummy variable x will be cancelled in the RHS of Eqn. (20.19), and hence both sides are numbers.

Proof. Note that g_μ can be written as

$$g_\mu(x\mathbf{1}+z) = f(z_2,\ldots,z_n)(x+z_1) + g(z_2,\ldots,z_n),$$

where

$$\begin{aligned} f(z_2,\ldots,z_n) &:= \sum_{\substack{S\subseteq[n]\backslash\{1\},\\ |S|=d_\mu-1}} \mu(S\cup\{1\})\prod_{i\in S}(x+z_i), \quad \text{and} \\ g(z_2,\ldots,z_n) &:= \sum_{\substack{S\subseteq[n]\backslash\{1\},\\ |S|=d_\mu}} \mu(S)\prod_{i\in S}(x+z_i). \end{aligned}$$

First, if $k=1$. We have

$$\mu_1(\{1\}) = \Pr_{T\sim\mu}[1\in T] = x^{-d_\mu+1}\cdot f(z_2,\ldots,z_n)\Big|_{z=0} = x^{-d_\mu+1}\cdot \partial_{z_1} g_\mu(x\mathbf{1}+z)\Big|_{z=0}.$$

Also,

$$\mu_1(\emptyset) = \Pr_{T\sim\mu}[1\notin T] = x^{-d_\mu}\cdot g(z_2,\ldots,z_n)\Big|_{z=0} = x^{-d_\mu}\cdot (1-x\partial_{z_1}) g_\mu(x\mathbf{1}+z)\Big|_{z=0}.$$

So, Eqn. (20.19) holds for $k=1$.

For the cases where $k>1$s, we can prove by induction on k. Suppose (20.19) holds for $1,\cdots,k-1$. Let $S\subseteq[k]$ be any subset of $[k]$. If $k\in S$, then consider $\partial_{z_k} g_\mu(x\mathbf{1}+z)$, which is the generating polynomial of μ' that restricts μ to the sets $T\subseteq[n]\backslash\{k\}$ with $\mu(T\cup\{k\})>0$. Let $S':=S\backslash\{k\}$. By the induction hypothesis, we have

$$\begin{aligned} \mu(S) = \mu'(S') &= x^{|S'|-d_{\mu'}}\cdot \prod_{i\in S'}\partial_{z_i}\prod_{i\in[k-1]\backslash S'}(1-x\partial_{z_i})\partial_{z_k} g_\mu(x\mathbf{1}+z)\Bigg|_{z=0} \\ &= x^{|S|-d_\mu}\cdot \prod_{i\in S}\partial_{z_i}\prod_{i\in[k]\backslash S}(1-x\partial_{z_i}) g_\mu(x\mathbf{1}+z)\Bigg|_{z=0}, \end{aligned}$$

where the second line follows from $|S'| = |S| - 1$ and $d_{\mu'} = d_\mu - 1$.

If $k \notin S$, then consider $(1 - x\partial_{z_k}) g_\mu(x\mathbf{1} + z)\Big|_{z_k=0}$, which is the generating polynomial of μ'' that restricts μ to the sets $T \subseteq [n]\backslash\{k\}$ with $\mu(T) > 0$. Then by the induction hypothesis, we have

$$\begin{aligned}\mu(S) = \mu''(S) &= x^{|S|-d_{\mu''}} \cdot \prod_{i\in S} \partial_{z_i} \prod_{i\in[k-1]\backslash S} (1 - x\partial_{z_i})(1 - x\partial_{z_k}) g_\mu(x\mathbf{1} + z)\Bigg|_{z=0} \\ &= x^{|S|-d_\mu} \cdot \prod_{i\in S} \partial_{z_i} \prod_{i\in[k]\backslash S} (1 - x\partial_{z_i}) g_\mu(x\mathbf{1} + z)\Bigg|_{z=0}.\end{aligned}$$

Hence, Eqn. (20.19) holds for all $k \in [n]$, which completes the proof of the fact. □

20.8.2 Defining interlacing family of characteristic polynomials

In this section, we consider the following family of polynomials:

Definition 20.17 (Interlacing Family of Theorem 20.54)**.** Let $h \in \mathbb{R}[x_1, \ldots, x_m]$ denote a degree-d hyperbolic polynomial with respect to hyperbolic direction $e \in \Gamma^h_{++}$. Let $\mu : 2^{[n]} \to \mathbb{R}$ be a homogeneous strongly Rayleigh probability distribution. Let $v_1, \ldots, v_n \in \Gamma^h_+$ be n vectors such that $rank_h(v_i) \le 1$ for all $i \in [n]$. Let $\mathcal{F} = \{S \subseteq [n] : \mu(S) > 0\}$ be the support of μ. For any $S \in \mathcal{F}$, let

$$q_S(x) = \mu(S) \cdot h\left(xe - \sum_{i\in S} v_i\right). \tag{20.20}$$

Let $\mathcal{Q}$ denote the following family of polynomials:

$$\mathcal{Q} := \Big\{ q_{s_1\cdots s_\ell}(s) = \sum_{\substack{t_{\ell+1},\cdots,t_n \\ (s_1\cdots s_\ell, t_{\ell+1},\cdots,t_n)\in\mathcal{F}}} q_{s_1,\cdots,s_\ell,t_{\ell+1},\cdots,t_n} : \forall \ell \in [n], (s_1, \cdots, s_\ell) \in \mathcal{F}|_{[\ell]} \Big\}.$$

where $\mathcal{F}|_{[\ell]}$ is $\mathcal{F}$ restricted to $[\ell]$, and a subset is represented by a binary indicator vector.

We show that the family defined above is an interlacing family. We will crucially use this fact to show Theorem 20.54.

Lemma 20.56 (Hyperbolic version of Theorem 3.3 in [AO14])**.** *Let $\mathcal{Q}$ denote the family of polynomials as in Definition 20.17. Then, if the polynomial*

$$\mathbb{E}_{\xi\sim\mu}\left[h(xe - (\sum_{i=1}^{n}\xi_i v_i))\right]$$

is real-rooted in x for any strongly Rayleigh distribution μ, then the polynomial family $\mathcal{Q}$ forms an interlacing family.

The proof of this lemma is the same as that of Theorem 3.3 in [AO14].

From Lemma 20.24 and Fact 20.19, we obtain the following corollary:

Corollary 20.57. *Let $h \in \mathbb{R}[x_1, \dots, x_m]$ denote a degree-d hyperbolic polynomial with respect to hyperbolic direction $e \in \Gamma^h_{++}$. Let $\mu : 2^{[n]} \to \mathbb{R}$ be a homogeneous strongly Rayleigh probability distribution. Let $v_1, \dots, v_n \in \Gamma^h_+$ be n vectors such that $\mathrm{rank}_h(v_i) \le 1$ for all $i \in [n]$. Let $\mathcal{F} = \{S \subseteq [n] : \mu(S) > 0\}$ be the support of μ. Then there exists $S \in \mathcal{F}$, such that the hyperbolic norm $\|\sum_{i\in S} v_i\|_h$ equals to the largest root of q_S, and is upper bounded by the largest root of*

$$q_\emptyset = \mathbb{E}_{\xi\sim\mu}\left[h(xe - (\sum_{i=1}^{n}\xi_i v_i))\right].$$

Proof. By Lemma 20.58 (we remark that this lemma does not depend on the results in this section. We will prove Lemma 20.58 in Section 20.8.3), the mixed characteristic polynomial

$$\mathbb{E}_{\xi\sim\mu}\left[h(xe - (\sum_{i=1}^{n}\xi_i v_i))\right]$$

is real-rooted. Then by Lemma 20.56, the polynomial family $\mathcal{Q}$ (see Definition 20.17) is an interlacing family. Therefore, by Lemma 20.23, there exists a subset $S \in \mathcal{F}$, such that the largest root of q_S, is upper bounded by the largest root of

$$q_\emptyset = \mathbb{E}_{\xi\sim\mu}\left[h(xe - (\sum_{i=1}^n \xi_i v_i))\right].$$

The corollary then follows from Fact 20.19. $\square$

20.8.3 From mixed characteristic polynomial to multivariate polynomial

In this section, we want to show that the mixed characteristic polynomial

$$\mathbb{E}_{\xi\sim\mu}\left[h(x^2 e - (\sum_{i=1}^n \xi_i v_i))\right]$$

has roots equals to the roots of the following multivariate polynomial after taking $z_1 = \cdots = z_n = 0$.

$$\prod_{i=1}^n (1 - \frac{1}{2}\partial_{z_i}^2)\Big(h(xe + \sum_{i=1}^n z_i v_i) g_\mu(x\mathbf{1} + z)\Big)$$

The largest root of this multivariate polynomial is relatively easy to upper-bound using barrier argument. We will describe the details in Section 20.8.4.

The main lemma of this section is as follows:

Lemma 20.58 (Hyperbolic version of Theorem 3.1 in [AO14])**.** *Let $h \in \mathbb{R}[x_1, \cdots, x_m]$ be a degree-d hyperbolic polynomial with hyperbolic direction $e \in \Gamma_{++}^h$. Let $\mu : 2^{[n]} \to \mathbb{R}$ be a d_μ-homogeneous strongly Rayleigh probability distribution with generating polynomial $g_\mu \in \mathbb{R}[z_1, \cdots, z_n]$. Let $v_1, \ldots, v_n \in \Gamma_+^h$ be n vectors such that $\mathrm{rank}_h(v_i) \le 1$ for all $i \in [n]$. Then, we have*

$$x^{d_\mu} \cdot \mathbb{E}_{\xi\sim\mu}\left[h(x^2 e - (\sum_{i=1}^n \xi_i v_i))\right] = x^d \cdot \prod_{i=1}^n (1 - \frac{1}{2}\partial_{z_i}^2)\Big(h(xe + \sum_{i=1}^n z_i v_i) g_\mu(x\mathbf{1} + z)\Big)\Big|_{z=0}. \tag{20.21}$$

Moreover,

$$\mathbb{E}_{\xi\sim\mu}\left[h(xe - (\sum_{i=1}^n \xi_i v_i))\right]$$

is real-rooted in x.

Proof. First, we rewrite the left-hand-side of (20.21) as the expectation over $T \sim \mu$ as an expectation over all indicator ξ of the subsets of $[n]$:

$$\begin{aligned}
& x^{d_\mu} \cdot 2^{-n} \cdot \mathbb{E}_{T\sim\mu}\left[h(x^2 e - (\sum_{i\in T} v_i))\right] \\
= {} & x^{d_\mu} \cdot 2^{-n} \cdot \sum_{\xi\in\{0,1\}^n}\left(h(x^2 e - (\sum_{i=1}^n \xi_i v_i)) \cdot \mu(\xi)\right) \\
= {} & x^{d_\mu} \cdot \mathbb{E}_{\xi\sim\{0,1\}^n}\left[h(x^2 e - \sum_{i=1}^n \xi_i v_i) \cdot \mu(\xi)\right] \\
= {} & x^{d_\mu} \cdot \mathbb{E}_{\xi\sim\{0,1\}^n}\left[h(x^2 e - \sum_{i=1}^n \xi_i v_i) \cdot x^{\sum_{i=1}^n \xi_i - d_\mu} \cdot \prod_{i=1}^n \big(\xi_i \partial_{z_i} + (1-\xi_i)(1 - z_i\partial_{z_i})\big) g(x\mathbf{1} + z)\Big|_{z=0}\right]
\end{aligned} \tag{20.22}$$

where in the third step, we let $\xi \in \{0,1\}^n$ be a random bit string uniformly sampled from $\{0,1\}^n$. In the last step we use Fact 20.55 and the fact that $\prod_{i=1}^n \big(\xi_i\partial_{z_i} + (1 - \xi_i)(1 - z_i\partial_{z_i})\big) = \prod_{i\in[n]:\xi_i=1} \partial_{z_i} \prod_{i\in[n]:\xi_i=0}(1 - z_i\partial_{z_i})$. Setting $g_2 \in \mathbb{R}[t]$ as

$$g_2(t) := x^{\sum_{i=2}^n \xi_i} \cdot \prod_{i=2}^n \big(\xi_i\partial_{z_i} + (1-\xi_i)(1 - x\partial_{z_i})\big) g(t, x + z_2, x + z_3, \cdots, x + z_n)\Big|_{z_2,\ldots,z_n=0}$$

and $x_2 = x^2 e - \sum_{i=2}^n \xi_i v_i$, we can simplify the above equation as

$$\begin{aligned}
& x^{d_\mu} \cdot 2^{-n} \cdot \mathbb{E}_{T\sim\mu}\left[h(x^2 e - (\sum_{i\in T} v_i))\right] \\
= {} & \frac{1}{2}\,\mathbb{E}_{\xi_2,\cdots,\xi_n\sim\{0,1\}^{n-1}}\left[h(x_2 - v_1)x\partial_{z_1} g_2(x + z_1) + h(x_2)(1 - x\partial z_1) g_2(x + z_1)\Big|_{z_1=0}\right] \\
= {} & \frac{1}{2}\,\mathbb{E}_{\xi_2,\cdots,\xi_n\sim\{0,1\}^{n-1}}\left[(1 - \partial_{z_1})h(x_2 + z_1 v_1)x\partial_{z_1} g_2(x + z_1) + h(x_2)(1 - x\partial_{z_1}) g_2(x + z_1)\Big|_{z_1=0}\right].
\end{aligned} \tag{20.23}$$

Now, we can expand the term inside the expectation of Eqn. (20.23), and get that

$$\begin{aligned}
&(1-\partial_{z_1})h(x_2+z_1v_1)x\partial_{z_1}g_2(x+z_1)+h(x_2)(1-x\partial_{z_1})g_2(x+z_1)\Big|_{z_1=0}\\
=\ & xh(x_2+z_1v_1)(\partial_{z_1}g_2(x+z_1))-x(\partial_{z_1}h(x_2+z_1v_1))(\partial_{z_1}g_2(x+z_1))\\
&+h(x_2)g_2(x+z_1)-h(x_2)(x\partial_{z_1}g_2(x+z_1))\Big|_{z_1=0}\\
=\ & xh(x_2)(Dg_2)(x)-x(D_{v_1}h)(x_2)(Dg_2)(x)+h(x_2)g_2(x)-xh(x_2)(Dg_2)(x)\\
=\ & h(x_2)g_2(x)-xD_{v_1}h(x_2)(Dg_2)(x)\\
=\ & (1-\frac{x}{2}\partial_{z_1}^2)\Big(h(x_2+z_1v_1)g_2(x+z_1)\Big)\Big|_{z_1=0},
\end{aligned}$$

where the last step follows from $rank_h(v_1)\le 1$ and $\deg(g_2)\le 1$.

Therefore, the left-hand-side of (20.21) equals to

$$\begin{aligned}
&x^{d_\mu}\cdot 2^{-n}\cdot \mathbb{E}_{\xi\sim\mu}\left[h(x^2e-(\sum_{i=1}^n\xi_iv_i))\right]\\
=\ & \frac{1}{2}\mathbb{E}_{\xi_2,\ldots,\xi_n}\left[(1-\frac{x}{2}\partial_{z_1}^2)\Big(h(x_2+z_1v_1)g_2(x+z_1)\Big)\Big|_{z_1=0}\right]\\
=\ & \frac{1}{2}(1-\frac{x}{2}\partial_{z_1}^2)\left(\mathbb{E}_{\xi_2,\ldots,\xi_n}\left[h(xe-\sum_{i=2}^n\xi_iv_i+z_1v_1)g_2(x+z_1)\right]\right)\Bigg|_{z_1=0}. \qquad (20.24)
\end{aligned}$$

If we repeat this process for n times, we will finally get

$$x^{d_\mu}\cdot\mathbb{E}_{\xi\sim\mu}\left[h(x^2e-(\sum_{i=1}^n\xi_iv_i))\right]=\prod_{i=1}^n(1-\frac{x}{2}\partial_{z_i}^2)\Big(h(x^2e+\sum_{i=1}^n z_iv_i)g_\mu(x\mathbf{1}+z)\Big)\Big|_{z=0}. \qquad (20.25)$$

Now we show that the right-hand-side of (20.25) equals to the right-hand-side of (20.21). First, we expand the product of partial operator $\prod_{i=1}^n(1-\frac{x}{2}\partial_{z_i}^2)$ and get that

$$\prod_{i=1}^n(1-\frac{x}{2}\partial_{z_i}^2)\Big(h(x^2e+\sum_{i=1}^n z_iv_i)g_\mu(x\mathbf{1}+z)\Big)\Big|_{z=0}=\sum_{T\subseteq[n]}(-\frac{x}{2})^{|T|}\partial_{z^T}^2\Big(h(x^2e+\sum_{i=1}^n z_iv_i)g_\mu(x\mathbf{1}+z)\Big)\Big|_{z=0}.$$

For any $T \subseteq [n]$ with $|T| = k$, we have

$$(-\frac{x}{2})^k \partial_{z^T}^2 \Big(h(x^2 e + \sum_{i=1}^n z_i v_i) g_\mu(x\mathbf{1} + z) \Big)\Big|_{z=0} = (-\frac{x}{2})^k \cdot 2^k \cdot \left(\prod_{i\in T} D_{v_i} \right) h(x^2 e) \cdot g_\mu^{(T)}(x\mathbf{1}),$$

where $g_\mu^{(T)}(x\mathbf{1}) = \prod_{i\in T} \partial_{z_i} g_\mu(x\mathbf{1} + z)\Big|_{z=0}$.

Since h is d-homogeneous, we know that $(\prod_{i\in T} D_{v_i})h$ is $(d-k)$-homogeneous. Hence, we get that

$$\begin{aligned}
(-\frac{x}{2})^k \partial_{z^T}^2 \Big(h(x^2 e + \sum_{i=1}^n z_i v_i) g_\mu(x\mathbf{1} + z) \Big)\Big|_{z=0} &= (-\frac{x}{2})^k \cdot 2^k \cdot x^{d-k} \cdot \left(\prod_{i\in T} D_{v_i} \right) h(xe) \cdot g_\mu^{(T)}(x\mathbf{1}) \\
&= x^d \cdot (-1)^k \cdot \left(\prod_{i\in T} D_{v_i} \right) h(xe) \cdot g_\mu^{(T)}(x\mathbf{1}) \\
&= x^d \cdot (-\frac{1}{2})^k \partial_{z^T}^2 \left(h(xe + \sum_{i=1}^n z_i v_i) g_\mu(x\mathbf{1} + z) \right)\Bigg|_{z=0}.
\end{aligned}$$

Therefore,

$$\begin{aligned}
x^{d_\mu} \cdot \mathbb{E}_{\xi\sim\mu} \left[h(x^2 e - (\sum_{i=1}^n \xi_i v_i)) \right] &= \sum_{T\subseteq[n]} x^d \cdot (-\frac{1}{2})^k \partial_{z^T}^2 \left(h(xe + \sum_{i=1}^n z_i v_i) g_\mu(x\mathbf{1} + z) \right)\Bigg|_{z=0} \\
&= x^d \cdot \prod_{i=1}^n (1 - \frac{1}{2}\partial_{z_i}^2) \Big(h(xe + \sum_{i=1}^n z_i v_i) g_\mu(x\mathbf{1} + z) \Big)\Big|_{z=0},
\end{aligned}$$

which completes the proof of Eqn. (20.21).

Finally, we show that Eqn. (20.21) is real-rooted in x. Since $e \in \Gamma_{++}^h$ and $v_1, \ldots, v_n \in \Gamma_+^h$, by Claim 20.51, we get that $h(xe + \sum_{i=1}^n z_i v_i)$ is real-stable. Furthermore, since $g_\mu(x\mathbf{1} + z)$ is real-stable, by Fact 20.17,

$$h(xe + \sum_{i=1}^n z_i v_i) g_\mu(x\mathbf{1} + z)$$

is also real-stable. Then by Fact 20.17 again, we get that

$$\prod_{i=1}^n (1 - \frac{1}{2}\partial_{z_i}^2) \Big(h(xe + \sum_{i=1}^n z_i v_i) g_\mu(x\mathbf{1} + z) \Big)\Big|_{z=0}$$

is real-stable. By Fact 20.14, it implies that it is real-rooted in x. Equivalently,

$$q_\emptyset(x^2) = \mathbb{E}_{\xi\sim\mu}\left[h(x^2 e - (\sum_{i=1}^n \xi_i v_i))\right]$$

is real-rooted in x. Then, it is easy to see that

$$q_\emptyset(x) = \mathbb{E}_{\xi\sim\mu}\left[h(xe - (\sum_{i=1}^n \xi_i v_i))\right]$$

is also real-rooted in x, since a complex root of $q_\emptyset(x)$ implies a complex root of $q_\emptyset(x^2)$.

The lemma is then proved. □

20.8.4 Applying barrier argument to bound the largest root of multivariate polynomial

In this section, we upper bound the largest root of the following multivariate polynomial:

$$\prod_{i=1}^n (1 - \frac{1}{2}\partial_{z_i}^2)\Big(h(xe + \sum_{i=1}^n z_i v_i) g_\mu(x\mathbf{1} + z)\Big)$$

using the real-stable version of the barrier method in [AO14].

Lemma 20.59 (Hyperbolic version of Theorem 4.1 in [AO14])**.** *Let $h \in \mathbb{R}[x_1, \cdots, x_m]$ denote a degree-d hyperbolic polynomial with hyperbolic direction $e \in \Gamma_{++}^h$. Let $v_1, \cdots, v_n \in \Gamma_+^h$ be n vectors such that $\sum_{i=1}^n v_i = e$ and $\mathrm{tr}_h[v_i] \leq \epsilon_2$ for all $i \in [n]$. Let $\mu : 2^{[n]} \to \mathbb{R}_{\geq 0}$ be a d_μ-homogeneous strongly Rayleigh probability distribution such that the marginal probability of each element $i \in [n]$ is at most ϵ_1. Then, all the roots of*

$$\prod_{i=1}^n (1 - \frac{1}{2}\partial_{z_i}^2)\Big(h(xe + \sum_{i=1}^n z_i v_i) g_\mu(x\mathbf{1} + z)\Big) \in \mathbb{R}[x, z_1, \cdots, z_n]$$

lie below $(\sqrt{4\epsilon + 2\epsilon^2}, 0, \cdots, 0) \in \mathbb{R}^{n+1}$, where $\epsilon = \epsilon_1 + \epsilon_2$.

Proof. Let $Q \in \mathbb{R}[x, z_1, \dots, z_n]$ be an $(n+1)$-variate polynomial:

$$Q(x,z) := h(xe + \sum_{i=1}^{n} z_i v_i) g_\mu(x\mathbf{1} + z).$$

We have already proved in Lemma 20.58 that $Q(x,z)$ is real-stable.

For $0 < t < \alpha$ where $\alpha = \alpha(t)$ is a parameter to be chosen later, we have

$$\begin{aligned} Q(\alpha, -t\mathbf{1}) &= h(\alpha e - t\sum_{i=1}^{n} v_i) g_\mu((\alpha - t)\mathbf{1}) \\ &= h((\alpha - t)e) g_\mu((\alpha - t)\mathbf{1}) \\ &= (\alpha - t)^{d + d_\mu} h(e) g_\mu(\mathbf{1}) \\ &> 0, \end{aligned}$$

where the second step follows from $\sum_{i=1}^{n} v_i = e$ and the last step follows from $\alpha > t$, $h(e) > 0$ and $g_\mu(\mathbf{1}) = 1$. This implies that $(\alpha, -t\mathbf{1}) \in \mathbf{Ab}_Q$.

We can upper bound $\Phi_Q^i(\alpha, -t\mathbf{1})$ as follows.

$$\begin{aligned} \Phi_Q^i(\alpha, -t\mathbf{1}) &= \frac{\partial_{z_i} Q}{Q} \bigg|_{x=\alpha, z=-t\mathbf{1}} \\ &= \frac{(\partial_{z_i} h(xe + \sum_{i=1}^{n} z_i v_i)) g_\mu(x\mathbf{1} + z) + h(xe + \sum_{i=1}^{n} z_i v_i)(\partial_{z_i} g_\mu(\mathbf{1} + z))}{h(xe + \sum_{i=1}^{n} z_i v_i) g_\mu(x\mathbf{1} + z)} \bigg|_{x=\alpha, z=-t\mathbf{1}} \\ &= \frac{\partial_{z_i} h(xe + \sum_{i=1}^{n} z_i v_i)}{h(xe + \sum_{i=1}^{n} z_i v_i)} + \frac{\partial_{z_i} g_\mu(x\mathbf{1} + z)}{g_\mu(x\mathbf{1} + z)} \bigg|_{x=\alpha, z=-t\mathbf{1}} \\ &= \frac{D_{v_i} h(xe + \sum_{i=1}^{n} z_i v_i)}{h(xe + \sum_{i=1}^{n} z_i v_i)} + \frac{\partial_{z_i} g_\mu(x\mathbf{1} + z)}{g_\mu(x\mathbf{1} + z)} \bigg|_{x=\alpha, z=-t\mathbf{1}} \\ &= \frac{D_{v_i} h(\alpha e - te)}{h(\alpha e - te)} + \frac{(\alpha - t)^{d_\mu - 1} \cdot \Pr_{S \sim \mu}[i \in S]}{(\alpha - t)^{d_\mu}} \\ &\leq \frac{tr[v_i]}{\alpha - t} + \frac{\epsilon_1}{\alpha - t} \\ &\leq \frac{\epsilon_1 + \epsilon_2}{\alpha - t}, \end{aligned}$$

where the first inequality follows from Fact 20.45.

Let $\epsilon := \epsilon_1 + \epsilon_2$. By choosing $\alpha = 2t = \sqrt{4\epsilon + 2\epsilon^2}$, we get that

$$\Phi_Q^i(\alpha, -t\mathbf{1}) \le \frac{\epsilon}{\sqrt{\epsilon + \epsilon^2/2}} = \frac{\sqrt{2}}{\sqrt{1 + 2/\epsilon}} < \sqrt{2}.$$

Then, by Lemma 20.26, we know that $(\alpha, -t\mathbf{1}) \in \mathbf{Ab}_{(1-\frac{1}{2}\partial_{z_i}^2)Q}$ for any $i \in [m]$.

Furthermore,

$$\frac{1}{t}\Phi_Q^i(\alpha, -t\mathbf{1}) + \frac{1}{2}\Phi_Q^i(\alpha, -t\mathbf{1})^2 \le \frac{\epsilon}{t^2} + \frac{1}{2}\frac{\epsilon^2}{t^2} = 1.$$

By the second part of Lemma 20.26, we have for all $j \in [m]$,

$$\Phi^j_{(1-\frac{1}{2}\partial_{z_i}^2)Q}(\alpha, -t\mathbf{1} + t\mathbf{1}_i) \le \Phi_Q^j(\alpha, -t\mathbf{1}).$$

By a similar induction process like in the proof of Lemma 20.53, we have

$$\begin{aligned}(\alpha, -t\mathbf{1} + \sum_{i=1}^n t\mathbf{1}_i) = (\sqrt{4\epsilon + 2\epsilon^2}, 0, 0, \dots, 0)\\ \in \mathbf{Ab}_{\prod_{i=1}^n (1-\partial_{z_i}^2/2)Q}\end{aligned}$$

i.e. $(\sqrt{4\epsilon + 2\epsilon^2}, 0, 0, \dots, 0)$ lies above the roots of

$$\prod_{i=1}^n \Big(1 - \frac{1}{2}\frac{\partial^2}{\partial z_i^2}\Big)\left[h\Big(xe + \sum_{i=1}^n \xi_i v_i\Big) \cdot g_\mu(x\mathbf{1} + z)\right]$$

as desired. $\square$

20.8.5 Combining together: proof of Theorem 20.54

Now we can combine the results from the previous section and prove Theorem 20.54:

Proof of Theorem 20.54. Let $\mathcal{F}$ be the support of μ.

By Lemma 20.59 and restricting to $z_i = 0$ for all $i \in [n]$, we have that $\sqrt{4(\epsilon_1 + \epsilon_2) + 2(\epsilon_1 + \epsilon_2)^2}$ lies above the largest root of the univariate polynomial

$$\prod_{i=1}^n (1 - \frac{1}{2}\partial_{z_i}^2)\Big(h(xe + \sum_{i=1}^n z_i v_i) g_\mu(x\mathbf{1} + z)\Big)\Big|_{z=0}.$$

We then conclude by Lemma 20.58 that $\sqrt{4(\epsilon_1+\epsilon_2)+2(\epsilon_1+\epsilon_2)^2}$ upper bounds the largest root of

$$\mathbb{E}_{\xi\sim\mu}\left[h(x^2e-(\sum_{i=1}^n \xi_i v_i))\right].$$

Therefore, the largest root of

$$q_\emptyset = \mathbb{E}_{\xi\sim\mu}\left[h(xe-(\sum_{i=1}^n \xi_i v_i))\right]$$

is upper bounded by $4(\epsilon_1+\epsilon_2)+2(\epsilon_1+\epsilon_2)^2$, where $q_\emptyset$ is the average of the polynomials in the interlacing family $\mathcal{Q}$ in Definition 20.17.

Finally, by Corollary 20.57, there exists $S\subseteq[n]$ in the support of μ, such that

$$\Big\|\sum_{i\in S} v_i\Big\|_h \le 4(\epsilon_1+\epsilon_2)+2(\epsilon_1+\epsilon_2)^2.$$

$\square$

20.9 Sub-Exponential Algorithms

20.9.1 Definitions

Definition 20.18 (k-th symmetric polynomials)**.** For any $n\in\mathbb{Z}_+, k\le n$, let $e_k\in\mathbb{R}[z_1,\cdots,z_n]$ denote the k-th elementary symmetric polynomial defined as

$$e_k(z_1,\cdots,z_n)=\sum_{T\in\binom{[n]}{k}}\prod_{i\in T} z_i.$$

Definition 20.19 (k-th power sum polynomials)**.** For any $n,k\in\mathbb{Z}_+$, let $p_k\in\mathbb{R}[z_1,\cdots,z_n]$ denote the k-th power sum polynomial defined as $p_k(z_1,\cdots,z_n)=\sum_{i=1}^n z_i^k$.

Fact 20.60 (Vieta's formulas)**.** *Let $f\in\mathbb{R}[x]$ be any degree n monic variate polynomial defined as $f(x)=x^n+c_1x^{n-1}+\cdots+c_n$. Then for any $k\in[n]$,*

$$c_k=(-1)^k e_k(\lambda_1,\cdots,\lambda_n).$$

Where $\lambda_1, \cdots, \lambda_n \in \mathbb{C}$ are the roots of f, and e_k is the k-th elementary symmetric polynomial defined in Definition 20.18.

Fact 20.61 (Newton's identities)**.** *For any $n \in \mathbb{Z}_+, k \in [n]$, let $e_k, p_k \in \mathbb{R}[z_1, \cdots, z_n]$ be defined in Definition 20.18 and Definition 20.19 respectively. Then there exists an $O(k^2)$-time algorithm* ELEMTOPOWER$(k, e_1, \cdots, e_k)$*, such that given any $k \in [n]$, and any $e_1 = e_1(x), \cdots, e_k = e_k(x)$ for some fixed $x \in \mathbb{R}^n$, outputs $p_k = p_k(x)$.*

20.9.2 Algorithm to approximate the largest root

Algorithm 116 Algorithm approximating the largest root.

```
1: procedure MaxRoot($n \in \mathbb{Z}_+, k \in [n], f \in \mathbb{R}[x], c_1, \cdots, c_k \in \mathbb{R}$)
2:     Precondition: $k \le n$, $c_1, \cdots, c_k \in \mathbb{R}$ are the top-$k$ coefficients of a degree $n$
       real-rooted monic-variate polynomial $f \in \mathbb{R}[x]$.
3:     Output: Return an approximate of the largest root of $f$.
4:     for $i = 1$ to $k$ do
5:         $e_k \leftarrow c_k \cdot (-1)^k$.
           ▷ $e_k = e_k(\lambda_1, \cdots, \lambda_n)$ is the $k$-th elementary polynomial of the roots of $f$ by
       Fact 20.60.
6:     end for
7:     $p_k \leftarrow$ ElemToPower($k, e_1, \cdots, e_k$)
           ▷ $p_k(\lambda) = \lambda_1^k + \cdots + \lambda_n^k$ is the power sum of the roots of $f$ by Fact 20.61.
8:     Return $(p_k)^{1/k}$.
9: end procedure
```

Lemma 20.62. *Let $f = x^n + c_1 x^{n-1} + \cdots + c_n \in R[x]$ be any degree n real-rooted monic-variate polynomial. Then* MAXROOT$(n, k, f, c_1, \cdots, c_k)$ *(Algorithm 116) returns an $(n^{1/k})$-approximate of the largest root of f in time $O(k^2 + k)$.*

Proof. Let $\lambda_1 \ge \lambda_2 \ge \lambda_n \in \mathbb{R}$ denote the roots of $f(x)$. By Fact 20.60 and Fact 20.61, Algorithm 116 returns $(p_k)^{1/k}$, where $p_k = \lambda_1^k + \cdots + \lambda_n^k$. Notice that

$$\frac{\lambda_1^k + \cdots + \lambda_n^k}{n} \le \lambda_1^k \le \lambda_1^k + \cdots + \lambda_n^k$$

we have

$$\lambda_1 \leq (p_k/n)^{1/k} \leq n^{1/k}\lambda_1.$$

$\square$

Remark 20.8. We remark that when $k > \log n$, the approximation factor $n^{1/k}$ is upper bounded by $1 + \frac{\log n}{k}$.

20.9.3 Reducing Kadison-Singer to finding leading coefficients of interlacing polynomial

We define an oracle that generates the top-k coefficients as follows:

Definition 20.20. Fix a family of degree n monic-variable polynomials $\mathcal{F} = \{f_{s_1,\cdots,s_m} \in \mathbb{R}[x] : s_1, \cdots s_\ell \in S_1 \times \cdots \times S_\ell\}$. We define oracle $\textsc{MaxCoeff}_{\mathcal{F}}(k, \ell, s_1, \cdots s_\ell)$ as follows: given any $k \in [n], \ell \in [m], s_1, \cdots s_\ell \in S_1 \times \cdots \times S_\ell$ as inputs, and it outputs the top-k coefficients of $f_{s_1 \cdots s_\ell}$ in $\mathcal{T}_{\mathrm{Coeff}}(\mathcal{F}, k)$ time.

Lemma 20.63 (Theorem 4.4 of [AOSS18]). *Let $S_1, \cdots, S_m$ be finite sets and let $\mathcal{F} = \{f_{s_1 \cdots s_m}(x) \in \mathbb{R}[x] : s_1 \in S_1, \cdots, s_m \in S_m\}$ be an interlacing family of degree n real-rooted monic-variate polynomials. Let $\textsc{MaxCoeff}_{\mathcal{F}}(k, \ell, s_1, \cdots s_\ell)$ be the oracle for finding the largest k coefficients defined as Definition 20.20.*

Then there is an algorithm $\textsc{KadisonSinger}(\delta, \mathcal{F} = \{f_{s_1 \cdots s_m}(x) : s_1 \in S_1, \cdots, s_m \in S_m\})$ (Algorithm 117) that, given any $\delta > 0$ and $\mathcal{F}$ as input, returns the elements $s'_1 \in S_1, \cdots, s'_m \in S_m$, such that the maximum root of $f_{s'_1 \cdots s'_m}$ is at most $(1+\delta)$ times the maximum root of $f_\emptyset$, in time

$$\left(\mathcal{T}_{\mathrm{Coeff}}\left(\mathcal{F}, O(\log(n) m \delta^{-1})\right) + k^2\right) \cdot \mathcal{S}^{\sqrt{m}} \cdot \sqrt{m},$$

where $\mathcal{S} = \max_{i \in [m]} |S_i|$.

Algorithm 117 Algorithm finding approximate solutions of Kadison-Singer problems

```
1: procedure KadisonSinger($\delta, \mathcal{F} = \{f_{s_1 \cdots s_m}(x) : s_1 \in S_1, \cdots, s_m \in S_m\}$)
2:     Precondition: $\delta > 0$, $S_1, \cdots, S_m$ are finite sets, $\mathcal{F} = \{f_{s_1 \cdots s_m}(x) \in \mathbb{R}[x] : s_1 \in S_1, \cdots, s_m \in S_m\}$ is an interlacing family of degree $n$ real-rooted monic-variate polynomials.
3:     Output: $s'_1 \in S_1, \cdots, s'_m \in S_m$, such that the maximum root of $f_{s'_1 \cdots s'_m}$ is at most $(1+\delta)$ times the maximum root of $f_\emptyset$.
4:     $M \leftarrow \sqrt{m}$
5:     $k \leftarrow \frac{M \log n}{\delta}$
6:     $\lambda_{\max} \leftarrow -\infty$
7:     for $i = 0$ to $\frac{m}{M} - 1$ do                              ▷ $\sqrt{m}$ iterations
8:         for $t_{iM+1} \in S_{iM+1}, \cdots, t_{iM+M} \in S_{iM+M}$ do      ▷ Brute force search on $\prod_{j=1}^{M} |S_{iM+j}|$ elements
9:             $(c_1, \cdots, c_k) \leftarrow$ MaxCoeff$_{\mathcal{F}}(k, M(i+1), s_1, \cdots, s_{iM}, t_{iM+1}, \cdots, t_{iM+M})$
10:            $\lambda \leftarrow$ MaxRoot$(n, k, f_{s_1, \cdots, s_{iM}, t_{iM+1}, \cdots, t_{iM+M}}, c_1, \cdots, c_k)$
11:            if $\lambda \le \lambda_{\min}$ then
12:                $\lambda_{\min} \leftarrow \lambda$
13:                $s_{iM+1} \leftarrow t_{iM+1}, \cdots, s_{iM+M} \leftarrow t_{iM+M}$
14:            end if
15:        end for
16:    end for
17:    Return $(s_1, \cdots, s_m)$
18: end procedure
```

Proof. Notice that Algorithm 117 runs in $\frac{m}{M} = \sqrt{M}$ iterations. Inside each iteration, it does a brute force search on at most $\mathcal{Q}^M = \mathcal{Q}^{\sqrt{m}}$ elements. For each element $(t_{iM+1}, \cdots, t_{iM+M})$, we query the oracles MaxCoeff$_{\mathcal{F}}$ and MaxRoot, which takes $\mathcal{T}_{\text{Coeff}}(\mathcal{F}, k) = \mathcal{T}_{\text{Coeff}}(\mathcal{F}, O(\log(n) m \delta^{-1})$ and $O(k + k^2)$ time respectively. Therefore, the running time of the algorithm is at most

$$\left(\mathcal{T}_{\text{Coeff}}\left(\mathcal{F}, O(\log(n) m \delta^{-1})\right) + k^2\right) \cdot \mathcal{S}^{\sqrt{m}} \cdot \sqrt{m}.$$

Now we show the correctness of the algorithm by induction on the number of iteration i. For any $0 \le i \le \frac{m}{M} - 1$, suppose we have selected $s_1, \cdots, s_{iM}$ in the previous iterations, and suppose

$$\lambda_{\max}\left(f_{s_1,\cdots,s_{iM}}\right) \leq (1+\frac{\delta}{2M})^{i+1} \cdot \lambda_{\max}(f_{\emptyset})$$

where $\lambda_{\max}(f)$ denotes the maximum root of the univariate polynomial f.

Suppose we fix $s_{iM+1}, \cdots, s_{iM+M}$ on the i-th iteration. Note that $k = \frac{2M\log n}{\delta} > \log n$, by Lemma 20.62, Line 10 of the algorithm returns an $1+\frac{\log n}{k} = (1+\frac{\delta}{2M})$-approximation of the largest root of $f_{s_1,\cdots,s_{iM},t_{iM+1},\cdots,t_{iM+M}}$. Therefore, for any $0 \leq i \leq \frac{m}{M}$, we have

$$\lambda_{\max}\left(f_{s_1,\cdots,s_{iM+M}}\right) \leq (1+\frac{\delta}{2M}) \cdot \min_{t_{iM+1}\in S_{iM+1},\cdots,t_{iM+M}\in S_{iM+M}} \lambda_{\max}\left(f_{s_1,\cdots,s_{M}i,t_{iM+1},\cdots,t_{iM+M}}\right)$$

Since $\{f_{s_1\cdots s_m}(x) : s_1 \in S_1, \cdots, s_m \in S_m\}$ is an interlacing family, we have

$$\min_{t_{iM+1}\in S_{iM+1},\cdots,t_{iM+M}\in S_{iM+M}} \lambda_{\max}\left(f_{s_1,\cdots,s_{M}i,t_{iM+1},\cdots,t_{iM+M}}\right) \leq \lambda_{\max}\left(f_{s_1,\cdots,s_{iM}}\right).$$

This proves the induction hypothesis. $\square$

Remark 20.9. It's important to note that our algorithm runs in $\mathcal{S}^{\widetilde{O}(\sqrt{n})}$ time. A paper by [AOSS18] provides an algorithm with a faster running time of $\mathcal{S}^{\widetilde{O}(\sqrt[3]{n})}$. However, this algorithm is limited to only the k-th largest root of *determinantal polynomials.*

Our algorithm faces a challenge in approximating the k-th largest root of *hyperbolic polynomials.* In order to achieve an $(1+\epsilon)$ approximation, our sub-exponential algorithm must run $\widetilde{O}(\sqrt{n}\epsilon)$ iterations and search $k = \widetilde{O}(\sqrt{n}/\epsilon)$ elements at each time. This ensures that the cumulative error does not exceed ϵ. During each iteration, we have to brute-force over $O(\sqrt{n})$ elements, which results in a $2^{\widetilde{O}(n)}$ search time.

20.9.4 Sub-exponential algorithm for Theorem 20.10

In this section, we want to describe the sub-exponential algorithm for constructing Theorem 20.10. Let $\mathcal{P}$ denote the interlacing family as defined in Definition 20.13.

Suppose each $p_{\mathbf{s}}(x) \in \mathcal{P}$ has degree d. Let $\textsc{MaxCoeff}_{\mathcal{P}}(k, \ell, s_1, \cdots, s_\ell)$ be the oracle defined in Definition 20.20, i.e. given any $k \in [d]$, $\ell \in [n]$, $(s_1, \cdots, s_\ell) \in \{\pm 1\}^\ell$, outputs the top-$k$ coefficients of $p_{s_1,\cdots,s_\ell}$ in at most $\mathcal{T}_{\mathrm{Coeff}}(\mathcal{P}, k)$ time. The following lemma states that if $\mathcal{T}_{\mathrm{Coeff}}(\mathcal{P}, k)$ is polynomial in k, then we can construct Theorem 20.10 in sub-exponential time:

Corollary 20.64 (Sub-exponential algorithm for Theorem 20.10, formal statement of Proposition 20.12)**.** *Let $h \in \mathbb{R}[x_1, \ldots, x_m]$ denote a hyperbolic polynomial with respect to $e \in \Gamma^h_{++}$. Let $u_1, \ldots, u_n \in \Gamma^h_+$ be n vectors such that*

$$\sigma = \Big\| \sum_{i=1}^n tr_h[u_i] u_i \Big\|_h .$$

Let $\mathcal{P}$ be the interlacing family defined in Definition 20.13. Let $\textsc{MaxCoeff}_{\mathcal{P}}$ be the oracle defined in Definition 20.20 with running time $\mathcal{T}_{\mathrm{Coeff}}(\mathcal{P}, k)$. Then for any $\delta > 0$, the algorithm KadisonSinger$(\delta, \mathcal{P})$ *(Algorithm 117) returns a sign assignment $(s_1, \cdots, s_n) \in \{\pm 1\}^n$, such that*

$$\Big\| \sum_{i=1}^n s_i u_i \Big\|_h \leq 4(1+\delta)\sigma$$

in time

$$\left(\mathcal{T}_{\mathrm{Coeff}}\left(\mathcal{P}, O(\log(n) m \delta^{-1})\right) + \frac{m \log^2 n}{\delta^2} \right) \cdot 2^{\sqrt{m}} \cdot \sqrt{m}.$$

Proof. For all $i \in [n]$, let $v_i = \frac{u_i}{\sqrt{\sigma}}$. Then we have

$$\left\| \sum_{i=1}^n tr_h[v_i] v_i \right\|_h = \left\| \sum_{i=1}^n \frac{tr_h[u_i] u_i}{\sigma} \right\|_h = 1.$$

Let $s_1, \cdots, s_m$ denote the output of $\textsc{KadisonSinger}(\sigma, \mathcal{P})$ (Algorithm 117). Then we have

$$\begin{aligned} \| \sum_{i=1}^m s_i v_i \|_h &= \lambda_{\max}(p_{s_1,\cdots,s_m}) \\ &\leq (1+\delta) \cdot \lambda_{\max}(p_\emptyset) && \text{(By Lemma 20.63)} \\ &\leq 4(1+\delta) && \text{(By Lemma 20.53)} \end{aligned}$$

Therefore we have

$$\| \sum_{i=1}^{m} s_i u_i \|_h \leq 4\sigma(1+\delta) \leq 8\sigma.$$

$\square$

20.9.5 Sub-exponential algorithm for Theorem 20.4

In particular, we can explicitly compute the running time of the sub-exponential algorithm for Theorem 20.4. We first define the interlacing family for Theorem 20.4 with the following statement:

Lemma 20.65 (Interlacing family for Theorem 20.4, Proposition 4.1 and 5.4 of [KLS20]). *Let $\xi_1, \cdots, \xi_n$ denote n i.i.d. random variables sampled uniformly at random from $\{\pm 1\}$. Let $u_1, \ldots, u_n \in \Gamma_+^h$ be n vectors such that $\sigma^2 = \| \sum_{i=1}^n (u_i u_i^*)^2 \|$*

For each $\mathbf{s} \in \{\pm 1\}^n$, let $f_{\mathbf{s}} \in \mathbb{R}[x]$ denote the following polynomial:

$$f_{\mathbf{s}}(x) := (\prod_{i=1}^{n} p_{i,s_i}) \cdot \det\left(x^2 - \left(\sum_{i=1}^{n} (s_i - \lambda_i) u_i u_i^* \right)^2 \right).$$

where $\forall i \in [n]$, $\lambda_i = \mathbb{E}[\xi_i]$, and $\forall i \in [n], \forall s_i \in \{\pm 1\}$, $p_{i,s_i} = \Pr_{\xi_i}[\xi_i = s_i]$. Let $\mathcal{F}$ denote the following family of polynomials:

$$\mathcal{F} := \left\{ f_{s_1 \cdots s_\ell}(x) = \sum_{t_{\ell+1} \in \{\pm 1\}, \cdots, t_n \in \{\pm 1\}} f_{s_1, \cdots, s_\ell, t_{\ell+1}, \cdots, t_n} : \forall \ell \in [n], s_1, \cdots, s_\ell \in \{\pm 1\}^\ell \right\}.$$

Then $\mathcal{F}$ is an interlacing family. Moreover, there exists a choice of outcomes $s_1, \cdots, s_n \in \{\pm 1\}^n$, such that

$$\| \sum_{i=1}^{n} (s_i - \lambda_i) u_i u_i^* \| = \lambda_{\max}(f_{s_1 \cdots s_n}) \leq \lambda_{\max}(f_\emptyset) \leq 4\sigma.$$

Lemma 20.66 (Computing the top k coefficients of interlacing polynomials for Theorem 20.4). *Let $\mathcal{F}$ denote the interlacing family defined in Lemma 20.65. Given independent random variable $\xi_1, \ldots, \xi_n \in \mathbb{R}$ with finite supports such that we know all moments of each random variable. There exists an algorithm* $\textsc{MaxCoeff}_{\mathcal{F}}(k, \ell, s_1, \cdots, s_\ell)$

such that for any $1 \leq \ell \leq n$ and $1 \leq k \leq m$, and for any $s_1, \ldots, s_\ell$ in the supports of $\xi_1, \ldots, \xi_\ell$ respectively, returns the top-k coefficients of $f_{s_1,\ldots,s_\ell}$ in time $O(n^k poly(m))$.

Proof. The leading constant $\prod_{i=1}^{\ell} p_{i,s_i}$ can be easily computed. It is easy to see that all odd-degree terms vanish. Thus, for the following expected polynomial:

$$\mathbb{E}_{\xi_{\ell+1},\ldots,\xi_n}\left[\det\left(x^2 I - \left(\sum_{i=1}^{\ell} s_i u_i u_i^* + \sum_{i=\ell+1}^{n} \xi_i u_i u_i^*\right)^2\right)\right],$$

consider the coefficient of $x^{2(n-k)}$, which is $(-1)^k$ times the sum of all principal k-by-k minors of the matrix $(\sum_{i=1}^{\ell} s_i u_i u_i^* + \sum_{i=\ell+1}^{n} \xi_i u_i u_i^*)^2$ in expectation. For any fixed value $\xi_{\ell+1}, \ldots, \xi_n$, we have

$$\begin{aligned}
&\sigma_k\left(\left(\sum_{i=1}^{\ell} s_i u_i u_i^* + \sum_{i=\ell+1}^{n} \xi_i u_i u_i^*\right)^2\right) \\
= {}&\sigma_k\left(\sum_{i,j\in[\ell]} s_i s_j \langle u_i, u_j\rangle u_i u_i^* + \sum_{i\in[\ell], j\in[n]\backslash[\ell]} s_i \xi_j \langle u_i, u_j\rangle (u_i u_j^* + u_j u_i^*) + \sum_{i,j\in[n]\backslash[\ell]} \xi_i \xi_j \langle u_i, u_j\rangle u_i u_i^*\right).
\end{aligned}$$

For simplicity, let

$$c_{i,j} := \begin{cases} s_i s_j \langle u_i, u_j\rangle & \text{if } i, j \in [\ell], \\ \xi_i s_j \langle u_i, u_j\rangle & \text{if } i \in [n]\backslash[\ell], j \in [\ell], \\ s_i \xi_j \langle u_i, u_j\rangle & \text{if } i \in [\ell], j \in [n]\backslash[\ell], \\ \xi_i \xi_j \langle u_i, u_j\rangle & \text{if } i, j \in [n]\backslash[\ell] \end{cases}, \ \forall i, j \in [n] \times [n].$$

Then, we have

$$\begin{aligned}
\sigma_k\left(\sum_{i,j\in[n]\times[n]} c_{i,j} u_i u_j^*\right) &= \sum_{S\in\binom{[n]\times[n]}{k}} \sigma_k\left(\sum_{(i,j)\in S} c_{i,j} u_i u_j^*\right) \\
&= \sum_{S\in\binom{[n]\times[n]}{k}} \prod_{(i,j)\in S} c_{i,j} \cdot \sigma_k\left(\sum_{(i,j)\in S} u_i u_j^*\right),
\end{aligned}$$

where the first step follows from Proposition 3.11 in [MSS15b] and the second step follows from Proposition 3.10 in [MSS15b].

Thus, the coefficient of the expected polynomial is

$$\sum_{S\in\binom{[n]\times[n]}{k}} \mathbb{E}_{\xi_{\ell+1},\ldots,\xi_n}\left[\prod_{(i,j)\in S} c_{i,j}\right]\cdot \sigma_k\left(\sum_{(i,j)\in S} u_i u_j^*\right).$$

Note that there are at most n^{2k} terms in the summation. And for each S, the expectation $\mathbb{E}_{\xi_{\ell+1},\ldots,\xi_n}\left[\prod_{(i,j)\in S} c_{i,j}\right]$ can be computed in $O(k)$-time, assuming we know all moments of each random variable, and the inner product $\langle u_i, u_j\rangle$ for all $i,j\in[n]$ can be pre-processed in $O(n^2m)$-time. The sum of minors $\sigma_k(\sum_{(i,j)\in S} u_i u_j^*)$ can be computed in $poly(m)$-time.

Therefore, the coefficient of $x^{2(m-k)}$ can be computed in $n^{2k}\cdot poly(m)$-time, which implies that the top-k coefficients can be computed in $O(n^k poly(m))$-time. The Lemma is then proved. $\square$

A similar proof of Corollary 20.64 yields the following corollary:

Corollary 20.67 (Sub-exponential algorithm for Theorem 20.4)**.** *Let $u_1,\ldots,u_n\in\Gamma_+^h$ be n vectors such that $\sigma^2 = \left\|\sum_{i=1}^n (u_i u_i^*)^2\right\|$. Let $\mathcal{F}$ be the interlacing family defined in Lemma 20.65. Then for any $\delta>0$, the algorithm* KadisonSinger$(\delta,\mathcal{F})$ *returns a sign assignment $(s_1,\cdots,s_n)\in\{\pm1\}^n$, such that*

$$\left\|\sum_{i=1}^n s_i u_i\right\|_h \le 4(1+\delta)\sigma$$

in time

$$\left(O(n^{O(\sqrt{m}\log n/\delta)}poly(m)) + \frac{m\log^2 n}{\delta^2}\right)\cdot 2^{\sqrt{m}}\cdot\sqrt{m}.$$

20.9.6 Sub-exponential algorithm for Theorem 20.11

In this section, we want to describe the sub-exponential algorithm for constructing Theorem 20.11. Let $\mathcal{Q}$ denote the interlacing family defined in Definition 20.17.

Suppose each $q_{\mathbf{s}}(x) \in \mathcal{Q}$ has degree d. Let $\textsc{MaxCoeff}_{\mathcal{Q}}(k, \ell, s_1, \cdots, s_\ell)$ be the oracle defined in Definition 20.20, i.e. given any $k \in [d]$, $\ell \in [n]$, $(s_1, \cdots, s_\ell)$ in the support of μ, outputs the top-k coefficients of $q_{s_1,\cdots,s_\ell}$ in at most $\mathcal{T}_{\mathrm{Coeff}}(\mathcal{Q}, k)$ time. The following corollary states that if $\mathcal{T}_{\mathrm{Coeff}}(\mathcal{Q}, k)$ is polynomial in k, then we can construct Theorem 20.11. The proof of this corollary is similar to that of Corollary 20.64.

Corollary 20.68 (Sub-exponential algorithm for Theorem 20.11, formal statement of Proposition 20.13)**.** *Let $h \in \mathbb{R}[x_1, \ldots, x_m]$ denote a hyperbolic polynomial with respect to $e \in \Gamma^h_{++}$. Let μ be a homogeneous strongly Rayleigh probability distribution on $[n]$ such that the marginal probability of each element is at most ϵ_1, and let $v_1, \cdots, v_n \in \Gamma^h_+$ be n vectors such that $\sum_{i=1}^n v_i = e$, and for all $i \in [n]$, $\|v_i\|_h \leq \epsilon_2$ and $\mathrm{rank}_h(v_i) \leq 1$.*

Let $\mathcal{Q}$ be the interlacing family defined in Definition 20.17 , and $\textsc{MaxCoeff}_{\mathcal{Q}}$ be the oracle defined in Definition 20.20 with running time $\mathcal{T}_{\mathrm{Coeff}}(\mathcal{Q}, k)$. Then for any $\delta > 0$, the algorithm KadisonSinger$(\delta, \mathcal{Q})$ *returns a set S in the support of μ, such that*

$$\Big\| \sum_{i \in S} u_i \Big\|_h \leq (1+\delta) \cdot \big(4(\epsilon_1 + \epsilon_2) + 2(\epsilon_1 + \epsilon_2)^2\big)$$

in time

$$\big(\mathcal{T}_{\mathrm{Coeff}}\left(\mathcal{Q}, O(\log(n) m \delta^{-1})\right) + k^2\big) \cdot 2^{\sqrt{m}} \cdot \sqrt{m}.$$

20.10 Examples and Discussions

Examples of real-stable polynomials

- *Spanning tree polynomial:* let $G = (V, E)$ be a connected undirected graph. Then its spanning tree polynomial

$$P_G(x) = \sum_{\substack{T \subset E, \\ T \text{ spanning tree}}} \prod_{e \in T} x_e \tag{20.26}$$

  is real-stable.

- *Elementary Symmetric Polynomials:* For any $n, k > 0$, the elementary symmetric polynomial

$$e_k(x) = \sum_{S \in \binom{[n]}{k}} \prod_{i \in S} x_i$$

is real-stable.

- *Vertex matching polynomial:* let $G = (V, E)$ be a undirected graph. Then its vertex matching polynomial

$$M_G(x) = \sum_{\substack{M \subset E, \\ M \text{ matching}}} \prod_{\{u,v\} \in M} -x_u x_v \tag{20.27}$$

is real-stable [BB09].

- *Vámos matroid polynomial:* let $\mathcal{B}$ consists of all subsets $B \subset [10]$ of size 4 except for $\{1,2,3,4\}, \{1,2,5,6\}, \{1,2,7,8\}, \{1,2,9,10\}, \{3,4,5,6\}, \{5,6,7,8\}, \{7,8,9,10\}$. Then, $\mathcal{B}$ is the collection of basis of a Vámos matroid. See Figure 20.1 as an illustration of $\mathcal{B}$. Its generating polynomial

$$f_{10}(x) = \sum_{B \in \mathcal{B}} \prod_{i \in B} x_i$$

is real-stable. Furthermore, for any $k > 0$, $(f_{10}(x))^k$ cannot be represented as a determinant of a linear matrix with positive semidefinite Hermitian forms [BVY14].

- *Determinant of the mixture of PSD matrices:* let $A_1, \ldots, A_n \in \mathbb{R}^{d \times d}$ be PSD matrix and $B \in \mathbb{R}^{d \times d}$ be symmetric. Then, the polynomial

$$p(x) = \det(x_1 A_1 + \cdots + x_n A_n + B)$$

is real-stable.

Figure 20.2 and Figure 20.3 illustrate an example of the spanning tree polynomial (Eqn. (20.26)) and vertex matching polynomial (Eqn. (20.27)) respectively.

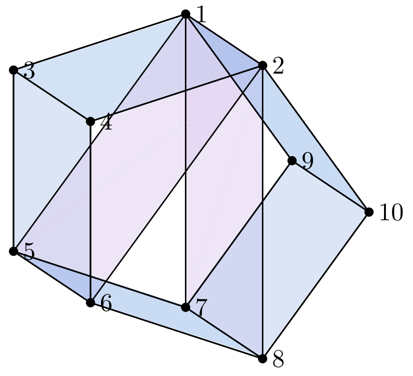


Figure 20.1: An example of the Vámos matroid $\mathcal{B}$. $\mathcal{B}$ contains all the sets in $\binom{[10]}{4}$, except the sets of size 4 represented by the colored faces.

Examples of hyperbolic polynomials

- *Lorentz polynomial:*

$$p(x) = x_n^2 - x_1^2 - \cdots - x_{n-1}^2$$

is hyperbolic with respect to $e = \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix}^\top$.

- *Determinant polynomial:*

$$p(x) = \det(\mathrm{mat}(x)) \in \mathbb{R}[\{x_{i,j}\}_{1 \le i \le j \le n}]$$

is hyperbolic with respect to $e = \mathrm{vec}(I)$, where $\mathrm{mat}(\cdot)$ packs an $(n(n+1)/2)$-dimensional vector to an n-by-n symmetric matrix, and $\mathrm{vec}(\cdot)$ vectorize a symmetric matrix to a vector.

- *Multivariate matching polynomial:* let $G = (V, E)$ be an undirected graph. Then

$$\mu_G(x, w) = \sum_{\substack{M \subset E, \\ M \text{ matching}}} (-1)^{|M|} \cdot \prod_{u \notin V(M)} x_u \cdot \prod_{e \in M} w_e^2$$

is hyperbolic [Ami19] with respect to $e = \begin{bmatrix} \mathbf{1}_V & \mathbf{0} \end{bmatrix}^\top$.

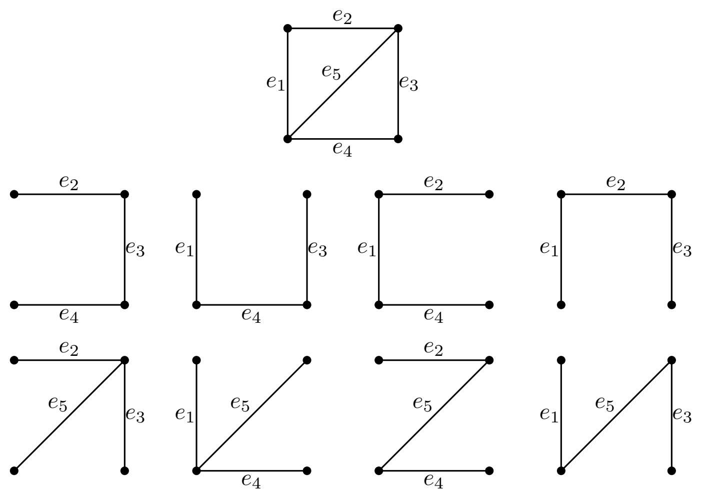


Figure 20.2: An example of the spanning tree polynomial. Above: the graph G; Below: the set of spanning trees of G. Therefore the spanning tree polynomial of G is $P_G(x) = x_2x_3x_4 + x_1x_3x_4 + x_1x_2x_4 + x_1x_2x_3 + x_2x_3x_5 + x_1x_4x_5 + x_2x_4x_5 + x_1x_3x_5$.

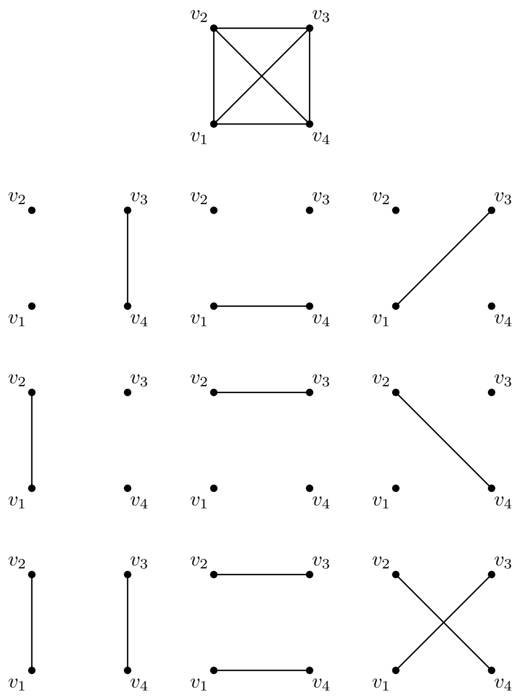


Figure 20.3: An example of the vertex matching polynomial. Above: the graph G; Below: the set of matchings of G. Therefore the matching polynomial of G is $M_G(x) = -(x_1x_2 + x_3x_4 + x_2x_3 + x_1x_4 + x_1x_3 + x_2x_4) + 3 \cdot x_1x_2x_3x_4$.

Chapter 21: Higher-Order Random Walk and Edge-Expansion on Posets

21.1 Introduction

Random walks on high dimensional expanders (HDX) have been the object of intense study in theoretical computer science in recent years. Starting with their original formulation by Kaufman and Mass [KM16], a series of works on the spectral structure of these walks [KO20, DDFH18, AL20a] led to significant breakthroughs in approximate sampling [ALGV19, AL20a, ALG20, CLV20, CLV21, CGŠV21, FGYZ21, JPV21, Liu21, BCC$^+$21], CSP-approximation [AJT19, BHKL20], error-correcting codes [JQST20, JST21], agreement testing [DK17, DD19, KM20b], and more. Most of these works focus on the structure of expansion in *hypergraphs* (typically called *simplicial complexes* in the HDX literature). On the other hand, it has become increasingly clear that hypergraphs are not always the right tool for the job—recent breakthroughs in locally testable [DEL$^+$22] and quantum LDPC codes [PK21, LH22, LZ22], for instance, all rely crucially on *cubical* structure not seen in hypergraphs, while many agreement testing results like the proof of the 2-2 Games Conjecture [SMS18] rely crucially on *linear algebraic* rather than simplicial structure.

In this chapter, we study a generalized notion of high dimensional expansion on *partially ordered sets* (posets) introduced by Dikstein, Dinur, Filmus, and Harsha (DDFH) [DDFH18] called *expanding posets* (eposets). Random walks on eposets capture a broad range of important structures beyond their hypergraph analogs, including natural sparsifications of the Grassmann graphs that recently proved crucial to the resolution of the 2-2 Games Conjecture [SMS18, KMS17, DKK$^+$18b, DKK$^+$18a, BKS18, KMMS18]. DDFH's notion of eposets is a *global* definition of high dimensional expansion based on a relaxation of Stanley's [Sta88] sequentially differential posets, a definition originally capturing both the Grassmanian and complete simplicial complex. More recently, Kaufman and Tessler (KT) [KT21a] have extended the study of eposets in two important aspects. First, in contrast to DDFH's original global definition, KT introduced the local-to-global study of high dimensional expansion in eposets. Second, they identified *regularity* as a key parameter controlling expansion. In particular, the authors showed strengthened local-to-global theorems for strongly regular posets like the Grassmann, giving the first general formulation for characterizing expansion based on an eposet's underlying architecture.

While analysis of the second eigenvalue is certainly an important consideration (e.g. for mixing applications), a deeper understanding of the spectral structure of eposets is required for applications like the proof of the 2-2 Games Conjecture. As such, our main focus in this chapter lies in characterizing the spectral and combinatorial behavior of random walks on eposets *beyond the second eigenvalue*. Strengthening DDFH and recent work of Bafna, Hopkins, Kaufman, and Lovett (BHKL) [BHKL20], we prove that at a coarse level (walks on) eposets indeed exhibit the same spectral and combinatorial characteristics as expanding hypergraphs (e.g. spectral stripping, expansion of pseudorandom sets). On the other hand, as in KT, we show that the finer-grained properties of these objects are actually controlled by the underlying poset's regularity, including the *rate of decay* of the spectrum and combinatorial expansion of associated random walks. This gives a stronger separation between structures like hypergraphs with weak (linear) eigenvalue decay, and Grassmann-based eposets with strong (exponential) eigenvalue decay (a crucial property in the proof of the 2-2 Games Conjecture [SMS18]).

In slightly more detail, we show that all eposets exhibit a behaviour called "eigenstripping" [KO20, DDFH18, BHKL20]: the spectrum of any associated random walk concentrates around a few unique approximate eigenvalues. Moreover, the approximate eigenvalues of walks on eposets are tightly controlled by the poset architecture's regularity[1] $R(j, i)$, which denotes the total number of rank-i elements[2]

[1]We will additionally assume a slightly stronger condition introduced in KT [KT21a] called *middle regularity* throughout. See Section 21.2.1 for details.

[2]We consider *regular graded posets*, where each element has a corresponding rank. In a hyper-

less than any fixed rank-j element (see Section 21.1.1 for standard definitions). For simplicity, we specialize our result below to the popular "lower" or "down-up" walk (this simply corresponds to taking a random step down and back up the poset, again see Section 21.1.1); a more involved version holds for higher order random walks in full generality.

Theorem 21.1 (Eigenstripping and Regularity (informal Corollary 21.19 and Theorem 21.21))**.** *The spectrum of the lower walk UD on a k-dimensional γ-eposet is concentrated in $(k+1)$ strips:*

$$Spec(UD) \in \{1\} \cup \bigcup_{i=1}^{k} [\lambda_i(UD) + O_k(\gamma), \lambda_i(UD) - O_k(\gamma)],$$

where the approximate eigenvalues $\lambda_i(UD)$ are determined by the poset's regularity:

$$\lambda_i(UD) = \frac{R(k-1,i)}{R(k,i)}.$$

Theorem 21.1 generalizes and tightens recent work on expanding hypergraphs of BHKL [BHKL20, Theorem 2.2] (which itself extended a number of earlier works on the topic [KO20, DDFH18, AJT19]). Additionally, our result on the connection between regularity and approximate eigenvalues generalizes the work of KT [KT21a], who show an analogous result for λ_2. Theorem 21.1 reveals a stark contrast between the spectral behavior of eposets with different regularity parameters. As a prototypical example, consider the case of hypergraphs versus subsets of the Grassmann (k-dimensional vector spaces over $\mathbb{F}_q^n$). In the former, each k-set contains $\binom{k}{i}$ i-sets, leading to approximate eigenvalues that decay *linearly* ($\lambda_i \approx (k-i)/k$). On the other hand, each k-dimensional vector space contains $\binom{k}{i}_q$ i-dimensional subspaces, which leads to eigenvalues that decay *exponentially* ($\lambda_i \approx q^{-i}$). The latter property, which we call *strong decay* is often crucial in applications (e.g. for hardness of approximation [SMS18] or fast algorithms [BHKL20]), and while it is possible to recover strong

graph, for instance, rank is given by the size of a set, while in the Grassmann poset it is given by subspace dimension.

decay on weaker posets by increasing the length of the walk [BHKL20], this is often untenable in application due to the additional degrees of freedom it affords.[3]

The spectral structure of walks on eposets is closely related to their *edge-expansion*, an important combinatorial property that has recently played a crucial role both in algorithms for [BBK+20, BHKL20] and hardness of unique games [SMS18]. The key insight in both cases lay in understanding *the structure of non-expanding sets*. We give a tight understanding of this phenomenon across all eposets in the so-called ℓ_2-regime [BHKL20], where we show that expansion is tightly controlled by the behavior of local restrictions called *links* (see Definition 21.2).

Theorem 21.2 (Expansion in the ℓ_2-Regime (informal Theorem 21.30))**.** *The expansion of any i-link is almost exactly $1-\lambda_i(M)$. Conversely, any set with expansion less than $1-\lambda_{i+1}(M)$ has **high variance** across i-links.*

In [BHKL20], it was shown this characterization allows for the application of a local-to-global algorithmic framework for unique games on such walks. This remains true on eposets, and it is an interesting open question whether there are significant applications beyond those given in BHKL's original work.[4]

Finally, as an application of our structure theorems, we give an in-depth analysis of the ℓ_2-structure of walks on expanding subsets of the Grassmann poset called q-eposets (first studied in [DDFH18]). We focus in particular on the natural q-analog of an important set of walks called *partial-swap walks* introduced by Alev, Jeronimo, and Tulsiani [AJT19] that generalize the Johnson graphs when applied to expanding hypergraphs. We show that applied to q-eposets, these objects give a natural set of walks generalizing the Grassmann graphs and further prove that our generic analysis

[3]For instance such a walk might take exponential time to implement, or correspond to a more complicated agreement test than desired.

[4]While one can apply the framework to playing unique games on the Grassmann poset (or sparsifications thereof), the spectral parameters are such that this does not give a substantial improvement over standard algorithms [AKK+08].

for eposets gives a tight characterization of non-expansion in this setting. We note that this does not recover the result used for the proof of the 2-2 Games Conjecture which lies in the ℓ_∞-*regime* (replacing variance above with maximum) and requires a *dimension-independent* bound. This issue was recently (and independently) resolved for simplicial complexes in [BHKL21] and [GLL22], and we view our work as an important step towards a more general understanding for families like the Grassmann beyond hypergraphs.

21.1.1 Background

Before jumping into our results in any further formality, we'll briefly overview the theory of expanding posets and higher order random walks. All definitions are covered in full formality in Section 21.2. A d-dimensional graded poset is a set X equipped with a partial order "$<$" and a ranking function $r : X \to [d]$ that respects the partial order and partitions X into levels $X(0) \cup \ldots \cup X(d)$. When $x < y$ and $r(y) = r(x) + 1$, we write $x \lessdot y$ or equivalently $y \gtrdot x$.[5] Finally, we will assume throughout this chapter that our posets are *downward regular*: there exists a regularity function $R(k, i)$ such that every k-dimensional element is greater than exactly $R(k, i)$ i-dimensional elements.[6]

Graded posets come equipped with a natural set of averaging operators called the *up* and *down* operators. Namely, for any function $f : X(i) \to \mathbb{R}$, these operators average f up or down one level of the poset respectively:

$$U_i f(x) = \mathop{\mathbb{E}}_{y \lessdot x}[f(y)],$$
$$D_i f(y) = \mathop{\mathbb{E}}_{x \gtrdot y}[f(x)].$$

Composing the averaging operators leads to a natural notion of random walks on the underlying poset called *higher order random walks* (HD-walks). The simplest example

[5]This is traditionally called a 'covering relation.'
[6]For notational convenience, we also define $R(i, i) = 1$ and $R(j, i) = 0$ whenever $j < i$.

of such a walk is the *upper walk* $D_{i+1}U_i$ which moves between elements $x, x' \in X(i)$ via a common element $y \in X(i+1)$ with $y > x, x'$. Similarly, the *lower walk* $U_{i-1}D_i$ walks between $x, x' \in X(i)$ via a common $y \in X(i-1)$ with $y < x, x'$. It will also be useful at points to consider longer variants of the upper and lower walks called *canonical walks* $\widehat{N}_k^i = D_{k+1} \circ \ldots \circ D_{k+i} \circ U_{k+i-1} \circ \ldots \circ U_k$ and $\widecheck{N}_k^i = U_k \circ \ldots \circ U_{k-i} \circ D_{k-i+1} \circ \ldots \circ D_k$ which similarly walk between k-dimensional elements in $X(k)$ via a shared element in $X(k+i)$ or $X(k-i)$ respectively.

Following DDFH [DDFH18], we call a poset a (δ, γ)-expander for $\delta \in [0,1]^{d-1}$ and $\gamma \in \mathbb{R}_+$ if the upper and lower walks are spectrally similar up to a laziness factor:

$$\|D_{i+1}U_i - (1-\delta_i)I - \delta_i U_{i-1}D_i\| \leq \gamma.$$

This generalizes standard spectral expansion which can be equivalently defined as looking at the spectral norm of $A_G - U_0D_1$, where A_G (the adjacency matrix) is exactly the non-lazy upper walk. We note that under reasonable regularity conditions (see [KT21a, DDFH18]), this definition is equivalent to *local-spectral expansion* [DK17], which requires every local restriction of the poset to be an expander graph. While most of our results hold more generally, it will also be useful to assume a weak non-laziness condition on our underlying posets throughout that holds in most cases of interest (see Definition 21.12).

21.1.2 Results

With these definitions in mind, we can now cover our results in somewhat more formality. We split this section into three parts for readability: spectral stripping, characterizing edge expansion, and applications to the Grassmann.

21.1.2.1 Eigenstripping

We start with our generalized spectral stripping theorem for walks on expanding posets.

Theorem 21.3 (Spectrum of HD-Walks (informal Corollary 21.19))**.** *Let M be an HD-walk on the kth level of a (δ, γ)-eposet. Then the spectrum of M is highly concentrated in $k+1$ strips:*

$$Spec(M) \in \{1\} \cup \bigcup_{i=1}^{k} [\lambda_i(M) - e, \lambda_i(M) + e]$$

where $e \leq O_{k,\delta}(\gamma)$. Moreover, the span of eigenvectors in the ith strip approximately correspond to functions lifted from $X(i)$ to $X(k)$.

This generalizes and improves an analogous result of BHKL [BHKL20] on expanding hypergraphs, which had sub-optimal error dependence of $O_k(\gamma^{1/2})$. The main improvement stems from an optimal spectral stripping result for arbitrary inner product spaces of independent interest.

Theorem 21.4 (Eigenstripping (informal Theorem 21.13))**.** *Let M be a self-adjoint operator over an inner product space V, and $V = V^1 \oplus \ldots \oplus V^k$ be an "approximate eigendecomposition" in the sense that there exist $\{\lambda_i\}_{i=1}^k$ and sufficiently small error factors $\{c_i\}_{i=1}^k$ such that for all $f_i \in V^i$:*

$$\|Mf_i - \lambda_i f_i\|_2 \leq c_i \|f_i\| .$$

Then the spectrum of M is concentrated around each λ_i:

$$Spec(M) \subseteq \bigcup_{i=1}^{k} [\lambda_i - c_i, \lambda_i + c_i] .$$

Note that this result is tight—when there is c_i "error" in our basis we cannot expect to have better than c_i error in the resulting spectral strips. Theorem 21.4 improves over a preliminary result to this effect in [BHKL20] which had substantially worse dependence on c_i and required much stronger assumptions.[7] Theorem 21.3

[7]It is also worth noting that the proof in this chapter is substantially simplified from [BHKL20], requiring no linear algebraic manipulations at all.

then follows by work of DDFH ([DDFH18, Theorem 8.6]), who introduced a natural approximate eigendecomposition on eposets we call the HD-Level-Set Decomposition.

In full generality, the approximate eigenvalues in Theorem 21.3 depend on the eposet parameters δ, and can be fairly difficult to interpret. However, we show that under weak assumptions (see Section 21.2) the eigenvalues can be associated with the regularity of the underlying poset. We focus on the lower walks for simplicity, though the result can be similarly extended to general walks on eposets.

Theorem 21.5 (Regularity Controls Spectral Decay (informal Theorem 21.21))**.** *The approximate eigenvalues of the lower walk $\check{N}_k^{k-i}$ on a (δ,γ)-eposet are controlled by the poset's regularity function:*

$$\lambda_j(\check{N}_k^{k-i}) \in \frac{R(i,j)}{R(k,j)} \pm O_{k,\delta}(\gamma).$$

As discussed in Section 21.1, this generalizes work of Kaufman and Tessler [KT21a] for the second eigenvalue of the upper/lower walks, and reveals a major distinction among poset architectures: posets with higher regularity enjoy faster decay of eigenvalues. We note that Theorem 21.1 can also be obtained by combining Theorem 21.4 with recent independent work of Dikstein, Dinur, Filmus, and Harsha on connections between eposets and regularity (namely in the recent update of their seminal eposet paper, see [DDFH18, Section 8.4.1]).

On a more concrete note, Theorem 21.5 gives a new method of identifying potential poset architectures exhibiting strong spectral decay in the sense that for any $\delta > 0$, the lower walk only contains $O_\delta(1)$ approximate eigenvalues larger than δ (rather than a number growing with dimension). This property, referred to as *constant ST-Rank* in the context of hypergraphs in [BHKL20], is an important factor not only for the run-time of approximation algorithms on HDX [BHKL20], but also for the soundness of the Grassmann-based agreement test in the proof of the 2-2 Games Conjecture [SMS18].

21.1.2.2 Characterizing Edge Expansion

Much of our motivation for studying the spectrum of HD-walks comes from the desire to understand a fundamental combinatorial quantity of graphs called *edge expansion*.

Definition 21.1 (Edge Expansion)**.** Let X be a graded poset and M an HD-Walk on $X(k)$. The edge expansion of a subset $S \subset X(k)$ with respect to M is

$$\Phi(S) = \mathop{\mathbb{E}}_{v \sim S}\left[M(v, X(k) \setminus S)\right],$$

where

$$M(v, X(k) \setminus S) = \sum_{y \in X(k) \setminus S} M(v, y)$$

and $M(v, y)$ denotes the transition probability from v to y.

As mentioned in the introduction, characterizing the edge-expansion of sets in HD-walks has recently proven crucial to understanding both algorithms for [BBK+20, BHKL20] and hardness of unique games [SMS18]. On expanding hypergraphs, it has long been known that *links* give the canonical example of small non-expanding sets.

Definition 21.2 (Link)**.** Let X be a d-dimensional graded poset. The k-dimensional link of an element $\sigma \in X$ is the set of rank k elements greater than σ:[8]

$$X_\sigma^k = \{y \in X(k) : y > \sigma\}.$$

We call the link of a rank-i element an "i-link." When the level k is clear from context, we write X_σ for X_σ^k for simplicity.

In greater detail, BHKL [BHKL20] proved that on hypergraphs, the expansion of links is exactly controlled by their corresponding spectral strip. While their proof of this fact relied crucially on simplicial structure, we show via a more general analysis that the result can be recovered for eposets.

[8]We note that in the literature a link is usually defined to be all such elements, not just those of rank k. We adopt this notation since we are mostly interested in working at a fixed level of the complex.

Theorem 21.6 (Expansion of Links (informal Theorem 21.27))**.** *Let X be a (δ, γ)-eposet and M an HD-walk on $X(k)$. Then for all $0 \leq i \leq k$ and $\tau \in X(i)$:*

$$\Phi(X_\tau) = 1 - \lambda_i(M) \pm O_{M,k,\delta}(\gamma).$$

As an immediate consequence, we get that when M is not a small-set expander, links are examples of small non-expanding sets. One might reasonably wonder whether the converse is true as well: *are all non-expanding sets explained by links*? This requires a bit of formalization. Following BHKL's exposition [BHKL20], given a set S consider the function $L_{S,i} : X(i) \to \mathbb{R}$ that encodes the behavior of $S \subset X(k)$ on links:

$$\forall \tau \in X(i) : L_{S,i}(\tau) = \underset{X_\tau}{\mathbb{E}}[\mathbb{1}_S] - \mathbb{E}[\mathbb{1}_S].$$

The statement "Non-expansion is explained by links" can then be interpreted as saying that a non-expanding set S should be detectable by some simple measure of $L_{S,i}$. There are two standard formalizations of this idea studied in the literature: the ℓ_2-regime, and the ℓ_∞-regime. These are captured by the following notion of *pseudorandomness* based on $L_{S,i}$.

Definition 21.3 (Pseudorandom Sets [BHKL20] (informal Definitions 21.17, 21.18))**.** We say a set S is (ε, ℓ)-ℓ_2-pseudorandom if[9]

$$\forall i \leq \ell : \|L_{S,i}\|_2^2 \leq \varepsilon \mathbb{E}[\mathbb{1}_S].$$

A set is (ε, ℓ)-ℓ_∞-pseudorandom if:

$$\forall i \leq \ell : \|L_{S,i}\|_\infty \leq \varepsilon.$$

In cases that ℓ_2 and ℓ_∞-pseudorandomness can be used interchangeably, we will simply write (ε, ℓ)-pseudorandom.

We prove that pseudorandom sets expand near-optimally.

[9]Throughout, $\|\cdot\|_2$ will always refer to the *expectation* norm $\|f\|_2 = \mathbb{E}[f^2]^{1/2}$.

Theorem 21.7 (Pseudorandom Sets Expand (informal Theorem 21.30))**.** *Let X be a (δ, γ)-eposet and M a walk on $X(k)$. Then the expansion of any (ε, i)-pseudorandom set S is at least:*

$$\Phi(S) \geq 1 - \lambda_{i+1} - O_\delta(R(k,i)\varepsilon) - O_{k,\delta,M}(\gamma).$$

In other words, any set with expansion less than $1-\lambda_{i+1}$ must have appreciable variance across links at level i. We note that the formal version of this result is essentially tight in the ℓ_2-regime, but can be improved in many important cases in the ℓ_∞-regime. We'll discuss this further in the next section, especially in the context of the Grassmann poset.

Before this, however, it is worth separately mentioning the main technical component behind Theorem 21.7, a result traditionally called a "level-i" inequality.

Theorem 21.8 (Level-i inequality (informal Theorem 21.25))**.** *Let X be a (δ, γ)-eposet and $S \subset X(k)$ a (ε, ℓ)-pseudorandom set. Then for all $1 \leq i \leq \ell$:*

$$|\langle \mathbb{1}_S, \mathbb{1}_{S,i} \rangle| \leq (R(k,i)\varepsilon + O_{k,\delta}(\gamma)) \langle \mathbb{1}_S, \mathbb{1}_S \rangle$$

where $\mathbb{1}_{S,i}$ is the projection of $\mathbb{1}_S$ onto the ith eigenstrip.[10]

In other words, pseudorandomness controls the projection of S onto eigenstrips. Theorems 21.7 and 21.8 recover the analogous optimal bounds for simplicial high dimensional expanders in [BHKL20], where the regularity parameter $R(k,i) = \binom{k}{i}$, and are tight in a number of other important settings such as the Grassmann (discussed below). Theorem 21.7 and Theorem 21.8 can also be viewed as another separation between eposet architectures, this time in terms of *combinatorial* rather than *spectral* properties.

[10]Note that since walks on eposets are simultaneously diagonalizable, the decomposition of X into eigenstrips is independent of the choice of walk.

21.1.2.3 Application: q-eposets and the Grassmann Graphs

Finally, we'll discuss the application of our results to a particularly important class of eposets called "q-eposets." Just like standard high dimensional expanders arise from expanding subsets of the complete complex (hypergraph), q-eposets arise from expanding subsets of the Grassmann Poset.

Definition 21.4 (Grassmann Poset)**.** The Grassmann Poset is a graded poset $(X, <)$ where X is the set of all subspaces of $\mathbb{F}_q^n$ of dimension at most d, the partial ordering "$<$" is given by inclusion, and the rank function is given by dimension.

We call a (downward-closed) subset of the Grassmann poset a q-simplicial complex, and an expanding q-simplicial complex a q-eposet (see Section 21.2.5 for exact details). Using our machinery for general eposets, we prove a tight level-i inequality for pseudorandom sets.

Corollary 21.9 (Grassmann level-i inequality (informal Theorem 21.37))**.** *Let X be a γ-q-eposet and $S \subseteq X(k)$. If S is (ε, ℓ)-pseudorandom, then for all $1 \leq i \leq \ell$:*

$$|\langle \mathbb{1}_S, \mathbb{1}_{S,i} \rangle| \leq \left(\binom{k}{i}_q \varepsilon + O_{q,k}(\gamma) \right) \langle \mathbb{1}_S, \mathbb{1}_S \rangle$$

where $\binom{k}{i}_q = \frac{(1-q^k)\cdots(1-q^{k-i+1})}{(1-q^i)\cdots(1-q)}$ is the Gaussian binomial coefficient.

Corollary 21.9 is tight in a few senses. First, we prove the bound cannot be improved by any constant factor, even in the ℓ_∞-regime. In other words, for every $c < 1$, it is always possible to find an (ε, i)-pseudorandom function satisfying:

$$|\langle \mathbb{1}_S, \mathbb{1}_{S,i} \rangle| > c \left(\binom{k}{i}_q \varepsilon + O_{q,k}(\gamma) \right) \langle \mathbb{1}_S, \mathbb{1}_S \rangle.$$

Furthermore, it is well known the dependence on k in this result is necessary [KMS17], even if one is willing to suffer a worse dependence on the pseudorandomness ε. This is different from the case of standard simplicial complexes, where the dependence can be removed in the ℓ_∞-regime [KMMS18, BHKL21, GLL22]. However, there is a crucial subtlety here. It is likely that the k-dependence in this result can be removed by *changing the definition of pseudorandomness.* On the Grassmann poset itself, for instance, it is known that this can be done by replacing links with a closely related but finer-grained local structure known as "zoom-in zoom-outs" [SMS18]. Indeed, more generally it is an interesting open problem whether there always exists a notion of locality based on the underlying poset structure that gives rise to k-independent bounds in the ℓ_∞-regime.

We close out the section by looking at an application of this level-i inequality to studying edge-expansion in an important class of walks that give rise to the well-studied *Grassmann graphs.*

Definition 21.5 (Grassmann Graphs)**.** The Grassmann Graph $J_q(n, k, t)$ is the graph on k-dimensional subspaces of $\mathbb{F}_q^n$ where $(V, W) \in E$ exactly when $\dim(V \cap W) = t$.

It is easy to see that the non-lazy upper walk on the Grassmann poset is exactly the Grassmann graph $J_q(n, k, k-1)$. In fact, it is possible to express any $J_q(n, k, t)$ as a sum of standard higher order random walks.

Proposition 21.10 (Grassmann Graphs are HD-Walks (informal Proposition 21.35))**.** *The Grassmann graphs are a hypergeometric sum of canonical walks:*

$$J_q(n,k,t) = \frac{1}{q^{(k-t)^2}\binom{k}{t}_q} \sum_{i=0}^{k-t} (-1)^{k-t-i} q^{\binom{k-t-i}{2}} \binom{k-t}{i}_q \binom{k+i}{i}_q N_k^i.$$

In Section 21.7 we prove a more general version of this result for any q-simplicial complex. This leads to a set of natural sparsifications of the Grassmann graphs that may be of independent interest for agreement testing, PCPs, and hardness of approximation. For simplicity, on a given q-simplicial complex X, we'll refer to these "sparsified" Grassmann graphs as $J_{X,q}(n, k, t)$ for the moment (more formally they are the "partial-swap walks," see Section 21.2.5). With this in mind, let's take a look at what our level-i inequality implies for the edge-expansion of these graphs.

Corollary 21.11 (q-eposets Edge-Expansion (informal Corollary 21.40))**.** *Let X be a d-dimensional γ-q-eposet and $S \subset X(k)$ a (ε, ℓ)-pseudorandom set. Then the expansion of S with respect to the sparsified Grassmann graph $J_{X,q}(n, k, t)$ is at least:*

$$\Phi(S) \geq 1 - \mathbb{E}[\mathbb{1}_S] - \varepsilon \sum_{i=1}^{\ell} \binom{t}{i}_q - q^{-(\ell+1)j} - O_{q,k}(\gamma).$$

In practice, t is generally thought of as being $\Omega(k)$ (or even $k - O(1)$), which results in a k-dependent bound. It remains an open problem whether a k-independent version can be proved for any q-eposet beyond the Grassmann poset itself. We conjecture such a result should indeed hold (albeit under a different notion of pseudorandomness), and may follow from q-analog analysis of recent work proving k-independent bounds for standard expanding hypergraphs [BHKL21, GLL22].

21.1.3 Related work

Higher Order Random Walks. Higher order random walks were first introduced in 2016 by Kaufman and Mass [KM16]. Their spectral structure was later elucidated in a series of works by Kaufman and Oppenheim [KO20], DDFH [DDFH18], Alev, Jeronimo, and Tulsiani [AJT19], Alev and Lau [AL20a], and finally BHKL [BHKL20]. With the exception of DDFH (who only worked with approximate eigenvectors without analyzing the true spectrum), all of these works focused on hypergraphs rather than general posets. Our spectral stripping theorem for eposets essentially follows from combining eposet machinery developed by DDFH with our improved variant of BHKL's general spectral stripping theorem.

Higher order random walks have also seen an impressive number of applications in recent years, frequently closely tied to analysis of their spectral structure. This has included breakthrough works on approximate sampling [ALGV19, AL20a, ALG20, CLV20, CLV21, CGŠV21, FGYZ21, JPV21, Liu21, BCC+21], CSP-approximation [AJT19, BHKL20], error-correcting codes [JQST20, JST21], and agreement testing [DK17, DD19, KM20b]. In this vein, our work is most closely related to that of Bafna, Barak, Kothari, Schramm, and Steurer [BBK$^+$20], and BHKL [BHKL20], who used the spectral and combinatorial structure of HD-walks to build new algorithms for unique games. As previously discussed, the generalized analysis in this chapter also lends itself to the algorithmic techniques developed in those works, but we do not know of any interesting examples beyond those covered in BHKL.

High Dimensional Expansion Beyond Hypergraphs. Most works listed above (and indeed in the high dimensional expansion literature in general) focus only on the setting of hypergraphs. However, recent years have also seen the nascent development and application of expansion beyond this setting [DEL$^+$22, PK21, LH22, LZ22, HL22], including the seminal work of DDFH [DDFH18] on expanding posets as well as more recent breakthroughs on locally testable and quantum codes [DEL$^+$22, PK21]. While DDFH largely viewed eposets as having similar structure (with the exception of the Grassmann), we strengthen the case that different underlying poset architectures exhibit different properties. This complements the recent result of Kaufman and Tessler [KT21a], who showed that expanding posets with strong regularity conditions such as the Grassmann exhibit more favorable properties with respect to the second eigenvalue. Our results provide a statement of the same flavor looking at the entire spectrum, along with additional separations in more combinatorial settings. We note that a related connection between poset regularity and the approximate spectrum of walks on eposets was independently developed by DDFH in a recent update of their seminal work [DDFH18].

Expansion and Unique Games. One of the major motivations behind this chapter is towards building a more general framework for understanding the structure underlying the Unique Games Conjecture [Kho02], a major open problem in complexity theory that implies optimal hardness of approximation results for a large swath of combinatorial optimization problems (see e.g. Khot's survey [KV05]). In 2018, Khot, Minzer, and Safra [SMS18] made a major breakthrough towards the UGC in proving a weaker variant known as the 2-2 Games Conjecture, completing a long line of work in this direction [KMS17, DKK+18b, DKK+18a, BKS18, KMMS18, SMS18]. The key to the proof lay in a result known as the "Grassmann expansion hypothesis," which stated that any non-expanding set in the Grassmann graph $J_q(d,k,k-1)$ had to be non-trivially concentrated inside a local-structure called "zoom-in zoom-outs." As noted in the previous section, this result differs from our analysis in two key ways: it lies in the ℓ_∞-regime, and must be totally independent of dimension.

Unfortunately, very little progress has been made towards the UGC since this result. This is in part because KMS' proof of the Grassmann expansion hypothesis, while a tour de force, is complicated and highly tailored to the exact structure of the Grassmann. To our knowledge, the same proof cannot be used, for instance, to resolve the related "shortcode expansion hypotheses" beyond degree-2, similar conjectures offered by Barak, Kothari, and Steurer [BKS18] in an effort to push beyond hardness of 2-2 Games. Just as the ℓ_2-regime analysis of DDFH and BHKL recently lead to a dimension independent bound in the ℓ_∞-regime for standard HDX [BHKL21, GLL22], we expect the groundwork laid in this chapter will be important for proving generalized dimension independent expansion hypotheses in the ℓ_∞-regime beyond the special case of the Grassmann graphs.

21.2 Preliminaries

Before jumping into the details in full formality, we give a more careful review of background definitions regarding expanding posets, higher order random walks, and the Grassmann.

21.2.1 Graded posets

We start with eposets' underlying structure, graded posets. A partially ordered set (poset) $P = (X, <)$ is a set of elements X endowed with a partial order "$<$". A graded poset comes equipped additionally with a rank function $r : X \to \mathbb{N}$ satisfying two properties:

1. r preserves "$<$": if $y < x$, then $r(y) < r(x)$.
2. r preserves cover relations: if x is the smallest element greater than y, then $r(x) = r(y) + 1$.

In other words, the function r partitions X into subsets by rank:

$$X(0) \cup \ldots \cup X(d),$$

where $\max_X(r) = d$, and $X(i) = r^{-1}(i)$. We refer to a poset with maximum rank d as "d-dimensional", and elements in $X(i)$ as "i-faces". Throughout this chapter, we will consider only d-dimensional graded posets with two additional restrictions:

1. They have a unique minimal element, i.e. $|X(0)| = 1$.
2. They are "pure": all maximal elements have rank d.

Finally, many graded posets of interest satisfy certain regularity conditions which will be crucial to our analysis. The first condition of interest is a natural notion called *downward* regularity.

Definition 21.6 (Downward Regularity)**.** We call a d-dimensional graded poset downward regular if for all $i \leq d$ there exists some constant $R(i)$ such that every element $x \in X(i)$ covers exactly $R(i)$ elements $y \in X(i-1)$.

Second, we will also need a useful notion called *middle regularity* that ensures uniformity across multiple levels of the poset.

Definition 21.7 (Middle Regularity)**.** We call a d-dimensional graded poset middle-regular if for all $0 \leq i \leq k \leq d$, there exists a constant $m(k,i)$ such that for any $x_k \in X(k)$ and $x_i \in X(i)$ satisfying $x_k > x_i$, there are exactly $m(k,i)$ chains[11] of elements $x_k > x_{k-1} > \ldots > x_{i+1} > x_i$ where each $x_j \in X(j)$.

[11]Such objects are sometimes called flags, e.g. in the case of the Grassmann poset.

We call a poset regular if it is both downward and middle regular. We note that regular posets also have the nice property that for any dimensions $i < k$, there exists a higher order regularity function $R(k, i)$ such that any $x \in X(k)$ is greater than exactly $R(k, i)$ elements in $X(i)$ (see Section 21.8). We will use this notation throughout. For notational convenience, we also define $R(i, i) = 1$ and $R(j, i) = 0$ whenever $j < i$.

Important examples of regular posets include pure simplicial complexes and the Grassmann poset (subspaces of $\mathbb{F}_q^n$ ordered by inclusion). We will assume all posets we discuss in this chapter are regular from this point forward.

21.2.2 Measured posets and the random walk operators

Higher order random walks may be defined over posets in a very similar fashion to simplicial complexes. The main difference is simply that "inclusion" is replaced with the poset order relation. Just as we might want these walks on HDX to have non-uniform weights, the same is true for posets, which can be analogously endowed with a distribution over levels. In slightly more detail, a *measured poset* is a graded poset X endowed with a distribution $\Pi = (\pi_0, \dots, \pi_d)$, where each marginal π_i is a distribution over $X(i)$. While measured posets may be defined in further generality (cf. [DDFH18, Definition 8.1]), we will focus on the case in which the distribution Π is induced entirely from π_d, analogous to weighted simplicial complexes. More formally, we have that for every $0 \leq i < d$:

$$\pi_i(x) = \frac{1}{R(i+1, i)} \sum_{y > x} \pi_{i+1}(y).$$

In other words, each lower dimensional distribution π_i may be induced through the following process: an element $y \in X(i+1)$ is selected with respect to π_{i+1}, and an element $x \in X(i)$ such that $x < y$ is then chosen uniformly at random.

The averaging operators U and D are defined analogously to their notions on simplicial complexes, with the main change being the use of the general regularity function $R(i+1,i)$:

$$U_i f(y) = \frac{1}{R(i+1,i)} \sum_{x<y} f(x),$$
$$D_{i+1} f(x) = \frac{1}{\pi_{i+1}(X_x)} \sum_{y>x} \pi_{i+1}(y) f(y),$$

where for $i < k$ and $x \in X(i)$,

$$\pi_k(X_x) = \sum_{y \in X(k): y > x} \pi_k(y) = R(k,i)\pi_i(x)$$

is the appropriate normalization factor (we will use this notation throughout). On regular posets, it is useful to note that the up operators compose nicely, and in particular that:

$$U_i^k f(y) \coloneqq U_{k-1} \circ \ldots \circ U_i f(y) = \frac{1}{R(k,i)} \sum_{x \in X(i): x<y} f(x)$$

(see Section 21.8). Furthermore, just like on simplicial complexes, the down and up operators are adjoint with respect to the standard inner product on measured posets:

$$\langle f, g \rangle_{X(k)} = \sum_{\tau \in X(k)} \pi_k(\tau) f(\tau) g(\tau),$$

that is to say for any $f : X(k) \to \mathbb{R}$ and $g : X(k-1) \to \mathbb{R}$:

$$\langle f, U_{k-1} g \rangle_{X(k)} = \langle D_k f, g \rangle_{X(k-1)}.$$

Note that we'll generally drop the $X(k)$ from the notation when clear from the context. This useful fact allows us to define basic self-adjoint notions of higher order random walks just like on simplicial complexes.

21.2.3 Higher order random walks

Let C_k denote the space of functions $f : X(k) \to \mathbb{R}$. We define a natural set of random walk operators via the averaging operators.

Definition 21.8 (k-Dimensional Pure Walk [KM16, DDFH18, AJT19])**.** Given a measured poset (X, Π), a k-dimensional pure walk $Y : C_k \to C_k$ on (X, Π) (of height $h(Y)$) is a composition:

$$Y = Z_{2h(Y)} \circ \cdots \circ Z_1,$$

where each Z_i is a copy of D or U, and there are $h(Y)$ of each type.

Following AJT and BHKL, we define general higher order random walks to be affine combinations[12] of pure walks.

Definition 21.9 (HD-walk)**.** Let X be a graded poset. Let $\mathcal{Y}$ be a family of pure walks $Y : C_k \to C_k$ on (X, Π). We call an affine combination

$$M = \sum_{Y \in \mathcal{Y}} \alpha_Y Y$$

a k-dimensional HD-walk on (X, Π) if it is stochastic and self-adjoint. The height of M, denoted $h(M)$, is the maximum height of any pure $Y \in \mathcal{Y}$ with a non-zero coefficient. The weight of M, denoted $w(M)$, is $|\alpha|_1$.

While most of our results will hold for general HD-walks (or at least some large subclass), we pay special attention to a basic class of pure walks that have seen the most study in the literature: *canonical walks*.

Definition 21.10 (Canonical Walk)**.** Given a d-dimensional measured poset (X, Π) and parameters $k + j \leq d$, the upper canonical walk $\widehat{N}_k^j$ is:

$$\widehat{N}_k^j = D_k^{k+j} U_k^{k+j},$$

and for $j \leq k$ the lower canonical walk $\check{N}_k^j$ is:

$$\check{N}_k^j = U_{k-j}^k D_{k-j}^k,$$

where $U_\ell^k = U_{k-1} \dots U_\ell$, and $D_\ell^k = D_{\ell+1} \dots D_k$.

[12]An affine combination is a linear combination whose coefficients sum to 1.

Since the non-zero spectrum of $\widehat{N}_k^j$ and $\widecheck{N}_{k+j}^j$ are equivalent (c.f. [AL20a]), we focus in this chapter mostly on the upper walks which we write simply as N_k^j.

For certain specially structured posets, we will also study an important class of HD-walks known as (partial) *swap walks*. We will introduce these well-studied walks in more detail in Section 21.2.5, and for now simply note that they give a direct generalization of the Johnson and Grassmann graphs when applied to the complete complex and Grassmann poset respectively.

21.2.4 Expanding posets and the HD-Level-Set decomposition

Dikstein, Dinur, Filmus, and Harsha [DDFH18] observed that one can use the averaging operators to define a natural extension of spectral expansion to graded posets. Their definition is inspired by the fact that γ-spectral expansion on a standard graph G can be restated as a bound on the spectral norm of the adjacency matrix minus its stationary operator:

$$\|A_G - UD\| \leq \gamma.$$

Informally, DDFH's definition can be thought of as stating that this relation holds for every level of a higher dimensional poset.

Definition 21.11 (eposet [DDFH18])**.** Let (X, Π) be a measured poset, $\delta \in [0,1]^{d-1}$, and $\gamma < 1$. X is an (δ, γ)-eposet if for all $1 \leq i \leq d-1$:

$$\|D_{i+1}U_i - (1-\delta_i)I - \delta_i U_{i-1}D_i\| \leq \gamma$$

.

We note that for a broad range of posets, this definition is actually equivalent (up to constants) to *local-spectral expansion*, a popular notion of high dimensional expansion introduced by Dinur and Kaufman [DK17]. This was originally proved for simplicial complexes by DDFH [DDFH18], and later extended to a more general class of posets by Kaufman and Tessler [KT21a]. It is also worth noting that when $\gamma = 0$, posets satisfying the guarantee in Definition 21.11 are known as *sequentially differential*, and were actually introduced much earlier by Stanley [Sta88] in the late 80s.

Much of our analysis in this chapter will be based off of an elegant approximate Fourier decomposition for eposets introduced by DDFH [DDFH18].

Theorem 21.12 (HD-Level-Set Decomposition, Theorem 8.2 [DDFH18])**.** *Let (X, Π) be a d-dimensional (δ, γ)-eposet with γ sufficiently small. For all $0 \leq k \leq d$, let*

$$H^0 = C_0, H^i = Ker(D_i), V_k^i = U_i^k H^i.$$

Then:

$$C_k = V_k^0 \oplus \ldots \oplus V_k^k.$$

In other words, every $f \in C_k$ has a unique decomposition $f = f_0 + \ldots + f_k$ such that $f_i = U_i^k g_i$ for $g_i \in Ker(D_i)$.

It is well known that the HD-Level-Set Decomposition is approximately an eigenbasis for HD-walks on simplicial complex [DDFH18, AJT19, BHKL20]. We show this statement extends to all eposets in Section 21.4 (extending DDFH's similar analysis of the upper walk N_k^1).

Finally, before moving on, we will assume for simplicity throughout this chapter an additional property of eposets we called (approximate) non-laziness.

Definition 21.12 (β-non-Lazy Eposets)**.** Let (X, Π) be a d-dimensional measured poset. We call (X, Π) β-non-lazy if for all $1 \leq i \leq d$, the laziness of the lower walk satisfies:

$$\max_{\sigma \in X(i)} \{\mathbb{1}_\sigma^T U_{i-1} D_i \mathbb{1}_\sigma\} \leq \beta.$$

Another way to think about this condition is that no element in the poset carries too much weight, even upon conditioning. All of our results hold for general eposets,[13] but their form is significantly more interpretable when the poset is additionally non-lazy. In fact, most γ-eposets of interest are $O(\gamma)$-non-lazy. It is easy to see for instance that any "γ-local-spectral" expander satisfies this condition, an equivalent notion of expansion to γ-eposets under suitable regularity conditions [KT21a]. We discuss this further in Section 21.8.

21.2.5 The Grassmann poset and q-eposets

At the moment, there are only two known families of expanding posets of significant interest in the literature: those based on pure simplicial complexes (the downward closure of a k-uniform hypergraph), and pure q-simplicial complexes (the analogous notion over subspaces). The ℓ_2-structure of the former set of objects is studied in detail in [BHKL20]. In this chapter, we will focus on the latter which has seen less attention in the literature, but is responsible for a number of important results including the resolution of the 2-to-2 Games Conjecture [SMS18].

Definition 21.13 (q-simplicial complex)**.** Let $G_q(n,d)$ denote the d-dimensional subspaces of $\mathbb{F}_q^n$. A weighted, pure q-simplicial complex (X,Π) is given by a family of subspaces $X \subseteq G_q(n,d)$ and a distribution Π over X. We will usually consider the downward closure of X in the following sense:

$$X = X(0) \cup \ldots \cup X(d),$$

where $X(i) \subseteq G_q(n,i)$ consists of all i-dimensional subspaces contained in some element in $X = X(d)$. Further, on each level $X(i)$, Π induces a natural distribution π_i:

$$\forall V \in X(i) : \pi_i(V) = \frac{1}{\binom{d}{i}_q} \sum_{W \in X(d) : W \supset V} \pi_d(W),$$

where $\pi_d = \Pi$ and $\binom{d}{i}_q = \frac{(1-q^d)\cdots(1-q^{d-i+1})}{(1-q^i)\cdots(1-q)}$ is the Gaussian binomial coefficient.

[13]The one exception is the lower bound of Theorem 21.6.

The most basic example of a q-simplicial complex is the Grassmann poset, which corresponds to taking $X = G_q(n,d)$. This is the q-analog of the complete simplicial complex. The Grassmann poset is well known to be a expander in this sense (see e.g. [Sta88])—in fact it is a sequentially differential poset with parameters

$$\delta_i = \frac{(q^i-1)(q^{n-i+1}-1)}{(q^{i+1}-1)(q^{n-i}-1)},$$

the q-analog of the eposet parameters for the complete complex [DDFH18]. With this in mind, let's define a special class of eposets based on q-simplicial complexes.

Definition 21.14 (γ-q-eposet [DDFH18])**.** A pure, d-dimensional weighted q-simplicial complex (X,Π) is a γ-q-eposet if it is a (δ,γ)-eposet satisfying $\delta_i = q\frac{q^i-1}{q^{i+1}-1}$ for all $1 \leq i \leq d-1$.

Constructing bounded-degree q-eposets (a problem proposed by DDFH [DDFH18]) remains an interesting open problem. Kaufman and Tessler [KT21a] recently made some progress in this direction, but the expansion parameter of their construction is fairly poor (around 1/2).

Finally, in our applications to the Grassmann we'll focus our attention on a particularly important class of walks called *partial-swap walks*. These should essentially be thought of as non-lazy variants of the upper canonical walks.

Definition 21.15 (Partial-Swap Walk)**.** Let (X,Π) be a weighted, d-dimensional q-simplicial complex. The partial-swap walk S^j_k is the restriction of the canonical walk N^j_k to faces whose intersection has dimension $k-j$. In other words, if $|V \cap W| > k-j$ then $S^j_k(V,W) = 0$, and otherwise $S^j_k(V,W) \propto N^j_k(V,W)$.

When applied to the Grassmann poset itself, it is clear by symmetry that the partial-swap walk S^j_k returns exactly the Grassmann graph $J_q(d,k,k-j)$. On the other hand, it is not immediately obvious these objects are even HD-walks when applied to a generic q-simplicial complex. We prove this is the case in Section 21.7.

21.3 Approximate Eigendecompositions and Eigenstripping

With preliminaries out of the way, we can move on to understanding HD-walks' spectral structure. It turns out that on expanding posets, these walks exhibit almost exactly the same properties as on the special case of simplicial complexes studied in [KO20, DDFH18, AJT19, BHKL20]: a walk's spectrum lies concentrated in strips corresponding to levels of the HD-Level-Set Decomposition. The key to proving this lies in a more general theorem characterizing the spectral structure of any inner product space admitting a "approximate eigendecomposition."

Definition 21.16 (Approximate Eigendecomposition [BHKL20])**.** Let M be an operator over an inner product space V. A decomposition $V = V^1 \oplus \ldots \oplus V^k$ is called a $(\{\lambda_i\}_{i=1}^k, \{c_i\}_{i=1}^k)$-approximate eigendecomposition if for all i and $v_i \in V^i$, Mv_i is close to $\lambda_i v_i$:

$$\|Mv_i - \lambda_i v_i\| \leq c_i \|v_i\| .$$

We will always assume for simplicity (and without loss of generality) that the λ_i are sorted: $\lambda_1 \geq \ldots \geq \lambda_k$.

BHKL [BHKL20] proved that as long as the c_i are sufficiently small, each V^i (loosely) corresponds to an "eigenstrip," the span of eigenvectors with eigenvalue closely concentrated around λ_i, and that these strips account of the entire spectrum of M. While sufficient for their purposes, their proof of this result was complicated and resulted in a variety of sub-optimal parameters. We give a tight variant of this result and significantly simplify the proof.

Theorem 21.13 (Eigenstripping)**.** *Let M be a self-adjoint operator over an inner product space V, and $V = V^1 \oplus \ldots \oplus V^k$ a $(\{\lambda_i\}_{i=1}^k, \{c_i\}_{i=1}^k)$-approximate eigendecomposition. Then as long as $c_i + c_{i+1} < \lambda_i - \lambda_{i+1}$, the spectrum of M is concentrated around each λ_i:*

$$\mathit{Spec}(M) \subseteq \bigcup_{i=1}^{k} [\lambda_i - c_i, \lambda_i + c_i]$$

Proof. The idea is to examine for each i the operator $M_i^2 = (M - \lambda_i I)^2$. In particular, we claim it is enough to show the following:

Claim 21.14. *For all $1 \leq i \leq k$, Spec(M_i^2) contains dim(V^i) eigenvalues less than c_i^2.*

Let's see why this implies the desired result. Notice that the eigenvalues of M_i^2 are exactly $(\mu - \lambda_i)^2$ for each μ in *Spec*(M) (with matching multiplicities), and therefore that any eigenvalue $\mu_i \in$ *Spec*(M_i^2) less than c_i^2 implies the existence of a corresponding eigenvalue of M in $[\lambda_i \pm c_i]$. If each M_i^2 has $\dim(V^i)$ eigenvalues less than c_i^2, then M has at least $\dim(V^i)$ eigenvalues in each interval $[\lambda_i \pm c_i]$. Moreover, since these intervals are disjoint by assumption and $\sum \dim(V^i) = \dim(V)$, this must account for all eigenvalues of M.

It remains to prove the claim, which is essentially an immediate application of Courant-Fischer theorem [Fis05].

Proof of Claim 21.14. The Courant-Fischer theorem states that the kth smallest eigenvalue of a self-adjoint operator A is:

$$\lambda_{n-k+1} = \min_U \left\{ \max_{f \in U} \left\{ \frac{\langle f, Af \rangle}{\langle f, f \rangle} \right\} \;\middle|\; \dim(U) = k \right\}.$$

Setting $U = V^i$, $A = M_i^2$ and $k = \dim(V^i)$ gives the claim:

$$\lambda_{n-k+1}(M_i^2) \leq \max_{f \in V^i} \left\{ \frac{\langle f, M_i^2 f \rangle}{\langle f, f \rangle} \right\} = \max_{f \in V^i} \left\{ \frac{\|(M - \lambda_i I) f\|_2^2}{\langle f, f \rangle} \right\} \leq c_i^2$$

since $(M - \lambda_i I)$ is self-adjoint and $\bigoplus_{i \in [k]} V^i$ is a $(\{\lambda_i\}_{i=1}^k, \{c_i\}_{i=1}^k)$-approximate eigendecomposition. $\square$

$\square$

Note that this result is also trivially tight, as any true eigendecomposition is also a $(\{\lambda_i \pm c_i\}, \{c_i\})$-approximate eigendecomposition. We also note that similar strategies have been used in the numerical analysis literature (see e.g. [HRW98]).

21.4 Spectra of HD-walks

Given Theorem 21.13, it is enough to prove that the HD-Level-Set Decomposition is an approximate eigenbasis for any HD-walk. This follows by the same inductive argument as for local-spectral expanders in [BHKL20], where the only difference is that somewhat more care is required to deal with general eposet parameters. To start, it will be useful to lay out some notation along with a simple observation from repeated application of Definition 21.11.

Lemma 21.15 ([DDFH18, Claim 8.8])**.** *Let (X,Π) be a d-dimensional (δ,γ)-eposet. Then*

$$\left\|D_{k+1}U^{k+1}_{k-j} - (1-\delta^k_j)U^k_{k-j} - \delta^k_j U^k_{k-j-1}D_{k-j}\right\| \le \gamma^k_j,$$

where

$$\delta^k_{-1}=1,\ \delta^k_j = \prod_{i=k-j}^{k}\delta_i,\ \gamma^k_j = \gamma\sum_{i=-1}^{j-1}\delta^k_i.$$

Applying this fact inductively implies that functions in the HD-Level-Set Decomposition are close to being eigenvectors.

Proposition 21.16. *Let (X,Π) be a (δ,γ)-eposet, and Y the pure balanced walk of height j, with down operators at positions $(i_1,\dots,i_j)$. For $1\le \ell\le k$, let $f_\ell = U^k_\ell g_\ell$ for some $g_\ell\in H^\ell$, and let*

$$\delta^k_j = \prod_{i=k-j}^{k}\delta_i,\ \gamma^k_j = \gamma\sum_{i=-1}^{j-1}\delta^k_i,$$

where $\delta^k_i = 1$ for any $i<0$ for notational convenience. Then f_ℓ is an approximate eigenvector of Y:

$$\left\|Yf_\ell - \prod_{s=1}^{j}\left(1-\delta^{k-2s+i_s}_{k-2s+i_s-\ell}\right)f_\ell\right\| \le \|g_\ell\|\sum_{s=1}^{j}\gamma^{k-2s+i_s}_{k-2s+i_s-\ell}\prod_{t=1}^{s-1}\left(1-\delta^{k-2t+i_t}_{k-2t+i_t-\ell}\right) \le (j+k)j\gamma\left\|g_\ell\right\|.$$

Proof. We prove a slightly stronger statement to simplify the induction. For $b>0$, let $Y^b_j : C_\ell \to C_{\ell+b}$ denote an unbalanced walk with j down operators, and $j+b$ up operators. If Y_j^b has down operators in positions $(i_1, \ldots, i_j)$ and $g_\ell \in H^\ell$, we claim:

$$\left\| Y_j^b g_\ell - \prod_{s=1}^{j} \left(1 - \delta_{i_s-2s}^{i_s+\ell-2s}\right) Y_0^b g_\ell \right\| \leq \|g_\ell\| \sum_{s=1}^{j} \gamma_{i_s-2s}^{i_s+\ell-2s} \prod_{t=1}^{s-1} \left(1 - \delta_{i_t-2t}^{i_t+\ell-2t}\right),$$

which implies the result (notice that the indices i_s shift by $b = k - \ell$). The base case $j = 0$ is trivial. Assume the inductive hypothesis holds for all $Y_i^b, i < j$. By Lemma 21.15 and recalling $g_\ell \in \ker(D_\ell)$, we have:

$$Y_j^b g_\ell = \left(1 - \delta_{i_1-2}^{i_1+\ell-2}\right) Y_{j-1}^b g_\ell + \Gamma g_\ell,$$

where Γ has spectral norm

$$\|\Gamma\| \leq \gamma_{i_1-2}^{i_1+\ell-2}.$$

Notice that Y_{j-1}^b has down operator indices $\{i_2 - 2, \ldots, i_j - 2\}$. The inductive hypothesis then implies:

$$\begin{aligned} Y_j^b g_\ell &= \left(1 - \delta_{i_1-2}^{i_1+\ell-2}\right) \prod_{s=2}^{j} \left(1 - \delta_{i_s-2s}^{i_s+\ell-2s}\right) Y_0^b g_\ell + \left(1 - \delta_{i_1-2}^{i_1+\ell-2}\right) \Gamma' g_\ell + \Gamma g_\ell \\ &= \prod_{s=1}^{j} \left(1 - \delta_{i_s-2s}^{i_s+\ell-2s}\right) g_\ell + \left(1 - \delta_{i_1-2}^{i_1+\ell-2}\right) \Gamma' g_\ell + \Gamma g_\ell, \end{aligned}$$

where $\Gamma' g_\ell$ has norm

$$\|\Gamma' g_\ell\| \leq \|g_\ell\| \sum_{s=2}^{j} \gamma_{i_s-2s}^{i_s+\ell-2s} \prod_{t=2}^{s-1} \left(1 - \delta_{i_t-2t}^{i_t+\ell-2t}\right).$$

Thus we may bound the norm of the righthand error term by:

$$\begin{aligned} \left\| \left(1 - \delta_{i_1-2}^{i_1+\ell-2}\right) \Gamma' g_\ell + \Gamma g_\ell \right\| &\leq \left(1 - \delta_{i_1-2}^{i_1+\ell-2}\right) \|\Gamma'\| \, \|g_\ell\| + \|\Gamma\| \, \|g_\ell\| \\ &\leq \sum_{s=1}^{j} \gamma_{i_s-2s}^{i_s+\ell-2s} \prod_{t=1}^{s-1} \left(1 - \delta_{i_t-2t}^{i_t+\ell-2t}\right) \|g_\ell\|, \end{aligned}$$

as desired. Recalling the shift in i_s by $k - \ell$, we can then bound the resulting error by $(j + k)j\gamma \|g_\ell\|$ since $\delta \in [0, 1]^{d-1}$. □

It is worth noting that when $\gamma = 0$, this implies that the HD-Level-Set decomposition is a true eigendecomposition. Since balanced walks are simply affine combinations of pure walks, this immediately implies a similar result for the more general case. To align with our definition of approximate eigendecompositions and Theorem 21.13, we'll also need the following general relation between $\|g_\ell\|$ and $\|f_\ell\|$ for eposets proved in [DDFH18] (albeit without the exact parameter dependence).

Lemma 21.17 ([DDFH18, Lemma 8.11])**.** *Let (X, Π) be a d-dimensional (δ, γ)-eposet, $0 \leq \ell \leq k < d$, and let*

$$\rho_\ell^k = \prod_{i=1}^{k-\ell} \left(1 - \delta_{k-\ell-i}^{k-i}\right), \quad \rho_{min} = \min_{0 \leq \ell \leq k} \{\rho_\ell^k\}.$$

Then for any $f_\ell = U_\ell^k g_\ell$ for $g_\ell \in Ker(D_\ell)$ we have:

$$\langle f_\ell, f_\ell \rangle \in (\rho_\ell^k \pm k^2\gamma)\langle g_\ell, g_\ell \rangle,$$

and for all $i \neq \ell$:

$$\langle f_\ell, f_i \rangle \leq O\left(\frac{k^2}{\rho_{min}}\gamma \|f_\ell\| \|f_i\|\right).$$

As an aside, we remark that the parameter ρ_ℓ^k turns out to be a crucial throughout much of our work, and while it is difficult to interpret on general eposets, we prove it has a very natural form as long as non-laziness holds.

Claim 21.18 (ρ_ℓ^k for regular eposets)**.** *Let (X, Π) be a regular, γ-non-lazy*[14] *d-dimensional (δ, γ)-eposet. Then for any $i \leq k < d$, we have:*

$$\rho_i^k \in \frac{1}{R(k,i)} \pm err,$$

where $err \leq O\left(\frac{i^3k^2R_{max}}{\delta_i(1-\delta_{i-1})}\gamma\right)$. Likewise as long as $\gamma \leq O\left(\frac{\max_i\{\delta_i(1-\delta_{i-1})\}}{i^3k^2R_{max}^2}\right)$ we have

$$\rho_{min}^{-1} \leq O(R_{max}),$$

where $R_{max} \coloneqq \max_{0 \leq i \leq k}\{R(k,i)\}$.

[14]One can prove this claim more generally for any β-non-laziness, but most γ-eposets of interest are additionally γ-non-lazy, so this simplified version is generally sufficient.

This gives a nice generalization of the interpretation of ρ_i^k on hypergraphs, which is well known to be $\frac{1}{\binom{k}{i}}$ [DDFH18]. We prove this claim in Section 21.8. For simplicity, we will assume throughout the rest of this chapter that our eposets are γ-non-lazy, which is true for most cases of interest (see Section 21.8). All results holds in the more general case using ρ_i^k unless otherwise noted.

Combining Proposition 21.16 and Lemma 21.17 immediately implies that the HD-Level-Set Decomposition is an approximate eigendecomposition in the sense of Definition 21.16.

Corollary 21.19. *Let (X,Π) be a (δ,γ)-eposet and let $M = \sum\limits_{Y \in \mathcal{Y}} \alpha_Y Y$ be an HD-walk. For $1 \leq \ell \leq k$, if $f_\ell = U_\ell^k g_\ell$ for some $g_\ell \in H^\ell$, then for $\gamma \leq O\left(\frac{\max_i\{\delta_i(1-\delta_{i-1})\}}{k^5 R_{max}^2}\right)$:*

$$\left\| M f_\ell - \left(\sum_{Y \in \mathcal{Y}} \alpha_Y \lambda_{Y,\delta,\ell}\right) f_\ell \right\| \leq c\gamma \left\| f_\ell \right\|,$$

where $\lambda_{Y,\delta,\ell}$ is the corresponding eigenvalues of the pure balanced walk Y on a $(\delta,0)$-eposet (the form of which are given in Proposition 21.16), and $c \leq O\left((h(M)+k)h(M)R(k,\ell)w(M)\right)$.

Thus as long as the walk in question is self-adjoint (e.g. canonical or swap walk), Theorem 21.13 immediately implies that the true spectrum is concentrated around these approximate eigenvalues.

Before moving on it is instructive (and as we will soon see quite useful) to give an example application of Corollary 21.19 to a basic higher order random walk.

Corollary 21.20 (Spectrum of Lower Canonical Walks)**.** *Let (X,Π) be a (δ,γ)-eposet. The approximate eigenvalues of the canonical lower walk $\check{N}_k^{k-\ell}$ are:*

$$\lambda_j(\check{N}_k^{k-\ell}) = \prod_{s=1}^{k-\ell}(1-\delta_{k-s-j}^{k-s}).$$

Proof. The lower canonical walk $\check{N}_k^{k-\ell} = U_\ell^k D_\ell^k$ is of height $k-\ell$, and has down operator at positions $\{1,\ldots,k-\ell\}$. In the language of Proposition 21.16 we therefore have $i_s = s$, which therefore gives:

$$\lambda_j(\check{N}_k^{k-\ell}) = \prod_{s=1}^{k-\ell}(1 - \delta_{k-s-j}^{k-s}).$$

Note this is 0 when $j > \ell$. $\square$

Similar to the case of ρ_i^k, while this is difficult to interpret in the general setting, the eigenvalues have a very natural form on non-lazy eposets given by the regularity parameters.

Theorem 21.21. *Let (X, Π) be a γ-non-lazy (δ, γ)-eposet. The approximate eigenvalues of the canonical lower walk $\check{N}_k^{k-i}$ are:*

$$\lambda_j(\check{N}_k^{k-i}) \in \frac{R(i,j)}{R(k,j)} \pm c\gamma,$$

where $c \leq O\left(\frac{i^4 k^2 R_{max}}{\delta_i(1-\delta_{i-1})}\gamma\right)$.

The proof requires machinery developed in Section 21.6 and Section 21.8, and is given in Section 21.8.

21.5 Pseudorandomness and the HD-Level-Set Decomposition

Now that we know the spectral structure of HD-walks, we shift to studying their combinatorial structure. In particular, we will focus on how natural notions of pseudorandomness control the projection of functions onto the HD-Level-Set Decomposition.

Before proceeding, we state a simple corollary of Lemma 21.17 that will prove useful going forward:

Corollary 21.22. *Let (X, Π) be a (δ, γ)-eposet and suppose $f \in C_k$ has HD-Level-Set Decomposition $f = f_0 + \ldots + f_k$. If $\gamma \leq \frac{c'\rho_{\min}}{k^3}$ for a sufficiently small constant $c' > 0$, then*

$$\sum_{j=0}^{k} \|f_j\| \leq O(\sqrt{k}\|f\|). \tag{21.1}$$

Moreover, for any subset of indices I, it holds that

$$-\sum_{j \in I} \langle f, f_j \rangle \leq O\left(\frac{k^3 \gamma \|f\|^2}{\rho_{min}}\right).$$

In particular, if $I = \{j : \langle f, f_j \rangle \leq 0\}$, then

$$\sum_{j \in I} |\langle f, f_j \rangle| \leq O\left(\frac{k^3 \gamma \|f\|^2}{\rho_{min}}\right).$$

Proof. For the first claim, recall that by the approximate orthogonality of the HD-Level-Set Decomposition (Lemma 21.17), we have for all $i \neq j$:

$$|\langle f_i, f_j \rangle| \leq O\left(\frac{k^2}{\rho_{\min}} \gamma \|f_i\| \|f_j\|\right).$$

Then, applying Cauchy-Schwarz gives:

$$\begin{aligned}
\left(\sum_{j=1}^{k} \|f_j\|\right)^2 &\leq k \sum_{j=1}^{k} \|f_j\|^2 \\
&\leq k\langle f, f \rangle - k \sum_{i \neq j \neq 0} \langle f_i, f_j \rangle \\
&\leq k\langle f, f \rangle + c\gamma \sum_{i \neq j \neq 0} \|f_i\| \|f_j\| \\
&\leq k\langle f, f \rangle + c\gamma \left(\sum_{j=1}^{k} \|f_j\|\right)^2
\end{aligned}$$

where $c \leq O\left(\frac{k^3}{\rho_{\min}}\right)$. By our assumption on γ, we have $c\gamma \leq \frac{1}{2}$, and therefore rearranging yields

$$\sum_{i=1}^{k} \|f_j\| \leq O(\sqrt{k} \|f\|).$$

We now show how the second claim is a consequence of the first. For any subset I, we have

$$\begin{aligned}
-\sum_{j\in I}\langle f, f_j\rangle &\leq -\sum_{j\in I}\sum_{i\neq j}\langle f, f_j\rangle \\
&\leq \frac{Ck^2\gamma}{\rho_{\min}}\sum_{i,j}\|f_i\|\,\|f_j\| \\
&= \frac{O(k^2\gamma)}{\rho_{\min}}\left(\sum_{i=0}^{k}\|f_i\|\right)^2 \\
&\leq O\left(\frac{k^3\gamma\,\|f\|}{\rho_{\min}}\right).
\end{aligned}$$

□

21.5.1 ℓ_2-pseudorandomness

We start with pseudorandomness in the ℓ_2-regime, which measures the variance of a set across links.

Definition 21.17 (ℓ_2-Pseudorandom functions [BHKL20]). A function $f \in C_k$ is $(\varepsilon_1, \ldots, \varepsilon_\ell)$-$\ell_2$-pseudorandom if its variance across i-links is small for all $1 \leq i \leq \ell$:

$$\mathrm{Var}(D_i^k f) \leq \varepsilon_i |\mathbb{E}[f]|.$$

In their work on simplicial complexes, BHKL [BHKL20] observed a close connection between ℓ_2-pseudorandomness, the HD-Level-Set Decomposition, and the spectra of the lower canonical walks. We'll show the same connection holds in general for eposets.

Theorem 21.23. *Let (X, Π) be a (δ, γ)-eposet with $\gamma \leq O\left(\frac{\max_i\{\delta_i(1-\delta_{i-1})\}}{k^5 R_{max}^2}\right)$. If $f \in C_k$ has HD-Level-Set Decomposition $f = f_0 + \ldots + f_k$, then for any $\ell \leq k$, $\mathrm{Var}(D_\ell^k f)$ is controlled by its projection onto $V_k^0 \oplus \ldots \oplus V_k^\ell$ in the following sense:*

$$\mathrm{Var}(D_\ell^k f) \in \sum_{j=1}^{\ell}\lambda_j(\check{N}_k^{k-\ell})\langle f, f_j\rangle \pm c_k\gamma\|f\|^2,$$

where $c_k \leq O(k^{5/2}R_{max})$ and $\lambda_j(\check{N}_k^{k-\ell}) = \prod_{s=1}^{k-\ell}(1-\delta_{k-s-j}^{k-s})$.

Proof. To start, notice that since $\langle D^k_\ell f, D^k_\ell f\rangle = \langle \check{N}^{k-\ell}_k f, f\rangle$ it is enough to analyze the application of $\check{N}^{k-\ell}_k$ to f. By Corollary 21.20, we know that each f_j is an approximate eigenvector satisfying:

$$\left\|\check{N}^{k-\ell}_k f_j - \lambda_j(\check{N}^{k-\ell}_k) f_j\right\| \leq O(k^2 R(k,\ell)\gamma) \left\|f_j\right\|,$$

where $\lambda_j(\check{N}^{k-\ell}_k) = 0$ for $j > \ell$. Combining these observations gives:

$$\begin{aligned}
\mathrm{Var}(D^k_\ell f) &= \left\langle D^k_\ell f, D^k_\ell f\right\rangle - \mathbb{E}[D^k_\ell f]^2 \\
&= \left\langle f, U^k_\ell D^k_\ell f\right\rangle - \langle f, f_0\rangle \\
&= \sum_{j=1}^{k} \langle f, U^k_\ell D^k_\ell f_j\rangle \\
&\in \sum_{j=1}^{\ell} \lambda_j(\check{N}^{k-\ell}_k)\langle f, f_j\rangle \pm O\left(\frac{k^2}{\rho_{\min}}\gamma \left\|f\right\| \sum_{j=1}^{k} \left\|f_j\right\|\right).
\end{aligned}$$

where we have additionally used the fact that $\langle f, f_0\rangle = \mathbb{E}[f]^2 = \mathbb{E}[D^k_\ell f]^2$ and $\lambda_0(\check{N}^{k-\ell}_k) = 1$. Applying Equation (21.1) from Corollary 21.22 to bound the sum in the error term and replacing ρ with the relevant regularity parameters by Claim 21.18 then gives the result. □

As an immediate corollary, we get a level-i inequality for pseudorandom functions.

Corollary 21.24. *Let (X,Π) be a (δ,γ)-eposet with $\gamma \leq O\left(\frac{\max_i\{\delta_i(1-\delta_{i-1})\}}{k^5 R^2_{max}}\right)$ and let $f \in C_k$ be an $(\varepsilon_1,\ldots,\varepsilon_\ell)$-$\ell_2$-pseudorandom function. Then for any $1 \leq i \leq \ell$:*

$$|\langle f, f_i\rangle| \leq R(k,i)\varepsilon_i|\mathbb{E}[f]| + c\gamma\|f\|^2,$$

where $c \leq O\left(\frac{k^5 R^2_{max}}{\max_i\{\delta_i(1-\delta_{i-1})\}}\right)$.

Proof. By Corollary 21.22, for any given $1 \leq i \leq k$, it holds that $-\sum_{j\neq i}\langle f, f_j\rangle \leq O\left(\frac{k^3}{\rho_{\min}}\gamma\|f\|^2\right)$. It follows from Theorem 21.23 that for all $0 \leq i \leq k$, the variance of $D^k_i f$ is lower bounded by its projection onto f_i:

$$\mathrm{Var}(D^k_i f) \geq \lambda_i(\check{N}^{k-i}_k)\langle f, f_i\rangle - c\gamma\langle f, f\rangle,$$

where $c \leq O(\frac{k^3}{\rho_{\min}})$. Noting that $\lambda_i(\check{N}_k^{k-i}) = \rho_i^k$, if $i \leq \ell$, re-arranging the above and applying the pseudorandomness assumption gives:

$$\begin{aligned}\langle f, f_i \rangle &\leq \frac{1}{\rho_i^k}\mathrm{Var}(D_i^k f) + c_2\gamma\langle f, f\rangle\\ &\leq \frac{1}{\rho_i^k}\varepsilon_i|\mathbb{E}[f]| + c_2\gamma\langle f, f\rangle,\end{aligned}$$

where $c_2 \leq O(\frac{k^3}{\rho_{\min}^2})$. The lower bound on $\langle f, f_i\rangle$ is immediate from Corollary 21.22 with the set $I = \{i\}$. Applying Claim 21.18 then gives the result. $\square$

As mentioned previously, this also recovers the tight inequality for simplicial complexes given in [BHKL20] where $R(k,i) = \binom{k}{i}$, as well as providing the natural q-analog for q-simplicial complexes where $R(k,i) = \binom{k}{i}_q$.

21.5.2 ℓ_∞-pseudorandomness

While ℓ_2-pseudorandomness is useful in its own right (e.g. for local-to-global algorithms for unique games [BBK+20, BHKL20]), there is also significant interest in a stronger ℓ_∞-variant in the hardness of approximation literature [KMMS18, SMS18].

Definition 21.18 (ℓ_∞-Pseudorandom functions)**.** A function $f \in C_k$ is $(\varepsilon_1, \ldots, \varepsilon_\ell)$-$\ell_\infty$-pseudorandom if for all $1 \leq i \leq \ell$ its local expectation is close to its global expectation:

$$\left\|D_i^k f - \mathbb{E}[f]\right\|_\infty \leq \varepsilon_i.$$

In their recent work on ℓ_2-structure of expanding simplicial complexes, BHKL prove a basic reduction from ℓ_∞ to ℓ_2-pseudorandomness that allows for an analogous level-i inequality for this notion as well. Here, we'll show the same result holds for general eposets. As in their work, we'll take advantage of a weak local-consistency property called locally-constant sign.

Definition 21.19 (locally-constant sign [BHKL20])**.** Let (X, Π) be a graded poset. We say a function $f \in C_k$ has ℓ-local constant sign if:

1. $\mathbb{E}[f] \neq 0$,
2. $\forall s \in X(\ell)$ s.t. $\underset{X_s}{\mathbb{E}}[f] \neq 0 : \operatorname{sign}\left(\underset{X_s}{\mathbb{E}}[f]\right) = \operatorname{sign}\left(\mathbb{E}[f]\right)$.

With this in mind, we now state ℓ_∞-variant of Corollary 21.24:

Theorem 21.25. *Let (X, Π) be a (δ, γ)-eposet with $\gamma \leq O\left(\frac{\max_i\{\delta_i(1-\delta_{i-1})\}}{k^5 R_{max}^2}\right)$ and let $f \in C_k$ have HD-Level-Set Decomposition $f = f_0 + \ldots + f_k$. If f is $(\varepsilon_1, \ldots, \varepsilon_\ell)$-$\ell_\infty$-pseudorandom, then for all $1 \leq i \leq \ell$:*

$$|\langle f, f_i \rangle| \leq (R(k,i) + c\gamma)\, \varepsilon_i^2 + c\gamma \|f\|^2,$$

and if f has i-local constant sign:

$$|\langle f, f_i \rangle| \leq (R(k,i) + c\gamma)\, \varepsilon_i |\mathbb{E}[f]| + c\gamma \|f\|^2$$

where in both cases $c \leq O\left(\frac{k^5 R_{max}^2}{\max_i\{\delta_i(1-\delta_{i-1})\}}\right)$.

We note that when f is boolean, this bound simplifies to

$$|\langle f, f_i \rangle| \leq (R(k,i)\varepsilon_i + c\gamma)\, \mathbb{E}[f],$$

which we'll see in the next section is a particularly useful form for analyzing edge expansion. The proof of Theorem 21.25 relies mainly on a reduction to the ℓ_2-variant for functions with locally-constant sign. This reduction is almost exactly the same as in [BHKL20], but we include it for completeness.

Lemma 21.26. *Let (X, Π) be a graded poset and $f \in C_k$ a $(\varepsilon_1, \ldots, \varepsilon_\ell)$-$\ell_\infty$-pseudorandom function with i-local constant sign for any $i \leq \ell$. Then f is $(\varepsilon_1, \ldots, \varepsilon_\ell)$-$\ell_2$-pseudorandom.*

Proof. As in [BHKL20], the idea is to notice that locally constant sign allows us to rewrite $\left\|D_i^k f\right\|_2^2$ as an expectation over some related distribution P_i:

$$\begin{aligned}\frac{1}{\mathbb{E}[f]}\langle D_i^k f, D_i^k f\rangle &= \sum_{s\in X(i)} \pi_i(s)\left(\frac{1}{\mathbb{E}[f]}\sum_{t\in X_s}\frac{\pi_k(t)f(t)}{\pi_k(X_s)}\right) D_i^k f(s)\\ &= \sum_{s\in X(i)}\left(\frac{1}{R(k,i)}\frac{\sum_{t\in X_s}\pi_k(t)f(t)}{\mathbb{E}[f]}\right) D_i^k f(s)\\ &= \mathop{\mathbb{E}}_{P_i}[D_i^k f],\end{aligned}$$

where P_i being a probability distribution follows from the locally-constant sign of f, and the second step follows from the fact that $\pi_k(X_s) = \sum_{t\in X_s}\pi_k(t) = R(k,i)\pi_i(s)$. The result then follows easily from averaging:

$$\left|\frac{1}{\mathbb{E}[f]}\mathrm{Var}(D_i^k f)\right| = \left|\mathop{\mathbb{E}}_{P_i}[D_i^k f] - \mathbb{E}[f]\right| \leq \|D_i^k f - \mathbb{E}[f]\|_\infty.$$

When $\mathbb{E}[f] > 0$, the ℓ_∞-norm here may be replaced with maximum. □

The proof of Theorem 21.25 now follows from reducing to the case of locally-constant sign. The argument is exactly as in the proof of [BHKL20, Theorem 8.7], but we include it for completeness.

Proof of Theorem 21.25. We focus on the general bound, since the result for functions with locally constant sign is immediate from Lemma 21.26 and Corollary 21.24. The argument for general functions f follows simply from noting that we can always shift f to have locally constant sign. With this in mind, assume $\mathbb{E}[f] \geq 0$ for simplicity (the negative case is similar). Let $f' = f + (\varepsilon_i - \mathbb{E}[f])\mathbb{1}$ be the aforementioned shift. As long as $\varepsilon_i > 0$, it is easy to see that f' has i-local constant sign and further that

$$f' = f'_0 + f_i + \ldots + f_k,$$

where $f'_0 = f_0 + (\varepsilon_i - \mathbb{E}[f])\mathbb{1}$. Since shifts have no effect on ℓ_∞-pseudorandomness, f' is $(\varepsilon_1, \ldots, \varepsilon_\ell)$-$\ell_\infty$-pseudorandom by assumption, and therefore $(\varepsilon_1, \ldots, \varepsilon_\ell)$-$\ell_2$-pseudorandom by Lemma 21.26. We can now apply Corollary 21.24 to get:

$$\begin{aligned}\langle f + (\varepsilon_i - \mathbb{E}[f])\mathbb{1}, f_i\rangle &\leq \frac{1}{\rho_i^k}\varepsilon_i \mathbb{E}[f + (\varepsilon_i - \mathbb{E}[f])\mathbb{1}] + c\gamma\langle f + (\varepsilon_i - \mathbb{E}[f])\mathbb{1}, f + (\varepsilon_i - \mathbb{E}[f])\mathbb{1}\rangle \\ &\leq \left(\frac{1}{\rho_i^k} + c\gamma\right)\varepsilon_i^2 + c\gamma\langle f, f\rangle,\end{aligned}$$

since $\langle f_i, \mathbb{1}\rangle = 0$ for all $i > 0$. Finally, as this holds for all $\varepsilon_i > 0$, a limiting argument implies the result for $\varepsilon_i = 0$. Applying Claim 21.18 completes the proof. □

21.6 Expansion of HD-walks

It is well known that higher order random walks on simplicial complexes (e.g. the Johnson graphs) are not small-set expanders. BHKL gave an exact characterization of this phenomenon for local-spectral expanders: they showed that the expansion of any i-link with respect to an HD-walk M is almost exactly $1 - \lambda_i(M)$. Moreover, using the level-i inequality from the previous section, BHKL proved a tight converse to this result in an ℓ_2-sense: *any* non-expanding set must have high variance across links. This gave a complete ℓ_2-characterization of non-expanding sets on local-spectral expanders, and lay the structural groundwork for new algorithms for unique games over HD-walks.

In this section, we'll show that these results extend to general expanding posets. To start, let's recall the definition of edge expansion.

Definition 21.20 (Weighted Edge Expansion)**.** Let (X, Π) be a graded poset and M a k-dimensional HD-Walk. The weighted edge expansion of a subset $S \subset X(k)$ with respect to M is

$$\Phi(S) = \mathop{\mathbb{E}}_{v \sim \pi_k|_S}\left[M(v, X(k) \setminus S)\right],$$

where

$$M(v, X(k) \setminus S) = \sum_{y \in X(k)\setminus S} M(v, y)$$

and $M(v, y)$ denotes the transition probability from v to y.

Before we prove the strong connections between links and expansion, we need to introduce an important property of HD-walks, monotonic eigenvalue decay.

Definition 21.21 (Monotonic HD-walk)**.** Let (X, Π) be a (δ, γ)-eposet. We call an HD-walk M monotonic if its approximate eigenvalues $\lambda_i(M)$ (given in Corollary 21.19) are non-increasing.

Most HD-walks of interest (e.g. pure walks, partial-swap walks on simplicial or q-simplicial complexes, etc.) are monotonic. This property will be crucial to understanding expansion. To start, let's see how it allows us to upper bound the expansion of links.

Theorem 21.27 (Local Expansion vs Global Spectra)**.** *Let (X, Π) be a (δ, γ)-eposet and M be a k-dimensional monotonic HD-walk. Then for all $0 \leq i \leq k$ and $\tau \in X(i)$:*

$$\Phi(X_\tau) \in 1 - \lambda_i(M) \pm c\gamma,$$

where $c \leq O\left(\frac{k^5 R_{max}^2 (h(M)+k) h(M) w(M)}{\delta_{k-i}^k (1-\delta_{i-1})}\right)$.

The key to proving Theorem 21.27 is to show that the weight of an i-link lies almost entirely on level i of the HD-Level-Set Decomposition. To show this, we'll rely another connection between regularity and eposet parameters for non-lazy posets.

Claim 21.28. *Let (X, Π) be a d-dimensional (δ, γ)-eposet. Then for every $1 \leq k \leq d$ and $0 \leq i \leq k$, the following approximate relation between the eposet and regularity parameters holds:*

$$\lambda_i(N_k^1) \in \frac{R(k,i)}{R(k+1,i)} \pm \left(\gamma_{k-i}^k + R(k,i)\delta_{k-i}^k \gamma\right)$$

where we recall $\lambda_i(N_k^1) = 1 - \prod_{j=i}^{k} \delta_j$.

We prove this relation in Section 21.8. With this in hand, we can show links project mostly onto their corresponding level.

Lemma 21.29. *Let (X,Π) be a d-dimensional (δ,γ)-eposet with $\gamma \leq O\left(\frac{\max_i\{\delta_i(1-\delta_{i-1})\}}{k^5 R_{max}^2}\right)$. Then for all $0 \leq i \leq k < d$ and $\tau \in X(i)$, $\mathbb{1}_{X_\tau}$ lies almost entirely in V_k^i. That is for all $j \neq i$:*

$$\left|\frac{\langle \mathbb{1}_{X_\tau,i}, \mathbb{1}_{X_\tau,j}\rangle}{\langle \mathbb{1}_{X_\tau}, \mathbb{1}_{X_\tau}\rangle}\right| \leq O\left(\frac{k^3 R_{max}}{\delta_{k-i}^k(1-\delta_{i-1})}\gamma\right).$$

Proof. We'll show that the expansion of $\mathbb{1}_{X_\tau}$ with respect to the upper walk N_k^1 is almost exactly $1-\lambda_i(N_k^1)$, which implies most of the weight must lie on V_i^k. We'll start by analyzing the expansion of $\mathbb{1}_{X_\tau}$ through a simple combinatorial argument. First, since D and U are adjoint we have:

$$\begin{aligned}\bar{\Phi}(\mathbb{1}_{X_\tau}) &= \frac{\langle \mathbb{1}_{X_\tau}, D_{k+1}U_k\mathbb{1}_{X_\tau}\rangle}{\langle \mathbb{1}_{X_\tau}, \mathbb{1}_{X_\tau}\rangle} \\ &= \frac{\langle U_k\mathbb{1}_{X_\tau}, U_k\mathbb{1}_{X_\tau}\rangle}{\langle \mathbb{1}_{X_\tau}, \mathbb{1}_{X_\tau}\rangle}.\end{aligned}$$

The trick is now to notice that $\langle \mathbb{1}_{X_\tau}, \mathbb{1}_{X_\tau}\rangle = R(k,i)\pi_i(\tau)$, and $\langle U_k\mathbb{1}_{X_\tau}, U_k\mathbb{1}_{X_\tau}\rangle = \frac{R(k,i)^2}{R(k+1,i)}\pi_i(\tau)$. As a result, applying Claim 21.28 gives:

$$\bar{\Phi}(\mathbb{1}_{X_\tau}) \in \lambda_i(N_k^1) \pm (c\gamma + R(k,i)\gamma),$$

for $c \leq k\gamma$. To see why this implies that most of the weight lies on V_i^k, note that we can also unfold the expansion of $\mathbb{1}_{X_\tau}$ in terms of the HD-Level-Set decomposition:

$$\begin{aligned}\bar{\Phi}(\mathbb{1}_{X_\tau}) &= \frac{1}{\langle \mathbb{1}_{X_\tau}, \mathbb{1}_{X_\tau}\rangle}\sum_{j=0}^{i}\langle \mathbb{1}_{X_\tau}, N_k^1\mathbb{1}_{X_\tau,j}\rangle \\ &\in \frac{1}{\langle \mathbb{1}_{X_\tau}, \mathbb{1}_{X_\tau}\rangle}\sum_{j=0}^{i}\lambda_i(N_k^1)\langle \mathbb{1}_{X_\tau}, \mathbb{1}_{X_\tau,j}\rangle \pm c_2\gamma\end{aligned}$$

where $c_2 \leq \frac{k\sqrt{k}}{\rho_{\min}}$. Recall from Corollary 21.22 that for the set I of indices with negative inner product, it holds that $-\sum_{j\in I}\langle \mathbb{1}_{X_\tau}, \mathbb{1}_{X_\tau,j}\rangle \leq O\left(\frac{k^3}{\rho_{\min}}\gamma\langle \mathbb{1}_{X_\tau}, \mathbb{1}_{X_\tau}\rangle\right)$. Moreover, the positive inner products (i.e. the indices not in I) must sum to at least $\langle \mathbb{1}_{X_\tau}, \mathbb{1}_{X_\tau}\rangle$. Then if there exists some $j \neq i$ such that $\langle \mathbb{1}_{X_\tau}, \mathbb{1}_{X_\tau,j}\rangle > c_3\langle \mathbb{1}_{X_\tau}, \mathbb{1}_{X_\tau}\rangle$ for large enough $c_3 \leq O\left(\frac{1}{\delta_{k-i}^k(1-\delta_{i-1})}\cdot\left(\frac{k^3}{\rho_{\min}}\gamma + R(k,i)\gamma\right)\right)$, the non-expansion would be strictly larger than $\lambda_i(N_k^1) + c\gamma + R(k,i)\gamma$ giving the desired contradiction (note that $(1-\delta_{i-1})\delta_{k-1}^k$

is the gap between the $i-1$st and ith approximate eigenvalue). The form in the theorem statement then follows from applying Claim 21.18. $\square$

We note that the above is the only result in our work that truly relies on non-laziness (it is used only to replace ρ with regularity in all other results). It is possible to recover the upper bound in Theorem 21.27 for general eposets via arguments used in [BHKL20], but the lower bound remains open for concentrated posets. With that in mind, we now prove Theorem 21.27.

Proof of Theorem 21.27. By the previous lemma, we have

$$\left|\frac{\langle \mathbb{1}_{X_\tau}, \mathbb{1}_{X_\tau,j}\rangle}{\langle \mathbb{1}_{X_\tau}, \mathbb{1}_{X_\tau}\rangle}\right| \leq O\left(\frac{1}{\delta_{k-i}^k(1-\delta_{i-1})}\cdot\left(\frac{k^3}{\rho_{\min}}\gamma + R(k,i)\gamma\right)\right).$$

Expanding out $\bar{\Phi}(\mathbb{1}_{X_\tau})$ then gives:

$$\begin{aligned}\bar{\Phi}(\mathbb{1}_{X_\tau}) &= \frac{1}{\langle \mathbb{1}_{X_\tau}, \mathbb{1}_{X_\tau}\rangle}\sum_{j=0}^{i}\langle \mathbb{1}_{X_\tau}, M\mathbb{1}_{X_\tau,j}\rangle \\ &\leq \frac{1}{\langle \mathbb{1}_{X_\tau}, \mathbb{1}_{X_\tau}\rangle}\sum_{j=0}^{i}\lambda_i(M)\langle \mathbb{1}_{X_\tau}, \mathbb{1}_{X_\tau,j}\rangle + c_2\gamma \\ &\leq \lambda_i(M)\frac{\langle \mathbb{1}_{X_\tau}, \mathbb{1}_{X_\tau,i}\rangle}{\langle \mathbb{1}_{X_\tau}, \mathbb{1}_{X_\tau}\rangle} + err_1 \\ &\leq \lambda_i(M) + err_2.\end{aligned}$$

where $c_2, err_1, err_2 \leq O\left(\frac{k}{\delta_{k-i}^k(1-\delta_{i-1})}\left(\frac{k^2(h(M)+k)h(M)w(M)}{\rho_{\min}}\gamma + R(k,i)\gamma\right)\right)$ and the last step follows from the approximate orthogonality. As usual, the form in the theorem statement then follows from applying Claim 21.18. $\square$

Altogether, we've seen that for sufficiently nice expanding posets, the expansion of any i-link with respect to an HD-walk is almost exactly $1-\lambda_i(M)$. Since HD-walks are generally poor expanders (have large $\lambda_1(M)$), Theorem 21.27 implies that links are examples of small, non-expanding sets. Following BHKL, we'll now prove a converse to this result: any non-expanding set must be explained by some structure inside links. To help give a precise statement, we first recall BHKL's notion of Stripped Threshold Rank (specialized to eposets for convenience).

Definition 21.22 (Stripped Threshold Rank [BHKL20])**.** Let (X, Π) be a (δ, γ)-eposet and M a k-dimensional HD-walk with γ small enough that Theorem 21.13 implies the HD-Level-Set Decomposition has a corresponding decomposition of disjoint eigenstrips $C_k = \bigoplus W_k^i$. The ST-Rank of M with respect to η is the number of strips containing an eigenvector with eigenvalue at least η:

$$R_\eta(M) = |\{W_k^i : \exists f \in V^i, Mf = \lambda f, \lambda > \eta\}|.$$

We often write just R_η when M is clear from context.

With this in mind, we'll show a converse to Theorem 21.27 in both ℓ_2 and ℓ_∞ senses (respectively that any non-expanding set must have high variance over links, and must be more concentrated than expected in some particular link). It is convenient to express these results through their contrapositives: that pseudorandom sets expand. The proof is the same as in [BHKL20] for simplicial complexes, but we include it for completeness.

Theorem 21.30. *Let (X, Π) (δ, γ)-eposet, M a k-dimensional, monotonic HD-walk, and γ small enough that the eigenstrip intervals of Theorem 21.13 are disjoint. For any $\eta > 0$, let $r = R_\eta(M) - 1$. Then the expansion of a set $S \subset X(k)$ of density α is at least:*

$$\Phi(S) \geq 1 - \alpha - (1-\alpha)\eta - \sum_{i=1}^{r} (\lambda_i(M) - \eta) R(k,i) \varepsilon_i - c\gamma$$

where S is $(\varepsilon_1, \ldots, \varepsilon_r)$-pseudorandom and $c \leq O\left(\frac{k^5 R_{max}^2 (h(M)+k) h(M) w(M)}{\max_i \{\delta_i (1-\delta_{i-1})\}}\right)$.

Proof. Let $\mathbb{1}_S = \mathbb{1}_{S,0} + \ldots + \mathbb{1}_{S,k}$ be the HD-Level-Set Decomposition of the indicator of S. By linearity of the inner product, we may then write:

$$\begin{aligned}
\Phi(S) &= 1 - \frac{1}{\mathbb{E}[\mathbb{1}_S]}\langle \mathbb{1}_{X_\tau}, M\mathbb{1}_{X_\tau}\rangle \\
&= 1 - \frac{1}{\mathbb{E}[\mathbb{1}_S]}\sum_{j=0}^{k}\langle \mathbb{1}_S, M\mathbb{1}_{S,j}\rangle \\
&= 1 - \frac{1}{\mathbb{E}[\mathbb{1}_S]}\sum_{j=0}^{k}\lambda_j(M)\langle \mathbb{1}_S, \mathbb{1}_{S,j}\rangle - \frac{1}{\mathbb{E}[\mathbb{1}_S]}\sum_{j=0}^{k}\langle \mathbb{1}_S, \Gamma_j\mathbb{1}_{S,j}\rangle
\end{aligned}$$

where $\|\Gamma_j\| \leq O\left((h(M)+k)h(M)\frac{w(M)}{\rho_{\min}}\right)$. The trick is now to notice we can bound the righthand error term using Cauchy-Schwarz:

$$\begin{aligned}
\left|\frac{1}{\mathbb{E}[\mathbb{1}_S]}\sum_{j=0}^{k}\langle \mathbb{1}_S, \Gamma_j\mathbb{1}_{S,j}\rangle\right| &\leq \frac{1}{\mathbb{E}[\mathbb{1}_S]}\sum_{j=0}^{k}|\langle \mathbb{1}_S, \Gamma_j\mathbb{1}_{S,j}\rangle| \\
&\leq \frac{1}{\mathbb{E}[\mathbb{1}_S]}\sum_{j=0}^{k}\|\Gamma_j\|\,\|\mathbb{1}_S\|\,\|\mathbb{1}_{S,j}\| \\
&\leq c\gamma\frac{\|\mathbb{1}_S\|}{\mathbb{E}[\mathbb{1}_S]}\sum_{j=0}^{k}\|\mathbb{1}_{S,j}\| \\
&\leq c_1\gamma,
\end{aligned}$$

where $c \leq O\left((h(M)+k)h(M)\frac{w(M)}{\rho_{\min}}\right)$ and $c_1 \leq O(\sqrt{k}c)$ by Equation (21.1). Since M is a monotonic walk, we can further write:

$$\begin{aligned}
\Phi(S) &\geq 1 - \frac{1}{\mathbb{E}[\mathbb{1}_S]}\sum_{i=0}^{r}\lambda_i(M)\langle \mathbb{1}_S, \mathbb{1}_{S,i}\rangle - \frac{1}{\mathbb{E}[\mathbb{1}_S]}\sum_{i=r+1}^{k}\lambda_i(M)\langle \mathbb{1}_S, \mathbb{1}_{S,i}\rangle - c_1\gamma \\
&\geq 1 - \frac{1}{\mathbb{E}[\mathbb{1}_S]}\sum_{i=0}^{r}\lambda_i(M)\langle \mathbb{1}_S, \mathbb{1}_{S,i}\rangle - \frac{\eta}{\mathbb{E}[\mathbb{1}_S]}\sum_{i=r+1}^{k}\langle \mathbb{1}_S, \mathbb{1}_{S,i}\rangle - c_2\gamma \\
&= 1 - \frac{1}{\mathbb{E}[\mathbb{1}_S]}\sum_{i=0}^{r}\lambda_i(M)\langle \mathbb{1}_S, \mathbb{1}_{S,i}\rangle - \eta\left(1 - \frac{1}{\mathbb{E}[\mathbb{1}_S]}\sum_{i=0}^{r}\langle \mathbb{1}_S, \mathbb{1}_{S,i}\rangle\right) - c_2\gamma \\
&= 1 - \eta - \frac{1}{\mathbb{E}[\mathbb{1}_S]}\sum_{i=0}^{r}(\lambda_i(M)-\eta)\langle \mathbb{1}_S, \mathbb{1}_{S,i}\rangle - c_2\gamma \\
&= 1 - \eta - (1-\eta)\alpha - \frac{1}{\mathbb{E}[\mathbb{1}_S]}\sum_{i=1}^{r}(\lambda_i(M)-\eta)\langle \mathbb{1}_S, \mathbb{1}_{S,i}\rangle - c_2\gamma,
\end{aligned}$$

where $c_2 \leq O\left(k^2(h(M)+k)h(M)\frac{w(M)}{\rho_{\min}}\right)$. To justify the second inequality, observe that for any $r < i \leq k$ such that $\langle \mathbb{1}_S, \mathbb{1}_{S,i} \rangle \geq 0$, replacing $\lambda_i(M)$ with η is valid. For the set I of $r < i \leq k$ with negative inner product, Corollary 21.22 implies that the sum over I is $O(k^3\gamma/\rho_{\min})$, so the inequality remains valid by absorbing the small error into c_2. Applying Corollary 21.24 to bound $\langle \mathbb{1}_S, \mathbb{1}_{S,i} \rangle$ then gives the ℓ_2-variant result, Theorem 21.25 gives the ℓ_∞-variant, and Claim 21.18 gives the form given in the theorem statement. $\square$

We note that Theorem 21.30 recovers the analogous result for simplicial complex in [BHKL20] by plugging in the appropriate value $R(k,i) = \binom{k}{i}$. BHKL also prove this special case is tight in two senses. First, they show that if one wants to retain linear dependence on the pseudorandomness parameter ε, Theorem 21.30 is tight in both the ℓ_2 and ℓ_∞-regimes. Second, they show that the dependence on k is necessary in the ℓ_2-regime, even if we allow sub-optimal dependence on ε. In the next section, we'll generalize this result to q-simplicial complexes as well. In both cases the proofs are highly structural and depend on the underlying structure of the poset—it remains an interesting open problem whether this bound is tight for all poset structures.

21.7 The Grassmann and q-eposets

In this section, we examine the specification of our results on eposets to expanding subsets of the Grassmann poset. We show that our analysis is tight in this regime via a classic example of a small non-expanding set in the Grassmann graphs called co-links.

21.7.1 Spectra

We'll start by examining the spectrum of HD-walks on the Grassmann and q-eposets. We'll focus our attention in this section on the most widely used walks in the literature, the canonical and partial-swap walks. To start, recall that the Grassmann poset itself is sequentially differential with parameters

$$\delta_i = \frac{(q^i - 1)(q^{n-i+1} - 1)}{(q^{i+1} - 1)(q^{n-i} - 1)}. \tag{21.2}$$

Plugging this into Proposition 21.16 gives a nice exact form for the spectra of canonical walks.

Corollary 21.31 (Grassmann Poset N_k^j Spectra)**.** *Let $X = G_q(n, d)$ be the Grassmann Poset, $k + j \leq d$, and $f_\ell = U_\ell^k g_\ell$ for some $g_\ell \in H^\ell$. Then:*

$$N_k^j f_\ell = \lambda_\ell f_\ell,$$

where,

$$\lambda_\ell = q^{\ell j} \frac{\binom{k+j-\ell}{j}_q}{\binom{k+j}{j}_q} \frac{\binom{n-k-\ell}{j}_q}{\binom{n-k}{j}_q} \approx q^{-\ell j}.$$

Proof. By Proposition 21.16 we have that

$$\begin{aligned}
\lambda_\ell(N_k^j) &= \prod_{s=1}^{j} \left(1 - \prod_{i=\ell}^{k-s+j} \delta_i\right) \\
&= \prod_{s=1}^{j} \left(1 - \prod_{i=\ell}^{k-s+j} \frac{(q^i - 1)(q^{n-i+1} - 1)}{(q^{i+1} - 1)(q^{n-i} - 1)}\right).
\end{aligned}$$

The result then follows from telescoping the interior product and simplifying:

$$\begin{aligned}
&= \prod_{s=1}^{j} \left(1 - \frac{(q^\ell - 1)(q^{n-\ell+1} - 1)}{(q^{k-s+j+1} - 1)(q^{n+s-k-j} - 1)}\right) \\
&= q^{\ell j} \left(\prod_{s=1}^{j} \frac{(q^{k+j-s-\ell+1} - 1)}{(q^{k+j-s+1} - 1)}\right) \left(\prod_{s=1}^{j} \frac{(q^{n+s-k-j-\ell} - 1)}{(q^{n+s-k-j} - 1)}\right) \\
&= q^{\ell j} \frac{\binom{k+j-\ell}{j}_q}{\binom{k+j}{j}_q} \frac{\binom{n-k-\ell}{j}_q}{\binom{n-k}{j}_q}
\end{aligned}$$

as desired. □

This recovers a very simple proof of classical results to this effect (see e.g. [Del76]). An analogous computation gives an approximate bound on the spectrum of N_k^j on q-eposets as well.

Corollary 21.32 (q-eposets N_k^j Spectra)**.** *Let (X, Π) be a d-dimensional γ-q-eposet with $\gamma \leq q^{-\Omega(k^2)}$, $k + j \leq d$, and $f_\ell = U_\ell^{k-1} g_\ell$ for some $g_\ell \in H^\ell$. Then:*

$$\left\| N_k^j f_\ell - \frac{\binom{k+j-\ell}{j}_q}{\binom{k+j}{j}_q} f_\ell \right\| \leq O\left(j(j+k) \binom{k}{\ell}_q \right) \gamma \left\| f_\ell \right\|$$

Note that for small enough γ, Theorem 21.13 implies that the true spectra is then concentrated around these values as well. It is also worth noting that these eigenvalues are, as one would expect, the natural q-analog of the corresponding eigenvalues on simplicial complexes.

It turns out that this fact will carry over to the important class of partial-swap walks as well. Partial-swap walks on simplicial complexes were originally analyzed by AJT [AJT19], who showed they can be written as a hypergeometric combination of canonical walks. Their proof is specific to the structure of simplicial complexes, and some work is required to generalize their ideas to the Grassmann case. Following the overall proof strategy of AJT, it will be helpful to first show that the canonical walks themselves can be written as an expectation of swap walks over a q-hypergeometric distribution, and then use the q-binomial inversion theorem to derive the desired result.

Lemma 21.33 (q-analog of [AJT19, Lemma 4.11])**.** *Let (X, Π) be a pure, measured q-simplicial complex. Then:*

$$N_k^j = \sum_{i=0}^{j} q^{i^2} \frac{\binom{j}{i}_q \binom{k}{k-i}_q}{\binom{k+j}{k}_q} S_k^i$$

Proof. We follow the structure and notation of [AJT19, Lemma 4.11]. Assume that the canonical walk starts at a subspace $V \in X(k)$, and walks up to $W \in X(k+j)$. We wish to analyze the probability that upon walking back down to level k, a subspace V' with intersection $k-i$ is chosen, that is:

$$dim(V \cap V') = k - i.$$

Let such an event be denoted $\mathcal{E}_i(W)$. It follows from elementary q-combinatorics (see e.g. [BH12, Lemma 9.3.2]) that

$$\Pr_{V' \subset W}[\mathcal{E}_i(W) \mid W] = q^{i^2} \frac{\binom{j}{i}_q \binom{k}{k-i}_q}{\binom{k+j}{k}_q},$$

where $V' \in X(k)$ is drawn uniformly from the k-dimensional subspaces of W. In essence, we wish to relate this process to the swap walk S^i_k. To do so, note that while the swap walk (as defined) only walks up to $X(k+i)$, walking up to $X(k+j)$ and conditioning on intersection i, a process called the *i-swapping j-walk* by [AJT19], is exactly the same due to symmetry (via the regularity condition, see [AJT19][Proposition 4.9] for a more detailed explanation). Thus consider the i-swapping j-walk, and let T'_i denote the variable standing for the subspace chosen by the walk. Conditioned on picking the same W as the canonical walk in its ascent, we may relate T'_i to the canonical walk:

$$\Pr[T'_i = T \mid W] = \Pr[V' = T \mid W \text{ and } \mathcal{E}_i(W)]$$

We may now decompose the canonical walk by intersection size:

$$\begin{aligned}
N_k^j(V,T) &= \sum_{i=0}^{j} \sum_{W \in X(k+j)} \Pr[W] \Pr[\mathcal{E}_i(W) \mid W] \Pr[V' = T \mid W \text{ and } \mathcal{E}_i(W)] \\
&= \sum_{i=0}^{j} \sum_{W \in X(k+j)} q^{i^2} \frac{\binom{j}{i}_q \binom{k}{k-i}_q}{\binom{k+j}{k}_q} \mathop{\mathbb{E}}_{W \supset V}[\Pr[V' = T \mid W \text{ and } \mathcal{E}_i(W)]] \\
&= \sum_{i=0}^{j} \sum_{W \in X(k+j)} q^{i^2} \frac{\binom{j}{i}_q \binom{k}{k-i}_q}{\binom{k+j}{k}_q} \mathop{\mathbb{E}}_{W \supset V}[\Pr[T_i' = T \mid W\ ]] \\
&= \sum_{i=0}^{j} \sum_{W \in X(k+j)} q^{i^2} \frac{\binom{j}{i}_q \binom{k}{k-i}_q}{\binom{k+j}{k}_q} \Pr[T_i' = T] \\
&= \sum_{i=0}^{j} \sum_{W \in X(k+j)} q^{i^2} \frac{\binom{j}{i}_q \binom{k}{k-i}_q}{\binom{k+j}{k}_q} S_k^i(V,T)
\end{aligned}$$

□

This results in the q-analog of the analogous result on simplicial complexes [AJT19, Lemma 4.11]. To recover the analogous statement writing partial-swap walks in terms of canonical walks, we can now apply a q-Binomial inversion theorem.

Lemma 21.34 (q-Binomial Inversion (Theorem 2.1 [Zou17]))**.** *Suppose* $\{a_i\}_{i\geq 1}$, $\{b_i\}_{i \geq 1}$ *are two sequences. If:*

$$a_j = \sum_{i=1}^{j} (-1)^i \binom{j}{i}_q b_i,$$

then

$$b_j = \sum_{i=1}^{j} (-1)^i q^{\binom{j-i}{2}} \binom{j}{i}_q a_i$$

We note that [Zou17, Theorem 2.1] is stated in slightly more generality in the original work, but the above lemma is an immediate application. With this in hand, we can finally prove that swap walks on the Grassmann poset are indeed HD-walks:

Proposition 21.35. *Let (X, Π) be a weighted pure q-simplicial complex. Then for $k + j \leq d$:*

$$S_k^j = \frac{1}{q^{j^2}\binom{k}{k-j}_q} \sum_{i=0}^{j} (-1)^{j-i} q^{\binom{j-i}{2}} \binom{j}{i}_q \binom{k+i}{i}_q N_k^i,$$

and similarly,

$$J_q(n,k,t) = S_k^{k-t} = \frac{1}{q^{(k-t)^2}\binom{k}{t}_q} \sum_{i=0}^{k-t} (-1)^{k-t-i} q^{\binom{k-t-i}{2}} \binom{k-t}{i}_q \binom{k+i}{i}_q N_k^i$$

Proof. The proof is an easy application of Lemma 21.34 and the q-Binomial theorem. In particular, for any $V, V' \in X(k)$, let

$$a_i = (-1)^i q^{i^2} \binom{k}{k-i}_q S_k^i(V, V').$$

Noting that $N_0^j = S_0^j = I$, Lemma 21.33 gives the following equality:

$$\binom{k+j}{k}_q \left(N_k^j(V,V') - \frac{1}{\binom{k+j}{k}_q} I(V,V') \right) = \sum_{i=1}^{j} (-1)^i \binom{j}{i}_q a_i.$$

Setting the second sequence $\{b_i\}_{i \geq 1}$ to

$$b_i = \binom{k+i}{k}_q \left(N_k^i(V,V') - \frac{1}{\binom{k+i}{k}_q} I(V,V') \right),$$

Lemma 21.34 then implies:

$$\begin{aligned}
q^{j^2}\binom{k}{k-j}_q S_k^j(V,V') &= \sum_{i=1}^{j} (-1)^{j-i} q^{\binom{j-i}{2}} \binom{k+i}{k} \left(N_k^i(V,V') - \frac{1}{\binom{k+i}{k}_q} I(V,V') \right) \\
&= \sum_{i=1}^{j} (-1)^{j-i} q^{\binom{j-i}{2}} \binom{k+i}{k} N_k^i(V,V') - \sum_{i=1}^{j} (-1)^{j-i} q^{\binom{j-i}{2}} I(V,V') \\
&= \sum_{i=0}^{j} (-1)^{j-i} q^{\binom{j-i}{2}} \binom{k+i}{k} N_k^i(V,V')
\end{aligned}$$

where the last step follows from the q-Binomial theorem. □

Once again, we note that this is unsurprisingly the q-analog of the analogous statement on simplicial complexes (see [AJT19, Corollary 4.13]). Finally, we'll use this to show that the eigenvalues of partial-swap walks on q-simplicial complexes are given by the natural q-analog of the simplicial complex case.

Corollary 21.36. *Let (X, Π) be a d-dimensional γ-q-eposet with γ sufficiently small, $k + j \leq d$, and $f_\ell = U_\ell^k g_\ell$ for some $g_\ell \in H^\ell$. Then:*

$$\left\| S_k^j f_\ell - \frac{\binom{k-j}{\ell}_q}{\binom{k}{\ell}_q} f_\ell \right\| \leq O\left(\left(\frac{q}{q-1}\right)^{\min(j,k-j)+2} k^2 \binom{k}{\ell}_q \right) \gamma \left\| f_\ell \right\|$$

Proof. This follows from combining Corollary 21.19, Corollary 21.31, and Proposition 21.35. Let $t = k - j$. In particular, it is sufficient to note that (in the notation of Corollary 21.19):

$$\begin{aligned}
\sum_{Y \in \mathcal{Y}} \alpha_Y \lambda_{Y,\delta,\ell} &= \frac{1}{q^{(k-t)^2} \binom{k}{t}_q} \sum_{i=0}^{k-t} (-1)^{k-t-i} q^{\binom{k-t-i}{2}} \binom{k-t}{i}_q \binom{k+i-\ell}{i}_q \\
&= \frac{\binom{k-j}{\ell}_q}{\binom{k}{\ell}_q}.
\end{aligned}$$

and further that:

$$\begin{aligned}
w(S_k^j) &= \frac{1}{q^{j^2} \binom{k}{k-j}_q} \sum_{i=0}^{j} q^{\binom{j-i}{2}} \binom{j}{i}_q \binom{k+i}{i}_q \\
&\leq \frac{q^{jk}}{q^{j^2} \binom{k}{k-j}_q} \sum_{i=0}^{j} q^{-i} \\
&\leq \left(\frac{q}{q-1}\right)^{\min(j,k-j)+1}
\end{aligned}$$

□

Again, since the swap walks are self-adjoint Theorem 21.13 implies that for small enough γ the true spectra is closely concentrated around these values as well. It is worth noting that if the above analysis is repeated using the exact eposet parameters for the Grassmann (see Equation (21.2)), this recovers the standard eigenvalues of the Grassmann graphs (see e.g. [Del76]).

21.7.2 Pseudorandom functions and small set expansion

With an understanding of the spectra of HD-walks on q-simplicial complexes, we move to studying its combinatorial structure. By direct computation, it is not hard to show that on q-eposets, $\rho_\ell^k = \frac{1}{\binom{k}{\ell}_q}$ (Claim 21.18 would only imply this is approximately true). As a result, we get a level-i inequality for q-simplicial complexes that is the natural q-analog of BHKL's inequality for basic simplicial complexes.

Theorem 21.37. *Let (X, Π) be a γ-q-eposet with $\gamma \leq q^{-\Omega(k^2)}$, and let $f : C_k \to \mathbb{R}$ be any function on k-faces with HD-Level-Set Decomposition $f = f_0 + \ldots + f_k$. If f is $(\varepsilon_1, \ldots, \varepsilon_\ell)$-$\ell_\infty$-pseudorandom, then for all $1 \leq i \leq \ell$:*

$$|\langle f, f_i \rangle| \leq \left(\binom{k}{i}_q + c\gamma \right) \varepsilon_i^2 + c\gamma \|f\|^2 .$$

If f additionally has i-local constant sign or is $(\varepsilon_1, \ldots, \varepsilon_\ell)$-$\ell_2$-pseudorandom, then

$$|\langle f, f_i \rangle| \leq \binom{k}{i}_q \varepsilon_i |\mathbb{E}[f]| + c\gamma \|f\|^2$$

where in both cases $c \leq q^{O(k^2)}$

For large enough q, γ^{-1}, this result is exactly tight. The key to showing this fact is to examine a local structure unique to the Grassmann called *co-links*. The co-link of an element $W \in X(k')$, is all of the subspaces *contained in* W:

$$\bar{X}_W = \{V \in X(k) : V \subseteq W\}.$$

Just like links, co-links of dimension i (that is $k' = d - i$) also come through levels 0 through i of the complex, although this is somewhat trickier to see.

Lemma 21.38 (HD-level-set decomposition of co-links)**.** *Let $X = G_q(d, k)$ and $\mathcal{S} = \overline{X}_W$ be a co-link of dimension i for $W \in X(d - i)$. Then, we have*

$$\mathbb{1}_{\mathcal{S}} \in V_k^0 \oplus \cdots \oplus V_k^i.$$

Proof. Since we know that $V_k^0 \oplus \cdots \oplus V_k^i = \mathrm{Im}(U_i^k C_i)$ (see e.g. [DDFH18]), all we need to do is to show that there exists an $f \in C_i$ such that $\mathbb{1}_{\mathbb{S}} = U_i^k f$. More specifically, we can write $f = \sum_{U \in X(i)} \alpha_U \mathbb{1}_U$. Then, we have

$$(U_i^k f)(V) = \sum_{U \in X(i)} \alpha_U (U_i^k \mathbb{1}_U)(V) = \frac{1}{R(k,i)} \sum_{U \in X(i), U \subset V} \alpha_U.$$

Suppose $\alpha_U = g(\dim(U \cap W))$ for some function $g : \{0, \ldots, i\} \to \mathbb{R}$. We will prove that there exists a unique g that satisfies the desired equations.

Consider the dimension of $V \cap W$. If $V \subset W$, i.e., $\dim(V \cap W) = k$, then for all $U \in X(i)$ s.t. $U \subset V$, $\dim(U \cap W) = i$. Then, for all $V \subset W$ we must have:

$$U_i^k \mathbb{1}_V = \frac{1}{R(k,i)} \sum_{U \in X(i), U \subset V} g(i) = g(i) = 1.$$

On the other hand, consider $V \not\subset W$. In this case we must have $\dim(V \cap W) = k - j$ for some $i \geq j > 0$ and further that $\dim(U \cap W) \in \{i - j, \ldots, i\}$ for all $U \in X(i)$ s.t. $U \subset V$. This gives the following set of linear equations:

$$U_i^k \mathbb{1}_V = \sum_{\ell = i-j}^{i-1} c_{j,\ell}\, g(\ell) + c_{j,i} = 0 \quad \forall 1 \leq j \leq i,$$

where $c_{j,\ell} := R(k,i)^{-1} \cdot |\{U \in X(i) : U \subset V, \dim(U \cap W) = \ell, \dim(V \cap W) = k - j\}|$ is a constant for all $\ell \in \{i - j, \ldots, i\}$. Since this system can be written as a triangular form with positive diagonal, it is invertible and there exists a unique solution for $g(0), \ldots, g(i-1)$ as desired. By definition, such a solution must satisfy $f = \sum_{U \in X(i)} g(\dim(U \cap W)) \mathbb{1}_U$, so we have constructed $f \in C_i$ such that $U_i^k f = \mathbb{1}_{\mathbb{S}}$, which completes the proof of the claim. $\square$

Using this fact, we can show that our level-i inequality is exactly tight.

Proposition 21.39. *Let $X = G_q(d,k)$ be the Grassmann poset. For any $i \leq k \in \mathbb{N}$ and $c < 1$, there exist large enough q, d and a set $S \subset X(k)$ such that*

$$\langle \mathbb{1}_S, \mathbb{1}_{S,i} \rangle > c \binom{k}{i}_q \varepsilon_i \langle \mathbb{1}_S, \mathbb{1}_S \rangle$$

where S is (i, ε_i)-pseudorandom.

Proof. The proof goes through examining a "co-link" of dimension i, that is for $W \in X(d-i)$:

$$\bar{X}_W = \{V \in X(k) : V \subset W\}.$$

For simplicity, let $S \coloneqq \bar{X}_W$. The density of the co-link S in any j-link X_V is:

$$\alpha_j = \frac{(q^{d-i-j}-1)\dots(q^{d-k+1-i}-1)}{(q^{d-j}-1)\dots(q^{d-k+1}-1)} = q^{-i(k-j)} + o_{q,d}(1).$$

The idea is now to examine the (non)-expansion of the co-link with respect to the lower walk $U_{k-1}D_k$. By direct computation, the probability of returning to $\bar{X}_W$ after moving to a $(k-1)$-dimensional subspace is exactly:

$$\bar{\Phi}(\bar{X}_W) = \frac{q^{d-i}-q^{k-1}}{q^d - q^{k-1}} = q^{-i} \pm q^{-\Omega(d)} \tag{21.3}$$

On the other hand, by Proposition 21.16 the approximate eigenvalues of the lower walk are given by

$$\lambda_j = \frac{q^{k-j}-1}{q^k-1} = q^{-j} - O(q^{-k})$$

Since a dimension-i co-link has no projection onto levels $i+1$ through k, we can also write the non-expansion as:

$$\bar{\Phi}(\bar{X}_W) = \frac{1}{\langle \mathbb{1}_S, \mathbb{1}_S \rangle} \sum_{j=0}^{i} q^{-j} \langle \mathbb{1}_S, \mathbb{1}_{S,j} \rangle - O(q^{-k})$$

for large enough q, d. Combining this with our previous formula for the non-expansion in Equation (21.3), we get that there exists a universal constant c' such that for large enough q and d, $\mathbb{1}_{\bar{X}_W}$ cannot have more than a $\frac{c'}{q}$ fraction of its mass on levels 1 through $i-1$. Finally, noticing that:

$$\binom{k}{i}_q \alpha_i = 1 + o_q(1)$$

we have

$$\frac{\langle \mathbb{1}_S, \mathbb{1}_{S,i} \rangle}{\langle \mathbb{1}_S, \mathbb{1}_S \rangle} \geq \frac{q-c'}{q} \geq c \binom{k}{i}_q \alpha_i$$

since the latter is strictly bounded away from 1 for large enough q. This completes the result since $\bar{X}_W$ is (α_i, i)-pseudorandom. □

We'll close the section by giving an immediate application of Theorem 21.37 to the expansion of pseudorandom sets, and briefly discuss connections with the proof of the 2-2 Games Conjecture and algorithms for unique games. Namely, as corollary of Theorem 21.37, we show that for both the canonical and partial-swap walks, sufficiently pseudorandom functions expand near perfectly.

Corollary 21.40 (q-eposets Edge-Expansion)**.** *Let (X, Π) be a d-dimensional γ-q-eposet, $S \subset X(k)$ a subset whose indicator function $\mathbb{1}_S$ is $(\varepsilon_1, \ldots, \varepsilon_\ell)$-pseudorandom. Then the edge expansion of S with respect to the canonical walk N_k^j is bounded by:*

$$\Phi_{\pi_k}(N_k^j, S) \geq 1 - \mathbb{E}[\mathbb{1}_S] - \sum_{i=1}^{\ell} \frac{\binom{k+j-i}{j}_q}{\binom{k+j}{j}_q} \binom{k}{i}_q \varepsilon_i - q^{-(\ell+1)j} - q^{O(k^2)}\gamma$$

Further, the edge expansion of S with respect to the partial-swap walk S_k^j is bounded by:

$$\Phi_{\pi_k}(S_k^j, S) \geq 1 - \mathbb{E}[\mathbb{1}_S] - \sum_{i=1}^{\ell} \binom{k-j}{i}_q \varepsilon_i - q^{-(\ell+1)j} - q^{O(k^2)}\gamma$$

Note that S_k^j on q-eposets is a generalization of the Grassmann Graphs (and are equivalent when X is the Grassmann Poset). While our definition of pseudorandomness is weaker than that of [SMS18] and therefore necessarily depends on the dimension k, we take the above as evidence that the framework of expanding posets may be important for making further progress on the Unique Games Conjecture. In particular, combined with recent works removing this k-dependence on simplicial complexes [BHKL21, GLL22], it seems plausible that the framework of expanding posets may lead to a more general understanding of the structure underlying the unique games conjecture.

21.8 Eposet Parameters and Regularity

In this appendix we will discuss connections between notions of regularity, the averaging operators, and eposet parameters. To start, we'll show that downward and middle regularity (which are defined only on adjacent levels of the poset) imply extended regularity between any two levels.

Proposition 21.41. *Let (X, Π) be a d-dimensional regular measured poset. Then for any $i < k \leq d$, there exist regularity constant $R(k, i)$ such that for any $x_k \in X(k)$, there are exactly $R(k, i)$ elements $x_i \in X(i)$ such that $x_k > x_i$.*

Proof. Given any element $x_k \in X(k)$, downward regularity promises there are exactly $\prod_{j=i+1}^{k} R(j)$ unique chains $x_k < x_{k-1} < \ldots < x_{i+1} < x_i$. By middle regularity, any fixed $x_i \in X(i)$ which appears in this fashion appears in exactly $m(k, i)$ chains. Noting that $x_i < x_k$ if and only if x_i appears in such a chain, the total number of $x_i < x_k$ must be exactly:

$$R(k, i) = \frac{\prod_{j=i+1}^{k} R(j)}{m(k, i)}.$$

□

A similar argument shows that regularity allows the up operators to compose in the natural way.

Proposition 21.42. *Let (X, Π) be a d-dimensional regular measured poset. Then for any $i < k \leq d$ we have:*

$$U_i^k f(x_k) = \frac{1}{R(k, i)} \sum_{x_i < x_k} f(x_i)$$

Proof. Expanding out $U_i^k f(y)$ gives:

$$U_i^k f(x_k) = \frac{1}{\prod\limits_{j=i+1}^{k} R(j)} \sum_{x_{k-1} < x_k} \cdots \sum_{x_i < x_{i+1}} f(x_i)$$

The number of times each x_i appears in this sum is exactly the number of chains starting at x_k and ending at x_i, so by middle regularity:

$$\begin{aligned}\frac{1}{\prod\limits_{j=i+1}^{k} R(j)} \sum_{x_{k-1}<x_k} \cdots \sum_{x_{i+1}<x_i} f(x_i) &= \frac{m(k,i)}{\prod\limits_{j=i+1}^{k} R(j)} \sum_{x_i<x_k} f(x_i) \\ &= \frac{1}{R(k,i)} \sum_{x_i<x_k} f(x_i).\end{aligned}$$

as desired. □

We'll now take a look at the connection between eposet parameters and regularity. It is convenient to first start with a lemma stating that non-laziness is equivalent to bounding the maximum transition probability of the lower walk.

Lemma 21.43. *Let (X,Π) be a d-dimensional measured poset. Then for any $0 < i \leq d$, the maximum laziness of the lower walk is also the maximum transition probability:*

$$\max_{\sigma \in X(i)} \left\{ \mathbb{1}_\sigma^T U_{i-1} D_i \mathbb{1}_\sigma \right\} = \max_{\sigma,\tau \in X(i)} \left\{ \mathbb{1}_\sigma^T U_{i-1} D_i \mathbb{1}_\tau \right\}.$$

Proof. Assume that $\tau \neq \sigma$. Then the transition probability from τ to σ is exactly

$$\begin{aligned}\mathbb{1}_\sigma^T U_{i-1} D_i \mathbb{1}_\tau &= \frac{\pi_\tau(\sigma \setminus \tau)}{R(i,i-1)} \\ &\leq \frac{1}{R(i,i-1)} \sum_{\tau<\sigma} \pi_\tau(\sigma \setminus \tau) \\ &= \mathbb{1}_\sigma^\tau U_{i-1} D_i \mathbb{1}_\sigma,\end{aligned}$$

which implies the result. □

We now prove our two claims relating the eposet parameters to regularity.

Claim 21.44. *Let (X,Π) be a d-dimensional (δ,γ)-eposet. Then for every $1 \leq k \leq d$ and $0 \leq i \leq k$, the following approximate relation between the eposet and regularity parameters holds:*

$$\lambda_i(N_k^1) \in \frac{R(k,i)}{R(k+1,i)} \pm \left(\gamma_{k-i}^k + R(k,i)\delta_{k-i}^k \gamma\right)$$

where we recall $\lambda_i(N_k^1) = 1 - \prod_{j=i}^{k} \delta_j$.

Proof. One of our main analytical tools so far has been the relation between the upper and lower walks given in Lemma 21.15:

$$\left\| D_{k+1}U_i^{k+1} - (1-\delta_{k-i}^k)U_i^k - \delta_{k-i}^k U_{i-1}^k D_i \right\| \leq \gamma_{k-i}^k.$$

For this result, we'll actually need a refinement of this result given in [BHKL20, Lemma A.1]:[15]

$$D_{k+1}U_i^{k+1} = (1-\delta_{k-i}^k)U_i^k + \delta_{k-i}^k U_{i-1}^k D_i + \sum_{j=-1}^{k-i-1} U_{k-j-1}^k \Gamma_j U_i^{k-j-1} \tag{21.4}$$

where $\sum \|\Gamma_j\| \leq \gamma_{k-i}^k$. The idea is now to examine the "laziness" of the two sides of this equality. In other words, given a starting k-face τ, what is the probability that the resulting i-face σ satisfies $\sigma < \tau$?

To start, we'll argue that the laziness of the lefthand side is exactly $\frac{R(k,i)}{R(k+1,i)}$. This follows from noting that there are $R(k,i)$ i-faces σ satisfying $\sigma < \tau$, and $R(k+1,i)$ options after taking the initial up-step of the walk to $\tau' > \tau$. After the down-steps, the resulting i-face is uniformly distributed over these $R(k+1,i)$ options $\sigma < \tau'$, and since every $\sigma < \tau < \tau'$, all original $R(k,i)$ lazy options are still viable after the up-step to τ'.

Analyzing the right-hand side is a bit trickier. The initial term $(1-\delta_{k-i}^k)U_i^k$ is completely lazy, so it contributes exactly $(1-\delta_{k-i}^k) = \lambda_i(N_k^1)$. We'll break the second term into two steps: walking from $X(k)$ to $X(i)$ via U_i^k, then from $X(i)$ to $X(i)$ via the lower walk $U_{i-1}D_i$. Starting at a k-face τ, notice that after applying the down step U_i^k we are uniformly spread over $\sigma < \tau$. Computing the laziness then amounts to asking what the probability of staying in this set is after the application of UD,

[15]Formally the result is only stated for simplicial complexes in [BHKL20], but the same proof holds for eposets.

which one can naively bound by the maximum transition probability times the set size $R(k,i)$. By non-laziness, the maximum transition probability is at most γ (see Lemma 21.43).

The third term can be handled similarly. The first down step U^k_{k-j-1} spreads τ evenly across $\sigma < \tau$ in $X(k-j-1)$. The resulting i-face σ' after applying $\Gamma_j U_i^{k-j-1}$ is less than τ if and only if the intermediary $(k-j-1)$-face after applying Γ_j is less than τ, which is bounded by the spectral norm $\|\Gamma_j\|$.[16]

Putting everything together, since both sides of Equation (21.4) must have equivalent laziness, we get that $\lambda_i(N_k^1)$ must be within $\sum \|\Gamma_j\| + \delta^k_{k-i} R(k,i)\gamma$ as desired. □

Claim 21.18 and Theorem 21.21 can both be proving an analogous theorem for the upper walk.

Claim 21.45 (Regularity and Upper Walk Spectrum)**.** *Let (X,Π) be a d-dimensional (δ,γ)-eposet. Then for any $j \leq i \leq k < d$, we have:*

$$\lambda_j(N_i^{k-i}) \in \frac{R(i,j)}{R(k,i)} \pm err,$$

where $err \leq O\left(\frac{i^4 k^2 R_{max}}{\delta_i(1-\delta_{i-1})}\gamma\right)$.

Proof. This follows almost immediately from the fact that i-links lie almost entirely on the ith eigenstrip (Lemma 21.29). In particular, it is enough to examine the expansion of i-links with respect to the upper canonical walk N_i^{k-i}. On the one hand,

[16]We note that Γ_j is not stochastic, but it is self-adjoint and an easy exercise to see that the analogous reasoning still holds.

for any $j \leq i$ and $\tau \in X(j)$ we have:

$$\begin{aligned}
\bar{\Phi}(X^i_\tau) &= \frac{\langle \mathbb{1}_{X^i_\tau}, N_i^{k-i}\mathbb{1}_{X^i_\tau}\rangle}{\langle \mathbb{1}_{X^i_\tau}, \mathbb{1}_{X^i_\tau}\rangle} \\
&= \frac{\langle U^k_j \mathbb{1}_\tau, U^k_j \mathbb{1}_\tau\rangle}{\langle U^i_j \mathbb{1}_\tau, U^i_j \mathbb{1}_\tau\rangle} \\
&= \frac{R(i,j)^2}{R(k,i)^2}\frac{\langle \mathbb{1}_{X^k_\tau}, \mathbb{1}_{X^k_\tau}\rangle}{\langle \mathbb{1}_{X^i_\tau}, \mathbb{1}_{X^i_\tau}\rangle} \\
&= \frac{R(i,j)}{R(k,i)}\frac{\langle \mathbb{1}_\tau, \mathbb{1}_\tau\rangle}{\langle \mathbb{1}_\tau, \mathbb{1}_\tau\rangle} \\
&= \frac{R(i,j)}{R(k,i)}.
\end{aligned}$$

where we have applied the fact that $\langle X^\ell_\tau, X^\ell_\tau\rangle = R(\ell, j)\langle \mathbb{1}_\tau, \mathbb{1}_\tau\rangle$. On the other hand, by Lemma 21.29 we also have that:

$$\begin{aligned}
\bar{\Phi}(\mathbb{1}_{X^i_\tau}) &= \frac{1}{\langle \mathbb{1}_{X^i_\tau}, \mathbb{1}_{X^i_\tau}\rangle}\sum_{\ell=0}^{i}\langle \mathbb{1}_{X^i_\tau}, N_i^{k-i}\mathbb{1}_{X^i_\tau,\ell}\rangle \\
&\in \frac{1}{\langle \mathbb{1}_\tau, \mathbb{1}_\tau\rangle}\sum_{\ell=0}^{i}\lambda_j(N_i^{k-i})\langle \mathbb{1}_{X^i_\tau}, \mathbb{1}_{X^i_\tau,\ell}\rangle + c\gamma \\
&\in \lambda_j(N_i^{k-i})\frac{\langle \mathbb{1}_\tau, \mathbb{1}_{\tau,j}\rangle}{\langle \mathbb{1}_\tau, \mathbb{1}_\tau\rangle} + \sum_{j=0}^{i} err_1 \\
&\in \lambda_j(N_i^{k-i}) + err_2
\end{aligned}$$

where as in the proof of Theorem 21.27, $c, err_1, err_2 \leq O\left(\frac{i^4k^2R_{\max}}{\delta^i_{i-j}(1-\delta_{j-1})}\gamma\right)$. $\square$

Claim 21.18 follows immediately from observing that $\rho^k_i = \lambda_i(N_i^{k-i})$ (by Proposition 21.16). Theorem 21.21 follows from observing that $\widehat{N}_i^{k-i}$ and $\check{N}_k^{k-i}$ have the same approximate eigenvalues (similarly by Proposition 21.16).

Finally we close out the section by discussing the connection between non-laziness and a variant of eposets called local-spectral expanders [KT21a]. To start, let's recall this latter definition.

Definition 21.23 (Local-Spectral Expander [DK17, KT21a])**.** A d-dimensional measured poset (X, Π) is a γ-local-spectral expander if the graph underlying every link[17] of dimension at most $d-2$ is a γ-spectral expander.[18]

Under suitable regularity conditions (see [KT21a]), local-spectral expansion is equivalent to the notion of expanding posets used in this chapter. A simple argument shows that γ-local-spectral expanders are γ-non-lazy.

Lemma 21.46. *Let (X, Π) be a d-dimensional γ-local-spectral expander, and $0 < i < d$. The laziness of the lower walk on level i is at most:*

$$\max_{\sigma \in X(i)} \left\{ \frac{\langle \mathbb{1}_\sigma, U_{i-1} D_i \mathbb{1}_\sigma \rangle}{\langle \mathbb{1}_\sigma, \mathbb{1}_\sigma \rangle} \right\} \leq \gamma.$$

Proof. Through direct computation, the laziness probability of the lower walk at $\sigma \in X(i)$ is exactly

$$\frac{\langle \mathbb{1}_\sigma, U_{i-1} D_i \mathbb{1}_\sigma \rangle}{\langle \mathbb{1}_\sigma, \mathbb{1}_\sigma \rangle} = \frac{1}{R(i, i-1)} \sum_{\tau \lessdot \sigma} \pi_\tau(\sigma \setminus \tau)$$

It is therefore enough to argue that $\pi_\tau(\sigma \setminus \tau) \leq \gamma$. This follows from the fact that the graph underlying the link X_τ is a γ-spectral expander. In particular, recall that an equivalent formulation of this definition states that:

$$\|A_\tau - UD_\tau\| \leq \gamma,$$

where A_τ is the standard (non-lazy upper) walk and UD_τ is the lower walk on the graph underlying X_τ. This implies that the weight of any vertex v in the graph is at most γ, as:

$$\frac{\langle \mathbb{1}_v, UD_\tau \mathbb{1}_v \rangle}{\langle \mathbb{1}_v, \mathbb{1}_v \rangle} = \frac{\langle \mathbb{1}_v, (UD_\tau - A_\tau) \mathbb{1}_v \rangle}{\langle \mathbb{1}_v, \mathbb{1}_v \rangle} \leq \|A_\tau - UD_\tau\| \leq \gamma$$

where we have used the fact that A_τ is non-lazy by definition. Since $\pi_\tau(\sigma \setminus \tau)$ is exactly the weight of the vertex $\sigma \setminus \tau$ in X_τ, this completes the proof. □

[17]Here the link of τ is not just its top level faces, but the complex given by taking this set, removing τ from each face, and downward closing.

[18]A graph is a γ-spectral expander if its weighted adjacency matrix has no non-trivial eigenvalues greater than γ in absolute value.

Appendix A: Probability Theory Toolbox

A.1 Concentration Inequalities

Lemma A.1 (Chebyshev's inequality)**.** *Let X be a random variable with finite expected value μ and finite non-zero variance σ^2. Then for any real number $k > 0$,*

$$\Pr[|X - \mu| \geq k\sigma] \leq \frac{1}{k^2}.$$

Lemma A.2 (Chernoff Bound [Che52])**.** *Let $X_1, X_2, \cdots, X_n$ be independent random variables. Assume that $0 \leq X_i \leq 1$ always, for each $i \in [n]$. Let $X = X_1 + X_2 + \cdots + X_n$ and $\mu = \mathbb{E}[X] = \sum\limits_{i=1}^{n} \mathbb{E}[X_i]$. Then for any $\epsilon > 0$,*

$$\mathsf{Pr}[X \geq (1+\epsilon)\mu] \leq \exp(-\frac{\epsilon^2}{2+\epsilon}\mu) \text{ and } \mathsf{Pr}[X \leq (1-\epsilon)\mu] \leq \exp(-\frac{\epsilon^2}{2}\mu).$$

Lemma A.3 (Hoeffding bound [Hoe63])**.** *Let $Z_1, \cdots, Z_n$ denote n independent bounded variables in $[a_i, b_i]$. Let $Z = \sum_{i=1}^n Z_i$, then we have*

$$\Pr[|Z - \mathbb{E}[Z]| \geq t] \leq 2\exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

Lemma A.4 (Bernstein inequality [Ber24])**.** *Assume $Z_1, \ldots, Z_n$ are n i.i.d. random variables with $\mathbb{E}[Z_i] = 0$ and $|Z_i| \leq M$ for all $i \in [n]$ almost surely. Let $Z = \sum_{i=1}^n Z_i$. Then, for all $t > 0$,*

$$\Pr[Z > t] \leq \exp\left(-\frac{t^2/2}{\sum_{j=1}^n \mathbb{E}[Z_j^2] + Mt/3}\right) \leq \exp\left(-\min\left\{\frac{t^2}{2\sum_{j=1}^n \mathbb{E}[Z_j^2]}, \frac{t}{2M}\right\}\right)$$

which implies with probability at least $1 - \delta$,

$$Z \leq \sqrt{2\sum_{j=1}^{n} \mathbb{E}[Z_j^2] \log\frac{1}{\delta}} + 2M\log\frac{1}{\delta}.$$

Lemma A.5 (Concentration of subgaussian random variables)**.** *Let $a \in \mathbb{R}^n$ be a vector where each coordinate of a is an independent subgaussian random variable with parameter σ^2. Then, for any vector $x \in \mathbb{R}^n$,*

$$\Pr[|\langle a, x\rangle| \geq t \cdot \|x\|_2] \leq 2\exp\left(-\frac{t^2}{2\sigma^2}\right).$$

Theorem A.6 (Theorem 5.1.1 in [Tro15]). *Let $X_1, \dots, X_m \in \mathbb{R}^{n\times n}$ be m independent random Hermitian matrices. Assume that $0 \preceq X_i \preceq L \cdot I$ for some $L > 0$ and for all $i \in [m]$. Let $X := \sum_{i=1}^m X_i$. Then, for $\epsilon \in (0, 1]$, we have*

$$\Pr[\lambda_{\min}(X) \le \epsilon\lambda_{\min}(\mathbb{E}[X])] \le n \cdot \exp(-(1-\epsilon)^2\lambda_{\min}(\mathbb{E}[X])/(2L)).$$

A.2 Anti-Concentration

Lemma A.7 (Anti-concentration of Gaussian distribution). *Let $Z \sim \mathcal{N}(0, \sigma^2)$. Then, for $t > 0$,*

$$\Pr[|Z| \le t] \le \frac{2t}{\sqrt{2\pi}\sigma}.$$

Lemma A.8 ([LS01, Theorem 3.1] with improved upper bound for Gaussian)). *Let $b > 0$ and $r > 0$. Then,*

$$\exp(-b^2/2) \Pr_{w\sim\mathcal{N}(0,1)}[|w| \le r] \le \Pr_{w\sim\mathcal{N}(0,1)}[|x-b| \le r] \le 2r \cdot \frac{1}{\sqrt{2\pi}} \exp(-(\max\{b-r, 0\})^2/2).$$

Proof. To prove the upper bound, we have

$$\Pr_{w\sim\mathcal{N}(0,1)}[|x-b| \le r] = \int_{b-r}^{b+r} \frac{1}{\sqrt{2\pi}} \exp(-x^2/2)\ dx \le 2r \cdot \frac{1}{\sqrt{2\pi}} \exp(-(\max\{b-r, 0\})^2/2).$$

$\square$

Lemma A.9 (Small ball probability, [RV09]). *Let $h \in \mathbb{R}^n$ be a vector such that $|h_i| \ge \delta$ for all $i \in [n]$. Let $a \in \{-1, 1\}^n$ be a random vector such that each coordinate is an independent Rademacher random variable. Then, for some absolute constants C_1, C_2, we have for any $t > 0$,*

$$\Pr[|\langle h, a\rangle| \le t] \le \min\left\{\frac{C_1 t}{\|h\|_2}, \frac{C_2 t}{\delta\sqrt{n}}\right\}.$$

Appendix B: Quantum States, Unitary Transformations, and Quantum Circuits

We provide a brief overview of quantum computation in this appendix. We recommend the standard textbook [NC11] for a more comprehensive treatment. We divide the computation into three parts: *input*, *process*, and *output*.

B.1 Input

In quantum computing, we represent information in quantum states using qubits. We first introduce pure quantum states (defined as follows). In general, people consider mixed states for quantum information, which we will introduce later.

Definition B.1 (Pure quantum state)**.** A pure quantum state $|\psi\rangle$ on n qubits is represented as a unit vector in $\mathbb{C}^{2^n}$, $|\psi\rangle = (\alpha_0, \alpha_1, \ldots, \alpha_{2^n-1})^\top$, where $\alpha_i \in \mathbb{C}$ for $i \in \{0, \ldots, 2^n - 1\}$ and $\sum_i |\alpha_i|^2 = 1$.

Remark B.1. The *Dirac notation* is widely used in quantum computing, where $|\psi\rangle$ (a column vector) is called a "ket". The complex conjugate of $|\psi\rangle$ is denoted as a row vector $\langle\psi| = (\alpha_0^*, \cdots, \alpha_{2^n-1}^*)$ (called a "bra").

For example, $|0\rangle, |1\rangle, \ldots, |2^n-1\rangle$ represent n-bit classical messages, $0, 1, \ldots, 2^n-1$. For convenience, we sometimes denote $N = 2^n$. In terms of linear algebra, one can think of $|i\rangle$ as the column vector with the $(i+1)$-th entry being 1 and 0 elsewhere. The input to quantum computers can be any quantum state. For classical problems, we can encode the classical input $x \in \{0,1\}^n$ as the quantum state $|x\rangle \in \mathbb{C}^{2^n}$. In general, any pure quantum state $|\psi\rangle$ can be represented as $\sum_i \alpha_i|i\rangle$ for some $\alpha_0, \ldots, \alpha_{2^n-1} \in \mathbb{C}$ with $\sum_i |\alpha_i|^2 = 1$. And α_i's are denoted as the *amplitudes* of $|\psi\rangle$.

The *tensor product* of quantum states is their Kronecker product: if $|\psi\rangle \in \mathbb{C}^{d_1}$

and $|\phi\rangle \in \mathbb{C}^{d_2}$, then we have $|\psi\rangle \otimes |\phi\rangle \in \mathbb{C}^{d_1} \otimes \mathbb{C}^{d_2}$ with

$$|\psi\rangle \otimes |\phi\rangle = (\alpha_0\beta_0, \alpha_0\beta_1, \dots, \alpha_{d_1-1}\beta_{d_2-1})^\top,$$

where α_i's and β_j's are the amplitudes of $|\psi\rangle$ and $|\phi\rangle$, respectively. We often omit the operator $\otimes$ and simply write $|\psi\rangle|\phi\rangle$ or $|\psi, \phi\rangle$ for being concise.

Mixed quantum states.

Definition B.2 (Mixed state)**.** A mixed state is a collection of pure states $|\phi_i\rangle$ for $i \in [n]$, each with associated probability p_i, with the condition $p_i \in [0, 1]$ and $\sum_{i=1}^{n} p_i = 1$.

A mixed state can also be represented by the density matrix:

$$\rho := \sum_{i=1}^{n} p_i |\phi_i\rangle\langle\phi_i|.$$

Partial Trace Notations. For a quantum state σ over two registers R_1, R_2 (i.e. Hilbert spaces $\mathscr{H}_{R_1}, \mathscr{H}_{R_2}$), the partial trace over the R_2 subsystem, denoted Tr_{R_2}, is defined as $\mathrm{Tr}_{R_2}[\rho] := \sum_j (I_{R_1} \otimes \langle j|_{R_2})\rho(I_{R_1} \otimes |j\rangle_{R_2})$, where $\{|j\rangle\}$ is any orthonormal basis for the Hilbert space of subsystem R_2.

We denote the state in R_1 as $\sigma[R_1]$, where $\sigma[R_1] = \mathrm{Tr}_2[\sigma]$ is a partial trace of σ. Similarly, we denote $\sigma[R_2] = \mathrm{Tr}_1[\sigma]$.

Purification of mixed states. For a mixed state ρ over $\mathscr{H}_A$, there exists another space $\mathscr{H}_B$ and a pure state $|\psi\rangle$ over $\mathscr{H}_A \otimes \mathscr{H}_B$ such that ρ is a partial trace of $|\psi\rangle\langle\psi|$ with respect to $\mathscr{H}_B$.

B.2 Quantum Process

Quantum process for quantum states is defined as a unitary transformation.

Definition B.3 (Unitary transformation)**.** A unitary transformation U for an n-qubit quantum state is an isomorphism in the 2^n-dimensional Hilbert space. For convenience, we view U as a $2^n \times 2^n$ matrix satisfying that $U \in \mathbb{C}^{N \times N}$ with $UU^\dagger = U^\dagger U = I$ where $U^\dagger$ is the Hermitian adjoint of U.

We can represent an n-qubit quantum process acting on an n-qubit state $|\psi\rangle$ as $|\psi\rangle \mapsto U|\psi\rangle$ as a unitary matrix U in $\mathbb{C}^{2^n \times 2^n}$. Note that a unitary matrix must preserve the norm of the input state. Thus any unitary transformation is reversible. To implement a unitary transformation, we pick a set of *local unitary operations* that can generate any unitary transformation with arbitrary precision.

Definition B.4 (Universal quantum gate set)**.** A quantum gate set $\mathcal{G}$ is a set of unitaries such that for any unitary transformation U, U can be approximated by a finite sequence of gates in $\mathcal{G}$.

For example, $\{\mathsf{Toffoli}, \mathsf{H}\}$ is a universal gate set [Shi02]. In this chapter, we consider gate sets which only contain unitaries with constant dimensions.

Note that choosing different universal gate sets may cause the circuit complexity of the same object to be different. However, the Solovay-Kitaev theorem shows that one universal gate set can approximate another one at a modest cost.

Theorem B.1 (Solovay-Kitaev Theorem)**.** *Let $\mathcal{G}$ and $\mathcal{G}'$ be two universal gate sets. Then, any s-gate circuit $\mathcal{C}$ using gates from $\mathcal{G}$ can be approximated to precision ϵ by a $s \,\mathsf{poly} \log \frac{s}{\epsilon}$-gates circuit $\mathcal{C}'$ using gates from $\mathcal{G}'$. We say $\mathcal{C}$ approximates to $\mathcal{C}'$ with precision ϵ if*

$$\|\mathcal{C} - \mathcal{C}'\| \leq \epsilon,$$

where $\|\cdot\|$ is L_2 norm.

We will formally state Solovay-Kitaev Theorem when defining the problems of quantum circuit complexity.

We can represent quantum algorithms as quantum circuits by using a sequence of quantum gates from a universal quantum gate set.

Definition B.5 (Quantum circuit $\mathsf{QC}(s,t,\mathcal{G})$)**.** Let $s,t:\mathbb{N}\to\mathbb{N}$ and $\mathcal{G}$ be a universal quantum gate set. A quantum circuit family $\{\mathcal{C}_n : n>0\}$ is in $\mathsf{QC}(s,t,\mathcal{G})$ if the following holds: For all $n>0$,

- the input to $\mathcal{C}_n$ is an n-qubit quantum state $|\psi\rangle$;
- $\mathcal{C}_n$ extends the input layer with $t(n)$ ancilla qubits, where these ancilla qubits are initiated to $|0^{t(n)}\rangle$;
- $\mathcal{C}_n$ applies $s(n)$ gates from $\mathcal{G}$ on the initial state $|\psi\rangle|0^{t(n)}\rangle$.

Here, in addition to the qubits for the input, the circuit can also have ancilla qubits as its working space. We say that a quantum algorithm is efficient if its corresponding circuit has circuit size at most polynomial in the input size. In the rest of the chapter, we may write $\mathsf{QC}(s,t,\mathcal{G})$ as $\mathsf{QC}(s)$ if the number of ancilla qubits is at most $O(s)$.

B.3 Output

The outputs of quantum circuits defined in Definition B.5 are quantum states. To extract useful information from a quantum state $|\psi\rangle$, one can *measure* the state. Mathematically, a measurement is simply a sampling process. For example, if we measure $|\psi\rangle$ in the *computational basis*, i.e., $\{|0\rangle\langle 0|,\dots,|2^n-1\rangle\langle 2^n-1|\}$, we get the output being index i with probability $|c_i|^2$. In general, we can measure a state $|\psi\rangle$ on any orthogonal basis $\mathcal{B}$ for $\mathbb{C}^{2^n}$. Mathematically, this is equivalent to a change of basis via a *unitary transformation*.

Therefore, a quantum algorithm for some Boolean function is as follows: Given $|x\rangle$, apply a quantum circuit $\mathcal{C}$ on state $|x,0^{t(n)}\rangle$, and then measure the state $\mathcal{C}|x,0^{t(n)}\rangle$ in the computational basis. If $\mathcal{C}$ computes f, then the measurement outcome will be $f(x)$ with probability good enough (e.g., $\geq 2/3$). Note that a quantum process can have measurements in the middle of the computation in general. In this case, the process is not reversible any more. However, we can always defer these measurements until all the unitaries have been applied by adding ancilla qubits.

Remark B.2 (Deferring measurements)*.* Let $\mathcal{M}_i$ be the computational-basis measurement on the i-th qubit. Let $|\psi\rangle$ be any n-qubit state and U, V be any n-qubit unitaries. Then, the process $U \circ \mathcal{M}_i \circ V$ operating on $|\psi\rangle$ is equivalent to $\mathcal{M}_{n+1} \circ U \circ \mathsf{CNOT}_{i,n+1} \circ V$, where $\mathsf{CNOT}_{i,n+1}$ has the i-th qubit as the control qubit and the $n+1$-th qubit as the target qubit.

More generally, we can do the following general form of measurement called POVM:

Definition B.6 (Positive operator-valued measure, POVM)**.** A positive operator-valued measure (POVM) $\mathcal{M}$ is specified by a finite index set $\mathcal{I}$ and a set $\{M_i\}_{i \in \mathcal{I}}$ of Hermitian positive semi-definite matrices M_i such that $\sum_{i \in \mathcal{I}} M_i = \mathbf{I}$.

When applying $\mathcal{M}$ to a quantum state ρ, the outcome is i with probability $p_i = \mathrm{Tr}[\rho P_i]$ for all $i \in \mathcal{I}$.

To characterize the post-measurement states, we define the quantum measurements as follows.

Definition B.7 (Quantum measurement)**.** A quantum measurement $\mathcal{E}$ is specified by a finite index set $\mathcal{I}$ and a set $\{E_i\}_{i \in \mathcal{I}}$ of measurement operators E_i such that $\sum_{i \in \mathcal{I}} E_i^\dagger E_i = \mathbf{I}$.

When applying $\mathcal{E}$ to a quantum state ρ, the outcome is i with probability $p_i = \mathrm{Tr}[\rho E_i^\dagger E_i]$ for all $i \in \mathcal{I}$. Furthermore, conditioned on the outcome being i, the post-measurement state is $E_i \rho E_i^\dagger / p_i$.

Note that POVM $\mathcal{M}$ and quantum measurement $\mathcal{E}$ are related by setting $M_i = E_i^\dagger E_i$. In this case, we say that $\mathcal{E}$ is an implementation of $\mathcal{M}$. The implementation of a POVM may not be unique.

Definition B.8 (Projective measurement and projective POVM)**.** A quantum measurement $\mathcal{E}$ is projective if for all $i \in \mathcal{I}$, E_i is a projection, i.e., E_i is Hermitian and $E_i^2 = E_i$.

Similarly, a POVM $\mathcal{M}$ is projective if each M_i is projection for $i \in \mathcal{I}$.

Appendix C: Omitted Materials from Chapter 5

C.1 A Review of Lin-Tong's Algorithm

In this section, we review the techniques in [LT22], which proposed a hybrid quantum/classical algorithm for estimating the ground state energy of a Hamiltonian. Compared with the algorithms in previous works, the algorithm in [LT22] uses fewer quantum resources and does not need to access the block-encoding of the Hamiltonian.

First of all, they assumed that the given initial state $|\phi_0\rangle$[1] has a nontrivial overlap with the ground state of H.

C.1.1 Quantum part of the algorithm

Fix $j \in \mathbb{Z}$. Suppose we want to estimate $\Re(\langle\phi_0| e^{-ij\tau H} |\phi_0\rangle)$. Then, we set $\mathrm{W} = I$ and define a random variable X_j as follows:

$$X_j := \begin{cases} 1 & \text{if the outcome is } 0 \\ -1 & \text{if the outcome is } 1 \end{cases}.$$

Since the state before the measurement is

$$\frac{1}{2}(|0\rangle \otimes (I + e^{-ij\tau H}) |\phi_0\rangle + |1\rangle \otimes (I - e^{-ij\tau H}) |\phi_0\rangle), \tag{C.1}$$

we have

$$\begin{aligned} \mathbb{E}[X_j] = {} & \Pr[X_j = 0] - \Pr[X_j = 1] \\ = {} & \frac{1}{4} \langle\phi_0| (I + e^{ij\tau H})(I + e^{-ij\tau H}) |\phi_0\rangle - \frac{1}{4} \langle\phi_0| (I - e^{ij\tau H})(I - e^{-ij\tau H}) |\phi_0\rangle \\ = {} & \frac{1}{2} \langle\phi_0| (e^{ij\tau H} + e^{-ij\tau H}) |\phi_0\rangle \\ = {} & \Re(\langle\phi_0| e^{-ij\tau H} |\phi_0\rangle). \end{aligned} \tag{C.2}$$

[1]In [LT22], they allowed the initial state to be a mixed state. For simplicity, we still denote it as $|\phi_0\rangle$.

For the imaginary part $\Im(\langle\phi_0| e^{-ij\tau H} |\phi_0\rangle)$, we can set W to be the phase gate $\begin{bmatrix}1 & 0\\ 0 & -i\end{bmatrix}$ and define the random variable Y_j similarly. Then, we have

$$\mathbb{E}[Y_j] = \Im(\langle\phi_0| e^{-ij\tau H} |\phi_0\rangle). \tag{C.3}$$

Therefore, Eqs. (C.2) and (C.3) implies the following claim:

Claim C.1 (Estimator of the Hamiltonian expectation)**.** *For any $j \in \mathbb{Z}$, the random variable $X_j + iY_j$ is an un-biased estimator for $\langle\phi_0| e^{-ij\tau H} |\phi_0\rangle$.*

C.1.2 Classical part of the algorithm

Let τ be a normalization factor such that $\|\tau H\| \leq \pi/3$. Suppose the initial state $|\phi_0\rangle$ can be decomposed in the eigenspace of H as $|\phi_0\rangle = \sum_k \sqrt{p_k} |\psi_k\rangle$. Let $p(x)$ be the following density function (spectral measure):

$$p(x) := \sum_k p_k \delta(x - \tau\lambda_k) \quad \forall x \in [-\pi, \pi]. \tag{C.4}$$

That is, $p(x)$ is the distribution of the state energy with respect to τH after we measure $|\phi_0\rangle$ in the eigenbasis of H.

Define the 2π-periodic Heaviside function by

$$H(x) = \begin{cases} 1 & x \in [2k\pi, (2k+1)\pi) \\ 0 & x \in [(2k-1)\pi, 2k\pi) \end{cases} \quad \forall k \in \mathbb{Z}. \tag{C.5}$$

Then, we define the 2π-periodic CDF of p as the convolution of H and p:

$$C(x) := (H * p)(x). \tag{C.6}$$

For any $x \in [-\pi/3, \pi/3]$, for any $w \in \mathbb{Z}$, we have

$$\begin{aligned} C(x+2w\pi) &= \int_{-\pi}^{\pi} H(x+2w\pi-t)p(t)dt \qquad (C.7) \\ &= \sum_k p_k \cdot \int_{-\pi}^{\pi} H(x+2w\pi-t)\delta(t-\tau\lambda_k)dt \\ &= \sum_k p_k \cdot H(x+2w\pi-\tau\lambda_k) \\ &= \sum_k p_k \cdot \mathbf{1}_{x\geq\tau\lambda_k} \\ &= \sum_{k:\lambda_k\leq x} p_k, \qquad (C.8) \end{aligned}$$

where the first step follows from the definition of convolution, the second step follows from Dirac delta function's property, and the third step follows from H has period 2π. We note that $C(x)$ is right continuous and non-decreasing in $[-\pi/3, \pi/3]$.

However, we cannot directly evaluate $C(x)$, but we can approximate it! Define the approximate CDF (ACDF) as

$$\widetilde{C}(x) := (F * p)(x), \qquad (C.9)$$

where $F(x) = \sum_{|j|\leq d} \hat{F}_j e^{ijx}$ is a low Fourier-degree approximation of the Heaviside function $H(x)$ such that

$$|F(x) - H(x)| \leq \epsilon \quad \forall x \in [-\pi+\delta, -\delta] \cup [\delta, \pi-\delta]. \qquad (C.10)$$

The construction of F is given by Lemma C.6. Furthermore, the approximation error of $\widetilde{C}(x)$ is bounded by

$$C(x-\delta) - \epsilon \leq \widetilde{C}(x) \leq C(x+\delta) + \epsilon, \qquad (C.11)$$

for any $x \in [-\pi/3, \pi/3]$, $\delta \in (0, \pi/6)$ and $\epsilon > 0$.

C.1.2.1 Estimating the ACDF

The goal of this section is to prove Lemma C.2, which constructs an estimator for $\widetilde{C}(x)$ (defined by Eq. (C.9)).

Lemma C.2 (Estimating the ACDF)**.** *For any $\sigma > 0$, for any $x \in [-\pi, \pi]$, there exists an un-biased estimator $\overline{G}(x)$ for the ACDF $\widetilde{C}(x)$ with variance at most σ^2.*

Furthermore, $\overline{G}(x)$ runs the quantum circuit (Figure 4.3) $O(\frac{\log^2 d}{\sigma^2})$ times with expected total evolution time $O(\frac{\tau d \log d}{\sigma^2})$.

Proof. $\widetilde{C}(x)$ can be expanded in the following way:

$$\begin{aligned}
\widetilde{C}(x) &= (F * p)(x) && \text{(C.12)}\\
&= \int_{-\pi}^{\pi} F(x-y)p(y)dy \\
&= \sum_{|j|\le d} \int_{-\pi}^{\pi} \hat{F}_j e^{ij(x-y)} p(y)dy \\
&= \sum_{|j|\le d} \hat{F}_j e^{ijx} \int_{-\pi}^{\pi} p(y) e^{-ijy} dy \\
&= \sum_{|j|\le d} \hat{F}_j e^{ijx} \sum_k p_k e^{-ij\tau\lambda_k} \\
&= \sum_{|j|\le d} \hat{F}_j e^{ijx} \cdot \langle \phi_0 | e^{-ij\tau H} | \phi_0 \rangle , && \text{(C.13)}
\end{aligned}$$

where the third step follows from the Fourier expansion of $F(x-y)$, the fifth step follows from the property of Dirac's delta function, and the last step follows from the definition of p_k and the eigenvalues of matrix exponential.

To estimate $\langle \phi_0 | e^{-ij\tau H} | \phi_0 \rangle$, we use the multi-level Monte Carlo method. Define a random variable J with support $\{-d, \cdots, d\}$ such that

$$\Pr[J = j] = \left| \hat{F}_j \right| / \mathcal{F}, \tag{C.14}$$

where $\mathcal{F} := \sum_{|j|\le d} |\hat{F}_j|$. Then, let $Z := X_J + iY_J \in \{\pm 1 \pm i\}$. Define an estimator $G(x; J, Z)$ as follows:

$$G(x; J, Z) := \mathcal{F} \cdot Z e^{i(\theta_J + Jx)},$$

where θ_j is defined by $\hat{F}_j = |\hat{F}_j|e^{i\theta_j}$. Then, we show that $G(x; J, Z)$ is un-biased:

$$\begin{aligned}\mathbb{E}[G(x; J, Z)] =& \sum_{|j|\le d} \mathbb{E}\left[(X_j + iY_j)e^{i(\theta_j + jx)}|\hat{F}_j|\right]\\ =& \sum_{|j|\le d} \hat{F}_j e^{ijx} \cdot \mathbb{E}\left[X_j + iY_j\right]\\ =& \sum_{|j|\le d} \hat{F}_j e^{ijx} \cdot \langle\phi_0|\, e^{-ij\tau H}\, |\phi_0\rangle\\ =& \widetilde{C}(x),\end{aligned}$$

where the third step follows from Claim C.1. Moreover, the variance of G can be upper-bounded by:

$$\begin{aligned}\mathrm{Var}[G(x; J, Z)] =& \mathbb{E}[|G(x; J, Z)|^2] - |\,\mathbb{E}[G(x; J, Z)]|^2\\ \le& \mathbb{E}[|G(x; J, Z)|^2]\\ =& \mathcal{F}^2 \cdot \mathbb{E}[|X_J + iY_J|^2]\\ =& 2\mathcal{F}^2,\end{aligned}$$

where the third step follows from $|e^{i(\theta_J + Jx)}| = 1$, and the last step follows from $X_j, Y_j \in \{\pm 1\}$.

Hence, we can take $N_s := \frac{2\mathcal{F}^2}{\sigma^2}$ independent samples of (J, Z), denoted by $\{(J_k, Z_k)\}_{k\in[N_s]}$ and compute

$$\overline{G}(x) := \frac{1}{N_s}\sum_{k=1}^{N_s} G(x; J_k, Z_k).$$

Then, we have

$$\mathbb{E}[\overline{G}(x)] = \widetilde{C}(x), \quad \text{and} \quad \mathrm{Var}[\overline{G}(x)] \le \sigma^2.$$

The expected total evolution time is

$$\mathcal{T}_{\mathsf{tot}} := N_s\tau\,\mathbb{E}[|J|] = \frac{2\mathcal{F}^2}{\sigma^2}\tau \sum_{|j|\le d} |j| \cdot \frac{|\hat{F}_j|}{\mathcal{F}} = \frac{2\mathcal{F}\tau}{\sigma^2}\sum_{|j|\le d} |j||\hat{F}_j|.$$

By Lemma C.6, we know that $|\hat{F}_j| = O(1/|j|)$. Hence, we have $\mathcal{F} = \sum_{|j|\le d} O(1/|j|) = O(\log d)$. Thus, the number of samples is

$$N_s = O\left(\frac{\log^2 d}{\sigma^2}\right).$$

And the expected total evolution time is

$$\mathcal{T}_{\mathsf{tot}} = O\left(\frac{\tau d \log d}{\sigma^2}\right).$$

The lemma is then proved. $\square$

C.1.2.2 Inverting the CDF

We first define the CDF inversion problem:

Definition C.1 (The CDF inversion problem)**.** For $0 < \delta < \pi/6$, $0 < \eta < 1$, find $x^\star \in (-\pi/3, \pi/3)$ such that

$$C(x^\star + \delta) > \eta/2, \quad C(x^\star - \delta) < \eta.$$

Remark C.1. The condition in Definition C.1 is weaker than $\eta/2 < C(x) < \eta$ due to the discontinuity of $C(x)$. For any CDF $C(x)$, such an $x^\star$ must exist: let $a := \sup\ \{x \in (-\pi/3, \pi/3) : C(x) \le \eta/2\}$ and $b := \inf\ \{x \in (-\pi/3, \pi/3) : C(x) \ge \eta\}$. Since $C(x)$ is non-decreasing, we have $a \le b$. And any $x \in (a - \delta, b + \delta)$ satisfies the condition in Definition C.1.

Then, we give an algorithm that solves the CDF inversion problem.

Lemma C.3 (Inverting the CDF, Theorem 2 in [LT22])**.** *There exists an algorithm that solves the CDF inversion problem (Definition C.1) with probability at least $1-\nu$ such that:*

1. *the number of independent samples of (J, Z) is*

$$O\left(\eta^{-2} \cdot (\log(\nu^{-1}) + \log\log(\delta^{-1})) \cdot (\log(\delta^{-1}) + \log\log(\delta^{-1}\eta^{-1}))^2\right)$$

Algorithm 118 Inverting the CDF

```
1: procedure InvertCDF($\eta, \delta, \{J_k, Z_k\}$)
2:     $x_L \leftarrow -\pi/3$, $X_R \leftarrow \pi/3$
3:     while $x_R - x_L > 2\delta$ do
4:         $x_M \leftarrow (x_L + x_R)/2$
5:         $u \leftarrow$ Certify($x_M, (2/3)\delta, \eta, \{J_k, Z_k\}$)
6:         if $u = 0$ then
7:             $x_R \leftarrow x_M + (2/3)\delta$
8:         else
9:             $x_L \leftarrow x_M - (2/3)\delta$
10:        end if
11:    end while
12:    return $(x_L + x_R)/2$
13: end procedure
```

2. *the expected total evolution time is*

$$O\left(\tau\eta^{-2} \cdot \delta^{-1}\log(\delta^{-1}\eta^{-1}) \cdot (\log(\delta^{-1}) + \log\log(\delta^{-1}\eta^{-1})) \cdot (\log(\nu^{-1}) + \log\log(\delta^{-1}))\right)$$

3. *the maximal evolution time is*

$$O\left(\tau\delta^{-1}\log(\delta^{-1}\eta^{-1})\right)$$

4. *the classical running time is*

$$O\left(\eta^{-2}\log(\delta^{-1}) \cdot (\log(\nu^{-1}) + \log\log(\delta^{-1})) \cdot (\log(\delta^{-1}) + \log\log(\delta^{-1}\eta^{-1}))^2\right).$$

Proof. For any $x \in [-\pi/3, \pi/3]$, at least one of the following conditions will hold:

$$C(x+\delta) > \eta/2, \quad \text{or} \quad C(x-\delta) < \eta. \tag{C.15}$$

Suppose we have a sub-routine Certify$(x, \delta, \eta, \{J_k, Z_k\})$ such that if $C(x+\delta) > \eta/2$, it returns 0; otherwise, it returns 1.

Then, we can solve the CDF inversion problem via the binary search (Algorithm 118).

In Line 3, x_L and x_R always satisfy the following conditions:

$$C(x_L) < \eta, \quad C(x_R) > \eta/2,$$

which is guaranteed by $\textsc{Certify}(x_M, (2/3)\delta, \eta, \{J_k, Z_k\})$. Then, when the while-loop ends, we have $x_R - x_L \le 2\delta$. Let $x^\star := (x_L + x_R)/2$ be the output of Algorithm 118. Then, we get that

$$\begin{aligned} C(x^\star + \delta) &\ge C(x_R) > \eta/2, \\ C(x^\star - \delta) &\le C(x_L) < \eta. \end{aligned}$$

And it is easy to see that Algorithm 118 will call Certify $L := O(\log(1/\delta))$ times. Then, by Lemma C.4 and union bound, Algorithm 118 will be correct with probability at least $1 - \nu$. We note that different runs of Certify can share a same set of samples $\{J_k, Z_k\}$, which does not affect the union bound. Hence, the number of samples and the total evolution time follows directly from Lemma C.4 and $d = O(\delta^{-1}\log(\delta^{-1}\eta^{-1}))$. $\square$

Lemma C.4 (Certify sub-routine)**.** *For any $\nu > 0$, there exists an algorithm that distinguishes the two cases in Eq.* (C.15) *for any $x \in [-\pi/3, \pi/3]$ with probability at least $1 - O(\nu/L)$ using*

$$O\left(\eta^{-2}\log^2(d)(\log(1/\nu) + \log\log(1/\delta))\right)$$

independent samples of (J, Z), and total evolution time

$$O\left(\eta^{-2}\tau d\log(d)(\log(1/\nu) + \log\log(1/\delta))\right)$$

in expectation.

Proof. To decide which one of the conditions holds for x, we can estimate the ACDF $\widetilde{C}(x)$. If we take $\epsilon = \eta/8$ in Lemma C.6, then the constructed ACDF satisfies

$$C(x - \delta) - \eta/8 \le \widetilde{C}(x) \le C(x + \delta) + \eta/8.$$

Algorithm 119 Distinguish the two cases in Eq. (C.15)

```
1: procedure CERTIFY($x, \eta, \delta, \{J_k, Z_k\}$)
2:     $c \leftarrow 0$, $N_b \leftarrow \Omega(\log(1/\nu) + \log\log(1/\delta))$
3:     for $1 \le r \le N_b$ do
4:         Compute $\overline{G}(x)$ using $\{J_k, Z_k\}_{k\in[(r-1)N_s+1, rN_s]}$    ▷ Lemma C.2
5:         if $\overline{G}(x) \ge (3/4)\eta$ then
6:             $c \leftarrow c + 1$
7:         end if
8:     end for
9:     return $\mathbf{1}_{c \le N_b/2}$
10: end procedure
```

Thus,

$$\begin{aligned}\widetilde{C}(x) > (5/8)\eta \quad &\Rightarrow \quad C(x+\delta) > \eta/2,\\ \widetilde{C}(x) < (7/8)\eta \quad &\Rightarrow \quad C(x-\delta) < \eta.\end{aligned}$$

Then, we can distinguish $\widetilde{C}(x) > (5/8)\eta$ or $\widetilde{C}(x) < (7/8)\eta$ by the estimator in Lemma C.2.

In Algorithm 119, we compute the estimator $\overline{G}(x)$ N_b times independently, where each time we use N_s samples of (J, Z). We note that an error occurs when $\widetilde{C}(x) > (7/8)\eta$ but $\overline{G}(x) < (3/4)\eta$, or $\widetilde{C}(x) < (5/8)\eta$ but $\overline{G}(x) > (3/4)\eta$ (when $(5/8)\eta \le \widetilde{C}(x) \le (7/8)\eta$, any output is correct). By Chebyshev's inequality, we have

$$\begin{aligned}\Pr[\overline{G}(x) \text{ has an error}] \le\ & \Pr\left[\overline{G}(x) < \frac{3}{4}\eta \;\Big|\; \widetilde{C}(x) > \frac{7}{8}\eta\right] + \Pr\left[\overline{G}(x) > \frac{3}{4}\eta \;\Big|\; \widetilde{C}(x) < \frac{5}{8}\eta\right]\\ \le\ & 2 \cdot \frac{\sigma^2}{\eta^2/64}\\ \le\ & \frac{1}{4},\end{aligned}$$

if we take $\sigma^2 = O(\eta^2)$ in Lemma C.2.

Then, by the Chernoff bound, we have

$$\Pr[\text{CERTIFY makes an error}] \le \exp(-\Omega(N_b)) \le \nu/L,$$

if we take $N_b := \Omega(\log(L/\nu)) = \Omega(\log(1/\nu) + \log\log(1/\delta))$. Thus, the total number of samples is

$$N_b N_s = O\left(\eta^{-2}\log^2(d)(\log(1/\nu) + \log\log(1/\delta))\right),$$

and the expected total evolution time is

$$O\left(\eta^{-2}\tau d\log(d)(\log(1/\nu) + \log\log(1/\delta))\right),$$

which complete the proof of the lemma. □

C.1.2.3 Estimating the ground state energy

Corollary C.5 (Ground state energy estimation, Corollary 3 in [LT22])**.** *If $p_0 \geq \eta$ for some known η, then with probability at least $1-\nu$, the ground state energy λ_0 can be estimated within additive error ϵ, such that:*

1. *the number of times running the quantum circuit (Figure 4.3) is $\widetilde{O}(\eta^{-2})$.*
2. *the expected total evolution time is $\widetilde{O}(\epsilon^{-1}\eta^{-2})$.*
3. *the maximal evolution time is $\widetilde{O}(\epsilon^{-1})$.*
4. *the classical running time is $\widetilde{O}(\eta^{-2})$.*

Proof. Suppose we can solve the CDF inversion problem (Definition C.1) for $\delta = \tau\epsilon$ and η, i.e., we find an $x^\star$ such that

$$C(x^\star + \tau\epsilon) > \eta/2 > 0, \quad C(x^\star - \tau\epsilon) < \eta \leq p_0.$$

Since $C(x)$ cannot take value between 0 and p_0, we have

$$x^\star + \tau\epsilon \geq \tau\lambda_0, \quad x^\star - \tau\epsilon < \tau\lambda_0,$$

which is

$$|x^\star/\tau - \lambda_0| \leq \epsilon.$$

The costs of this algorithm follows from Lemma C.3. □

C.1.3 Low Fourier degree approximation of the Heaviside function

We construct the low degree approximation of the Heaviside function in this section.[2]

Lemma C.6 (Constructing low degree approximation of H)**.** *Let $H(x)$ be the 2π-period Heaviside function (Eq. (C.5)). For any $\delta \in (0, \pi/2)$ such that $\tan(\delta/2) \le 1 - 1/\sqrt{2}$, there exists a $d = O(\delta^{-1}\log(\delta^{-1}\epsilon^{-1}))$ and a 2π-period function $F_{d,\delta}(x)$ of the form:*

$$F_{d,\delta}(x) = \frac{1}{\sqrt{2\pi}} \sum_{j=-d}^{d} \widehat{F}_{d,\delta,j} \cdot e^{ijx} \tag{C.16}$$

such that

1. *$F_{d,\delta}(x) \in [0, 1]$ for all $x \in \mathbb{R}$.*
2. *$|F_{d,\delta}(x) - H(x)| \le \epsilon$ for $x \in [-\pi + \delta, -\delta] \cup [\delta, \pi - \delta]$.*
3. *$|\widehat{F}_{d,\delta,j}| = \Theta(1/|j|)$ for $j \neq 0$.*

Proof. We first construct $F'_{d,\delta}(x)$ by mollifying the Heaviside function with $M_{d,\delta}(x)$ in Lemma C.8:

$$F'_{d,\delta}(x) := (M_{d,\delta} * H)(x) = \int_{-\pi}^{\pi} M_{d,\delta}(y) H(x - y) dy. \tag{C.17}$$

We can verify that $F'_{d,\delta}$ has Fourier degree at most d. It follows from the Chebyshev polynomial $T_d(x)$ is of degree d. Hence, the Fourier coefficients of $M_{d,\delta}(x)$:

$$\widehat{M}_{d,\delta,j} = \frac{1}{\sqrt{2\pi}} \int_{-\pi}^{\pi} M_{d,\delta}(x) e^{-ijx} dx \neq 0 \tag{C.18}$$

only if $j \in \{-d, \dots, d\}$. Since $F_{d,\delta}$ is a convolution of $M_{d,\delta}$ and H, we have

$$\widehat{F'}_{d,\delta,j} = \sqrt{2\pi} \widehat{M}_{d,\delta,j} \widehat{H}_j \quad \forall |j| \le d. \tag{C.19}$$

[2]The construction in [LT22] is not enough to prove Lemma C.7 because the range of $F_{d,\delta}$ is $[-\epsilon/2, 1 + \epsilon]$ while Lemma C.7 requires the range to be $[0, 1]$. We fix this issue in Lemma C.6.

Then, we define

$$F_{d,\delta}(x) := \frac{1}{\sqrt{2\pi}} \sum_{j=-d}^{d} \widehat{F}_{d,\delta,j} \cdot e^{ijx}, \tag{C.20}$$

where

$$\widehat{F}_{d,\delta,j} = \begin{cases} \frac{1}{1+(5/4)\epsilon}\left(\widehat{F'}_{d,\delta,j} + \sqrt{2\pi}\epsilon/4\right) & \text{if } j = 0, \\ \frac{1}{1+(5/4)\epsilon}\widehat{F'}_{d,\delta,j} & \text{otherwise.} \end{cases} \tag{C.21}$$

It is easy see that

$$F_{d,\delta}(x) = \frac{F'_{d,\delta}(x) + \epsilon/4}{1 + (5/4)\epsilon} \quad \forall x \in \mathbb{R}. \tag{C.22}$$

Then, we will show that taking $d = O(\delta^{-1}\log(\delta^{-1}\epsilon^{-1}))$ is enough to satisfy (1)-(3).

Part (1): We first compute the range of $F'_{d,\delta}(x)$:

$$F'_{d,\delta}(x) \le \int_{-\pi}^{\pi} |M_{d,\delta}(y)| dy \le 1 + \frac{4\pi}{\mathcal{N}_{d,\delta}}, \tag{C.23}$$

where the second step follows from (2) in Lemma C.8. On the other hand,

$$F'_{d,\delta}(x) \ge -\frac{1}{\mathcal{N}_{d,\delta}} \int_{-\pi}^{\pi} H(y) dy = \frac{-\pi}{\mathcal{N}_{d,\delta}}.$$

Hence, if we take $d = O(\delta^{-1}\log(\delta^{-1}\epsilon^{-1}))$ such that

$$\mathcal{N}_{d,\delta} \ge C_1 e^{d\delta/\sqrt{2}} \sqrt{\frac{\delta}{d}} \cdot \mathrm{erf}(C_2\sqrt{d}\delta) \ge \frac{4\pi}{\epsilon} \tag{C.24}$$

holds, we will have

$$-\epsilon/4 \le F'_{d,\delta} \le 1 + \epsilon. \tag{C.25}$$

Therefore, for all $x \in \mathbb{R}$,

$$F_{d,\delta}(x) = \frac{F'_{d,\delta}(x) + \epsilon/4}{1 + (5/4)\epsilon} \in [0, 1]. \tag{C.26}$$

Part (2): The approximation error of $F'_{d,\delta}$ is

$$\begin{aligned}|F'_{d,\delta}(x) - H(x)| &\le \left| \int_{-\pi}^{\pi} M_{d,\delta}(y)(H(x-y) - H(x))dy \right| \\ &\le \int_{-\pi}^{\pi} |M_{d,\delta}(y)||H(x-y) - H(x)|dy, \end{aligned} \tag{C.27}$$

where the first step follows from (2) in Lemma C.8, and the second step follows from the triangle inequality.

Fix $x \in [-\pi + \delta, -\delta] \cup [\delta, \pi - \delta]$. If $y \in (-\delta, \delta)$, then $H(x - y) = H(x)$ and

$$\int_{-\delta}^{\delta} |M_{d,\delta}(y)||H(x-y) - H(x)|dy = 0. \tag{C.28}$$

If $|y| \ge \delta$, by (1) in Lemma C.8, we have $|M_{d,\delta}| \le \frac{1}{\mathcal{N}_{d,\delta}}$. Since $|H(x-y) - H(x)| \le 1$, we have

$$\left(\int_{-\pi}^{-\delta} + \int_{\delta}^{\pi} \right) |M_{d,\delta}(y)||H(x-y) - H(x)|dy \le \frac{2\pi}{\mathcal{N}_{d,\delta}} \le \epsilon/2, \tag{C.29}$$

where the last step follows from Eq. (C.24). Therefore,

$$|F'_{d,\delta}(x) - H(x)| \le \epsilon/2 \quad \forall |x| \in [\delta, \pi - \delta]. \tag{C.30}$$

Thus,

$$\begin{aligned}|F_{d,\delta}(x) - H(x)| &= \left| \frac{F'_{d,\delta}(x) + \epsilon/4}{1 + (5/4)\epsilon} - H(x) \right| && \text{(C.31)} \\ &\le |F'_{d,\delta}(x) - H(x)| + \frac{(5/4)\epsilon}{1 + (5/4)\epsilon} |F'_{d,\delta}(x)| + \frac{\epsilon/4}{1 + (5/4)\epsilon} \\ &\le \epsilon/2 + \frac{(5/4)\epsilon}{1 + (5/4)\epsilon}(1 + \epsilon) + \frac{\epsilon/4}{1 + (5/4)\epsilon} \\ &\le 2\epsilon, && \text{(C.32)}\end{aligned}$$

where the second step follows from the triangle inequality, the third step follows from Eq. (C.25). By scaling for ϵ, we can make the approximation error at most ϵ.

Part (3): Since $|\widehat{F'}_{d,\delta,j}| = \sqrt{2\pi}|\widehat{M}_{d,\delta,j}||\widehat{H}_j|$, we first bound $|\widehat{M}_{d,\delta,j}|$:

$$\left|\widehat{M}_{d,\delta,j}\right| \leq \frac{1}{\sqrt{2\pi}}\int_{-\pi}^{\pi}|M_{d,\delta}(x)|dx \leq \frac{1}{\sqrt{2\pi}}\left(1+\frac{4\pi}{\mathcal{N}_{d,\delta}}\right) \leq \frac{1+\epsilon}{\sqrt{2\pi}}, \tag{C.33}$$

where the second step follows from (2) in Lemma C.8 and the last step follows from Eq. (C.24).

For $|\widehat{H}_j|$, if $j \neq 0$, we have

$$\widehat{H}_j = \frac{1}{\sqrt{2\pi}}\int_{-\pi}^{\pi}H(x)e^{-ijx}dx = \frac{1}{\sqrt{2\pi}}\int_{0}^{\pi}e^{-ijx}dx = \begin{cases}\frac{\sqrt{2}}{i\sqrt{\pi}j} & \text{if } j \text{ is odd,}\\ 0 & \text{if } j \text{ is even.}\end{cases} \tag{C.34}$$

Hence, for $j \neq 0$,

$$|\widehat{F'}_{d,\delta,j}| \leq \sqrt{2\pi}\cdot\frac{1+\epsilon}{\sqrt{2\pi}}\cdot\sqrt{\frac{2}{\pi}}\frac{1}{|j|} = \frac{1+\epsilon}{\sqrt{\pi/2}|j|}. \tag{C.35}$$

Then, by definition, we get that

$$|\widehat{F}_{d,\delta,j}| \leq \frac{1+\epsilon}{\sqrt{\pi/2}(1+(5/4)\epsilon)|j|} = \Theta(1/|j|). \tag{C.36}$$

The proof of the lemma is completed. $\square$

The following lemma shows the approximation ratio of the ACDF $\widetilde{C}(x)$ constructed from the low degree approximated Heaviside function $F(x)$ by Lemma C.6.

Lemma C.7 (Approximation ratio of the ACDF)**.** *For any $\epsilon > 0$, $0 < \delta < \pi/6$, let $F(x) := F_{d,\delta}(x)$ constructed by Lemma C.6. Then, for any $x \in [-\pi/3, \pi/3]$, the ACDF $\widetilde{C}(x) = (F * p)(x)$ satisfies:*

$$C(x-\delta) - \epsilon \leq \widetilde{C}(x) \leq C(x+\delta) + \epsilon.$$

Proof. By (2) in Lemma C.6, we have

$$|F(x) - H(x)| \leq \epsilon \quad \forall x \in [-\pi+\delta, -\delta] \cup [\delta, \pi-\delta]. \tag{C.37}$$

Define $F_L := F(x-\delta)$ such that

$$|F_L(x) - H(x)| \leq \epsilon \quad \forall x \in [-\pi + 2\delta, 0] \cup [2\delta, \pi]. \tag{C.38}$$

For $\widetilde{C}_L(x) := (F_L * p)(x)$, we have $\widetilde{C}_L(x) = \widetilde{C}(x-\delta)$, and for $x \in [-\pi/3, \pi/3]$,

$$\begin{aligned}
|C(x) - \widetilde{C}_L(x)| &= \left| \int_{-\pi}^{\pi} p(x-y)(H(y) - F_L(y)) dy \right| \qquad (\text{C.39}) \\
&\leq \int_{-\pi}^{\pi} p(x-y)|H(y) - F_L(y)| dy \\
&= \left(\int_{-\pi}^{0} + \int_{2\delta}^{\pi} \right) p(x-y)|H(y) - F_L(y)| dy + \int_{0}^{2\delta} p(x-y)|H(y) - F_L(y)| dy \\
&\leq \epsilon \cdot \left(\int_{-\pi}^{0} + \int_{2\delta}^{\pi} \right) p(x-y) dy + \int_{0}^{2\delta} p(x-y)|H(y) - F_L(y)| dy \\
&\leq \epsilon + \int_{0}^{2\delta} p(x-y)|H(y) - F_L(y)| dy \\
&\leq \epsilon + \int_{0}^{2\delta} p(x-y) dy \\
&= \epsilon + \int_{x-2\delta}^{x} p(y) dy \\
&= \epsilon + C(x) - C(x-2\delta), \qquad (\text{C.40})
\end{aligned}$$

where the second step follows from Cauchy-Schwarz inequality, the forth step follows from Eq. (C.38), the fifth step follows from $p(x)$ is a density function, the sixth step follows from $H(y) = 1$ and $F_L(y) \in [0,1]$ for $y \in [0, 2\delta]$, the last step follows from $C(x)$ is the CDF of $p(x)$ in $[-\pi, \pi]$.

Hence, we have

$$\widetilde{C}_L(x) \geq C(x) - (\epsilon + C(x) - C(x-2\delta)) = C(x-2\delta) - \epsilon, \tag{C.41}$$

which proves the first inequality:

$$\widetilde{C}(x-\delta) \geq C(x-2\delta) - \epsilon. \tag{C.42}$$

Similarly, we can define $F_R := F(x+\delta)$ and $\widetilde{C}_R(x) := (F_R * p)(x)$. We can show that

$$|C(x) - \widetilde{C}_R(x)| \leq \epsilon + C(x+2\delta) - C(x), \tag{C.43}$$

which gives

$$\widetilde{C}(x+\delta) \leq C(x+2\delta) + \epsilon. \tag{C.44}$$

The lemma is then proved. □

C.1.3.1 Technical lemma

Lemma C.8 (Mollifier, Lemma 5 in [LT22]). *Define $M_{d,\delta}(x)$ to be*

$$M_{d,\delta} := \frac{1}{\mathcal{N}_{d,\delta}} T_d\left(1 + 2\frac{\cos(x) - \cos(\delta)}{1+\cos(\delta)}\right) \tag{C.45}$$

where $T_d(x)$ is the d-th Chebyshev polynomial of the first kind, and

$$\mathcal{N}_{d,\delta} := \int_{-\pi}^{\pi} T_d\left(1 + 2\frac{\cos(x) - \cos(\delta)}{1+\cos(\delta)}\right) dx. \tag{C.46}$$

Then

1. $|M_{d,\delta}(x)| \leq \frac{1}{\mathcal{N}_{d,\delta}}$ *for* $x \in [-\pi, -\delta] \cup [\delta, \pi]$*, and* $M_{d,\delta}(x) \geq \frac{1}{\mathcal{N}_{d,\delta}}$ *for* $x \in [-\delta, \delta]$.
2. $\int_{-\pi}^{\pi} M_{d,\delta}(x)dx = 1$, $1 \leq \int_{-\pi}^{\pi} |M_{d,\delta}(x)|dx \leq 1 + \frac{4\pi}{\mathcal{N}_{d,\delta}}$.
3. *When* $\tan(\delta/2) \leq 1 - 1/\sqrt{2}$, *we have*

$$\mathcal{N}_{d,\delta} \geq C_1 e^{d\delta/\sqrt{2}} \sqrt{\frac{\delta}{d}} \cdot \operatorname{erf}(C_2\sqrt{d}\delta), \tag{C.47}$$

for some universal constant C_1, C_2.

The proof can be found in Appendix A in [LT22], and we omit it here.

C.2 Technical Details of the Hadamard Test of Block-Encoded Observable

In this section, we give a detailed analysis of the Hadamard test for block-encodings which plays a crucial role in the proof of Theorem 5.17.

We first note that the quantum state before the final measurements is as follows:

$$|\phi_1\rangle = \begin{cases} \frac{1}{\sqrt{2}}\left(|+\rangle|0^m\rangle|\phi_0\rangle + |-\rangle(I\otimes e^{-iHt_2})U(I\otimes e^{-iHt_1})|0^m\rangle|\phi_0\rangle\right) & \text{if } W = I, \\ \frac{1}{\sqrt{2}}\left(|+\rangle|0^m\rangle|\phi_0\rangle + i|-\rangle(I\otimes e^{-iHt_2})U(I\otimes e^{-iHt_1})|0^m\rangle|\phi_0\rangle\right) & \text{if } W = S. \end{cases} \tag{C.48}$$

Case 1: $W = I$ We measure the first two registers. If the outcome is $(0, 0^m)$, the (un-normalized) remaining state is:

$$\begin{aligned} &(\langle 0|\langle 0^m|\otimes I)\frac{1}{\sqrt{2}}\left(|+\rangle|0^m\rangle|\phi_0\rangle + |-\rangle(I\otimes e^{-iHt_2})U(I\otimes e^{-iHt_1})|0^m\rangle|\phi_0\rangle\right) \\ &= \frac{1}{2}|\phi_0\rangle + \frac{1}{2\alpha}e^{-iHt_2}Oe^{-iHt_1}|\phi_0\rangle \end{aligned} \tag{C.49}$$

Hence, this event happens with the following probability:

$$\begin{aligned} &\Pr[\text{the outcome is } (0,0^m)|W=I] && \text{(C.50)} \\ &= \langle\phi_0|\left(\frac{1}{2}I + \frac{1}{2\alpha}e^{iHt_1}O^\dagger e^{iHt_2}\right)\left(\frac{1}{2}I + \frac{1}{2\alpha}e^{-iHt_2}Oe^{-iHt_1}\right)|\phi_0\rangle \\ &= \frac{1}{4}\left(1 + \frac{1}{\alpha}\langle\phi_0|e^{iHt_1}O^\dagger e^{iHt_2}|\phi_0\rangle + \frac{1}{\alpha}\langle\phi_0|e^{-iHt_2}Oe^{-iHt_1}|\phi_0\rangle + \frac{1}{\alpha^2}\langle\phi_0|e^{iHt_1}O^\dagger Oe^{-iHt_1}|\phi_0\rangle\right). && \text{(C.51)} \end{aligned}$$

Similarly, if the outcome is $(1, 0^m)$, the remaining (un-normalized) state is

$$\frac{1}{2}|\phi_0\rangle - \frac{1}{2\alpha}e^{-iHt_2}Oe^{-iHt_1}|\phi_0\rangle, \tag{C.52}$$

and the probability is

$$\begin{aligned} &\Pr[\text{the outcome is } (1,0^m)|W=I] \\ &= \frac{1}{4}\left(1 - \frac{1}{\alpha}\langle\phi_0|e^{iHt_1}O^\dagger e^{iHt_2}|\phi_0\rangle - \frac{1}{\alpha}\langle\phi_0|e^{-iHt_2}Oe^{-iHt_1}|\phi_0\rangle + \frac{1}{\alpha^2}\langle\phi_0|e^{iHt_1}O^\dagger Oe^{-iHt_1}|\phi_0\rangle\right). \end{aligned} \tag{C.53}$$

Hence, the expectation of X is

$$\mathbb{E}[X] = \alpha \cdot (\Pr[\text{the outcome is } (0, 0^m)|W = I] - \Pr[\text{the outcome is } (1, 0^m)|W = I]) \tag{C.54}$$

$$\begin{aligned}
&= \frac{1}{2}(\langle\phi_0| e^{-iHt_2} O e^{-iHt_1} |\phi_0\rangle + \langle\phi_0| e^{iHt_1} O^\dagger e^{iHt_2} |\phi_0\rangle) \\
&= \frac{1}{2}(\langle\phi_0| e^{-iHt_2} O e^{-iHt_1} |\phi_0\rangle + \overline{\langle\phi_0| e^{-iHt_2} O e^{-iHt_1} |\phi_0\rangle}) \\
&= \Re \langle\phi_0| e^{-iHt_2} O e^{-iHt_1} |\phi_0\rangle .
\end{aligned} \tag{C.55}$$

Case 2: $W = S$ Similar to the case 1, we have

$$\Pr[\text{the outcome is } (0, 0^m)|W = S] \tag{C.56}$$

$$\begin{aligned}
&= \langle\phi_0| \left(\frac{1}{2}I - \frac{i}{2\alpha} e^{iHt_1} O^\dagger e^{iHt_2}\right) \left(\frac{1}{2}I + \frac{i}{2\alpha} e^{-iHt_2} O e^{-iHt_1}\right) |\phi_0\rangle \\
&= \frac{1}{4}\left(1 - \frac{i}{\alpha}\langle\phi_0| e^{iHt_1} O^\dagger e^{iHt_2} |\phi_0\rangle + \frac{i}{\alpha}\langle\phi_0| e^{-iHt_2} O e^{-iHt_1} |\phi_0\rangle + \frac{1}{\alpha^2}\langle\phi_0| e^{iHt_1} O^\dagger O e^{-iHt_1} |\phi_0\rangle\right).
\end{aligned} \tag{C.57}$$

And

$$\begin{aligned}
&\Pr[\text{the outcome is } (1, 0^m)|W = S] \\
&= \frac{1}{4}\left(1 + \frac{i}{\alpha}\langle\phi_0| e^{iHt_1} O^\dagger e^{iHt_2} |\phi_0\rangle - \frac{i}{\alpha}\langle\phi_0| e^{-iHt_2} O e^{-iHt_1} |\phi_0\rangle + \frac{1}{\alpha^2}\langle\phi_0| e^{iHt_1} O^\dagger O e^{-iHt_1} |\phi_0\rangle\right).
\end{aligned} \tag{C.58}$$

Hence,

$$\mathbb{E}[Y] = \alpha \cdot (\Pr[\text{the outcome is } (1, 0^m)|W = S] - \Pr[\text{the outcome is } (0, 0^m)|W = S]) \tag{C.59}$$

$$\begin{aligned}
&= \frac{i}{2}(-\langle\phi_0| e^{-iHt_2} O e^{-iHt_1} |\phi_0\rangle + \overline{\langle\phi_0| e^{-iHt_2} O e^{-iHt_1} |\phi_0\rangle}) \\
&= \Im \langle\phi_0| e^{-iHt_2} O e^{-iHt_1} |\phi_0\rangle .
\end{aligned} \tag{C.60}$$

Therefore,

$$\mathbb{E}[X + iY] = \langle\phi_0| e^{-iHt_2} O e^{-iHt_1} |\phi_0\rangle . \tag{C.61}$$

C.2.1 Generalized Hadamard test

In this subsection, we study the generalized Hadamard test for block-encodings and we will show that the estimator's variance can be reduced by replacing the first Hadamard gate with an α-dependent single-qubit gate.

Suppose $W = I$ and we replace the first Hadamard gate with the following single-qubit gate:

$$G(a,b,\theta) := \begin{bmatrix} a & b \\ -e^{i\theta}\overline{b} & e^{i\theta}\overline{a} \end{bmatrix}, \tag{C.62}$$

where $\theta \in \mathbb{R}$, $a, b \in \mathbb{C}$ with $|a|^2 + |b|^2 = 1$.

Then, we have

$$\begin{aligned}
&\left|0\right\rangle \left|0^m\right\rangle \left|\phi_0\right\rangle \\
\xrightarrow{G(a,b,\theta)}\ & a\left|0\right\rangle \left|0^m\right\rangle \left|\phi_0\right\rangle - e^{i\theta}\overline{b}\left|1\right\rangle \left|0^m\right\rangle \left|\phi_0\right\rangle \\
\xrightarrow{C\text{-}e^{-iHt_1}}\ & a\left|0\right\rangle \left|0^m\right\rangle \left|\phi_0\right\rangle - e^{i\theta}\overline{b}\left|1\right\rangle (I\otimes e^{-iHt_1})\left|0^m\right\rangle \left|\phi_0\right\rangle \\
\xrightarrow{C\text{-}U}\ & a\left|0\right\rangle \left|0^m\right\rangle \left|\phi_0\right\rangle - e^{i\theta}\overline{b}\left|1\right\rangle U(I\otimes e^{-iHt_1})\left|0^m\right\rangle \left|\phi_0\right\rangle \\
\xrightarrow{C\text{-}e^{-iHt_2}}\ & a\left|0\right\rangle \left|0^m\right\rangle \left|\phi_0\right\rangle - e^{i\theta}\overline{b}\left|1\right\rangle (I\otimes e^{-iHt_2})U(I\otimes e^{-iHt_1})\left|0^m\right\rangle \left|\phi_0\right\rangle \\
\xrightarrow{G(p,q,\rho)}\ & a(p\left|0\right\rangle - e^{i\rho}\overline{q}\left|1\right\rangle)\left|0^m\right\rangle \left|\phi_0\right\rangle - e^{i\theta}\overline{b}(q\left|0\right\rangle + e^{i\rho}\overline{p}\left|1\right\rangle)(I\otimes e^{-iHt_2})U(I\otimes e^{-iHt_1})\left|0^m\right\rangle \left|\phi_0\right\rangle \\
=:\ & \left|\phi_1\right\rangle.
\end{aligned}$$

Hence, the un-normalized remaining state after the measurement with outcome $(0, 0^m)$ is:

$$\left(\left\langle 0\right| \left\langle 0^m\right| \otimes I\right)\left|\phi_1\right\rangle = ap\left|\phi_0\right\rangle - \frac{e^{i\theta}\overline{b}q}{\alpha} e^{-iHt_2} O e^{-iHt_1}\left|\phi_0\right\rangle. \tag{C.63}$$

It implies that

$$\Pr[\text{the outcome is } (0,0^m)|W = I] \tag{C.64}$$

$$\begin{aligned}
&= \left\langle \phi_0\right| \left(\overline{ap}I - \frac{e^{-i\theta}b\overline{q}}{\alpha} e^{iHt_1}O^\dagger e^{iHt_2}\right)\left(apI - \frac{e^{i\theta}\overline{b}q}{\alpha}e^{-iHt_2}Oe^{-iHt_1}\right)\left|\phi_0\right\rangle \\
&= |a|^2|p|^2 + \frac{|b|^2|q|^2}{\alpha^2}\left\langle \phi_0\right| e^{iHt_1}O^\dagger O e^{-iHt_1}\left|\phi_0\right\rangle \\
&-\frac{e^{i\theta}\overline{a}\overline{b}\overline{p}q}{\alpha}\left\langle \phi_0\right| e^{-iHt_2}Oe^{-iHt_1}\left|\phi_0\right\rangle - \frac{e^{-i\theta}abp\overline{q}}{\alpha}\overline{\left\langle \phi_0\right| e^{-iHt_2}Oe^{-iHt_1}\left|\phi_0\right\rangle}.
\end{aligned} \tag{C.65}$$

On the other hand, the un-normalized state for the outcome $(1,0^m)$ is

$$(\langle 1| \langle 0^m| \otimes I) |\phi_1\rangle = -e^{i\rho} a\overline{q} |\phi_0\rangle - \frac{e^{i(\theta+\rho)}\overline{b}\overline{p}}{\alpha} e^{-iHt_2} O e^{-iHt_1} |\phi_0\rangle , \tag{C.66}$$

and the probability is

$$\begin{aligned}
&\Pr[\text{the outcome is } (1,0^m)|W=I] \qquad\qquad (\text{C.67})\\
=\ & \langle\phi_0| \left(-e^{-i\rho}\overline{a}qI - \frac{e^{-i(\theta+\rho)}bp}{\alpha} e^{iHt_1} O^\dagger e^{iHt_2}\right)\left(-e^{i\rho}a\overline{q}I - \frac{e^{i(\theta+\rho)}\overline{b}\overline{p}}{\alpha}e^{-iHt_2}Oe^{-iHt_1}\right)|\phi_0\rangle\\
=\ & |a|^2|q|^2 + \frac{|b|^2|p|^2}{\alpha^2}\langle\phi_0| e^{iHt_1}O^\dagger O e^{-iHt_1}|\phi_0\rangle\\
&+\frac{e^{i\theta}\overline{a}\overline{b}\overline{p}q}{\alpha}\langle\phi_0|e^{-iHt_2}Oe^{-iHt_1}|\phi_0\rangle + \frac{e^{-i\theta}abp\overline{q}}{\alpha}\overline{\langle\phi_0|e^{-iHt_2}Oe^{-iHt_1}|\phi_0\rangle} \qquad (\text{C.68})
\end{aligned}$$

If we choose $|p| = |q| = \frac{1}{\sqrt{2}}$, then we have

$$\begin{aligned}
&\Pr[\text{the outcome is } (1,0^m)|W=I] - \Pr[\text{the outcome is } (0,0^m)|W=I]\\
=\ & \Re\frac{4e^{i\theta}\overline{a}\overline{b}\overline{p}q}{\alpha}\langle\phi_0|e^{-iHt_2}Oe^{-iHt_1}|\phi_0\rangle .
\end{aligned} \tag{C.69}$$

Notice that to make the Hadamard test work, we need the coefficient $\frac{4e^{i\theta}\overline{a}\overline{b}\overline{p}q}{\alpha}$ to be a real or an imaginary number.

Now, we show how to choose the parameters to minimize the variance. Without loss of generality, we may assume $a, b \in (0,1)$ such that $a^2+b^2=1$ and use p, q to cancel the phase factor, i.e., $e^{i\theta}\overline{a}\overline{b}\overline{p}q = \frac{1}{2}ab$. It gives that:

$$\begin{aligned}
&\Pr[\text{the outcome is } (1,0^m)|W=I] - \Pr[\text{the outcome is } (0,0^m)|W=I]\\
=\ & \frac{2ab}{\alpha}\Re\langle\phi_0|e^{-iHt_2}Oe^{-iHt_1}|\phi_0\rangle ,
\end{aligned} \tag{C.70}$$

and

$$\begin{aligned}
&\Pr[\text{the outcome is } (1,0^m)|W=I] + \Pr[\text{the outcome is } (0,0^m)|W=I]\\
=\ & a^2 + \frac{b^2}{\alpha^2}\langle\phi_0|e^{iHt_1}O^\dagger Oe^{-iHt_1}|\phi_0\rangle
\end{aligned} \tag{C.71}$$

Now, define the random variable as follows:

$$X := \begin{cases} \frac{\alpha}{2ab} & \text{if the outcome is } (1, 0^m), \\ -\frac{\alpha}{2ab} & \text{if the outcome is } (0, 0^m), \\ 0 & \text{otherwise.} \end{cases} \tag{C.72}$$

Then, we have

$$\mathbb{E}[X] = \Re \langle \phi_0 | e^{-iHt_2} O e^{-iHt_1} | \phi_0 \rangle . \tag{C.73}$$

And we have

$$\begin{aligned} \mathbf{Var}[X] = & \ \mathbb{E}[X^2] - \mathbb{E}[X]^2 \\ = & \ \frac{\alpha^2}{4a^2b^2} \left(a^2 + \frac{b^2}{\alpha^2} \langle \phi_0 | e^{iHt_1} O^\dagger O e^{-iHt_1} | \phi_0 \rangle \right) - \left(\Re \langle \phi_0 | e^{-iHt_2} O e^{-iHt_1} | \phi_0 \rangle \right)^2 . \end{aligned} \tag{C.74}$$

The second term is fixed for any parameters. And for the first term, we have

$$\frac{\alpha^2}{4a^2b^2} \left(a^2 + \frac{b^2}{\alpha^2} \langle \phi_0 | e^{iHt_1} O^\dagger O e^{-iHt_1} | \phi_0 \rangle \right) = \frac{\alpha^2}{4b^2} + \frac{1}{4a^2} \langle \phi_0 | e^{iHt_1} O^\dagger O e^{-iHt_1} | \phi_0 \rangle \tag{C.75}$$

$$\begin{aligned} &= \frac{\alpha^2}{4(1-a^2)} + \frac{\| O e^{-iHt_1} | \phi_0 \rangle \|^2}{4a^2} \\ &\geq \frac{1}{4} (\alpha + \| O e^{-iHt_1} | \phi_0 \rangle \|)^2, \end{aligned} \tag{C.76}$$

where the minimizer is at $a := \sqrt{\frac{\| O e^{-iHt_1} | \phi_0 \rangle \|}{\alpha + \| O e^{-iHt_1} | \phi_0 \rangle \|}}$. However, since we do not know the value of $\| O e^{-iHt_1} | \phi_0 \rangle \|$, there are two approaches to resolve this issue: (1) use another quantum circuit to estimate $\| O e^{-iHt_1} | \phi_0 \rangle \|$ and then set the parameters; (2) just take $a := \sqrt{\frac{1}{\alpha+1}}$. Notice that when the first gate is the Hadamard gate, i.e., $a = \frac{1}{\sqrt{2}}$, we have

$$\mathbf{Var}\left[X \ \middle| \ a = \frac{1}{\sqrt{2}} \right] = \frac{1}{2} (\alpha^2 + \| O e^{-iHt_1} | \phi_0 \rangle \|^2). \tag{C.77}$$

When $a = \sqrt{\frac{1}{\alpha+1}}$, we have

$$\begin{aligned}
\mathbf{Var}\left[X \;\middle|\; a = \frac{1}{\sqrt{\alpha+1}}\right] &= \frac{1}{4}\alpha(\alpha+1) + \frac{1}{4}\|Oe^{-iHt_1}\left|\phi_0\right\rangle\|^2(\alpha+1) \qquad \text{(C.78)} \\
&= \frac{1}{2}(\alpha^2 + \|Oe^{-iHt_1}\left|\phi_0\right\rangle\|^2) - \frac{1}{4}(\alpha-1)(\alpha - \|Oe^{-iHt_1}\left|\phi_0\right\rangle\|^2) \\
&\leq \mathbf{Var}\left[X \;\middle|\; a = \frac{1}{\sqrt{2}}\right], \qquad \text{(C.79)}
\end{aligned}$$

where the last step follows from $\alpha \geq 1$ and $\|Oe^{-iHt_1}\left|\phi_0\right\rangle\|^2 \leq 1$. Therefore, we can reduce the estimator's variance by choosing $a = \sqrt{\frac{1}{\alpha}}$. Moreover, if α is large, the new variance is about half of the variance using the Hadamard gate.

Similar strategy can also be used to reduce the variance of the random variable Y.

Appendix D: Omitted Materials from Chapter 6

D.1 Proof of the QED-mixer's universality for planar graphs

In this section we prove that QED-mixer is able to evolve between two arbitrary paths in $O(n)$ loop operations on a planar graph. *Given a undirected graph $G(V, E)$ with k pairs (s_i, t_i). The goal is to find k paths connecting s_i and t_i for all $i \in [k]$ such that the maximum of congestion in each edge is minimized.*

Proof: We first assume that P_1 and P_2 do not have any common vertex. Then, $s \xrightarrow{P_1} t \xrightarrow{-P_2} s$ forms a closed simple region, where $-P_2$ means the inverse direction of path P_2. By Jordan's theorem [Hal07], we can take the "interior" of this region, which is a subgraph G' of G. It's easy to see that every cycle in G' is also a cycle in G. Hence, we can apply a loop operation for every cycle in G', and doing so in some specific directions will transform P_1 to P_2. Wlog., suppose $s \xrightarrow{P_1} t \xrightarrow{-P_2} s$ is in clockwise direction. Then, for every cycle, we apply a counter-clock loop operator, which is equivalent to transfer 1 unit of flow counter-clockwise through the cycle. Let $e = (u, v)$ be an edge in G'. If e is contained in P_1 and initially there is 1 unit flow from u to v. After the loop operation, another 1 unit flow from v to u is introduced so that the total flow on e is 0. Similarly, if e is contained in P_2 and is in the same direction as P_2, then the flow on e is 1. For all the interior edges, the flow on them is 0 because each edge is contained in two cycles and the loop operation on each cycle will introduce 1 unit flow through e in opposite directions, which will be cancelled by each other. Therefore, after these loop operations, the flow from s to t through P_1 will be transformed to P_2.

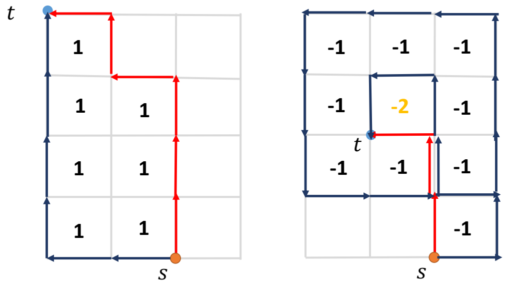
In general, let $v_1 = s, \dots, v_l = t$ be l common vertices between P_1 and P_2, sorted by their appearance orders in the path. Then, we can see that for all $i \in [t-1]$, $v_i \to v_{i+1} \to v_i$ forms a closed simple region and we take the subgraph G_i'. For each G_i', we can apply a series of loop operations to transform $v_i \xrightarrow{P_1} v_{i+1}$ to $v_i \xrightarrow{P_2} v_{i+1}$.

Therefore, after processing $t-1$ subgraphs, P_1 will be transformed to P_2.

Lastly, we show that the number of loop operations we applied is $O(n)$. For each cycle, we only apply the corresponding loop operation once. Hence, the number of loop operations is upper bounded by the number of cycles in G. Since G is planar, Euler characteristic for planar graph gives $n-m+f=2$, where m is the number of edges in G and f is the number of cycles. Thus, we have $f=m-n+2$. We also know that, for planar graph, $m \leq 3n-6$. Hence, $f \leq 2n-4 = O(n)$. Therefore, we can transform P_1 to P_2 by $O(n)$ loop operations.

D.2 Dual “height-model" formulation

Here we consider a canonical (for a detailed description, see, [Fis04], for example) dual picture description of the algorithm on plane graphs that might be useful for implementation sometimes. In graph theory, the dual of any plane graph G is obtained by taking each of its faces as a vertex, and drawing an edge between any two neighboring faces. In this dual representation, an initial configuration is chosen, and the states are defined by the relative “loop distance" to the initial configuration. The amount of flow on each path is then equal to the initial flow plus the difference between states of adjacent faces with the direction perpendicular counterclockwise to the gradient direction. Namely, a “1" state on some elementary loop adds a counterclockwise flow loop to the initial configuration, and vice versa. Naively, one would think that the total Hilbert size for EDP problem becomes 3^{kf}, where f stands for the number of faces; this makes the Hilbert space a polynomial order less than the original picture, considering $e > f$ on planar graphs. In addition, we could prepare equal superposition states in the dual picture. Nevertheless, there are cases which require more levels on each face, as shown in Figure. D.1. For larger graphs, the dual encoding thus becomes even more expensive. On the other hand, the encoding still cannot get rid of isolated loops, although a direct interpretation of RQED-mixer is possible. In conclusion, the dual description can be a useful alternative when consid-



.

Figure D.1: **Examples of dual picture description** We consider a single-pair flow instance: red represents initial/reference configuration, and blue stands for the final configuration. Numbers on each faces stands for the states encoded in dual picture language; every unlabeled face has state 0. In some cases (left) the range of dual state could be simply $-1, 0, 1$ whereas more complicated paths (right) needs greater range, which could be proportional to the radius of the graph, namely $O(\sqrt{n})$.

ering the ordinary QED-mixer on small graphs, but adds significant qubit resource overheads for larger graphs.

Appendix E: Omitted Materials from Chapter 9

E.1 Initialization

Let us state an initialization result which is very standard in literature, see Section 10 in [LSW15], Appendix A in [CLS19], and Section 9 in [JKL+20]. It can be easily proved using the property of special matrix/Kronecker product (Fact 9.14).

Lemma E.1. *For an SDP instance defined in Definition 9.1 (m $n \times n$ constraint matrices, let X^* be any optimal solution to SDP.), assume it has two properties :*

1. *Bounded diameter: for any feasible solution $X \in \mathbb{R}^{n\times n}_{\succeq 0}$, it has $\|X\|_2 \leq R$.*
2. *Lipschitz objective: the objective matrix $C \in \mathbb{R}^{n\times n}$ has bounded spectral norm, i.e., $\|C\|_2 \leq L$.*

Given $\epsilon \in (0, 1/2]$, we can construct the following modified *SDP instance in dimension $n+2$ with $m+1$ constraints:*

$$\max_{\overline{X}\succeq 0} \ \langle \overline{C}, \overline{X} \rangle$$
$$\text{s.t. } \langle \overline{A}_i, \overline{X} \rangle = \overline{b}_i, \ \forall i \in [m+1],$$

where

$$\overline{A}_i = \begin{bmatrix} A_i & 0_n & 0_n \\ 0_n^\top & 0 & 0 \\ 0_n^\top & 0 & \frac{b_i}{R} - \operatorname{tr}[A_i] \end{bmatrix} \quad \forall i \in [m], \ \textit{and} \ \overline{A}_{m+1} = \begin{bmatrix} I_n & 0_n & 0_n \\ 0_n^\top & 1 & 0 \\ 0_n^\top & 0 & 0 \end{bmatrix}.$$

$$\overline{b} = \begin{bmatrix} \frac{1}{R} b \\ n+1 \end{bmatrix}, \ \overline{C} = \begin{bmatrix} \frac{\epsilon}{L}\cdot C & 0_n & 0_n \\ 0_n^\top & 0 & 0 \\ 0_n^\top & 0 & -1 \end{bmatrix}.$$

Moreover, it has three properties:

1. *$(\overline{X}_0, \overline{y}_0, \overline{S}_0)$ are feasible primal and dual solutions of the modified instance, where*

$$\overline{X}_0 = I_{n+2} \ , \ \overline{y}_0 = \begin{bmatrix} 0_m \\ 1 \end{bmatrix} \ , \ \overline{S}_0 = \begin{bmatrix} I_n - C \cdot \frac{\epsilon}{L} & 0_n & 0 \\ 0_n^\top & 1 & 0 \\ 0_n^\top & 0 & 1 \end{bmatrix} . \tag{E.1}$$

2. *For any feasible primal and dual solutions $(\overline{X}, \overline{y}, \overline{S})$ with duality gap at most ϵ^2, the matrix $\widehat{X} = R \cdot \overline{X}_{[n]\times[n]}$, where $\overline{X}_{[n]\times[n]}$ is the top-left n-by-n block submatrix of $\overline{X}$. The matrix $\widehat{X}$ has three properties*

$$\langle C, \widehat{X} \rangle \geq \langle C, X^* \rangle - LR \cdot \epsilon,$$
$$\widehat{X} \succeq 0,$$
$$\sum_{i \in [m]} |\langle A_i, \widehat{X} \rangle - b_i| \leq 4n\epsilon \cdot \Big(R \sum_{i \in [m]} \|A_i\|_1 + \|b\|_1 \Big),$$

3. *If we take $\epsilon \leq \epsilon_N^2$ in Eq. (E.1), then the initial dual solution satisfies the induction invariant:*

$$g(\overline{y}_0, \eta) H(\overline{y}_0)^{-1} g(\overline{y}_0, \eta) \leq \epsilon_N^2.$$

E.2 From Dual to Primal

We state a lemma about transforming a nearly optimal dual solution to a primal solution for SDP, which is very standard in literature and follows directly from Section 10 in [LSW15] and Fact 9.14.

Lemma E.2. *Given parameter $\eta_{\text{final}} = \frac{1}{n+2}(1 + \frac{\epsilon_N}{20\sqrt{n}})^T$ where $T = 40\epsilon_N^{-1}\sqrt{n}\log(n/\epsilon)$, dual variable $y \in \mathbb{R}^m$ and slack variable $S \in \mathbb{R}^{n \times n}$ such that*

$$g(y, \eta_{\text{final}}) H(y)^{-1} g(y, \eta_{\text{final}}) \leq \epsilon_N^2$$
$$\sum_{i=1}^m y_i A_i - C = S$$
$$b^\top y \leq b^\top y^* + \frac{n}{\eta_{\text{final}}}(1 + 2\epsilon_N)$$
$$S \succ 0$$

with $\epsilon_N \leq 1/10$. Then there is an algorithm that finds a primal variable $X \in \mathbb{R}^{n \times n}$ in $O(n^{\omega+o(1)})$ time such that

$$\begin{aligned} \langle C, X\rangle &\geq \langle C, X^*\rangle - LR\cdot\epsilon \\ X &\succeq 0 \\ \sum_{i\in[m]} |\langle A_i, X\rangle - b_i| &\leq 4n\epsilon \cdot (R\sum_{i\in[m]} \|A_i\|_1 + \|b\|_1) \end{aligned} \tag{E.2}$$

E.3 Our Straightforward Implementation of the Hybrid Barrier SDP Solver

Theorem E.3 (Our straight forward implementation of the hybrid algorithm [Ans00])**.** *The original hybrid barrier algorithm [Ans00] use*

$$O^*(m^2 n^\omega + m^4 n^2 + m^{\omega+2})$$

cost per iteration.

Remark E.1. A naive implementation of hybrid barrier (e.g. Table 1.1 of [JKL+20][1]) takes time

$$O^*(m^3 n^\omega + m^4 n^2 + m^{\omega+2}).$$

Our implementation (Theorem E.3) improves it to $m^2 n^\omega + m^4 n^2 + m^{\omega+2}$ by reusing the computations in the Hessian matrix.

Proof. In each iteration, the computation workload is comprised of the following:

- The slack variable $S \in \mathbb{R}^{n\times n}$, given by

$$S = S(y) = \sum_{i=1}^m y_i A_i - C.$$

[1]The bound claimed in [JKL+20] is $O^*(m^3 n^\omega + m^4 n^2)$, since they want to consider the special parameter regime where $n \leq m \leq n^2$. In that regime, $m^{\omega+2}$ is dominated by the first two terms.

- The gradient of ϕ_{vol}, denote by $\nabla\phi_{\mathrm{vol}}(y) \in \mathbb{R}^m$, given by:

$$\nabla\phi_{\mathrm{vol}}(y)_i = -\mathrm{tr}[H(S)^{-1} \cdot \mathsf{A}(S^{-1}A_iS^{-1} \otimes S^{-1})\mathsf{A}^\top].$$

- The gradient of $\phi_{\log}$, denote by $\nabla\phi_{\log}(y) \in \mathbb{R}^m$, given by:

$$\nabla\phi_{\log}(y)_i = -\mathrm{tr}[S^{-1} \cdot A_i].$$

- The Hessian matrix of $\phi_{\log}$, denoted by $H(S) \in \mathbb{R}^{m\times m}$, given by:

$$H(S) = \mathsf{A} \cdot (S^{-1} \otimes S^{-1}) \cdot \mathsf{A}^\top$$

- The first component of the Hessian matrix of ϕ_{vol}, denoted by $Q(S) \in \mathbb{R}^{m\times m}$, (recall from [Ans00], $\nabla^2\phi_{\mathrm{vol}}(y) = 2Q(S) + R(S) - 2T(S) \in \mathbb{R}^{m\times m}$), given by:

$$Q(S)_{i,j} = \mathrm{tr}[H(S)^{-1}\mathsf{A}(S^{-1}A_iS^{-1}A_jS^{-1} \otimes_S S^{-1})\mathsf{A}^\top].$$

- The second component of Hessian matrix of ϕ_{vol}, denoted by $R(S) \in \mathbb{R}^{m\times m}$, given by:

$$R(S)_{i,j} = \mathrm{tr}[H(S)^{-1}\mathsf{A}(S^{-1}A_iS^{-1} \otimes_S S^{-1}A_jS^{-1})\mathsf{A}^\top].$$

- The third component of Hessian matrix of ϕ_{vol}, denoted by $T(S) \in \mathbb{R}^{m\times m}$, given by:

$$T(S)_{i,j} = \mathrm{tr}[H(S)^{-1}\mathsf{A}(S^{-1}A_iS^{-1} \otimes_S S^{-1})\mathsf{A}^\top H(S)^{-1}\mathsf{A}(S^{-1}A_jS^{-1} \otimes_S S^{-1})\mathsf{A}^\top].$$

- Newton direction, denoted by $\delta_y \in \mathbb{R}^m$, given by

$$\delta_y = -(\nabla^2\phi(y))^{-1}(\eta b - \nabla\phi(y)).$$

To find all these items, we carry out the following computations.

Step 1. We compute $S \in \mathbb{R}^{n\times n}$ in mn^2 time.

Step 2. We compute $S^{-1}A_iS^{-1}A_j \in \mathbb{R}^{n\times n}$ and $S^{-1}A_i$ for all $i \in \{1, \cdots, m\}$, $\forall j \in \{1, \cdots, m\}$. This step costs $n^\omega m^2$. Since $H(S)_{i,j} = \mathrm{tr}[S^{-1}A_iS^{-1}A_j]$, it costs an additional nm^2 time to compute $H(S)$, and n^ω time to find $H(S)^{-1}$ correspondingly. Since $\nabla\phi_{\log}(y)_i = -\mathrm{tr}[S^{-1}\cdot A_j]$, we already find $\nabla\phi_{\log}(y)$. In total, this step costs $n^\omega m^2$ time.

Step 3. We compute $\mathrm{tr}[S^{-1}A_kS^{-1}A_lS^{-1}A_i]$ and $\mathrm{tr}[S^{-1}A_kS^{-1}A_lS^{-1}A_iS^{-1}A_j]$ for all $i, j, k, l \in [m]$. For the former, we only need to sum up all diagonal terms of $S^{-1}A_kS^{-1}A_lS^{-1}A_i$, and each diagonal term is computed by multiplying one column of $S^{-1}A_k$ and one column of $S^{-1}A_lS^{-1}A_i$. Therefore, the cost of finding $\mathrm{tr}[S^{-1}A_kS^{-1}A_lS^{-1}A_i]$ for all $i, k, l \in [m]$ is m^3n^2. For the latter, we only need to sum up all diagonal terms of $S^{-1}A_kS^{-1}A_lS^{-1}A_iS^{-1}A_j$, and each diagonal term is computed by multiplying one column of $S^{-1}A_kS^{-1}A_l$ and one column of $S^{-1}A_iS^{-1}A_j$. Therefore, the cost of finding $\mathrm{tr}[S^{-1}A_kS^{-1}A_lS^{-1}A_iS^{-1}A_j]$ for all $i, j, k, l \in [m]$ is m^4n^2.[2] In total, this step costs m^4n^2 time.

Step 4. We compute $\nabla\phi_{\mathrm{vol}}(y)$, $Q(S)$, $R(S)$, and $T(S)$. We note

$$(\mathsf{A}(S^{-1}A_iS^{-1} \otimes S^{-1})\mathsf{A}^\top)_{k,l} = \mathrm{tr}[A_kS^{-1}A_lS^{-1}A_iS^{-1}]$$
$$(\mathsf{A}(S^{-1} \otimes S^{-1}A_iS^{-1})\mathsf{A}^\top)_{k,l} = \mathrm{tr}[A_kS^{-1}A_iS^{-1}A_lS^{-1}]$$

and

$$(\mathsf{A}(S^{-1}A_iS^{-1}A_jS^{-1} \otimes S^{-1})\mathsf{A}^\top)_{k,l} = \mathrm{tr}[A_kS^{-1}A_lS^{-1}A_iS^{-1}A_jS^{-1}]$$
$$(\mathsf{A}(S^{-1} \otimes S^{-1}A_iS^{-1}A_jS^{-1})\mathsf{A}^\top)_{k,l} = \mathrm{tr}[A_kS^{-1}A_iS^{-1}A_jS^{-1}A_\ell S^{-1}]$$

and

$$(\mathsf{A}(S^{-1}A_iS^{-1} \otimes S^{-1}A_jS^{-1})\mathsf{A}^\top)_{k,l} = \mathrm{tr}[A_kS^{-1}A_jS^{-1}A_lS^{-1}A_iS^{-1}]$$

[2]If we batch them together and use matrix multiplication, this term will be $\mathcal{T}_{\mathrm{mat}}(m^2, n^2, m^2)$.

Thus each coordinate of $\nabla\phi_{\mathrm{vol}}(y)$, $Q(S)$, $R(S)$, and $T(S)$ can be computed by multiplying the terms in the second step (i.e. $\mathrm{tr}[S^{-1}A_kS^{-1}A_lS^{-1}A_i]$ and $\mathrm{tr}[S^{-1}A_kS^{-1}A_lS^{-1}A_iS^{-1}A_j]$ for all $i,j,k,l \in [m]$) with $H(S)^{-1}$, and then taking trace. These cost $O(m^{\omega+2})$ time, in total. More specifically, for the gradient $\nabla\phi_{\mathrm{vol}}(y)$,

$$\begin{aligned}(\nabla\phi_{\mathrm{vol}}(y))_i = & -\mathrm{tr}[H(S)^{-1}\cdot \mathsf{A}(S^{-1}A_iS^{-1}\otimes S^{-1})\mathsf{A}^\top] \\ = & \sum_{k=1}^{m}\sum_{l=1}^{m} -H(S)^{-1}_{k,l}\cdot \mathrm{tr}[A_kS^{-1}A_lS^{-1}A_iS^{-1}].\end{aligned}$$

Hence, it takes $O(m^3)$-time to compute $\nabla\phi_{\mathrm{vol}}(y)$.

For $Q(S)$,

$$\begin{aligned}Q(S)_{i,j} &= \mathrm{tr}[H(S)^{-1}\mathsf{A}(S^{-1}A_iS^{-1}A_jS^{-1}\otimes_S S^{-1})\mathsf{A}^\top] \\ &= \sum_{k=1}^{m}\sum_{l=1}^{m} -H(S)^{-1}_{k,l}\cdot\frac{1}{2}\left(\mathrm{tr}[A_kS^{-1}A_lS^{-1}A_iS^{-1}A_jS^{-1}] + \mathrm{tr}[A_kS^{-1}A_iS^{-1}A_jS^{-1}A_\ell S^{-1}]\right).\end{aligned}$$

Hence, it takes $O(m^4)$-time to compute $Q(S)$.

For $R(S)$,

$$\begin{aligned}R(S)_{i,j} &= \mathrm{tr}[H(S)^{-1}\mathsf{A}(S^{-1}A_iS^{-1}\otimes_S S^{-1}A_jS^{-1})\mathsf{A}^\top] \\ &= \sum_{k=1}^{m}\sum_{l=1}^{m} -H(S)^{-1}_{k,l}\cdot\frac{1}{2}\left(\mathrm{tr}[A_kS^{-1}A_jS^{-1}A_lS^{-1}A_iS^{-1}] + \mathrm{tr}[A_kS^{-1}A_iS^{-1}A_lS^{-1}A_jS^{-1}]\right).\end{aligned}$$

Hence, it takes $O(m^4)$-time to compute $Q(S)$. For $T(S)$,

$$\begin{aligned}T(S)_{i,j} &= \mathrm{tr}[H(S)^{-1}\mathsf{A}(S^{-1}A_iS^{-1}\otimes_S S^{-1})\mathsf{A}^\top H(S)^{-1}\mathsf{A}(S^{-1}A_jS^{-1}\otimes_S S^{-1})\mathsf{A}^\top] \\ &= \mathrm{vec}[\mathsf{A}(S^{-1}A_jS^{-1}\otimes_S S^{-1})\mathsf{A}^\top]^\top (H(S)^{-1}\otimes H(S)^{-1})\mathrm{vec}[\mathsf{A}(S^{-1}A_iS^{-1}\otimes_S S^{-1})\mathsf{A}^\top].\end{aligned}$$

The matrix $\mathsf{A}(S^{-1}A_iS^{-1}\otimes_S S^{-1})\mathsf{A}^\top$ can be computed in $O(m^2)$-time, via

$$(\mathsf{A}(S^{-1}A_iS^{-1}\otimes_S S^{-1})\mathsf{A}^\top)_{k,l} = \frac{1}{2}\left(\mathrm{tr}[A_kS^{-1}A_lS^{-1}A_iS^{-1}] + \mathrm{tr}[A_kS^{-1}A_iS^{-1}A_lS^{-1}]\right).$$

Then, we vectorize this matrix and multiply with the Kronecker product $H(S)^{-1}\otimes H(S)^{-1}$. It takes $O(m^\omega)$-time to obtain the vector

$$(H(S)^{-1}\otimes H(S)^{-1})\mathrm{vec}[\mathsf{A}(S^{-1}A_iS^{-1}\otimes_S S^{-1})\mathsf{A}^\top] \in \mathbb{R}^{m^2}.$$

Next, we do the inner product and get $T(S)_{i,j}$ in $O(m^2)$-time. Thus, $T(S)$ can be computed in $O(m^{\omega+2})$-time. Therefore, this step takes $m^{\omega+2}$ time in total.

Step 5. We compute $\delta_y \in \mathbb{R}^m$. Since

$$\nabla^2 \phi(y) = 225\sqrt{\frac{n}{m}} \cdot \left(Q(S) + R(S) + T(S) + \frac{m-1}{n-1} \cdot H(S) \right),$$

it can be found in m^2 time using the terms in the second and fourth step. Notice $\eta b - \nabla\phi(y) = \eta b - 225\sqrt{\frac{n}{m}} \cdot \left(\nabla\phi_{\mathrm{vol}}(y) + \frac{m-1}{n-1} \cdot \nabla\phi_{\log}(y)\right)$, it can be computed in m time. Then, δ_y can be computed in m^ω time. In total, this step costs m^ω time.

Summing up, the total cost per iteration is given by

$$mn^2 + n^\omega m^2 + m^4 n^2 + n^\omega m^2 + m^{\omega+2} = m^2 n^\omega + m^4 n^2 + m^{\omega+2}.$$

$\square$

E.4 Maintain the Leverage Score Matrix of the Volumetric Barrier

It is observed in the LP that the volumetric barrier is roughly like the log barrier weighted by the leverage score of the matrix $A_x := S^{-1}A$, which is defined to be the diagonal elements of the orthogonal projection matrix $A_x^\top (A_x^\top A_x)^\top A_x$. Similar phenomenon also appears in the SDP. we first consider the orthogonal projection matrix $P \in \mathbb{R}^{n^2 \times n^2}$ on the image of $\mathsf{A}(S^{-1/2} \otimes S^{-1/2})$:

$$P(S) := (S^{-1/2} \otimes S^{-1/2})\mathsf{A}^\top \left(\mathsf{A}(S^{-1} \otimes S^{-1})\mathsf{A}^\top\right)^{-1} \mathsf{A}(S^{-1/2} \otimes S^{-1/2}) \in \mathbb{R}^{n^2 \times n^2}.$$

Then, we define the leverage score matrix as the block-trace of $P(S)$:

Definition E.1 (Leverage score matrix)**.** For $S \succ 0$, define the leverage score matrix $\Sigma(S) \in \mathbb{R}^{n \times n}$ as:

$$\Sigma(S)_{i,j} := \mathrm{tr}\left[(e_i \otimes I_n)^\top P(e_j \otimes I_n)\right] \quad \forall i, j \in [n].$$

In this section, we show how to efficiently compute the leverage score matrix in each iteration of the IPM via low-rank update and amortization, which may be of independent interest.

E.4.1 Basic facts on the leverage score matrix

Fact E.4 (Gradient of the volumetric barrier)**.** *For $S \succ 0$, we have*

$$\nabla \phi_{\mathrm{vol}}(y) = -\mathsf{A}(S^{-1/2} \otimes S^{-1/2})\mathrm{vec}(\Sigma).$$

The computation cost is $mn^2 + n^\omega$.

Proof. The computation cost is $mn^2 + \mathcal{T}_{\mathrm{kron}}(n)$, where $\mathcal{T}_{\mathrm{kron}}(n)$ is the time to compute $(A \otimes B)v$ for $A, B \in \mathbb{R}^{n\times n}, v \in \mathbb{R}^{n^2}$.

By Fact 9.12, we have $\mathcal{T}_{\mathrm{kron}}(n) = n^\omega$. □

Fact E.5 ("Proxy" Hessian of the volumetric barrier)**.** *For $S \succ 0$, we have*

$$Q(S) = \mathsf{A}(S \otimes (S^{-1/2}\Sigma S^{-1/2}))\mathsf{A}^\top.$$

Fact E.6 (Trace of the leverage score matrix)**.** *It holds that*

$$\mathrm{tr}[\Sigma] = m.$$

E.4.2 Efficient algorithm for the leverage score matrix

This section shows how to maintain Σ efficiently.

Lemma E.7. *Let $\Sigma(S)_{i,j} := \mathrm{tr}\left[(e_i \otimes I_n)^\top P(S)(e_j \otimes I_n)\right]$. $S^{-1/2} \in \mathbb{R}^{n\times n}$, $S^{-1} \in \mathbb{R}^{n\times n}$, and $H(S)^{-1} \in \mathbb{R}^{n^2\times n^2}$ are known. Then we can compute $\Sigma(S) \in \mathbb{R}^{n\times n}$ it in*

$$\min\left\{n^4 + \mathcal{T}_{M(S)}, n^\omega m^2\right\}$$

Proof. We provide two different approaches for computing the matrix $\Sigma(S)$.

Approach 1. Each entry of $\Sigma(S)$ can be expressed as follows:

$$\begin{aligned}
\Sigma(S)_{i,j} &:= \mathrm{tr}\left[(e_i \otimes I_n)^\top P(S)(e_j \otimes I_n)\right] \\
&= \mathrm{tr}\left[(e_i \otimes I_n)^\top (S^{-1/2} \otimes S^{-1/2})\mathsf{A}^\top H(S)^{-1}\mathsf{A}(S^{-1/2} \otimes S^{-1/2})(e_j \otimes I_n)\right] \\
&= \mathrm{tr}\left[\left((S^{-1/2})_i^\top \otimes S^{-1/2}\right)\mathsf{A}^\top H(S)^{-1}\mathsf{A}\left((S^{-1/2})_j \otimes S^{-1/2}\right)\right] \\
&= \mathrm{tr}\left[\left((S^{-1/2})_i^\top \otimes S^{-1/2}\right)M(S)\left((S^{-1/2})_j \otimes S^{-1/2}\right)\right] \\
&= \mathrm{tr}\left[\left((S^{-1/2})_j \otimes S^{-1/2}\right)\left((S^{-1/2})_i^\top \otimes S^{-1/2}\right)M(S)\right] \\
&= \mathrm{tr}\left[\left((S^{-1/2})_j(S^{-1/2})_i^\top \otimes S^{-1}\right)M(S)\right],
\end{aligned}$$

where $M(S) := \mathsf{A}^\top H(S)^{-1}\mathsf{A} \in \mathbb{R}^{n^2 \times n^2}$. Notice that

$$\begin{aligned}
&\left((S^{-1/2})_j(S^{-1/2})_i^\top \otimes S^{-1}\right)M(S) \\
=\ &\left((S^{-1/2})_j(S^{-1/2})_i^\top \otimes S^{-1}\right)\left[\mathrm{vec}[M(S)_{(1,1)}] \quad \mathrm{vec}[M(S)_{(1,2)}] \quad \cdots \quad \mathrm{vec}[M(S)_{(n,n)}]\right] \\
=\ &\left[\mathrm{vec}[S^{-1}M_{(1,1)}(S^{-1/2})_i(S^{-1/2})_j^\top] \quad \cdots \quad \mathrm{vec}[S^{-1}M_{(n,n)}(S^{-1/2})_i(S^{-1/2})_j^\top]\right] \in \mathbb{R}^{n^2\times n^2},
\end{aligned}$$

where $M_{(k,\ell)} \in \mathbb{R}^{n\times n} := \mathrm{mat}[M(S)_{(k,\ell)}]$ is the matrix form of the (k,ℓ)-th column of $M(S) \in \mathbb{R}^{n^2\times n^2}$ for $(k,\ell) \in [n]\times[n]$. Hence,

$$\begin{aligned}
\Sigma(S)_{i,j} &= \sum_{k=1}^n\sum_{\ell=1}^n \mathrm{vec}\left[S^{-1}M_{(k,\ell)}(S^{-1/2})_i(S^{-1/2})_j^\top\right]_{(k-1)n+\ell} \\
&= \sum_{k=1}^n\sum_{\ell=1}^n \left(S^{-1}M_{(k,\ell)}(S^{-1/2})_i(S^{-1/2})_j^\top\right)_{\ell,k} \\
&= \sum_{k=1}^n\sum_{\ell=1}^n e_\ell^\top \cdot S^{-1}M_{(k,\ell)}(S^{-1/2})_i(S^{-1/2})_j^\top \cdot e_k \\
&= \sum_{k=1}^n\sum_{\ell=1}^n (S^{-1})_\ell^\top \cdot M_{(k,\ell)} \cdot (S^{-1/2})_i \cdot (S^{-1/2})_{j,k}.
\end{aligned}$$

Then, we get that

$$\begin{aligned}
\Sigma(S) &= \sum_{i=1}^n \sum_{j=1}^n e_i e_j^\top \sum_{k=1}^n \sum_{\ell=1}^n (S^{-1})_\ell^\top \cdot M_{(k,\ell)} \cdot (S^{-1/2})_i \cdot (S^{-1/2})_{j,k} \\
&= \sum_{i=1}^n \sum_{k=1}^n \sum_{\ell=1}^n \left((S^{-1})_\ell^\top \cdot M_{(k,\ell)} \cdot (S^{-1/2})_i\right) e_i \cdot \left(\sum_{j=1}^n (S^{-1/2})_{j,k} \cdot e_j^\top\right) \\
&= \sum_{i=1}^n \sum_{k=1}^n \sum_{\ell=1}^n \left((S^{-1})_\ell^\top \cdot M_{(k,\ell)} \cdot (S^{-1/2})_i\right) e_i \cdot (S^{-1/2})_k^\top \\
&= \sum_{k=1}^n \sum_{\ell=1}^n \left(\sum_{i=1}^n (S^{-1})_\ell^\top \cdot M_{(k,\ell)} \cdot (S^{-1/2})_i \cdot e_i\right) \cdot (S^{-1/2})_k^\top \\
&= \sum_{k=1}^n \sum_{\ell=1}^n \left((S^{-1})_\ell^\top \cdot M_{(k,\ell)} \cdot S^{-1/2}\right)^\top \cdot (S^{-1/2})_k^\top \\
&= \sum_{k=1}^n \sum_{\ell=1}^n S^{-1/2} \cdot M_{(k,\ell)}^\top \cdot (S^{-1})_\ell \cdot (S^{-1/2})_k^\top \\
&= S^{-1/2} \cdot \sum_{k=1}^n \sum_{\ell=1}^n M_{(k,\ell)}^\top \cdot (S^{-1})_\ell \cdot e_k^\top \cdot S^{-1/2}. \qquad \text{(E.3)}
\end{aligned}$$

Therefore, once we have $S^{-1/2} \in \mathbb{R}^{n\times n}$, $S^{-1} \in \mathbb{R}^{n\times n}$, and $M(S) \in \mathbb{R}^{n^2\times n^2}$, then the $\Sigma(S)$ can be computed exactly in $O(n^4)$-time using Eq. (E.3). More specifically,

Step 1. For $k, \ell \in [n]$, we form $M_{(k,\ell)}$ from $M(S)$, which takes $O(n^2)$-time.

Step 2. We compute the matrix

$$W_{k,\ell} := M_{(k,\ell)}^\top \cdot (S^{-1})_\ell \cdot e_k^\top \in \mathbb{R}^{n\times n}$$

in $O(n^2)$ time. And it takes $O(n^4)$-time to compute $\{W_{k,\ell}\}_{k,\ell\in[m]}$.

Step 3. We sum all the $W_{k,\ell}$ together in $O(n^4)$-time. And

$$\Sigma(S) = S^{-1/2} \cdot \left(\sum_{k=1}^n \sum_{\ell=1}^n W_{k,\ell}\right) \cdot S^{-1/2},$$

which can be done in $O(n^\omega)$-time.

Hence, $\Sigma(S)$ can be computed in

$$O\left(n^\omega + n^4 + \mathcal{T}_{M(S)}\right) = O\left(n^4 + \mathcal{T}_{M(S)}\right)$$

time, where $\mathcal{T}_{M(S)}$ is the computation cost for computing $M(S)$.

Approach 2. Another approach for computing $\Sigma(S)$ can be done in $O\left(n^\omega m^2\right)$-time, without maintaining $M(S)$.

Recall that

$$\begin{aligned}\Sigma(S)_{i,j} &= \operatorname{tr}\left[\left((S^{-1/2})_i^\top \otimes S^{-1/2}\right)\mathsf{A}^\top \cdot H(S)^{-1} \cdot \mathsf{A}\left((S^{-1/2})_j \otimes S^{-1/2}\right)\right] \\ &= \operatorname{tr}\left[M_i \cdot H(S)^{-1} \cdot M_j^\top\right] \\ &= \left\langle M_j^\top M_i, H(S)^{-1}\right\rangle\end{aligned}$$

Consider the matrix M_i:

$$\begin{aligned}M_i :=&\ \left((S^{-1/2})_i^\top \otimes S^{-1/2}\right)\mathsf{A}^\top \\ =&\ \left((S^{-1/2})_i^\top \otimes S^{-1/2}\right)\begin{bmatrix}\operatorname{vec}[A_1] & \operatorname{vec}[A_2] & \cdots & \operatorname{vec}[A_m]\end{bmatrix} \\ =&\ \begin{bmatrix}S^{-1/2}A_1(S^{-1/2})_i & \cdots & S^{-1/2}A_m(S^{-1/2})_i\end{bmatrix} \in \mathbb{R}^{n\times m}.\end{aligned}$$

For $k,l \in [m]$, the (k,l)-entry of $M_j^\top M_i$ is

$$\begin{aligned}(S^{-1/2}A_k(S^{-1/2})_j)^\top \cdot (S^{-1/2}A_l(S^{-1/2})_i) &= (S^{-1/2})_j^\top A_k S^{-1/2} \cdot S^{-1/2} A_l (S^{-1/2})_i \\ &= (S^{-1/2})_j^\top A_k S^{-1} A_l (S^{-1/2})_i \\ &= (S^{-1/2}A_k S^{-1} A_l S^{-1/2})_{j,i} \\ &= (S^{-1/2}A_l S^{-1} A_k S^{-1/2})_{i,j}.\end{aligned}$$

Hence, (i,j)-entry of $\Sigma(S)$ is

$$\Sigma(S)_{i,j} = \sum_{k=1}^{m}\sum_{l=1}^{m}(S^{-1/2}A_l S^{-1} A_k S^{-1/2})_{i,j} \cdot (H(S)^{-1})_{k,l}.$$

It implies that

$$\Sigma(S) = \sum_{k=1}^{m}\sum_{l=1}^{m} S^{-1/2}A_l S^{-1} A_k S^{-1/2} \cdot (H(S)^{-1})_{k,l}.$$

We use the following steps to compute $\Sigma(S)$:

Step 1. We compute $S^{-1/2}A_l S^{-1} A_k S^{-1/2}$ for all $k, l \in [m]$. It can be done in $O(n^\omega m^2)$-time.

Step 2. We compute $\Sigma(S)$ by summing the m^2 matrices with weights $(H(S)^{-1})_{k,l}$. It can be done in $O(m^2 n^2)$-time.

Hence, it takes $O(n^\omega m^2)$-time in total to compute $\Sigma(S)$, assuming $H(S)^{-1}$ is given. $\square$

E.4.3 Maintain intermediate matrix

The following lemma shows how to efficiently compute $M(S)$ in each iteration:

Lemma E.8 (Compute $M(S)$)**.** *The matrix $M(S) := \mathsf{A}^\top H(S)^{-1}\mathsf{A} \in \mathbb{R}^{n^2\times n^2}$ can be computed as follows:*

- **Part 1.** *In the initialization, if $H(S)^{-1} \in \mathbb{R}^{m\times m}$ is already known, then $M(S)$ can be computed in $\mathcal{T}_{\mathrm{mat}}(n^2, m, n^2)$ time.*
- **Part 2.** *In each iteration, $M(\widetilde{S})$ can be computed in $\mathcal{T}_{\mathrm{mat}}(n^2, n^2, nr_t)$ time.*

Proof. **Part 1.** Given $H(S)^{-1}$, it costs $\mathcal{T}_{\mathrm{mat}}(n^2, m, n^2)$ to compute $M(S) = \mathsf{A}^\top H(S)^{-1}\mathsf{A} \in \mathbb{R}^{n^2\times n^2}$.

Part 2. We can maintain $M(S)$ in each iteration. Let $\widetilde{S} \in \mathbb{R}^{n\times n}$ be the approximated slack variable in the previous iteration and $\widetilde{S}^{\mathrm{new}} \in \mathbb{R}^{n\times n}$ be the current approximated slack variable. Let $G := H(\widetilde{S})^{-1} \in \mathbb{R}^{m\times m}$ and $G^{\rm new} := H(\widetilde{S}^{\rm new})^{-1} \in \mathbb{R}^{m\times m}$. By Lemma 9.23, we have

$$G^{\rm new} = G - G \cdot \mathsf{A}Y_1 \cdot (I + Y_2^\top \mathsf{A}^\top \cdot \mathsf{A}Y_1)^{-1} \cdot Y_2^\top \mathsf{A}^\top \cdot G$$

Then,

$$\begin{aligned} M(\widetilde{S}^{\rm new}) &= \mathsf{A}^\top G^{\rm new} \mathsf{A} \\ &= \mathsf{A}^\top G \mathsf{A} - \mathsf{A}^\top G \cdot \mathsf{A}Y_1 \cdot (I + Y_2^\top \mathsf{A}^\top \cdot \mathsf{A}Y_1)^{-1} \cdot Y_2^\top \mathsf{A}^\top \cdot G\mathsf{A} \\ &= M(\widetilde{S}) - M(\widetilde{S}) \cdot Y_1 \cdot (I + Y_2^\top \mathsf{A}^\top \cdot \mathsf{A}Y_1)^{-1} \cdot Y_2^\top \cdot M(\widetilde{S}), \end{aligned}$$

where $Y_1, Y_2 \in \mathbb{R}^{n^2\times(2nr_t + r_t^2)}$ and $(I + Y_2^\top \mathsf{A}^\top \cdot \mathsf{A}Y_1)^{-1} \in \mathbb{R}^{(2nr_t+r_t^2)\times(2nr_t+r_t^2)}$. Hence, we first compute $M(\widetilde{S})Y_1 \in \mathbb{R}^{n^2\times nr_t}$ and $Y_2^\top M(\widetilde{S}) \in \mathbb{R}^{nr_t\times n^2}$ in $\mathcal{T}_{\rm mat}(n^2, n^2, nr_t)$. $M(\widetilde{S}^{\rm new}) \in \mathbb{R}^{n^2\times n^2}$ can be directly computed from $G^{\rm new} \in \mathbb{R}^{m\times m}$ in $\mathcal{T}_{\rm mat}(n^2, nr_t, n^2)$-time.

□

E.4.4 Amortized running time

Theorem E.9. *There is an algorithm that compute $\Sigma(S)$ in each iteration of Algorithm 45 with amortized cost-per-iteration*

$$\min\left\{n^{2\omega-\frac{1}{2}}, n^\omega m^2\right\}.$$

Proof. By Lemma E.7, we know that the cost-per-iteration to compute $Sigma(S)$ is

$$\min\left\{n^4 + \mathcal{T}_{M(S)}, n^\omega m^2\right\},$$

where $\mathcal{T}_{M(S)} = \mathcal{T}_{\rm mat}(n^2, nr_t, n^2)$ by Lemma E.8.

Then, by Corollary 9.57, we have

$$\sum_{t=1}^{T} \mathcal{T}_{\rm mat}(n^2, nr_t, n^2) = O^*\left(T \cdot n^{2\omega-\frac{1}{2}}\right).$$

Therefore, the amortized running time per iteration is

$$\min\left\{n^4 + n^{2\omega-\frac{1}{2}}, n^\omega m^2\right\} = \min\left\{n^{2\omega-\frac{1}{2}}, n^\omega m^2\right\}.$$

□

Remark E.2. The second term $n^\omega m^2$ in the running time represents the approach that does not use the maintenance technique. When $m = \Omega(n^{0.94})$, the first approach using low-rank update and amortization is faster.

Appendix F: Theoretical Analysis of Sparsely Activated Wide Neural Networks

This appendix is supplementary to Chapter 14.

F.1 Introduction

In this appendix, we follow the line of theoretical studies of sparsely trained overparameterized neural networks and address the two main research limitations in the previous studies (see e.g., Chapters 14 and 15).

1. The bias parameters used in the previous works are not trainable, contrary to what people are doing in practice.

2. The previous works only provided the convergence guarantee, while lacking the generalization performance which is of the central interest in deep learning theory.

Thus, our study will fill the above important gaps, by providing a comprehensive study of training one-hidden-layer sparsely activated neural networks in the NTK regime with (a) trainable biases incorporated in the analysis; (b) finer analysis of the convergence; and (c) first generalization bound for such sparsely activated neural networks after training with sharp bound on the restricted smallest eigenvalue of the limiting NTK. We further elaborate our technical contributions are follows:

1. **Convergence.** Surprisingly, Theorem F.1 shows that the network after sparsification can achieve as fast convergence as the original network. It further provides the required width to ensure that gradient descent can drive the training error towards zero at a linear rate. Our convergence result contains two novel ingredients compared to the existing study. (1) Our analysis handles trainable

bias, and shows that even though the biases are allowed to be updated from its initialization, the network's activation remains sparse during the entire training. This relies on our development of a new result showing that the change of bias is also diminishing with a $O(1/\sqrt{m})$ dependence on the network width m. (2) A finer analysis is provided such that the required network width to ensure the convergence can be much smaller, with an improvement upon the previous result by a factor of $\widetilde{\Theta}(n^{8/3})$ under appropriate bias initialization, where n is the sample size. This relies on our novel development of (1) a better characterization of the activation flipping probability via an analysis of the Gaussian anti-concentration based on the location of the strip and (2) a finer analysis of the initial training error.

2. **Generalization.** Theorem F.5 studies the generalization of the network after gradient descent training where we characterize how the network width should depend on activation sparsity, which lead to a sparsity-dependent localized Rademacher complexity and a generalization bound matching previous analysis (up to logarithmic factors). To our knowledge, this is the first sparsity-dependent generalization result via localized Rademacher complexity. In addition, compared with previous works, our result yields a better width's dependence by a factor of n^{10}. This relies on (1) the usage of symmetric initialization and (2) a finer analysis of the weight matrix change in Frobenius norm in Lemma F.8.

3. **Restricted Smallest Eigenvalue.** Theorem F.5 shows that the generalization bound heavily depends on the smallest eigenvalue $\lambda_{\min}$ of the limiting NTK. However, the previously known worst-case lower bounds on $\lambda_{\min}$ under data separation have a $1/n^2$ explicit dependence in [OS20, SYZ21], making the generalization bound vacuous. Instead, our Theorem F.7 establishes a much sharper lower bound restricted to a data-dependent region, which is sample-size-independent. This hence yields a desirable generalization bound that vanishes

as fast as $O(1/\sqrt{n})$, given that the label vector is in this region, which can be done with simple label-shifting.

F.1.1 Related works

A work related to ours is [LK22] where they also considered training a one-hidden-layer neural network with sparse activation and studied its convergence. However, different from our work, their sparsity is induced by sampling a random mask at each step of gradient descent whereas our sparsity is induced by non-zero initialization of the bias terms. Also, their network has no bias term, and they only focus on studying the training convergence but not generalization. We discuss additional related works here.

Training Overparameterized Neural Networks. Over the past few years, a tremendous amount of efforts have been made to study training overparameterized neural networks. A series of works have shown that if the neural network is wide enough (polynomial in depth, number of samples, etc), gradient descent can drive the training error towards zero in a fast rate either explicitly [DZPS19, DLL+19, JT19] or implicitly [AZLS19a, ZG19, ZCZG20] using the neural tangent kernel (NTK) [JGH18]. Further, under some conditions, the networks can generalize [CG19]. Under the NTK regime, the trained neural network can be well-approximated by its first order Taylor approximation from the initialization and [LZB20] showed that this transition to linearity phenomenon is a result from a diminishing Hessian 2-norm with respect to width. Later on, [FG21] and [LZB22] showed that closeness to initialization is sufficient but not necessary for gradient descent to achieve fast convergence as long as the non-linear system satisfies some variants of the Polyak-Łojasiewicz condition. On the other hand, although NTK offers good convergence explanation, it contradicts the practice since (1) the neural networks need to be unrealistically wide and (2) the neuron weights merely change from the initialization. As [COB19] pointed out, this "lazy training" regime can be explained by a mere effect of scaling. Other works have considered the mean-field limit [CB18, MMM19, CCGZ20], feature learning [AZL20, AZL22, SWL21, Tel22] which allow the weights to travel far away from the initialization.

Sparse Neural Networks in Practice. Besides finding a fixed sparse mask at the initialization as we mentioned in introduction, on the other hand, dynamic sparse training allows the sparse mask to be updated during training, e.g., [MMS+18, MW19, EGM+20, JPR+20, LCC+21, LMM+21, LYMP21].

F.2 Preliminaries

Notations. We use $\|\cdot\|_2$ to denote vector or matrix 2-norm and $\|\cdot\|_F$ to denote the Frobenius norm of a matrix. When the subscript of $\|\cdot\|$ is unspecified, it is default to be the 2-norm. For matrices $A \in \mathbb{R}^{m\times n_1}$ and $B \in \mathbb{R}^{m\times n_2}$, we use $[A, B]$ to denote the row concatenation of A, B and thus $[A, B]$ is a $m \times (n_1 + n_2)$ matrix. For matrix $X \in \mathbb{R}^{m\times n}$, the row-wise vectorization of X is denoted by $\vec{X} = [x_1, x_2, \ldots, x_m]^\top$ where x_i is the i-th row of X. For a given integer $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{0, \ldots, n\}$, i.e., the set of integers from 0 to n. For a set S, we use $\overline{S}$ to denote the complement of S. We use $\mathcal{N}(\mu, \sigma^2)$ to denote the Gaussian distribution with mean μ and standard deviation σ. In addition, we use $\widetilde{O}, \widetilde{\Theta}, \widetilde{\Omega}$ to suppress (poly-)logarithmic factors in O, Θ, Ω.

F.2.1 Problem formulation

Let the training set to be (X, y) where $X = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^{d\times n}$ denotes the feature matrix consisting of n d-dimensional vectors, and $y = (y_1, y_2, \ldots, y_n) \in \mathbb{R}^n$ consists of the corresponding n response variables. We assume $\|x_i\|_2 \le 1$ and $y_i = O(1)$ for all $i \in [n]$. We use one-hidden-layer neural network and consider the regression problem with the square loss function:

$$f(x;W,b) := \frac{1}{\sqrt{m}}\sum_{r=1}^{m} a_r \sigma(\langle w_r, x\rangle - b_r), \qquad L(W,b) := \frac{1}{2}\sum_{i=1}^{n}(f(x_i;W,b) - y_i)^2,$$

where $W \in \mathbb{R}^{m\times d}$ with its r-th row being w_r, $b \in \mathbb{R}^m$ is a vector with b_r being the bias of r-th neuron, a_r is the second layer weight, and $\sigma(\cdot)$ denotes the ReLU activation function. We initialize the neural network by $W_{r,i} \sim \mathcal{N}(0,1)$ and $a_r \sim \text{Uniform}(\{\pm 1\})$ and $b_r = B$ for some value $B \geq 0$ of choice, for all $r \in [m],\ i \in [d]$. We train only the parameters W and b (i.e., the linear layer a_r for $r \in [m]$ is not trained) via gradient descent, the update of which are given by

$$w_r(t+1) = w_r(t) - \eta\frac{\partial L(W(t), b(t))}{\partial w_r}, \qquad b_r(t+1) = b_r(t) - \eta\frac{\partial L(W(t), b(t))}{\partial b_r}.$$

By the chain rule, we have $\frac{\partial L}{\partial w_r} = \frac{\partial L}{\partial f}\frac{\partial f}{\partial w_r}$. The gradient of the loss with respect to the network is $\frac{\partial L}{\partial f} = \sum_{i=1}^{n}(f(x_i;W,b) - y_i)$ and the network gradients with respect to weights and bias are

$$\frac{\partial f(x;W,b)}{\partial w_r} = \frac{1}{\sqrt{m}} a_r x \mathbb{I}(w_r^\top x \geq b_r), \qquad \frac{\partial f(x;W,b)}{\partial b_r} = -\frac{1}{\sqrt{m}} a_r \mathbb{I}(w_r^\top x \geq b_r),$$

where $\mathbb{I}(\cdot)$ is the indicator function. We further define H as the **NTK** matrix of this network with

$$\begin{aligned} H_{i,j}(W,b) &:= \left\langle \frac{\partial f(x_i;W,b)}{\partial W}, \frac{\partial f(x_j;W,b)}{\partial W}\right\rangle + \left\langle \frac{\partial f(x_i;W,b)}{\partial b}, \frac{\partial f(x_j;W,b)}{\partial b}\right\rangle \\ &= \frac{1}{m}\sum_{r=1}^{m}(\langle x_i, x_j\rangle + 1)\mathbb{I}(w_r^\top x_i \geq b_r, w_r^\top x_j \geq b_r) \end{aligned} \tag{F.1}$$

and the **infinite-width version** $H^\infty(B)$ of the NTK matrix H is given by

$$H^\infty_{ij}(B) := \mathbb{E}_{w\sim\mathcal{N}(0,I)}\left[(\langle x_i, x_j\rangle + 1)\mathbb{I}(w^\top x_i \geq B, w^\top x_j \geq B)\right].$$

Let $\lambda(B) := \lambda_{\min}(H^\infty(B))$. We define $\mathbb{I}_{r,i}(W,b) := \mathbb{I}(w_r^\top x_i \geq b_r)$ and the matrix $Z(W,b)$ as

$$Z(W,b) := \frac{1}{\sqrt{m}}\begin{bmatrix} \mathbb{I}_{1,1}(W,b)a_1[x_1^\top, -1]^\top & \dots & \mathbb{I}_{1,n}(W,b)a_1[x_n^\top, -1]^\top \\ \vdots & \ddots & \vdots \\ \mathbb{I}_{m,1}(W,b)a_m[x_1^\top, -1]^\top & \dots & \mathbb{I}_{m,n}(W,b)a_m[x_n^\top, -1]^\top \end{bmatrix} \in \mathbb{R}^{m(d+1)\times n}.$$

Note that $H(W,b) = Z(W,b)^\top Z(W,b)$. Hence, the gradient descent step can be written as

$$[W,b]\vec{(t}+1) = [W,\vec{b}](t) - \eta Z(t)(f(t)-y)$$

where $[W,b](t) \in \mathbb{R}^{m\times(d+1)}$ denotes the row-wise concatenation of $W(t)$ and $b(t)$ at the t-th step of gradient descent, and $Z(t) := Z(W(t), b(t))$.

F.3 Main Theory

F.3.1 Convergence and sparsity

We present the convergence of gradient descent for the sparsely activated neural networks. Compared to the convergence result in Chapter 14, our study handles the trainable bias with constant initialization in the convergence analysis (which is the first of such a type). Also, our bound is sharper and yields a much smaller bound on the width of neural networks to guarantee the convergence.

Theorem F.1 (Convergence)**.** *Let the learning rate* $\eta \le O(\frac{\lambda(B)\exp(B^2)}{n^2})$*, and the bias initialization* $B \in [0, \sqrt{0.5\log m}]$*. Assume* $\lambda(B) = \lambda_0 \exp(-B^2/2)$ *for some* $\lambda_0 > 0$ *independent of* B*. Then, if the network width satisfies* $m \ge \widetilde{\Omega}\left(\lambda_0^{-4} n^4 \exp(B^2)\right)$*, over the randomness in the initialization,*

$$\mathbb{P}\left[\forall t : L(W(t), b(t)) \le (1-\eta\lambda(B)/4)^t L(W(0), b(0))\right] \ge 1 - \delta - e^{-\Omega(n)}.$$

This theorem show that the training loss decreases linearly, and its rate depends on the smallest eigenvalue of the NTK. The assumption on $\lambda(B)$ in Theorem F.1 can be justified by Theorem 14.31 which shows that under some mild conditions, the NTK's least eigenvalue $\lambda(B)$ is positive and has an $\exp(-B^2/2)$ dependence. This further implies that the network after sparsification can achieve as fast convergence as the original network.

Remark F.1. Theorem F.1 establishes a much sharper bound on the width of the neural network than previous work to guarantee the linear convergence. To elaborate, our bound only requires $m \geq \widetilde{\Omega}\left(\lambda_0^{-4} n^4 \exp(B^2)\right)$, as opposed to the bound $m \geq \widetilde{\Omega}(\lambda_0^{-4} n^4 B^2 \exp(2B^2))$ in Lemma 14.28. If we take $B = \sqrt{0.25 \log m}$ (as allowed by the theorem), then our lower bound yields a polynomial improvement by a factor of $\widetilde{\Theta}(n/\lambda_0)^{8/3}$, which implies that the neural network width can be much smaller to achieve the same linear convergence.

Key ideas in the proof of Theorem F.1. The proof mainly consists of developing a novel bound on activation flipping probability and a novel upper bound on initial error, as we elaborate below.

Like previous works, in order to prove convergence, we need to show that the NTK during training is close to its initialization. Inspecting the expression of NTK in Equation (F.1), observe that the training will affect the NTK by changing the output of each indicator function. We say that the r-th neuron flips its activation with respect to input x_i at the k-th step of gradient descent if $\mathbb{I}(w_r(k)^\top x_i - b_r(k) > 0) \neq \mathbb{I}(w_r(k-1)^\top x_i - b_r(k-1) > 0)$ for all $r \in [m]$. The central idea is that for each neuron, as long as the weight and bias movement R_w, R_b from its initialization is small, then the probability of activation flipping (with respect to random initialization) should not be large. We first present the bound on the probability that a given neuron flips its activation.

Lemma F.2 (Bound on Activation flipping probability)**.** *Let $B \geq 0$ and $R_w, R_b \leq \min\{1/B, 1\}$. Let $\tilde{W} = (\tilde{w}_1, \ldots, \tilde{w}_m)$ be vectors generated i.i.d. from $\mathcal{N}(0, I)$ and $\tilde{b} = (\tilde{b}_1, \ldots, \tilde{b}_m) = (B, \ldots, B)$, and weights $W = (w_1, \ldots, w_m)$ and biases $b = (b_1, \ldots, b_m)$ that satisfy for any $r \in [m]$, $\|\tilde{w}_r - w_r\|_2 \leq R_w$ and $|\tilde{b}_r - b_r| \leq R_b$. Define the event*

$$A_{i,r} = \{\exists w_r, b_r : \|\tilde{w}_r - w_r\|_2 \leq R_w,\ |b_r - \tilde{b}_r| \leq R_b,\ \mathbb{I}(x_i^\top \tilde{w}_r \geq \tilde{b}_r) \neq \mathbb{I}(x_i^\top w_r \geq b_r)\}.$$

Then, for some constant c,

$$\mathbb{P}\left[A_{i,r}\right] \leq c(R_w + R_b) \exp(-B^2/2).$$

Claim 14.19 presents a $O(\min\{R, \exp(-B^2/2)\})$ bound on $\mathbb{P}[A_{i,r}]$. The reason that their bound involving the min operation is because $\mathbb{P}[A_{i,r}]$ can be bounded by the standard Gaussian tail bound and Gaussian anti-concentration bound separately and then, take the one that is smaller. On the other hand, our bound replaces the min operation by the product which creates a more convenient (and tighter) interpolation between the two bounds. Later, we will show that the maximum movement of neuron weights and biases, R_w and R_b, both have a $O(1/\sqrt{m})$ dependence on the network width, and thus our bound offers a $\exp(-B^2/2)$ improvement where $\exp(-B^2/2)$ can be as small as $1/m^{1/4}$ when we take $B = \sqrt{0.5 \log m}$.

Proof idea of Lemma F.2. First notice that $\mathbb{P}[A_{i,r}] = \mathbb{P}_{x\sim\mathcal{N}(0,1)}[|x - B| \leq R_w + R_b]$. Thus, here we are trying to solve a fine-grained Gaussian anti-concentration problem with the strip centered at B. The problem with the standard Gaussian anti-concentration bound is that it only provides a worst case bound and, thus, is location-oblivious. Centered in our proof is a nice Gaussian anti-concentration bound based on the location of the strip, which we describe as follows: Let's first assume $B > R_w + R_b$. A simple probability argument yields a bound of $2(R_w+R_b)\frac{1}{\sqrt{2\pi}}\exp(-(B-R_w-R_b)^2)$. Since later in the Appendix we can show that R_w and R_b have a $O(1/\sqrt{m})$ dependence (Lemma F.15 bounds the movement for gradient descent and Lemma F.16 for gradient flow) and we only take $B = O(\sqrt{\log m})$, by making m sufficiently large, we can safely assume that R_w and R_b is sufficiently small. Thus, the probability can be bounded by $O((R_w + R_b)\exp(-B^2/2))$. However, when $B < R_w + R_b$ the above bound no longer holds. But a closer look tells us that in this case B is close to zero, and thus $(R_w + R_b)\frac{1}{\sqrt{2\pi}}\exp(-B^2/2) \approx \frac{R_w+R_b}{\sqrt{2\pi}}$ which yields roughly the same bound as the standard Gaussian anti-concentration.

Next, our proof of Theorem F.1 develops the following initial error bound.

Lemma F.3 (Initial error upper bound)**.** *Let $B > 0$ be the initialization value of the biases and all the weights be initialized from standard Gaussian. Let $\delta \in (0, 1)$ be the* *failure probability. Then, with probability at least $1-\delta$ over the randomness in the initialization, we have*

$$L(W(0), b(0)) = O\left(n + n\left(\exp(-B^2/2) + 1/m\right)\log^3(2mn/\delta)\right).$$

Claim 14.20 gives a rough estimate of the initial error with $O(n(1+B^2)\log^2(n/\delta)\log(m/\delta))$ bound. When we set $B = C\sqrt{\log m}$ for some constant C, our bound improves the previous result by a polylogarithmic factor. The previous bound is not tight in the following two senses: (1) the bias will only decrease the magnitude of the neuron activation instead of increasing and (2) when the bias is initialized as B, only roughly $O(\exp(-B^2/2)) \cdot m$ neurons will activate. Thus, we can improve the B^2 dependence to $\exp(-B^2/2)$.

By combining the above two improved results, we can prove our convergence result with improved lower bound of m as in Remark F.1. We provide the complete proof in Section F.6.

Lastly, since the total movement of each neuron's bias has a $O(1/\sqrt{m})$ dependence (shown in Lemma F.15), combining with the number of activated neurons at the initialization, we can show that during the entire training, the number of activated neurons is small.

Lemma F.4 (Number of Activated Neurons per Iteration)**.** *Assume the parameter settings in Theorem F.1. With probability at least $1-e^{-\Omega(n)}$ over the random initialization, we have*

$$|\mathcal{S}_{\text{on}}(i,t)| = O(m \cdot \exp(-B^2/2))$$

for all $0 \le t \le T$ and $i \in [n]$, where $\mathcal{S}_{\text{on}}(i,t) = \{r \in [m] : \ w_r(t)^\top x_i \ge b_r(t)\}$.

F.3.2 Generalization and restricted least eigenvalue

In this section, we present the sparsity-dependent generalization of our neural networks after gradient descent training. However, for technical reasons stated in

Section F.3.3, we use symmetric initialization defined below. Further, we adopt the setting in [ADH+19a] and use a non-degenerate data distribution to make sure the infinite-width NTK is positive definite.

Definition F.1 (Symmetric Initialization)**.** For a one-hidden layer neural network with $2m$ neurons, the network is initialized as the following:

1. For $r \in [m]$, independently initialize $w_r \sim \mathcal{N}(0, I)$ and $a_r \sim \text{Uniform}(\{-1, 1\})$.
2. For $r \in \{m+1, \ldots, 2m\}$, let $w_r = w_{r-m}$ and $a_r = -a_{r-m}$.

Definition F.2 ((λ_0, δ, n)-non-degenerate distribution, [ADH+19a])**.** A distribution $\mathcal{D}$ over $\mathbb{R}^d \times \mathbb{R}$ is (λ_0, δ, n)-non-degenerate, if for n i.i.d. samples $\{(x_i, y_i)\}_{i=1}^n$ from $\mathcal{D}$, with probability $1 - \delta$ we have $\lambda_{\min}(H^\infty(B)) \geq \lambda_0 > 0$.

Theorem F.5. *Fix a failure probability $\delta \in (0, 1)$ and an accuracy parameter $\epsilon \in (0, 1)$. Suppose the training data $S = \{(x_i, y_i)\}_{i=1}^n$ are i.i.d. samples from a (λ, δ, n)-non-degenerate distribution $\mathcal{D}$ defined in Definition F.2. Assume the one-hidden layer neural network is initialized by symmetric initialization in Definition F.1. Further, assume the parameter settings in Theorem F.1 except we let $m \geq \widetilde{\Omega}\left(\lambda(B)^{-6} n^6 \exp(-B^2)\right)$. Consider any loss function $\ell : \mathbb{R} \times \mathbb{R} \to [0, 1]$ that is 1-Lipschitz in its first argument. Then with probability at least $1 - 2\delta - e^{-\Omega(n)}$ over the randomness in symmetric initialization of $W(0) \in \mathbb{R}^{m \times d}$ and $a \in \mathbb{R}^m$ and the training samples, the two layer neural network $f(W(t), b(t), a)$ trained by gradient descent for $t \geq \Omega(\frac{1}{\eta\lambda(B)} \log \frac{n \log(1/\delta)}{\epsilon})$ iterations has empirical Rademacher complexity (see its formal definition in Definition F.6 in Appendix) bounded as*

$$\mathcal{R}_S(\mathcal{F}) \leq \sqrt{\frac{y^\top (H^\infty(B))^{-1} y \cdot 8 \exp(-B^2/2)}{n}} + \tilde{O}\left(\frac{\exp(-B^2/4)}{n^{1/2}}\right)$$

and the population loss $L_\mathcal{D}(f) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(f(x), y)]$ can be upper bounded as

$$L_\mathcal{D}(f(W(t), b(t), a)) \leq \sqrt{\frac{y^\top (H^\infty(B))^{-1} y \cdot 32 \exp(-B^2/2)}{n}} + \tilde{O}\left(\frac{1}{n^{1/2}}\right). \tag{F.2}$$

To show good generalization, we need a larger width: the second term in the Rademacher complexity bound is diminishing with m and to make this term $O(1/\sqrt{n})$, the width needs to have $(n/\lambda(B))^6$ dependence as opposed to $(n/\lambda(B))^4$ for convergence. Now, at the first glance of our generalization result, it seems we can make the Rademacher complexity arbitrarily small by increasing B. Recall from the discussion of Theorem F.1 that the smallest eigenvalue of $H^\infty(B)$ also has an $\exp(-B^2/2)$ dependence. Thus, in the worst case, the $\exp(-B^2/2)$ factor gets canceled and sparsity will not hurt the network's generalization.

Before we present the proof, we make a corollary of Theorem F.5 for the zero-initialized bias case.

Corollary F.6. *Take the same setting as in Theorem F.5 except now the biases are initialized as zero, i.e., $B = 0$. Then, if we let $m \geq \widetilde{\Omega}(\lambda(0)^{-6}n^6)$, the empirical Rademacher complexity and population loss are both bounded by*

$$\mathcal{R}_S(\mathcal{F}),\ L_{\mathcal{D}}(f(W(t), b(t), a)) \leq \sqrt{\frac{y^\top (H^\infty(0))^{-1} y \cdot 32}{n}} + \tilde{O}\left(\frac{1}{n^{1/2}}\right).$$

Corollary F.6 requires the network width $m \geq \widetilde{\Omega}((n/\lambda(0))^6)$ which significantly improves upon the previous result in [SY19, Theorem G.7] $m \geq \widetilde{\Omega}(n^{16}\mathrm{poly}(1/\lambda(0)))$ (including the dependence on the rescaling factor κ) which is a much wider network.

Generalization Bound via Least Eigenvalue. Note that in Theorem F.5, the worst case of the first term in the generalization bound in Equation (F.2) is given by $\widetilde{O}(\sqrt{1/(\lambda(B)\cdot n)})$. Hence, the least eigenvalue $\lambda(B)$ of the NTK matrix can significantly affect the generalization bound. Previous works [OS20, SYZ21] established lower bounds on $\lambda(B)$ with an explicit $1/n^2$ dependence on n under the δ data separation assumption (see Theorem F.7), which clearly makes a vacuous generalization bound of $\widetilde{O}(\sqrt{n})$. This thus motivates us to provide a tighter bound (desirably independent on n) on the least eigenvalue of the infinite-width NTK in order to make the generalization bound in Theorem F.5 valid and useful. However, it turns out that there are major difficulties in proving a better lower bound in the general case and thus, we are only able to present a better lower bound when we restrict the domain to some (data-dependent) regions.

Definition F.3 (Data-dependent Region)**.** Let $p_{ij} = \mathbb{P}_{w \sim \mathcal{N}(0,I)}[w^\top x_i \geq B,\ w^\top x_j \geq B]$ for $i \neq j$. Define the (data-dependent) region $\mathcal{R} = \{a \in \mathbb{R}^n : \sum_{i \neq j} a_i a_j p_{ij} \geq \min_{i' \neq j'} p_{i'j'} \sum_{i \neq j} a_i a_j\}$.

Notice that $\mathcal{R}$ is non-empty for any input data-set since $\mathbb{R}^n_+ \subset \mathcal{R}$ where $\mathbb{R}^n_+$ denotes the set of vectors with non-negative entries, and $\mathcal{R} = \mathbb{R}^n$ if $p_{ij} = p_{i'j'}$ for all $i \neq i', j \neq j'$.

Theorem F.7 (Restricted Least Eigenvalue)**.** *Let $X = (x_1, \ldots, x_n)$ be points in $\mathbb{R}^d$ with $\|x_i\|_2 = 1$ for all $i \in [n]$ and $w \sim \mathcal{N}(0, I_d)$. Suppose that there exists $\delta \in [0, \sqrt{2}]$ such that*

$$\min_{i \neq j \in [n]} (\|x_i - x_j\|_2, \|x_i + x_j\|_2) \geq \delta.$$

Let $B \geq 0$. Consider the minimal eigenvalue of H^∞ over the data-dependent region $\mathcal{R}$ defined above, i.e., let $\lambda := \min_{\|a\|_2=1,\ a \in \mathcal{R}} a^\top H^\infty a$. Then, $\lambda \geq \max(0, \lambda')$ where

$$\lambda' \geq \max\left(\frac{1}{2} - \frac{B}{\sqrt{2\pi}},\ \left(\frac{1}{B} - \frac{1}{B^3}\right)\frac{e^{-B^2/2}}{\sqrt{2\pi}}\right) - e^{-B^2/(2-\delta^2/2)} \frac{\pi - \arctan\left(\frac{\delta\sqrt{1-\delta^2/4}}{1-\delta^2/2}\right)}{2\pi}. \tag{F.3}$$

To demonstrate the usefulness of our result, if we take the bias initialization $B = 0$ in Equation (F.3), this bound yields $1/(2\pi) \cdot \arctan((\delta\sqrt{1-\delta^2/4})/(1-\delta^2/2)) \approx \delta/(2\pi)$, when δ is close to 0 whereas [SYZ21] yields a bound of δ/n^2. On the other hand, if the data has maximal separation, i.e., $\delta = \sqrt{2}$, we get a $\max\left(\frac{1}{2} - \frac{B}{\sqrt{2\pi}},\ \left(\frac{1}{B} - \frac{1}{B^3}\right)\frac{e^{-B^2/2}}{\sqrt{2\pi}}\right)$ lower bound, whereas [SYZ21] yields a bound of $\exp(-B^2/2)\sqrt{2}/n^2$. Connecting to our convergence result in Theorem F.1, if $f(t) - y \in \mathcal{R}$, then the error can be reduced at a much faster rate than the (pessimistic) rate with $1/n^2$ dependence in the previous studies as long as the error vector lies in the region.

Remark F.2. The lower bound on the restricted smallest eigenvalue λ in Theorem F.7 is **independent on** n, which makes that the generalization bound in Theorem F.5 vanishes as fast as $O(1/\sqrt{n})$. Such a lower bound is much sharper than the previous results with a $1/n^2$ explicit dependence which yields vacuous generalization. This improvement relies on a fact that the label vector should lie in the region $\mathcal{R}$, which can be justified by a simple label-shifting strategy as follows. Since $\mathbb{R}^n_+ \subset \mathcal{R}$, the condition can be easily achieved by training the neural network on the shifted labels $y + C$ (with appropriate broadcast) where C is a constant such that $\min_i y_i + C \geq 0$.

Careful readers may notice that in the proof of Theorem F.7 in Section F.7, the restricted least eigenvalue on $\mathbb{R}^n_+$ is always positive even if the data separation is zero. However, we would like to point out that the generalization bound in Theorem F.5 is meaningful only when the training is successful: when the data separation is zero, the limiting NTK is no longer positive definite and the training loss cannot be minimized toward zero.

F.3.3 Key ideas in the proof of Theorem F.5

Since each neuron weight and bias move little from their initialization, a natural approach is to bound the generalization via localized Rademacher complexity. After that, we can apply appropriate concentration bounds to derive generalization. The main effort of our proof is devoted to bounding the weight movement to bound the localized Rademacher complexity. If we directly take the setting in Theorem F.1 and compute the network's localized Rademacher complexity, we will encounter a non-diminishing (with the number of samples n) term which can be as large as $O(\sqrt{n})$ since the network outputs non-zero values at the initialization. [ADH$^+$19a] and [SY19] resolved this issue by initializing the neural network weights instead by $\mathcal{N}(0, \kappa^2 I)$ to force the neural network output something close to zero at the initialization. The magnitude of κ is chosen to balance different terms in the Rademacher complexity bound in the end. Similar approach can also be adapted to our case by initializing the weights by $\mathcal{N}(0, \kappa^2 I)$ and the biases by κB. However, the drawback of such an approach is that the effect of κ to all the previously established results for convergence need to be carefully tracked or derived. In particular, in order to guarantee convergence, the neural network's width needs to have a polynomial dependence on $1/\kappa$ where $1/\kappa$ has a polynomial dependence on n and $1/\lambda$, which means their network width needs to be larger to compensate for the initialization scaling. We resolve this issue by symmetric initialization Definition F.1 which yields no effect (up to constant factors) on previously established convergence results, see [MOSW22]. Symmetric initialization allows us to organically combine the results derived for convergence to be reused for generalization, which leads to a more succinct analysis. Further, we replace the ℓ_1-ℓ_2 norm upper bound by finer inequalities in various places in the original analysis. All these improvements lead to the following upper bound of the weight matrix change in Frobenius norm. Further, combining our sparsity-inducing initialization, we present our sparsity-dependent Frobenius norm bound on the weight matrix change.

Lemma F.8. *Assume the one-hidden layer neural network is initialized by symmetric initialization in Definition F.1. Further, assume the parameter settings in Theorem F.1. Then with probability at least $1-\delta-e^{-\Omega(n)}$ over the random initialization, we have for all $t \geq 0$,*

$$\|[W,b](t)-[W,b](0)\|_F \leq \sqrt{y^\top (H^\infty)^{-1} y} + O\left(\frac{n}{\lambda}\left(\frac{\exp(-B^2/2)\log(n/\delta)}{m}\right)^{1/4}\right)$$
$$+ O\left(\frac{n\sqrt{R\exp(-B^2/2)}}{\lambda}\right) + \frac{n}{\lambda^2}\cdot O\left(\exp(-B^2/4)\sqrt{\frac{\log(n^2/\delta)}{m}} + R\exp(-B^2/2)\right)$$

where $R = R_w + R_b$ denote the maximum magnitude of neuron weight and bias change.

By Lemma F.15 and Lemma F.17 in the Appendix, we have $R = \widetilde{O}(\frac{n}{\lambda\sqrt{m}})$. Plugging in and setting $B=0$, we get $\|[W,b](t)-[W,b](0)\|_F \leq \sqrt{y^\top (H^\infty)^{-1} y} + \widetilde{O}(\frac{n}{\lambda m^{1/4}} + \frac{n^{3/2}}{\lambda^{3/2} m^{1/4}} + \frac{n}{\lambda^2\sqrt{m}} + \frac{n^2}{\lambda^3\sqrt{m}})$. On the other hand, taking $\kappa = 1$, [SY19, Lemma G.6] yields a bound of $\|W(t)-W(0)\|_F \leq \sqrt{y^\top (H^\infty)^{-1} y} + \widetilde{O}(\frac{n}{\lambda} + \frac{n^{7/2}\text{poly}(1/\lambda)}{m^{1/4}})$. Notice that the $\widetilde{O}(\frac{n}{\lambda})$ term has no dependence on $1/m$ and is removed by symmetric initialization in our analysis and we improve the upper bound's dependence on n by a factor of n^2.

We defer the full proof of Theorem F.5 and Lemma F.8 to Section F.8.

F.3.4 Key ideas in the proof of Theorem F.7

In this section, we analyze the smallest eigenvalue $\lambda := \lambda_{\min}(H^\infty)$ of the limiting NTK H^∞ with δ data separation. We first note that $H^\infty \succeq \mathbb{E}_{w\sim\mathcal{N}(0,I)}\left[\mathbb{I}(Xw \geq B)\mathbb{I}(Xw \geq B)^\top\right]$ and for a fixed vector a, we are interested in the lower bound of $\mathbb{E}_{w\sim\mathcal{N}(0,I)}[|a^\top \mathbb{I}(Xw \geq B)|^2]$. In previous works, [OS20] showed a lower bound $\Omega(\delta/n^2)$ for zero-initialized bias, and later [SYZ21] generalized this result to a lower bound $\Omega(e^{-B^2/2}\delta/n^2)$ for non-zero initialized bias. Both lower bounds have a dependence of $1/n^2$. Their approach is by using an intricate Markov's inequality argument and then proving an lower bound of $\mathbb{P}[|a^\top \mathbb{I}(Xw \geq B)| \geq c\,\|a\|_\infty]$. The lower bound is proved by only considering the contribution from the largest coordinate of a and treating all other values as noise. It is non-surprising that the lower bound has a factor of $1/n$ since a can have identical entries. On the other hand, the diagonal entries can give a $\exp(-B^2/2)$ upper bound and thus there is a $1/n^2$ gap between the two. Now, we give some evidence suggesting the $1/n^2$ dependence may not be tight in some cases. Consider the following scenario: Assume $n \ll d$ and the data set is orthonormal. For a fixed a, we have

$$\begin{aligned} & a^\top \mathbb{E}_{w\sim\mathcal{N}(0,I)}\left[\mathbb{I}(Xw \geq B)\mathbb{I}(Xw \geq B)^\top\right] a \\ &\qquad = \textstyle\sum_{i,j\in[n]} a_i a_j \mathbb{P}[w^\top x_i \geq B,\ w^\top x_j \geq B] = p_0 \|a\|_2^2 + p_1 \sum_{i\neq j} a_i a_j \\ &\qquad = p_0 - p_1 + p_1 \left(\textstyle\sum_i a_i\right)^2 > p_0 - p_1 \end{aligned}$$

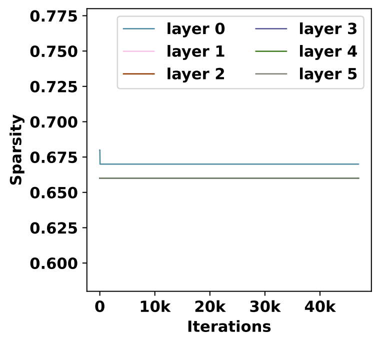
where $p_0, p_1 \in [0,1]$ are defined such that due to the spherical symmetry of the standard Gaussian we are able to let $p_0 = \mathbb{P}[w^\top x_i \geq B]$, $\forall i \in [n]$ and $p_1 = \mathbb{P}[w^\top x_i \geq B, w^\top x_j \geq B]$, $\forall i,j \in [n]$, $i \neq j$. Notice that $p_0 > p_1$. Since this is true for all $a \in \mathbb{R}^n$, we get a lower bound of $p_0 - p_1$ with no explicit dependence on n and this holds for all $n \leq d$. When d is large and $n = d/2$, this bound is better than previous bound by a factor of $\Theta(1/d^2)$. However, it turns out that the product terms with $i \neq j$ above creates major difficulties in analyzing the general case. Due to such technical difficulties, we are only able to prove a better lower bound by utilizing the extra constant factor in the NTK thanks to the trainable bias, when we restrict the domain to some data-dependent region. We defer the proof of Theorem F.7 to Section F.7.

F.4 Experiments

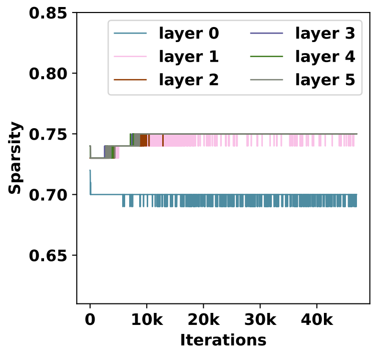
In this section, we study how the activation sparsity patterns of multi-layer neural networks change during training when the bias parameters are initialized as non-zero.

Settings. We train a 6-layer multi-layer perceptron (MLP) of width 1024 with trainable bias terms on MNIST image classification [LCB10]. The biases of the fully-connected layers are initialized as $0, -0.5$ and -1. For the weights in the linear layer, we use Kaiming Initialization [HZRS15] which is sampled from an appropriately scaled Gaussian distribution. The traditional MLP architecture only has linear layers with ReLU activation. However, we found out that using the sparsity-inducing initialization, the magnitude of the activation will decrease geometrically layer-by-layer, which leads to vanishing gradients and that the network cannot be trained. Thus, we made a slight modification to the MLP architecture to include an extra Batch Normalization after ReLU to normalize the activation. Our MLP implementation is based on [ZDZ+21]. We train the neural network by stochastic gradient descent with a small learning rate 5e-3 to make sure the training is in the NTK regime. The sparsity is measured as the total number of activated neurons (i.e., ReLU outputs some positive values) divided by total number of neurons, averaged over every SGD batch. We plot how the sparsity patterns changes for different layers during training.

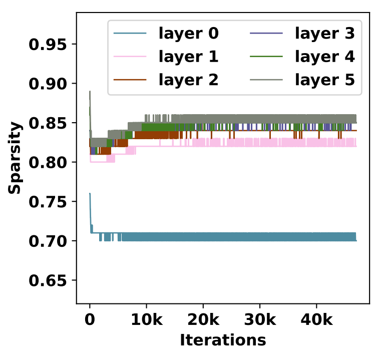
Observation and Implication. As demonstrated at Figure F.1, when we initialize the bias with three different values, the sparsity patterns are stable across all layers



(a) Init Bias as 0.



(b) Init Bias as -0.5.



(c) Init Bias as -1.0.

Figure F.1: Sparsity pattern on different layers across different training iterations for three different bias initialization. The x and y axis denote the iteration number and sparsity level, respectively. The models can achieve $97.9\%, 97.7\%$ and 97.3% accuracy after training, respectively. Note that, in Figure (a), the lines of layers 1-5 overlap together except layer 0.

during training: when the bias is initialized as 0 and -0.5, the sparsity change is within 2.5%; and when the bias is initialized as -1.0, the sparsity change is within 10%. Meanwhile, by increasing the initialization magnitude for bias, the sparsity level increases with only marginal accuracy dropping. This implies that our theory can be extended to the multi-layer setting (with some extra care for coping with vanishing gradient) and multi-layer neural networks can also benefit from the sparsity-inducing initialization and enjoy reduction of computational cost. Another interesting observation is that the input layer (layer 0) has a different sparsity pattern from other layers while all the rest layers behave similarly.

F.5 Discussion

In this chapter, we study training one-hidden-layer overparameterized ReLU networks in the NTK regime with its biases being trainable and initialized as some constants rather than zero. We showed sparsity-dependent results on convergence, restricted least eigenvalue, and generalization. A future direction is to generalize our analysis to multi-layer neural networks. In practice, label shifting is unnecessary for achieving good generalization. An open problem is whether it is possible to improve the dependence on the sample size of the lower bound of the infinite-width NTK's least eigenvalue, or even whether a lower bound purely dependent on the data separation is possible so that the generalization bound is no longer vacuous for all labels.

F.6 Convergence

Notation simplification. Since the smallest eigenvalue of the limiting NTK appeared in this proof all has dependence on the bias initialization parameter B, for the ease of notation of our proof, we suppress its dependence on B and use λ to denote $\lambda := \lambda(B) = \lambda_{\min}(H^\infty(B))$.

F.6.1 Difference between limit NTK and sampled NTK

Lemma F.9. *For a given bias vector $b \in \mathbb{R}^m$ with $b_r \geq 0,\ \forall r \in [m]$, the limit NTK H^∞ and the sampled NTK H are given as*

$$\begin{aligned} H^\infty_{ij} &:= \mathbb{E}_{w\sim\mathcal{N}(0,I)}\left[(\langle x_i, x_j\rangle + 1)\mathbb{I}(w_r^\top x_i \geq b_r, w_r^\top x_j \geq b_r)\right], \\ H_{ij} &:= \frac{1}{m}\sum_{r=1}^m (\langle x_i, x_j\rangle + 1)\mathbb{I}(w_r^\top x_i \geq b_r, w_r^\top x_j \geq b_r). \end{aligned}$$

Let's define $\lambda := \lambda_{\min}(H^\infty)$ and assume $\lambda > 0$. If the network width $m = \Omega(\lambda^{-1} n \cdot \log(n/\delta))$, then

$$\mathbb{P}\left[\lambda_{\min}(H) \geq \frac{3}{4}\lambda\right] \geq 1 - \delta.$$

Proof. Let $H_r := \frac{1}{m}\widetilde{X}(w_r)^\top \widetilde{X}(w_r)$, where $\widetilde{X}(w_r) \in \mathbb{R}^{(d+1)\times n}$ is defined as

$$\widetilde{X}(w_r) := [\mathbb{I}(w_r^\top x_1 \geq b) \cdot (x_1, 1), \ldots, \mathbb{I}(w_r^\top x_n \geq b) \cdot (x_n, 1)],$$

where $(x_i, 1)$ denotes appending the vector x_i by 1. Hence $H_r \succeq 0$. Since for each entry H_{ij} we have

$$(H_r)_{ij} = \frac{1}{m}(\langle x_i, x_j\rangle + 1)\mathbb{I}(w_r^\top x_i \geq b_r, w_r^\top x_j \geq b_r) \leq \frac{1}{m}(\langle x_i, x_j\rangle + 1) \leq \frac{2}{m},$$

and naively, we can upper bound $\|H_r\|_2$ by:

$$\|H_r\|_2 \leq \|H_r\|_F \leq \sqrt{n^2 \frac{4}{m^2}} = \frac{2n}{m}.$$

Then $H = \sum_{r=1}^m H_r$ and $\mathbb{E}[H] = H^\infty$. Hence, by the Matrix Chernoff Bound in Theorem A.6 and choosing $m = \Omega(\lambda^{-1} n \cdot \log(n/\delta))$, we can show that

$$\begin{aligned}\mathbb{P}\left[\lambda_{\min}(H) \leq \frac{3}{4}\lambda\right] &\leq n \cdot \exp\left(-\frac{1}{16}\lambda/(4n/m)\right) \\ &= n \cdot \exp\left(-\frac{\lambda m}{64n}\right) \\ &\leq \delta.\end{aligned}$$

□

Lemma F.10. *Assume $m = n^{O(1)}$ and $\exp(B^2/2) = O(\sqrt{m})$ where we recall that B is the initialization value of the biases. With probability at least $1 - \delta$, we have $\|H(0) - H^\infty\|_F \leq 4n \exp(-B^2/4)\sqrt{\frac{\log(n^2/\delta)}{m}}$.*

Proof. First, we have $\mathbb{E}[((\langle x_i, x_j\rangle + 1)\mathbb{I}_{r,i}(0)\mathbb{I}_{r,j}(0))^2] \leq 4\exp(-B^2/2)$. Then, by Bernstein's inequality in Lemma A.4, with probability at least $1 - \delta/n^2$,

$$|H_{ij}(0) - H_{ij}^\infty| \leq 2\exp(-B^2/4)\sqrt{2\frac{\log(n^2/\delta)}{m}} + 2\frac{2}{m}\log(n^2/\delta) \leq 4\exp(-B^2/4)\sqrt{\frac{\log(n^2/\delta)}{m}}.$$

By a union bound, the above holds for all $i, j \in [n]$ with probability at least $1 - \delta$, which implies

$$\|H(0) - H^\infty\|_F \leq 4n \exp(-B^2/4)\sqrt{\frac{\log(n^2/\delta)}{m}}.$$

□

F.6.2 Bounding the number of flipped neurons

Definition F.4 (No-flipping set)**.** For each $i \in [n]$, let $S_i \subset [m]$ denote the set of neurons that are never flipped during the entire training process,

$$S_i := \{r \in [m] : \ \forall t \in [T] \ \mathrm{sign}(\langle w_r(t), x_i\rangle - b_r(t)) = \mathrm{sign}(\langle w_r(0), x_i\rangle - b_r(0))\}.$$

Thus, the flipping set is $\overline{S}_i$ for $i \in [n]$.

Lemma F.11 (Bound on flipping probability)**.** *Let $B \geq 0$ and $R_w, R_b \leq \min\{1/B, 1\}$. Let $\widetilde{W} = (\widetilde{w}_1, \ldots, \widetilde{w}_m)$ be vectors generated i.i.d. from $\mathcal{N}(0, I)$ and $\tilde{b} = (\tilde{b}_1, \ldots, \tilde{b}_m) = (B, \ldots, B)$, and weights $W = (w_1, \ldots, w_m)$ and biases $b = (b_1, \ldots, b_m)$ that satisfy for any $r \in [m]$, $\|\tilde{w}_r - w_r\|_2 \leq R_w$ and $|\tilde{b}_r - b_r| \leq R_b$. Define the event*

$$A_{i,r} = \{\exists w_r, b_r : \|\tilde{w}_r - w_r\|_2 \leq R_w,\ |b_r - \tilde{b}_r| \leq R_b,\ \mathbb{I}(x_i^\top \tilde{w}_r \geq \tilde{b}_r) \neq \mathbb{I}(x_i^\top w_r \geq b_r)\}.$$

Then,

$$\mathbb{P}\left[A_{i,r}\right] \leq c(R_w + R_b)\exp(-B^2/2)$$

for some constant c.

Proof. Notice that the event $A_{i,r}$ happens if and only if $|\tilde{w}_r^\top x_i - \tilde{b}_r| < R_w + R_b$. First, if $B > 1$, then by Lemma A.8, we have

$$\mathbb{P}\left[A_{i,r}\right] \leq (R_w + R_b)\frac{1}{\sqrt{2\pi}}\exp(-(B - R_w - R_b)^2/2) \leq c_1(R_w + R_b)\exp(-B^2/2)$$

for some constant c_1. If $0 \leq B < 1$, then the above analysis doesn't hold since it is possible that $B - R_w - R_b \leq 0$. In this case, the probability is at most $\mathbb{P}[A_{i,r}] \leq 2(R_w + R_b)\frac{1}{\sqrt{2\pi}}\exp(-0^2/2) = \frac{2(R_w+R_b)}{\sqrt{2\pi}}$. However, since $0 \leq B < 1$ in this case, we have $\exp(-1^2/2) \leq \exp(-B^2/2) \leq \exp(-0^2/2)$. Therefore, $\mathbb{P}[A_{i,r}] \leq c_2(R_w + R_b)\exp(-B^2/2)$ for $c_2 = \frac{2\exp(1/2)}{\sqrt{2\pi}}$. Take $c = \max\{c_1, c_2\}$ finishes the proof. $\square$

Corollary F.12. *Let $B > 0$ and $R_w, R_b \leq \min\{1/B, 1\}$. Assume that $\|w_r(t) - w_r(0)\|_2 \leq R_w$ and $|b_r(t) - b_r(0)| \leq R_b$ for all $t \in [T]$. For $i \in [n]$, the flipping set $\overline{S}_i$ satisfies that*

$$\mathbb{P}[r \in \overline{S}_i] \leq c(R_w + R_b)\exp(-B^2/2)$$

for some constant c, which implies

$$\mathbb{P}[\forall i \in [n]:\ |\overline{S}_i| \leq 2mc(R_w + R_b)\exp(-B^2/2)] \geq 1 - n \cdot \exp\left(-\frac{2}{3}mc(R_w + R_b)\exp(-B^2/2)\right).$$

Proof. The proof is by observing that $\mathbb{P}[r \in \overline{S}_i] \le \mathbb{P}[A_{i,r}]$. Then, by Bernstein's inequality,

$$\mathbb{P}[|\overline{S}_i| > t] \le \exp\left(-\frac{t^2/2}{mc(R_w + R_b)\exp(-B^2/2) + t/3}\right).$$

Take $t = 2mc(R_w + R_b)\exp(-B^2/2)$ and a union bound over $[n]$, we have

$$\mathbb{P}[\forall i \in [n] :\ |\overline{S}_i| \le 2mc(R_w + R_b)\exp(-B^2/2)] \ge 1 - n \cdot \exp\left(-\frac{2}{3}mc(R_w + R_b)\exp(-B^2/2)\right).$$

□

F.6.3 Bounding NTK if perturbing weights and biases

Lemma F.13. *Assume $\lambda > 0$. Let $B > 0$ and $R_b, R_w \le \min\{1/B, 1\}$. Let $\tilde{W} = (\tilde{w}_1, \ldots, \tilde{w}_m)$ be vectors generated i.i.d. from $\mathcal{N}(0, I)$ and $\tilde{b} = (\tilde{b}_1, \ldots, \tilde{b}_m) = (B, \ldots, B)$. For any set of weights $W = (w_1, \ldots, w_m)$ and biases $b = (b_1, \ldots, b_m)$ that satisfy for any $r \in [m]$, $\|\tilde{w}_r - w_r\|_2 \le R_w$ and $|\tilde{b}_r - b_r| \le R_b$, we define the matrix $H(W, b) \in \mathbb{R}^{n\times n}$ by*

$$H_{ij}(W, b) = \frac{1}{m}\sum_{r=1}^{m}(\langle x_i, x_j\rangle + 1)\mathbb{I}(w_r^\top x_i \ge b_r, w_r^\top x_j \ge b_r).$$

It satisfies that for some small positive constant c,

1. *With probability at least $1 - n^2 \exp\left(-\frac{2}{3}cm(R_w + R_b)\exp(-B^2/2)\right)$, we have*

$$\left\|H(\tilde{W}, \tilde{b}) - H(W, b)\right\|_F \le n \cdot 8c(R_w + R_b)\exp(-B^2/2),$$
$$\left\|Z(\tilde{W}, \tilde{b}) - Z(W, b)\right\|_F \le \sqrt{n \cdot 8c(R_w + R_b)\exp(-B^2/2)}.$$

2. *With probability at least $1 - \delta - n^2 \exp\left(-\frac{2}{3}cm(R_w + R_b)\exp(-B^2/2)\right)$,*

$$\lambda_{\min}(H(W, b)) > 0.75\lambda - n \cdot 8c(R_w + R_b)\exp(-B^2/2).$$

Proof. We have

$$\begin{aligned}\left\|Z(W,b) - Z(\tilde{W},\tilde{b})\right\|_F^2 &= \sum_{i\in[n]} \left(\frac{2}{m} \sum_{r\in[m]} \left(\mathbb{I}(w_r^\top x_i \geq b_r) - \mathbb{I}(\tilde{w}_r^\top x_i \geq \tilde{b}_r) \right)^2 \right) \\ &= \sum_{i\in[n]} \left(\frac{2}{m} \sum_{r\in[m]} t_{r,i} \right)\end{aligned}$$

and

$$\begin{aligned}&\left\|H(W,b) - H(\tilde{W},\tilde{b})\right\|_F^2 \\ &= \sum_{i\in[n],\ j\in[n]} (H_{ij}(W,b) - H_{ij}(\tilde{W},\tilde{b}))^2 \\ &\leq \frac{4}{m^2} \sum_{i\in[n],\ j\in[n]} \left(\sum_{r\in[m]} |\mathbb{I}(w_r^\top x_i \geq b_r, w_r^\top x_j \geq b_r) - \mathbb{I}(\tilde{w}_r^\top x_i \geq \tilde{b}_r, \tilde{w}_r^\top x_j \geq \tilde{b}_r)| \right)^2 \\ &= \frac{4}{m^2} \sum_{i,j\in[n]} \left(\sum_{r\in[m]} s_{r,i,j} \right)^2,\end{aligned}$$

where we define

$$\begin{aligned}s_{r,i,j} &:= |\mathbb{I}(w_r^\top x_i \geq b_r, w_r^\top x_j \geq b_r) - \mathbb{I}(\tilde{w}_r^\top x_i \geq \tilde{b}_r, \tilde{w}_r^\top x_j \geq \tilde{b}_r)|, \\ t_{r,i} &:= (\mathbb{I}(w_r^\top x_i \geq b_r) - \mathbb{I}(\tilde{w}_r^\top x_i \geq \tilde{b}_r))^2.\end{aligned}$$

Notice that $t_{r,i} = 1$ only if the event $A_{i,r}$ happens (recall the definition of $A_{i,r}$ in Lemma F.11) and $s_{r,i,j} = 1$ only if the event $A_{i,r}$ or $A_{j,r}$ happens. Thus,

$$\sum_{r\in[m]} t_{r,i} \leq \sum_{r\in[m]} \mathbb{I}(A_{i,r}), \quad \sum_{r\in[m]} s_{r,i,j} \leq \sum_{r\in[m]} \mathbb{I}(A_{i,r}) + \mathbb{I}(A_{j,r}).$$

By Lemma F.11, we have

$$\mathbb{E}_{\tilde{w}_r}[s_{r,i,j}] \leq \mathbb{E}_{\tilde{w}_r}[s_{r,i,j}^2] \leq \mathbb{P}_{\tilde{w}_r}[A_{i,r}] + \mathbb{P}_{\tilde{w}_r}[A_{j,r}] \leq 2c(R_w + R_b)\exp(-B^2/2).$$

Define $s_{i,j} = \sum_{r=1}^m \mathbb{I}(A_{i,r}) + \mathbb{I}(A_{j,r})$. By Bernstein's inequality in Lemma A.4,

$$\begin{aligned}&\mathbb{P}\left[s_{i,j} \geq m \cdot 2c(R_w + R_b)\exp(-B^2/2) + mt\right] \\ &\leq \exp\left(-\frac{m^2t^2/2}{m \cdot 2c(R_w + R_b)\exp(-B^2/2) + mt/3} \right), \quad \forall t \geq 0.\end{aligned}$$

Let $t = 2c(R_w + R_b)\exp(-B^2/2)$. We get

$$\mathbb{P}[s_{i,j} \geq m \cdot 4c(R_w + R_b)\exp(-B^2/2)] \leq \exp\left(-\frac{2}{3}cm(R_w + R_b)\exp(-B^2/2)\right).$$

Thus, we obtain with probability at least $1 - n^2 \exp\left(-\frac{2}{3}cm(R_w + R_b)\exp(-B^2/2)\right)$,

$$\left\|H(\tilde{W}, \tilde{b}) - H(W, b)\right\|_F \leq n \cdot 8c(R_w + R_b)\exp(-B^2/2),$$
$$\left\|Z(\tilde{W}, \tilde{b}) - Z(W, b)\right\|_F \leq \sqrt{n \cdot 8c(R_w + R_b)\exp(-B^2/2)}.$$

For the second result, by Lemma F.9, $\mathbb{P}[\lambda_{\min}(H(\tilde{W}, \tilde{b})) \geq 0.75\lambda] \geq 1 - \delta$. Hence, with probability at least $1 - \delta - n^2 \exp\left(-\frac{2}{3}cm(R_w + R_b)\exp(-B^2/2)\right)$,

$$\begin{aligned}\lambda_{\min}(H(W, b)) &\geq \lambda_{\min}(H(\tilde{W}, \tilde{b})) - \left\|H(W, b) - H(\tilde{W}, \tilde{b})\right\| \\ &\geq \lambda_{\min}(H(\tilde{W}, \tilde{b})) - \left\|H(W, b) - H(\tilde{W}, \tilde{b})\right\|_F \\ &\geq 0.75\lambda - n \cdot 8c(R_w + R_b)\exp(-B^2/2).\end{aligned}$$

□

F.6.4 Total movement of weights and biases

Definition F.5 (NTK at time t). For $t \geq 0$, let $H(t)$ be an $n \times n$ matrix with (i, j)-th entry

$$H_{ij}(t) := \left\langle \frac{\partial f(x_i; \theta(t))}{\partial \theta(t)}, \frac{\partial f(x_j; \theta(t))}{\partial \theta(t)} \right\rangle = \frac{1}{m}\sum_{r=1}^{m}(\langle x_i, x_j\rangle + 1)\mathbb{I}(w_r(t)^\top x_i \geq b_r(t), w_r(t)^\top x_j \geq b_r(t)).$$

We follow the proof strategy from [DZPS19]. Now we derive the total movement of weights and biases. Let $f(t) = f(X; \theta(t))$ where $f_i(t) = f(x_i; \theta(t))$. The dynamics of each prediction is given by

$$\begin{aligned}\frac{d}{dt} f_i(t) &= \left\langle \frac{\partial f(x_i; \theta(t))}{\partial \theta(t)}, \frac{d\theta(t)}{dt} \right\rangle \\ &= \sum_{j=1}^{n}(y_j - f_j(t)) \left\langle \frac{\partial f(x_i; \theta(t))}{\partial \theta(t)}, \frac{\partial f(x_j; \theta(t))}{\partial \theta(t)} \right\rangle \\ &= \sum_{j=1}^{n}(y_j - f_j(t)) H_{ij}(t),\end{aligned}$$

which implies

$$\frac{d}{dt}f(t) = H(t)(y - f(t)). \tag{F.4}$$

Lemma F.14 (Gradient Bounds)**.** *For any* $0 \le s \le t$*, we have*

$$\left\| \frac{\partial L(W(s), b(s))}{\partial w_r(s)} \right\|_2 \le \sqrt{\frac{n}{m}} \left\| f(s) - y \right\|_2 ,$$
$$\left\| \frac{\partial L(W(s), b(s))}{\partial b_r(s)} \right\|_2 \le \sqrt{\frac{n}{m}} \left\| f(s) - y \right\|_2 .$$

Proof. We have:

$$\begin{aligned}
\left\| \frac{\partial L(W(s), b(s))}{\partial w_r(s)} \right\|_2 &= \left\| \frac{1}{\sqrt{m}} \sum_{i=1}^{n} (f(x_i; W(s), b(s)) - y_i) a_r x_i \mathbb{I}(w_r(s)^\top x_i \ge b_r) \right\|_2 \\
&\le \frac{1}{\sqrt{m}} \sum_{i=1}^{n} |f(x_i; W(s), b(s)) - y_i| \\
&\le \sqrt{\frac{n}{m}} \left\| f(s) - y \right\|_2 ,
\end{aligned}$$

where the first inequality follows from triangle inequality, and the second inequality follows from Cauchy-Schwarz inequality.

Similarly, we also have:

$$\begin{aligned}
\left\| \frac{\partial L(W(s), b(s))}{\partial b_r(s)} \right\|_2 &= \left\| \frac{1}{\sqrt{m}} \sum_{i=1}^{n} (f(x_i; W(s), b(s)) - y_i) a_r \mathbb{I}(w_r(s)^\top x_i \ge b_r) \right\|_2 \\
&\le \frac{1}{\sqrt{m}} \sum_{i=1}^{n} |f(x_i; W(s), b(s)) - y_i| \\
&\le \sqrt{\frac{n}{m}} \left\| f(s) - y \right\|_2 .
\end{aligned}$$

□

F.6.4.1 Gradient descent

Lemma F.15. *Assume $\lambda > 0$. Assume $\|y - f(k)\|_2^2 \le (1 - \eta\lambda/4)^k \|y - f(0)\|_2^2$ holds for all $k' \le k$. Then for every $r \in [m]$,*

$$\|w_r(k+1) - w_r(0)\|_2 \le \frac{8\sqrt{n}\,\|y - f(0)\|_2}{\sqrt{m}\lambda} := D_w,$$
$$|b_r(k+1) - b_r(0)| \le \frac{8\sqrt{n}\,\|y - f(0)\|_2}{\sqrt{m}\lambda} := D_b.$$

Proof.

$$\begin{aligned}
\|w_r(k+1) - w_r(0)\|_2 &\le \eta \sum_{k'=0}^{k} \left\| \frac{\partial L(W(k'))}{\partial w_r(k')} \right\|_2 \\
&\le \eta \sum_{k'=0}^{k} \sqrt{\frac{n}{m}} \|y - f(k')\|_2 \\
&\le \eta \sum_{k'=0}^{k} \sqrt{\frac{n}{m}} (1 - \eta\lambda/4)^{k'/2} \|y - f(0)\|_2 \\
&\le \eta \sum_{k'=0}^{k} \sqrt{\frac{n}{m}} (1 - \eta\lambda/8)^{k'} \|y - f(0)\|_2 \\
&\le \eta \sum_{k'=0}^{\infty} \sqrt{\frac{n}{m}} (1 - \eta\lambda/8)^{k'} \|y - f(0)\|_2 \\
&\le \frac{8\sqrt{n}}{\sqrt{m}\lambda} \|y - f(0)\|_2,
\end{aligned}$$

where the first inequality is by Triangle inequality, the second inequality is by Lemma F.14, the third inequality is by our assumption and the fourth inequality is by $(1-x)^{1/2} \le 1 - x/2$ for $x \ge 0$.

The proof for b is similar. $\square$

F.6.4.2 Gradient flow

Lemma F.16. *Suppose for $0 \le s \le t$, $\lambda_{\min}(H(s)) \ge \frac{\lambda_0}{2} > 0$. Then we have $\|y - f(t)\|_2^2 \le \exp(-\lambda_0 t) \|y - f(0)\|_2^2$ and for any $r \in [m]$, $\|w_r(t) - w_r(0)\|_2 \le \frac{\sqrt{n}\|y - f(0)\|_2}{\sqrt{m}\lambda_0}$ and $|b_r(t) - b_r(0)| \le \frac{\sqrt{n}\|y - f(0)\|_2}{\sqrt{m}\lambda_0}$.*

Proof. By the dynamics of prediction in Equation (F.4), we have

$$\begin{aligned}\frac{d}{dt}\|y - f(t)\|_2^2 &= -2(y - f(t))^\top H(t)(y - f(t))\\ &\leq -\lambda_0 \|y - f(t)\|_2^2,\end{aligned}$$

which implies

$$\|y - f(t)\|_2^2 \leq \exp(-\lambda_0 t)\|y - f(t)\|_2^2.$$

Now we bound the gradient norm of the weights

$$\begin{aligned}\left\|\frac{d}{ds}w_r(s)\right\|_2 &= \left\|\sum_{i=1}^{n}(y_i - f_i(s))\frac{1}{\sqrt{m}}a_r x_i \mathbb{I}(w_r(s)^\top x_i \geq b(s))\right\|_2\\ &\leq \frac{1}{\sqrt{m}}\sum_{i=1}^{n}|y_i f_i(s)| \leq \frac{\sqrt{n}}{\sqrt{m}}\|y - f(s)\|_2 \leq \frac{\sqrt{n}}{\sqrt{m}}\exp(-\lambda_0 s)\|y - f(0)\|_2.\end{aligned}$$

Integrating the gradient, the change of weight can be bounded as

$$\|w_r(t) - w_r(0)\|_2 \leq \int_0^t \left\|\frac{d}{ds}w_r(s)\right\|_2 ds \leq \frac{\sqrt{n}\|y - f(0)\|_2}{\sqrt{m}\lambda_0}.$$

For bias, we have

$$\begin{aligned}\left\|\frac{d}{ds}b_r(s)\right\|_2 &= \left\|\sum_{i=1}^{n}(y_i - f_i(s))\frac{1}{\sqrt{m}}a_r \mathbb{I}(w_r(s)^\top x_i \geq b(s))\right\|_2\\ &\leq \frac{1}{\sqrt{m}}\sum_{i=1}^{n}|y_i - f_i(s)| \leq \frac{\sqrt{n}}{\sqrt{m}}\|y - f(s)\|_2 \leq \frac{\sqrt{n}}{\sqrt{m}}\exp(-\lambda_0 s)\|y - f(0)\|_2.\end{aligned}$$

Now, the change of bias can be bounded as

$$\|b_r(t) - b_r(0)\|_2 \leq \int_0^t \left\|\frac{d}{ds}w_r(s)\right\|_2 ds \leq \frac{\sqrt{n}\|y - f(0)\|_2}{\sqrt{m}\lambda_0}.$$

□

F.6.5 Gradient descent convergence analysis

F.6.5.1 Upper bound of the initial error

Lemma F.17 (Initial error upper bound)**.** *Let $B > 0$ be the initialization value of the biases and all the weights be initialized from standard Gaussian. Let $\delta \in (0, 1)$ be* *the failure probability. Then, with probability at least* $1-\delta$*, we have*

$$\|f(0)\|_2^2 = O(n(\exp(-B^2/2)+1/m)\log^3(mn/\delta)),$$
$$\|f(0)-y\|_2^2 = O\left(n + n\left(\exp(-B^2/2)+1/m\right)\log^3(2mn/\delta)\right).$$

Proof. Since we are only analyzing the initialization stage, for notation ease, we omit the dependence on time without any confusion. We compute

$$\begin{aligned}\|y-f\|_2^2 &= \sum_{i=1}^n (y_i - f(x_i))^2 \\ &= \sum_{i=1}^n \left(y_i - \frac{1}{\sqrt{m}}\sum_{r=1}^m a_r \sigma(w_r^\top x_i - B)\right)^2 \\ &= \sum_{i=1}^n \left(y_i^2 - 2\frac{y_i}{\sqrt{m}}\sum_{r=1}^m a_r\sigma(w_r^\top x_i - B) + \frac{1}{m}\left(\sum_{r=1}^m a_r\sigma(w_r^\top x_i - B)\right)^2\right).\end{aligned}$$

Since $w_r^\top x_i \sim \mathcal{N}(0,1)$ for all $r \in [m]$ and $i \in [n]$, by Gaussian tail bound and a union bound over r, i, we have

$$\mathbb{P}[\forall i \in [n],\ j \in [m] : w_r^\top x_i \le \sqrt{2\log(2mn/\delta)}] \ge 1 - \delta/2.$$

Let E_1 denote this event. Conditioning on the event E_1, let

$$z_{i,r} := \frac{1}{\sqrt{m}} \cdot a_r \cdot \min\left\{\sigma(w_r^\top x_i - B), \sqrt{2\log(2mn/\delta)}\right\}.$$

Notice that $z_{i,r} \neq 0$ with probability at most $\exp(-B^2/2)$. Thus,

$$\mathbb{E}_{a_r,w_r}[z_{i,r}^2] \le \exp(-B^2/2)\frac{1}{m}2\log(2mn/\delta).$$

By randomness in a_r, we know $\mathbb{E}[z_{i,r}] = 0$. Now apply Bernstein's inequality in Lemma A.4, we have for all $t > 0$,

$$\mathbb{P}\left[\left|\sum_{r=1}^m z_{i,r}\right| > t\right] \le \exp\left(-\min\left(\frac{t^2/2}{4\exp(-B^2/2)\log(2mn/\delta)}, \frac{\sqrt{m}t/2}{2\sqrt{2\log(2mn/\delta)}}\right)\right).$$

Thus, by a union bound, with probability at least $1-\delta/2$, for all $i \in [n]$,

$$\begin{aligned}\left|\sum_{r=1}^{m} z_{i,r}\right| &\leq \sqrt{2\log(2mn/\delta)\exp(-B^2/2)2\log(2n/\delta)} + 2\sqrt{\frac{2\log(2mn/\delta)}{m}}\log(2n/\delta) \\ &\leq \left(2\exp(-B^2/4) + 2\sqrt{2/m}\right)\log^{3/2}(2mn/\delta).\end{aligned}$$

Let E_2 denote this event. Thus, conditioning on the events E_1, E_2, with probability $1-\delta$,

$$\|f(0)\|_2^2 = \sum_{i=1}^{n}\left(\sum_{r=1}^{m} z_{i,r}\right)^2 = O(n(\exp(-B^2/2) + 1/m)\log^3(mn/\delta))$$

and

$$\begin{aligned}&\|y - f(0)\|_2^2 \\ &= \sum_{i=1}^{n} y_i^2 - 2\sum_{i=1}^{n} y_i \sum_{r=1}^{m} z_{i,r} + \sum_{i=1}^{n}\left(\sum_{r=1}^{m} z_{i,r}\right)^2 \\ &\leq \sum_{i=1}^{n} y_i^2 + 2\sum_{i=1}^{n} |y_i| \left(2\exp(-B^2/4) + 2\sqrt{2/m}\right)\log^{3/2}(2mn/\delta) \\ &\quad + \sum_{i=1}^{n}\left(\left(2\exp(-B^2/4) + 2\sqrt{2/m}\right)\log^{3/2}(2mn/\delta)\right)^2 \\ &= O\left(n + n\left(\exp(-B^2/2) + 1/m\right)\log^3(2mn/\delta)\right),\end{aligned}$$

where we assume $y_i = O(1)$ for all $i \in [n]$. □

F.6.5.2 Error Decomposition

We follow the proof outline in [SY19, SYZ21] and we generalize it to networks with trainable b. Let us define matrix $H^\perp$ similar to H except only considering flipped neurons by

$$H_{ij}^\perp(k) := \frac{1}{m}\sum_{r\in\overline{S}_i}(\langle x_i, x_j\rangle + 1)\mathbb{I}(w_r(k)^\top x_i \geq b_r(k), w_r(k)^\top x_j \geq b_r(k))$$

and vector v_1, v_2 by

$$v_{1,i} := \frac{1}{\sqrt{m}} \sum_{r \in S_i} a_r(\sigma(\langle w_r(k+1), x_i \rangle - b_r(k+1)) - \sigma(\langle w_r(k), x_i \rangle - b_r(k))),$$
$$v_{2,i} := \frac{1}{\sqrt{m}} \sum_{r \in \overline{S}_i} a_r(\sigma(\langle w_r(k+1), x_i \rangle - b_r(k+1)) - \sigma(\langle w_r(k), x_i \rangle - b_r(k))).$$

Now we give out our error update.

Claim F.18.

$$\|y - f(k+1)\|_2^2 = \|y - f(k)\|_2^2 + B_1 + B_2 + B_3 + B_4,$$

where

$$B_1 := -2\eta(y - f(k))^\top H(k)(y - f(k)),$$
$$B_2 := 2\eta(y - f(k))^\top H^\perp(k)(y - f(k)),$$
$$B_3 := -2(y - f(k))^\top v_2,$$
$$B_4 := \|f(k+1) - f(k)\|_2^2.$$

Proof. First we can write

$$\begin{aligned}
v_{1,i} &= \frac{1}{\sqrt{m}} \sum_{r \in S_i} a_r \left(\sigma \left(\left\langle w_r(k) - \eta \frac{\partial L}{\partial w_r}, x_i \right\rangle - \left(b_r(k) - \eta \frac{\partial L}{\partial b_r} \right) \right) - \sigma(\langle w_r(k), x_i \rangle - b_r(k)) \right) \\
&= \frac{1}{\sqrt{m}} \sum_{r \in S_i} a_r \left(\left\langle -\eta \frac{\partial L}{\partial w_r}, x_i \right\rangle + \eta \frac{\partial L}{\partial b_r} \right) \mathbb{I}(\langle w_r(k), x_i \rangle - b_r(k) \geq 0) \\
&= \frac{1}{\sqrt{m}} \sum_{r \in S_i} a_r \left(\eta \frac{1}{\sqrt{m}} \sum_{j=1}^n (y_j - f_j(k)) a_r(\langle x_j, x_i \rangle + 1) \mathbb{I}(w_r(k)^\top x_j \geq b_r(k)) \right) \\
&\quad \cdot \mathbb{I}(\langle w_r(k), x_i \rangle - b_r(k) \geq 0) \\
&= \eta \sum_{j=1}^n (y_j - f_j(k))(H_{ij}(k) - H_{ij}^\perp(k))
\end{aligned}$$

which means

$$v_1 = \eta(H(k) - H^\perp(k))(y - f(k)).$$

Now we compute

$$\begin{aligned}\|y - f(k+1)\|_2^2 &= \|y - f(k) - (f(k+1) - f(k))\|_2^2 \\ &= \|y - f(k)\|_2^2 - 2(y - f(k))^\top (f(k+1) - f(k)) + \|f(k+1) - f(k)\|_2^2 .\end{aligned}$$

Since $f(k+1) - f(k) = v_1 + v_2$, we can write the cross product term as

$$\begin{aligned}&(y - f(k))^\top (f(k+1) - f(k)) \\ &= (y - f(k))^\top (v_1 + v_2) \\ &= (y - f(k))^\top v_1 + (y - f(k))^\top v_2 \\ &= \eta (y - f(k))^\top H(k)(y - f(k)) \\ &\quad - \eta (y - f(k))^\top H^\perp(k)(y - f(k)) + (y - f(k))^\top v_2.\end{aligned}$$

□

F.6.5.3 Bounding the decrease of the error

Lemma F.19. *Assume $\lambda > 0$. Assume we choose R_w, R_b, B where $R_w, R_b \le \min\{1/B, 1\}$ such that $8cn(R_w + R_b)\exp(-B^2/2) \le \lambda/8$. Denote $\delta_0 = \delta + n^2 \exp(-\frac{2}{3}cm(R_w + R_b)\exp(-B^2/2))$. Then,*

$$\mathbb{P}[B_1 \le -\eta 5\lambda \|y - f(k)\|_2^2 /8] \ge 1 - \delta_0.$$

Proof. By Lemma F.13 and our assumption,

$$\lambda_{\min}(H(W)) > 0.75\lambda - n \cdot 8c(R_w + R_b)\exp(-B^2/2) \ge 5\lambda/8$$

with probability at least $1 - \delta_0$. Thus,

$$(y - f(k))^\top H(k)(y - f(k)) \ge \|y - f(k)\|_2^2 5\lambda/8.$$

□

F.6.5.4 Bounding the effect of flipped neurons

Here we bound the term B_2, B_3. First, we introduce a fact.

Fact F.20.

$$\left\|H^{\perp}(k)\right\|_F^2 \leq \frac{4n}{m^2} \sum_{i=1}^{n} |\overline{S}_i|^2.$$

Proof.

$$\begin{aligned}\left\|H^{\perp}(k)\right\|_F^2 &= \sum_{i,j \in [n]} \left(\frac{1}{m} \sum_{r \in \overline{S}_i} (x_i^\top x_j + 1) \mathbb{I}(w_r(k)^\top x_i \geq b_r(k),\ w_r(k)^\top x_j \geq b_r(k)) \right)^2 \\ &\leq \sum_{i,j \in [n]} \left(\frac{1}{m} 2 |\overline{S}_i| \right)^2 \leq \frac{4n}{m^2} \sum_{i=1}^{n} |\overline{S}_i|^2.\end{aligned}$$

□

Lemma F.21. *Denote $\delta_0 = n \exp(-\frac{2}{3} cm(R_w + R_b) \exp(-B^2/2))$. Then,*

$$\mathbb{P}[B_2 \leq 8\eta nc(R_w + R_b) \exp(-B^2/2) \cdot \|y - f(k)\|_2^2] \geq 1 - \delta_0.$$

Proof. First, we have

$$B_2 \leq 2\eta \left\|y - f(k)\right\|_2^2 \left\|H^{\perp}(k)\right\|_2.$$

Then, by Fact F.20,

$$\left\|H^{\perp}(k)\right\|_2^2 \leq \left\|H^{\perp}(k)\right\|_F^2 \leq \frac{4n}{m^2} \sum_{i=1}^{n} |\overline{S}_i|^2.$$

By Corollary F.12, we have

$$\mathbb{P}[\forall i \in [n] :\ |\overline{S}_i| \leq 2mc(R_w + R_b) \exp(-B^2/2)] \geq 1 - \delta_0.$$

Thus, with probability at least $1 - \delta_0$,

$$\left\|H^{\perp}(k)\right\|_2 \leq 4nc(R_w + R_b) \exp(-B^2/2).$$

□

Lemma F.22. *Denote* $\delta_0 = n \exp(-\frac{2}{3} cm(R_w + R_b)\exp(-B^2/2))$. *Then,*

$$\mathbb{P}[B_3 \leq 4c\eta n(R_w + R_b)\exp(-B^2/2)\left\|y - f(k)\right\|_2^2] \geq 1 - \delta_0.$$

Proof. By Cauchy-Schwarz inequality, we have $B_3 \leq 2\left\|y - f(k)\right\|_2 \left\|v_2\right\|_2$. We have

$$\begin{aligned}
\left\|v_2\right\|_2^2 &\leq \sum_{i=1}^n \left(\frac{\eta}{\sqrt{m}} \sum_{r \in \overline{S}_i} \left|\left\langle \frac{\partial L}{\partial w_r}, x_i \right\rangle\right| + \left|\frac{\partial L}{\partial b_r}\right|\right)^2 \\
&\leq \sum_{i=1}^n \frac{\eta^2}{m} \max_{i \in [n]} \left(\left|\left\langle \frac{\partial L}{\partial w_r}, x_i \right\rangle\right| + \left|\frac{\partial L}{\partial b_r}\right|\right)^2 |\overline{S}_i|^2 \\
&\leq n\frac{\eta^2}{m}\left(2\sqrt{\frac{n}{m}}\left\|f(k) - y\right\|_2 2mc(R_w + R_b)\exp(-B^2/2)\right)^2 \\
&= 16c^2\eta^2 n^2 \left\|y - f(k)\right\|_2^2 (R_w + R_b)^2 \exp(-B^2),
\end{aligned}$$

where the last inequality is by Lemma F.14 and Corollary F.12 which holds with probability at least $1 - \delta_0$. □

F.6.5.5 Bounding the network update

Lemma F.23.

$$B_4 \leq C_2^2 \eta^2 n^2 \left\|y - f(k)\right\|_2^2 \exp(-B^2).$$

for some constant C_2*.*

Proof. Recall that the definition that $\mathcal{S}_{\text{on}}(i,t) = \{r \in [m] : w_r(t)^\top x_i \geq b_r(t)\}$, i.e., the set of neurons that activates for input x_i at the t-th step of gradient descent.

$$\begin{aligned}
\left\|f(k+1) - f(k)\right\|_2^2 &\leq \sum_{i=1}^n \left(\frac{\eta}{\sqrt{m}} \sum_{r: r \in \mathcal{S}_{\text{on}}(i,k+1) \cup \mathcal{S}_{\text{on}}(i,k)} \left|\left\langle \frac{\partial L}{\partial w_r}, x_i \right\rangle\right| + \left|\frac{\partial L}{\partial b_r}\right|\right)^2 \\
&\leq n\frac{\eta^2}{m}(|\mathcal{S}_{\text{on}}(i,k+1)| + |\mathcal{S}_{\text{on}}(i,k)|)^2 \max_{i \in [n]} \left(\left|\left\langle \frac{\partial L}{\partial w_r}, x_i \right\rangle\right| + \left|\frac{\partial L}{\partial b_r}\right|\right)^2 \\
&\leq n\frac{\eta^2}{m}\left(C_2 m \exp(-B^2/2) \cdot \sqrt{\frac{n}{m}}\left\|y - f(k)\right\|_2\right)^2 \\
&\leq C_2^2 \eta^2 n^2 \left\|y - f(k)\right\|_2^2 \exp(-B^2).
\end{aligned}$$

where the third inequality is by Lemma F.25 for some C_2. □

F.6.5.6 Putting it all together

Theorem F.24 (Convergence). *Assume $\lambda > 0$. Let $\eta \le \frac{\lambda \exp(B^2)}{5C_2^2 n^2}$, $B \in [0, \sqrt{0.5 \log m}]$ and*

$$m \ge \widetilde{\Omega}\left(\lambda^{-4} n^4 \left(1 + \left(\exp(-B^2/2) + 1/m\right) \log^3(2mn/\delta)\right) \exp(-B^2)\right).$$

Assume $\lambda = \lambda_0 \exp(-B^2/2)$ for some constant λ_0. Then,

$$\mathbb{P}\left[\forall t : \|y - f(t)\|_2^2 \le (1 - \eta\lambda/4)^t \|y - f(0)\|_2^2\right] \ge 1 - \delta - e^{-\Omega(n)}.$$

Proof. From Lemma F.19, Lemma F.21, Lemma F.22 and Lemma F.23, we know with probability at least $1 - 2n^2 \exp(-\frac{2}{3} cm(R_w + R_b)\exp(-B^2/2)) - \delta$, we have

$$\begin{aligned}\|y - f(k+1)\|_2^2 \le \|y - f(k)\|_2^2 \Big(&1 - 5\eta\lambda/8 + 12\eta nc(R_w + R_b)\exp(-B^2/2) \\ &+ C_2^2 \eta^2 n^2 \|y - f(k)\|_2^2 \exp(-B^2)\Big).\end{aligned}$$

By Lemma F.15, we need

$$\begin{aligned} D_w &= \frac{8\sqrt{n}\,\|y - f(0)\|_2}{\sqrt{m}\lambda} \le R_w, \\ D_b &= \frac{8\sqrt{n}\,\|y - f(0)\|_2}{\sqrt{m}\lambda} \le R_b. \end{aligned}$$

By Lemma F.17, we have

$$\mathbb{P}[\|f(0) - y\|_2^2 = O\left(n + n\left(\exp(-B^2/2) + 1/m\right)\log^3(2mn/\delta)\right)] \ge 1 - \delta.$$

Let $R = \min\{R_w, R_b\}$, $D = \max\{D_w, D_b\}$. Combine the results we have

$$R > \Omega(\lambda^{-1} m^{-1/2} n \sqrt{1 + (\exp(-B^2/2) + 1/m)\log^3(2mn/\delta)}).$$

Lemma F.19 requires

$$\begin{aligned} 8cn(R_w + R_b)\exp(-B^2/2) &\le \lambda/8 \\ \Rightarrow R &\le \frac{\lambda \exp(B^2/2)}{128cn}. \end{aligned}$$

which implies a lower bound on m

$$m \geq \Omega\left(\lambda^{-4} n^4 \left(1 + \left(\exp(-B^2/2) + 1/m\right) \log^3(2mn/\delta)\right) \exp(-B^2)\right).$$

Lemma F.9 further requires a lower bound of $m = \Omega(\lambda^{-1} n \cdot \log(n/\delta))$ which can be ignored.

Lemma F.13 further requires $R < \min\{1/B, 1\}$ which implies

$$B < \frac{128cn}{\lambda \exp(B^2/2)},$$
$$m \geq \widetilde{\Omega}\left(\lambda^{-4} n^4 \left(1 + \left(\exp(-B^2/2) + 1/m\right) \log^3(2mn/\delta)\right) \exp(-B^2)\right).$$

From Theorem F.1 in [SYZ21] we know that $\lambda = \lambda_0 \exp(-B^2/2)$ for some λ_0 with no dependence on B and $\lambda \exp(B^2/2) \leq 1$. Thus, by our constraint on m and B, this is always satisfied.

Finally, to require

$$12\eta nc(R_w + R_b) \exp(-B^2/2) + C_2^2 \eta^2 n^2 \exp(-B^2) \leq \eta\lambda/4,$$

we need $\eta \leq \frac{\lambda \exp(B^2)}{5C_2^2 n^2}$. By our choice of m, B, we have

$$2n^2 \exp(-\frac{2}{3} cm(R_w + R_b) \exp(-B^2/2)) = e^{-\Omega(n)}.$$

$\square$

F.6.6 Bounding the number of activated neurons per iteration

First, we define the set of activated neurons at iteration t for training point x_i to be

$$\mathcal{S}_{\mathrm{on}}(i,t) = \{r \in [m] : \; w_r(t)^\top x_i \geq b_r(t)\}.$$

Lemma F.25 (Number of Activated Neurons at Initialization)**.** *Assume the choice of m in Theorem F.24. With probability at least $1 - e^{-\Omega(n)}$ over the random initialization, we have*

$$|\mathcal{S}_{\mathrm{on}}(i,t)| = O(m \cdot \exp(-B^2/2)),$$

for all $0 \le t \le T$ *and* $i \in [n]$. *And As a by-product,*

$$\|Z(0)\|_F^2 \le 8n \exp(-B^2/2).$$

Proof. First we bound the number of activated neuron at the initialization. We have $\mathbb{P}[w_r^\top x_i \ge B] \le \exp(-B^2/2)$. By Bernstein's inequality,

$$\mathbb{P}[|\mathcal{S}_{\text{on}}(i,0)| \ge m\exp(-B^2/2) + t] \le \exp\left(-\frac{t^2}{m\exp(-B^2/2) + t/3}\right).$$

Take $t = m\exp(-B^2/2)$ we have

$$\mathbb{P}[|\mathcal{S}_{\text{on}}(i,0)| \ge 2m\exp(-B^2/2)] \le \exp\left(-m\exp(-B^2/2)/4\right).$$

By a union bound over $i \in [n]$, we have

$$\mathbb{P}[\forall i \in [n] : \ |\mathcal{S}_{\text{on}}(i,0)| \le 2m\exp(-B^2/2)] \ge 1 - n\exp\left(-m\exp(-B^2/2)/4\right).$$

Notice that

$$\|Z(0)\|_F^2 \le \frac{4}{m}\sum_{r=1}^{m}\sum_{i=1}^{n} \mathbb{I}_{r,i}(0) \le 8n\exp(-B^2/2).$$

□

Lemma F.26 (Number of Activated Neurons per Iteration)**.** *Assume the parameter settings in Theorem F.24. With probability at least* $1 - e^{-\Omega(n)}$ *over the random initialization, we have*

$$|\mathcal{S}_{\text{on}}(i,t)| = O(m \cdot \exp(-B^2/2))$$

for all $0 \le t \le T$ *and* $i \in [n]$.

Proof. By Corollary F.12 and Theorem F.24, we have

$$\mathbb{P}[\forall i \in [n] : \ |\overline{S}_i| \le 4mc\exp(-B^2/2)] \ge 1 - e^{-\Omega(n)}.$$

Recall $\overline{S}_i$ is the set of flipped neurons during the entire training process. Notice that $|\mathcal{S}_{\text{on}}(i,t)| \le |\mathcal{S}_{\text{on}}(i,0)| + |\overline{S}_i|$. Thus, by Lemma F.25

$$\mathbb{P}[\forall i \in [n] : \ |\mathcal{S}_{\text{on}}(i,t)| = O(m\exp(-B^2/2))] \ge 1 - e^{-\Omega(n)}.$$

□

F.7 Bounding the Restricted Smallest Eigenvalue with Data Separation

Theorem F.27. *Let $X = (x_1, \ldots, x_n)$ be points in $\mathbb{R}^d$ with $\|x_i\|_2 = 1$ for all $i \in [n]$ and $w \sim \mathcal{N}(0, I_d)$. Suppose that there exists $\delta \in [0, \sqrt{2}]$ such that*

$$\min_{i \neq j \in [n]} (\|x_i - x_j\|_2, \|x_i + x_j\|_2) \geq \delta.$$

Let $B \geq 0$. Recall the limit NTK matrix H^∞ defined as

$$H^\infty_{ij} := \mathbb{E}_{w \sim \mathcal{N}(0,I)} \left[(\langle x_i, x_j \rangle + 1) \mathbb{I}(w^\top x_i \geq B, w^\top x_j \geq B) \right].$$

Define $p_0 = \mathbb{P}[w^\top x_1 \geq B]$ and $p_{ij} = \mathbb{P}[w^\top x_i \geq B,\ w^\top x_j \geq B]$ for $i \neq j$. Define the (data-dependent) region $\mathcal{R} = \{a \in \mathbb{R}^n : \ \sum_{i \neq j} a_i a_j p_{ij} \geq \min_{i' \neq j'} p_{i'j'} \sum_{i \neq j} a_i a_j\}$ and let $\lambda := \min_{\|a\|_2 = 1,\ a \in \mathcal{R}} a^\top H^\infty a$. Then, $\lambda \geq \max(0, \lambda')$ where

$$\begin{aligned} \lambda' &\geq p_0 - \min_{i \neq j} p_{ij} \\ &\geq \max\left(\frac{1}{2} - \frac{B}{\sqrt{2\pi}}, \left(\frac{1}{B} - \frac{1}{B^3} \right) \frac{e^{-B^2/2}}{\sqrt{2\pi}} \right) - e^{-B^2/(2-\delta^2/2)} \frac{\pi - \arctan\left(\frac{\delta\sqrt{1-\delta^2/4}}{1-\delta^2/2} \right)}{2\pi}. \end{aligned}$$

Proof. Define $\Delta := \max_{i \neq j} | \langle x_i, x_j \rangle |$. Then by our assumption,

$$\begin{aligned} &1 - \Delta = 1 - \max_{i \neq j} | \langle x_i, x_j \rangle | = \frac{\min_{i \neq j}(\|x_i - x_j\|_2^2, \|x_i + x_j\|_2^2)}{2} \geq \delta^2/2 \\ &\Rightarrow \Delta \leq 1 - \delta^2/2. \end{aligned}$$

Further, we define

$$Z(w) := [x_1 \mathbb{I}(w^\top x_1 \geq B), x_2 \mathbb{I}(w^\top x_2 \geq B), \ldots, x_n \mathbb{I}(w^\top x_n \geq B)] \in \mathbb{R}^{d \times n}.$$

Notice that $H^\infty = \mathbb{E}_{w \sim \mathcal{N}(0,I)} \left[Z(w)^\top Z(w) + \mathbb{I}(Xw \geq B) \mathbb{I}(Xw \geq B)^\top \right]$. We need to lower bound

$$\begin{aligned} \min_{\|a\|_2 = 1, a \in \mathcal{R}} a^\top H^\infty a &= \min_{\|a\|_2 = 1, a \in \mathcal{R}} a^\top \mathbb{E}_{w \sim \mathcal{N}(0,I)} \left[Z(w)^\top Z(w) \right] a \\ &\quad + a^\top \mathbb{E}_{w \sim \mathcal{N}(0,I)} \left[\mathbb{I}(Xw \geq B) \mathbb{I}(Xw \geq B)^\top \right] a \\ &\geq \min_{\|a\|_2 = 1, a \in \mathcal{R}} a^\top \mathbb{E}_{w \sim \mathcal{N}(0,I)} \left[\mathbb{I}(Xw \geq B) \mathbb{I}(Xw \geq B)^\top \right] a. \end{aligned}$$

Now, for a fixed a,

$$\begin{aligned} a^\top \mathbb{E}_{w\sim\mathcal{N}(0,I)}\left[\mathbb{I}(Xw \geq B)\mathbb{I}(Xw \geq B)^\top\right] a &= \sum_{i=1}^n a_i^2 \mathbb{P}[w^\top x_i \geq B] + \sum_{i\neq j} a_i a_j \mathbb{P}[w^\top x_i \geq B,\ w^\top x_j \geq B] \\ &= p_0 \left\|a\right\|_2^2 + \sum_{i\neq j} a_i a_j p_{ij}, \end{aligned}$$

where the last equality is by $\mathbb{P}[w^\top x_1 \geq B] = \ldots = \mathbb{P}[w^\top x_n \geq B] = p_0$ which is due to spherical symmetry of standard Gaussian. Notice that $\max_{i\neq j} p_{ij} \leq p_0$. Since $a \in \mathcal{R}$,

$$\begin{aligned} \mathbb{E}_{w\sim\mathcal{N}(0,I)}\left[(a^\top \mathbb{I}(Xw \geq B))^2\right] &\geq (p_0 - \min_{i\neq j} p_{ij}) \left\|a\right\|_2^2 + (\min_{i\neq j} p_{ij}) \left\|a\right\|_2^2 + (\min_{i\neq j} p_{ij}) \sum_{i\neq j} a_i a_j \\ &= (p_0 - \min_{i\neq j} p_{ij}) \left\|a\right\|_2^2 + (\min_{i\neq j} p_{ij}) \left(\sum_i a_i\right)^2. \end{aligned}$$

Thus,

$$\begin{aligned} \lambda &\geq \min_{\left\|a\right\|_2=1, a\in\mathcal{R}} \mathbb{E}_{w\sim\mathcal{N}(0,I)}\left[(a^\top \mathbb{I}(Xw \geq B))^2\right] \\ &\geq \min_{\left\|a\right\|_2=1, a\in\mathcal{R}} (p_0 - \min_{i\neq j} p_{ij}) \left\|a\right\|_2^2 + \min_{\left\|a\right\|_2=1, a\in\mathcal{R}} (\min_{i\neq j} p_{ij}) \left(\sum_i a_i\right)^2 \\ &\geq p_0 - \min_{i\neq j} p_{ij}. \end{aligned}$$

Now we need to upper bound

$$\min_{i\neq j} p_{ij} \leq \max_{i\neq j} p_{ij}.$$

We divide into two cases: $B = 0$ and $B > 0$. Consider two fixed examples x_1, x_2. Then, let $v = (I - x_1 x_1^\top)x_2 / \left\|(I - x_1 x_1^\top)x_2\right\|$ and $c = |\langle x_1, x_2\rangle|$ [1].

Case 1: $B = 0$**.** First, let us define the region $\mathcal{A}_0$ as

$$\mathcal{A}_0 = \left\{(g_1, g_2) \in \mathbb{R}^2 :\ g_1 \geq 0,\ g_1 \geq -\frac{\sqrt{1-c^2}}{c} g_2\right\}.$$

[1]Here we force c to be positive. Since we are dealing with standard Gaussian, the probability is exactly the same if $c < 0$ by symmetry and therefore, we force $c > 0$.

Then,

$$\begin{aligned}
\mathbb{P}[w^\top x_1 \geq 0,\ w^\top x_2 \geq 0] &= \mathbb{P}[w^\top x_1 \geq 0,\ w^\top (cx_1 + \sqrt{1-c^2}v) \geq 0] \\
&= \mathbb{P}[g_1 \geq 0,\ cg_1 + \sqrt{1-c^2}g_2 \geq 0] \\
&= \mathbb{P}[\mathcal{A}_0] \\
&= \frac{\pi - \arctan\left(\frac{\sqrt{1-c^2}}{|c|}\right)}{2\pi} \\
&\leq \frac{\pi - \arctan\left(\frac{\sqrt{1-\Delta^2}}{|\Delta|}\right)}{2\pi},
\end{aligned}$$

where we define $g_1 := w^\top x_1$ and $g_2 := w^\top v$ and the second equality is by the fact that since x_1 and v are orthonormal, g_1 and g_2 are two independent standard Gaussian random variables; the last inequality is by arctan is a monotonically increasing function and $\frac{\sqrt{1-c^2}}{|c|}$ is a decreasing function in $|c|$ and $|c| \leq \Delta$. Thus,

$$\min_{i \neq j} p_{ij} \leq \max_{i \neq j} p_{ij} \leq \frac{\pi - \arctan\left(\frac{\sqrt{1-\Delta^2}}{|\Delta|}\right)}{2\pi}.$$

Case 2: $B > 0$. First, let us define the region

$$\mathcal{A} = \left\{(g_1, g_2) \in \mathbb{R}^2 :\ g_1 \geq B,\ g_1 \geq \frac{B}{c} - \frac{\sqrt{1-c^2}}{c} g_2\right\}.$$

Then, following the same steps as in case 1, we have

$$\mathbb{P}[w^\top x_1 \geq B,\ w^\top x_2 \geq B] = \mathbb{P}[g_1 \geq B,\ cg_1 + \sqrt{1-c^2}g_2 \geq B] = \mathbb{P}[\mathcal{A}].$$

Let $B_1 = B$ and $B_2 = B\sqrt{\frac{1-c}{1+c}}$. Further, notice that $\mathcal{A} = \mathcal{A}_0 + (B_1, B_2)$. Then,

$$\begin{aligned}
\mathbb{P}[\mathcal{A}] &= \iint_{(g_1,g_2)\in\mathcal{A}} \frac{1}{2\pi} \exp\left\{-\frac{g_1^2 + g_2^2}{2}\right\}\, dg_1\, dg_2 \\
&= \iint_{(g_1,g_2)\in\mathcal{A}_0} \frac{1}{2\pi} \exp\left\{-\frac{(g_1+B_1)^2 + (g_2+B_2)^2}{2}\right\}\, dg_1\, dg_2 \\
&= e^{-(B_1^2+B_2^2)/2} \iint_{(g_1,g_2)\in\mathcal{A}_0} \frac{1}{2\pi} \exp\{-B_1 g_1 - B_2 g_2\} \exp\left\{-\frac{g_1^2+g_2^2}{2}\right\}\, dg_1\, dg_2.
\end{aligned}$$

Now, $B_1 g_1 + B_2 g_2 = Bg_1 + B\sqrt{\frac{1-c}{1+c}} g_2 \geq 0$ always holds if and only if $g_1 \geq -\sqrt{\frac{1-c}{1+c}} g_2$. Define the region $\mathcal{A}_+$ to be

$$\mathcal{A}_+ = \left\{ (g_1, g_2) \in \mathbb{R}^2 : \; g_1 \geq 0, \; g_1 \geq -\sqrt{\frac{1-c}{1+c}} g_2 \right\}.$$

Observe that

$$\sqrt{\frac{1-c}{1+c}} \leq \frac{\sqrt{1-c^2}}{c} = \frac{\sqrt{(1-c)(1+c)}}{c} \Leftrightarrow c \leq 1 + c.$$

Thus, $\mathcal{A}_0 \subset \mathcal{A}_+$. Therefore,

$$\begin{aligned}
\mathbb{P}[\mathcal{A}] &\leq e^{-(B_1^2+B_2^2)/2} \iint_{(g_1,g_2)\in\mathcal{A}_0} \frac{1}{2\pi} \exp\left\{ -\frac{g_1^2+g_2^2}{2} \right\} \, dg_1 \, dg_2 \\
&= e^{-(B_1^2+B_2^2)/2} \mathbb{P}[\mathcal{A}_0] \\
&= e^{-(B_1^2+B_2^2)/2} \frac{\pi - \arctan\left(\frac{\sqrt{1-c^2}}{|c|} \right)}{2\pi} \\
&\leq e^{-B^2/(1+\Delta)} \frac{\pi - \arctan\left(\frac{\sqrt{1-\Delta^2}}{|\Delta|} \right)}{2\pi}.
\end{aligned}$$

Finally, we need to lower bound p_0. This can be done in two ways: when B is small, we apply Gaussian anti-concentration bound and when B is large, we apply Gaussian tail bounds. Thus,

$$p_0 = \mathbb{P}[w^\top x_1 \geq B] \geq \max\left(\frac{1}{2} - \frac{B}{\sqrt{2\pi}}, \; \left(\frac{1}{B} - \frac{1}{B^3} \right) \frac{e^{-B^2/2}}{\sqrt{2\pi}} \right).$$

Combining the lower bound of p_0 and upper bound on $\max_{i\neq j} p_{ij}$ we have

$$\lambda \geq p_0 - \min_{i\neq j} p_{ij} \geq \max\left(\frac{1}{2} - \frac{B}{\sqrt{2\pi}}, \; \left(\frac{1}{B} - \frac{1}{B^3} \right) \frac{e^{-B^2/2}}{\sqrt{2\pi}} \right) - e^{-B^2/(1+\Delta)} \frac{\pi - \arctan\left(\frac{\sqrt{1-\Delta^2}}{|\Delta|} \right)}{2\pi}.$$

Applying $\Delta \leq 1 - \delta^2/2$ and noticing that H^∞ is positive semi-definite gives our final result. $\square$

F.8 Generalization

F.8.1 Rademacher complexity

In this section, we would like to compute the Rademacher Complexity of our network. Rademacher complexity is often used to bound the deviation from empirical risk and true risk (see, e.g. [SSBD14].)

Definition F.6 (Empirical Rademacher Complexity)**.** Given n samples S, the empirical Rademacher complexity of a function class $\mathcal{F}$, where $f : \mathbb{R}^d \to \mathbb{R}$ for $f \in \mathcal{F}$, is defined as

$$\mathcal{R}_S(\mathcal{F}) = \frac{1}{n}\mathbb{E}_\epsilon\left[\sup_{f\in\mathcal{F}}\sum_{i=1}^{n}\epsilon_i f(x_i)\right]$$

where $\epsilon = (\epsilon_1, \ldots, \epsilon_n)^\top$ and ϵ_i is an i.i.d Rademacher random variable.

Theorem F.28 ([SSBD14])**.** *Suppose the loss function $\ell(\cdot,\cdot)$ is bounded in $[0, c]$ and is ρ-Lipschitz in the first argument. Then with probability at least $1 - \delta$ over sample S of size n:*

$$\sup_{f\in\mathcal{F}} L_{\mathcal{D}}(f) - L_S(f) \le 2\rho\mathcal{R}_S(\mathcal{F}) + 3c\sqrt{\frac{\log(2/\delta)}{2n}}.$$

In order to get meaningful generalization bound via Rademacher complexity, previous results, such as [ADH$^+$19a, SY19], multiply the neural network by a scaling factor κ to make sure the neural network output something small at the initialization, which requires at least modifying all the previous lemmas we already established. We avoid repeating our arguments by utilizing symmetric initialization to force the neural network to output exactly zero for any inputs at the initialization. [2]

[2]While preparing the manuscript, the authors notice that this can be alternatively solved by reparameterized the neural network by $f(x; W) - f(x; W_0)$ and thus minimizing the following objective $L = \frac{1}{2}\sum_{i=1}^{n}(f(x_i; W) - f(x_i; W_0) - y_i)^2$. The corresponding generalization is the same since Rademacher complexity is invariant to translation. However, since the symmetric initialization is widely adopted in theory literature, we go with symmetric initialization here.

Definition F.7 (Symmetric Initialization)**.** For a one-hidden layer neural network with $2m$ neurons, the network is initialized as the following

1. For $r \in [m]$, initialize $w_r \sim \mathcal{N}(0, I)$ and $a_r \sim \text{Uniform}(\{-1, 1\})$.
2. For $r \in \{m+1, \dots, 2m\}$, let $w_r = w_{r-m}$ and $a_r = -a_{r-m}$.

It is not hard to see that all of our previously established lemmas hold including expectation and concentration. The only effect this symmetric initialization brings is to worse the concentration by a constant factor of 2 which can be easily addressed. For detailed analysis, see [MOSW22].

In order to state our final theorem, we need to use Definition F.2. Now we can state our theorem for generalization.

Theorem F.29. *Fix a failure probability $\delta \in (0, 1)$ and an accuracy parameter $\epsilon \in (0, 1)$. Suppose the training data $S = \{(x_i, y_i)\}_{i=1}^n$ are i.i.d. samples from a (λ, δ, n)-non-degenerate distribution $\mathcal{D}$. Assume the settings in Theorem F.24 except now we let*

$$m \geq \widetilde{\Omega}\left(\lambda^{-4} n^6 \left(1 + \left(\exp(-B^2/2) + 1/m\right) \log^3(2mn/\delta)\right) \exp(-B^2)\right).$$

Consider any loss function $\ell : \mathbb{R} \times \mathbb{R} \to [0, 1]$ that is 1-Lipschitz in its first argument. Then with probability at least $1 - 2\delta - e^{-\Omega(n)}$ over the symmetric initialization of $W(0) \in \mathbb{R}^{m \times d}$ and $a \in \mathbb{R}^m$ and the training samples, the two layer neural network $f(W(k), b(k), a)$ trained by gradient descent for $k \geq \Omega(\frac{1}{\eta\lambda} \log \frac{n \log(1/\delta)}{\epsilon})$ iterations has population loss $L_{\mathcal{D}}(f) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(f(x), y)]$ upper bounded as

$$L_{\mathcal{D}}(f(W(k), b(k), a)) \leq \sqrt{\frac{y^\top (H^\infty)^{-1} y \cdot 32 \exp(-B^2/2)}{n}} + \tilde{O}\left(\frac{1}{n^{1/2}}\right).$$

Proof. First, we need to bound L_S. After training, we have $\|f(k) - y\|_2 \leq \epsilon < 1$, and thus

$$\begin{aligned}L_S(f(W(k),b(k),a)) &= \frac{1}{n}\sum_{i=1}^{n}[\ell(f_i(k),y_i)-\ell(y_i,y_i)]\\ &\le \frac{1}{n}\sum_{i=1}^{n}|f_i(k)-y_i|\\ &\le \frac{1}{\sqrt{n}}\|f(k)-y\|_2\\ &\le \frac{1}{\sqrt{n}}.\end{aligned}$$

By Theorem F.28, we know that

$$\begin{aligned}L_{\mathcal{D}}(f(W(k),b(k),a)) &\le L_S(f(W(k),b(k),a)) + 2\mathcal{R}_S(\mathcal{F}) + \tilde{O}(n^{-1/2})\\ &\le 2\mathcal{R}_S(\mathcal{F}) + \tilde{O}(n^{-1/2}).\end{aligned}$$

Then, by Theorem F.30, we get that for sufficiently large m,

$$\begin{aligned}\mathcal{R}_S(\mathcal{F}) &\le \sqrt{\frac{y^\top (H^\infty)^{-1} y \cdot 8\exp(-B^2/2)}{n}} + \tilde{O}\left(\frac{\exp(-B^2/4)}{n^{1/2}}\right)\\ &\le \sqrt{\frac{y^\top (H^\infty)^{-1} y \cdot 8\exp(-B^2/2)}{n}} + \tilde{O}\left(\frac{1}{n^{1/2}}\right),\end{aligned}$$

where the last step follows from $B > 0$.

Therefore, we conclude that:

$$L_{\mathcal{D}}(f(W(k),b(k),a)) \le \sqrt{\frac{y^\top (H^\infty)^{-1} y \cdot 32\exp(-B^2/2)}{n}} + \tilde{O}\left(\frac{1}{n^{1/2}}\right).$$

□

Theorem F.30. *Fix a failure probability $\delta \in (0,1)$. Suppose the training data $S = \{(x_i,y_i)\}_{i=1}^n$ are i.i.d. samples from a (λ,δ,n)-non-degenerate distribution $\mathcal{D}$. Assume the settings in Theorem F.24 except now we let*

$$m \ge \widetilde{\Omega}\left(\lambda^{-6} n^6 \left(1 + \left(\exp(-B^2/2) + 1/m\right)\log^3(2mn/\delta)\right)\exp(-B^2)\right).$$

Denote the set of one-hidden-layer neural networks trained by gradient descent as $\mathcal{F}$. Then with probability at least $1-2\delta-e^{-\Omega(n)}$ over the randomness in the symmetric initialization and the training data, the set $\mathcal{F}$ has empirical Rademacher complexity bounded as

$$\mathcal{R}_S(\mathcal{F}) \le \sqrt{\frac{y^\top (H^\infty)^{-1} y \cdot 8\exp(-B^2/2)}{n}} + \tilde{O}\left(\frac{\exp(-B^2/4)}{n^{1/2}}\right).$$

Note that the only extra requirement we make on m is the $(n/\lambda)^6$ dependence instead of $(n/\lambda)^4$ which is needed for convergence. The dependence of m on n is significantly better than previous work [SYZ21] where the dependence is n^{14}. We take advantage of our initialization and new analysis to improve the dependence on n.

Proof. Let R_w (R_b) denotes the maximum distance moved any any neuron weight (bias), the same role as D_w (D_b) in Lemma F.15. From Lemma F.15 and Lemma F.17, and we have

$$\max(R_w, R_b) \le O\left(\frac{n\sqrt{1+(\exp(-B^2/2)+1/m)\log^3(2mn/\delta)}}{\sqrt{m}\lambda}\right).$$

The rest of the proof depends on the results from Lemma F.31 and Lemma F.33. Let $R := \|[W,b](k) - [W,b](0)\|_F$. By Lemma F.31 we have

$$\begin{aligned}\mathcal{R}_S(\mathcal{F}_{R_w,R_b,R}) &\le R\sqrt{\frac{8\exp(-B^2/2)}{n}} + 4c(R_w+R_b)^2\sqrt{m}\exp(-B^2/2)\\ &\le R\sqrt{\frac{8\exp(-B^2/2)}{n}} + O\left(\frac{n^2(1+(\exp(-B^2/2)+1/m)\log^3(2mn/\delta))\exp(-B^2/2)}{\sqrt{m}\lambda^2}\right).\end{aligned}$$

Lemma F.33 gives that

$$\begin{aligned}R \le{}& \sqrt{y^\top (H^\infty)^{-1} y} + O\left(\frac{n}{\lambda}\left(\frac{\exp(-B^2/2)\log(n/\delta)}{m}\right)^{1/4}\right) + O\left(\frac{n\sqrt{(R_w+R_b)\exp(-B^2/2)}}{\lambda}\right)\\ &+ \frac{n}{\lambda^2}\cdot O\left(\exp(-B^2/4)\sqrt{\frac{\log(n^2/\delta)}{m}} + (R_w+R_b)\exp(-B^2/2)\right).\end{aligned}$$

Combining the above results and using the choice of m, R, B in Theorem F.24 gives us

$$\mathcal{R}(\mathcal{F}) \leq \sqrt{\frac{y^\top (H^\infty)^{-1} y \cdot 8\exp(-B^2/2)}{n}} + O\left(\frac{\sqrt{n\exp(-B^2/2)}}{\lambda}\left(\frac{\exp(-B^2/2)\log(n/\delta)}{m}\right)^{1/4}\right)$$
$$+ O\left(\frac{\sqrt{n(R_w+R_b)}}{\lambda\exp(B^2/2)}\right) + \frac{\sqrt{n}}{\lambda^2}\cdot O\left(\exp(-B^2/2)\sqrt{\frac{\log(n^2/\delta)}{m}} + (R_w+R_b)\exp(-3B^2/4)\right)$$
$$+ O\left(\frac{n^2(1+(\exp(-B^2/2)+1/m)\log^3(2mn/\delta))\exp(-B^2/2)}{\sqrt{m}\lambda^2}\right).$$

Now, we analyze the terms one by one by plugging in the bound of m and R_w, R_b and show that they can be bounded by $\tilde{O}(\exp(-B^2/4)/n^{1/2})$. For the second term, we have

$$O\left(\frac{\sqrt{n\exp(-B^2/2)}}{\lambda}\left(\frac{\exp(-B^2/2)\log(n/\delta)}{m}\right)^{1/4}\right) = O\left(\frac{\sqrt{\lambda}\exp(-B^2/8)\log^{1/4}(n/\delta)}{n}\right).$$

For the third term, we have

$$\begin{aligned} O\left(\frac{\sqrt{n(R_w+R_b)}}{\lambda\exp(B^2/2)}\right) &= O\left(\frac{\sqrt{n}}{\lambda\exp(B^2/2)}\frac{\sqrt{n}(1+(\exp(-B^2/2)+1/m)\log^3(2mn/\delta))^{1/4}}{m^{1/4}\lambda^{1/2}}\right) \\ &= O\left(\frac{n}{\exp(B^2/2)n^{6/4}\exp(-B^2/4)}\right) \\ &= O\left(\frac{\exp(-B^2/4)}{n^{1/2}}\right). \end{aligned}$$

For the fourth term, we have

$$\begin{aligned} &\frac{\sqrt{n}}{\lambda^2}\cdot O\left(\exp(-B^2/2)\sqrt{\frac{\log(n^2/\delta)}{m}} + (R_w+R_b)\exp(-3B^2/4)\right) \\ &= O\left(\frac{\lambda\sqrt{\log(n/\delta)}}{n^{2.5}}\right) + O\left(\frac{\exp(-B^2/4)}{n^{1.5}}\right). \end{aligned}$$

For the last term, we have

$$\begin{aligned} &O\left(\frac{n^2(1+(\exp(-B^2/2)+1/m)\log^3(2mn/\delta))\exp(-B^2/2)}{\sqrt{m}\lambda^2}\right) \\ &= O\left(\frac{\lambda\sqrt{1+(\exp(-B^2/2)+1/m)\log^3(2mn/\delta)}}{n}\right). \end{aligned}$$

Recall our discussion on λ in Section F.3.4 that $\lambda = \lambda_0 \exp(-B^2/2) \le 1$ for some λ_0 independent of B. Putting them together, we get the desired upper bound for $\mathcal{R}(\mathcal{F})$, and the theorem is then proved. $\square$

Lemma F.31. *Assume the choice of R_w, R_b, m in Theorem F.24. Given $R > 0$, with probability at least $1 - e^{-\Omega(n)}$ over the random initialization of $W(0), a$, the following function class*

$$\begin{aligned}\mathcal{F}_{R_w,R_b,R} = \{f(W,a,b) :\ & \|W - W(0)\|_{2,\infty} \le R_w,\ \|b - b(0)\|_\infty \le R_b, \\ & \left\|[W, b] - [\vec{W}(0), b(0)]\right\| \le R\}\end{aligned}$$

has empirical Rademacher complexity bounded as

$$\mathcal{R}_S(\mathcal{F}_{R_w,R_b,R}) \le R\sqrt{\frac{8\exp(-B^2/2)}{n}} + 4c(R_w + R_b)^2\sqrt{m}\exp(-B^2/2).$$

Proof. We need to upper bound $\mathcal{R}_S(\mathcal{F}_{R_w,R_b,R})$. Define the events

$$A_{r,i} = \{|w_r(0)^\top x_i - b_r(0)| \le R_w + R_b\},\ i \in [n],\ r \in [m]$$

and a shorthand $\mathbb{I}(w_r(0)^\top x_i - B \geq 0) = \mathbb{I}_{r,i}(0)$. Then,

$$\begin{aligned}
&\sum_{i=1}^n \epsilon_i \sum_{r=1}^m a_r \sigma(w_r^\top x_i - b_r) - \sum_{i=1}^n \epsilon_i \sum_{r=1}^m a_r \mathbb{I}_{r,i}(0)(w_r^\top x_i - b_r) \\
&= \sum_{i=1}^n \sum_{r=1}^m \epsilon_i a_r \left(\sigma(w_r^\top x_i - b_r) - \mathbb{I}_{r,i}(0)(w_r^\top x_i - b_r)\right) \\
&= \sum_{i=1}^n \sum_{r=1}^m \mathbb{I}(A_{r,i}) \epsilon_i a_r \left(\sigma(w_r^\top x_i - b_r) - \mathbb{I}_{r,i}(0)(w_r^\top x_i - b_r)\right) \\
&= \sum_{i=1}^n \sum_{r=1}^m \mathbb{I}(A_{r,i}) \epsilon_i a_r \Big(\sigma(w_r^\top x_i - b_r) - \mathbb{I}_{r,i}(0)(w_r(0)^\top x_i - b_r(0)) \\
&\quad - \mathbb{I}_{r,i}(0)((w_r - w_r(0))^\top x_i - (b_r - b_r(0)))\Big) \\
&= \sum_{i=1}^n \sum_{r=1}^m \mathbb{I}(A_{r,i}) \epsilon_i a_r \Big(\sigma(w_r^\top x_i - b_r) - \sigma(w_r(0)^\top x_i - b_r(0)) \\
&\quad - \mathbb{I}_{r,i}(0)((w_r - w_r(0))^\top x_i - (b_r - b_r(0)))\Big) \\
&\leq \sum_{i=1}^n \sum_{r=1}^m \mathbb{I}(A_{r,i}) 2(R_w + R_b),
\end{aligned}$$

where the second equality is due to the fact that $\sigma(w_r^\top x_i - b_r) = \mathbb{I}_{r,i}(0)(w_r^\top x_i - b_r)$ if

$r \notin A_{r,i}$. Thus, the Rademacher complexity can be bounded as

$$
\begin{aligned}
&\mathcal{R}_S(\mathcal{F}_{R_w,R_b,R}) \\
&= \frac{1}{n}\mathbb{E}_\epsilon\left[\sup_{\substack{\|W-W(0)\|_{2,\infty}\le R_w,\ \|b-b(0)\|_\infty \le R_b,\\ \left\|[W,b]-[\vec{W}(0),b(0)]\right\|\le R}} \sum_{i=1}^n \epsilon_i \sum_{r=1}^m \frac{a_r}{\sqrt{m}}\sigma(w_r^\top x_i - b_r)\right] \\
&\le \frac{1}{n}\mathbb{E}_\epsilon\left[\sup_{\substack{\|W-W(0)\|_{2,\infty}\le R_w,\ \|b-b(0)\|_\infty \le R_b,\\ \left\|[W,b]-[\vec{W}(0),b(0)]\right\|\le R}} \sum_{i=1}^n \epsilon_i \sum_{r=1}^m \frac{a_r}{\sqrt{m}}\mathbb{I}_{r,i}(0)(w_r^\top x_i - b_r)\right] + \frac{2(R_w+R_b)}{n\sqrt{m}}\sum_{i=1}^n\sum_{r=1}^m \mathbb{I}(A_{r,i}) \\
&= \frac{1}{n}\mathbb{E}_\epsilon\left[\sup_{\left\|[W,b]-[\vec{W}(0),b(0)]\right\|\le R} [\vec{W},b]^\top Z(0)\epsilon\right] + \frac{2(R_w+R_b)}{n\sqrt{m}}\sum_{i=1}^n\sum_{r=1}^m \mathbb{I}(A_{r,i}) \\
&= \frac{1}{n}\mathbb{E}_\epsilon\left[\sup_{\left\|[W,b]-[\vec{W}(0),b(0)]\right\|\le R} [W,b]-[\vec{W}(0),b(0)]^\top Z(0)\epsilon\right] + \frac{2(R_w+R_b)}{n\sqrt{m}}\sum_{i=1}^n\sum_{r=1}^m \mathbb{I}(A_{r,i}) \\
&\le \frac{1}{n}\mathbb{E}_\epsilon[R\left\|Z(0)\epsilon\right\|_2] + \frac{2(R_w+R_b)}{n\sqrt{m}}\sum_{i=1}^n\sum_{r=1}^m \mathbb{I}(A_{r,i}) \\
&\le \frac{R}{n}\sqrt{\mathbb{E}_\epsilon[\left\|Z(0)\epsilon\right\|_2^2]} + \frac{2(R_w+R_b)}{n\sqrt{m}}\sum_{i=1}^n\sum_{r=1}^m \mathbb{I}(A_{r,i}) \\
&= \frac{R}{n}\left\|Z(0)\right\|_F + \frac{2(R_w+R_b)}{n\sqrt{m}}\sum_{i=1}^n\sum_{r=1}^m \mathbb{I}(A_{r,i}),
\end{aligned}
$$

where we recall the definition of the matrix

$$
Z(0) = \frac{1}{\sqrt{m}}\begin{bmatrix} \mathbb{I}_{1,1}(0)a_1[x_1^\top,-1]^\top & \dots & \mathbb{I}_{1,n}(0)a_1[x_n^\top,-1]^\top \\ \vdots & & \vdots \\ \mathbb{I}_{m,1}(0)a_m[x_1^\top,-1]^\top & \dots & \mathbb{I}_{m,n}(0)a_m[x_n^\top,-1]^\top \end{bmatrix} \in \mathbb{R}^{m(d+1)\times n}.
$$

By Lemma F.25, we have $\left\|Z(0)\right\|_F \le \sqrt{8n\exp(-B^2/2)}$ and by Corollary F.12, we have

$$
\mathbb{P}\left[\forall i \in [n]:\ \sum_{r=1}^m \mathbb{I}(A_{r,i}) \le 2mc(R_w+R_b)\exp(-B^2/2)\right] \ge 1 - e^{-\Omega(n)}.
$$

Thus, with probability at least $1 - e^{-\Omega(n)}$, we have

$$\mathcal{R}_S(\mathcal{F}_{R_w,R_b,R}) \le R\sqrt{\frac{8\exp(-B^2/2)}{n}} + 4c(R_w + R_b)^2\sqrt{m}\exp(-B^2/2).$$

$\square$

F.8.2 Analysis of radius

Theorem F.32. *Assume the parameter settings in Theorem F.24. With probability at least* $1 - \delta - e^{-\Omega(n)}$ *over the initialization we have*

$$f(k) - y = -(I - \eta H^{\infty})^k y \pm e(k),$$

where

$$\|e(k)\|_2 = k(1 - \eta\lambda/4)^{(k-1)/2}\eta n^{3/2} \cdot O\left(\exp(-B^2/4)\sqrt{\frac{\log(n^2/\delta)}{m}} + (R_w + R_b)\exp(-B^2/2)\right).$$

Proof. Before we start, we assume all the events needed in Theorem F.24 succeed, which happens with probability at least $1 - \delta - e^{-\Omega(n)}$.

Recall the no-flipping set S_i in Definition F.4. We have

$$\begin{aligned} f_i(k+1) - f_i(k) &= \frac{1}{\sqrt{m}}\sum_{r=1}^{m} a_r[\sigma(w_r(k+1)^\top x_i - b_r(k+1)) - \sigma(w_r(k)^\top x_i - b_r(k))] \\ &= \frac{1}{\sqrt{m}}\sum_{r\in S_i} a_r[\sigma(w_r(k+1)^\top x_i - b_r(k+1)) - \sigma(w_r(k)^\top x_i - b_r(k))] \\ &\quad + \underbrace{\frac{1}{\sqrt{m}}\sum_{r\in \overline{S}_i} a_r[\sigma(w_r(k+1)^\top x_i - b_r(k+1)) - \sigma(w_r(k)^\top x_i - b_r(k))]}_{\epsilon_i(k)}. \end{aligned} \tag{F.5}$$

Now, to upper bound the second term $\epsilon_i(k)$,

$$\begin{aligned}
|\epsilon_i(k)| &= \left|\frac{1}{\sqrt{m}}\sum_{r\in\overline{S}_i} a_r[\sigma(w_r(k+1)^\top x_i - b_r(k+1)) - \sigma(w_r(k)^\top x_i - b_r(k))]\right| \\
&\le \frac{1}{\sqrt{m}}\sum_{r\in\overline{S}_i} |w_r(k+1)^\top x_i - b_r(k+1) - (w_r(k)^\top x_i - b_r(k))| \\
&\le \frac{1}{\sqrt{m}}\sum_{r\in\overline{S}_i} \|w_r(k+1) - w_r(k)\|_2 + |b_r(k+1) - b_r(k)| \\
&= \frac{1}{\sqrt{m}}\sum_{r\in\overline{S}_i} \left\|\frac{\eta}{\sqrt{m}} a_r \sum_{j=1}^{n} (f_j(k) - y_j)\mathbb{I}_{r,j}(k) x_j\right\|_2 + \left|\frac{\eta}{\sqrt{m}} a_r \sum_{j=1}^{n} (f_j(k) - y_j)\mathbb{I}_{r,j}(k)\right| \\
&\le \frac{2\eta}{m}\sum_{r\in\overline{S}_i}\sum_{j=1}^{n} |f_j(k) - y_j| \\
&\le \frac{2\eta\sqrt{n}|\overline{S}_i|}{m}\|f(k) - y\|_2 \\
\Rightarrow \|\epsilon\|_2 &= \sqrt{\sum_{i=1}^{n}\frac{4\eta^2 n|\overline{S}_i|^2}{m^2}\|f(k) - y\|_2^2} \le \eta n O((R_w + R_b)\exp(-B^2/2))\|f(k) - y\|_2
\end{aligned} \tag{F.6}$$

where we apply Corollary F.12 in the last inequality. To bound the first term,

$$\begin{aligned}
&\frac{1}{\sqrt{m}}\sum_{r\in S_i} a_r[\sigma(w_r(k+1)^\top x_i - b_r(k+1)) - \sigma(w_r(k)^\top x_i - b_r(k))]\\
&= \frac{1}{\sqrt{m}}\sum_{r\in S_i} a_r \mathbb{I}_{r,i}(k)\left((w_r(k+1)-w_r(k))^\top x_i - (b_r(k+1)-b_r(k))\right)\\
&= \frac{1}{\sqrt{m}}\sum_{r\in S_i} a_r \mathbb{I}_{r,i}(k)\left(\left(-\frac{\eta}{\sqrt{m}}a_r\sum_{j=1}^n (f_j(k)-y_j)\mathbb{I}_{r,j}(k)x_j\right)^\top x_i - \frac{\eta}{\sqrt{m}}a_r\sum_{j=1}^n (f_j(k)-y_j)\mathbb{I}_{r,j}(k)\right)\\
&= \frac{1}{\sqrt{m}}\sum_{r\in S_i} a_r \mathbb{I}_{r,i}(k)\left(-\frac{\eta}{\sqrt{m}}a_r\sum_{j=1}^n (f_j(k)-y_j)\mathbb{I}_{r,j}(k)(x_j^\top x_i+1)\right)\\
&= -\eta\sum_{j=1}^n (f_j(k)-y_j)\frac{1}{m}\sum_{r\in S_i}\mathbb{I}_{r,i}(k)\mathbb{I}_{r,j}(k)(x_j^\top x_i+1)\\
&= -\eta\sum_{j=1}^n (f_j(k)-y_j)H_{ij}(k) + \underbrace{\eta\sum_{j=1}^n (f_j(k)-y_j)\frac{1}{m}\sum_{r\in \overline{S}_i}\mathbb{I}_{r,i}(k)\mathbb{I}_{r,j}(k)(x_j^\top x_i+1)}_{\epsilon_i'(k)} \quad \text{(F.7)}
\end{aligned}$$

where we can upper bound $|\epsilon_i'(k)|$ as

$$|\epsilon_i'(k)| \le \frac{2\eta}{m}|\overline{S}_i|\sum_{j=1}^n |f_j(k)-y_j| \le \frac{2\eta\sqrt{n}|\overline{S}_i|}{m}\|f(k)-y\|_2$$

$$\Rightarrow \|\epsilon'\|_2 = \sqrt{\sum_{i=1}^n \frac{4\eta^2 n|\overline{S}_i|^2}{m^2}\|f(k)-y\|_2^2} \le \eta n O((R_w+R_b)\exp(-B^2/2))\|f(k)-y\|_2. \tag{F.8}$$

Combining Equation (F.5), Equation (F.6), Equation (F.7) and Equation (F.8), we have

$$\begin{aligned}
&f_i(k+1) - f_i(k) = -\eta \sum_{j=1}^{n} (f_j(k) - y_j) H_{ij}(k) + \epsilon_i(k) + \epsilon_i'(k) \\
\Rightarrow\ & f(k+1) - f(k) = -\eta H(k)(f(k) - y) + \epsilon(k) + \epsilon'(k) \\
&\qquad = -\eta H^\infty (f(k) - y) + \underbrace{\eta(H^\infty - H(k))(f(k) - y) + \epsilon(k) + \epsilon'(k)}_{\zeta(k)} \\
&\Rightarrow f(k) - y = (I - \eta H^\infty)^k (f(0) - y) + \sum_{t=0}^{k-1} (I - \eta H^\infty)^t \zeta(k-1-t) \\
&\qquad = -(I - \eta H^\infty)^k y + \underbrace{(I - \eta H^\infty)^k f(0) + \sum_{t=0}^{k-1} (I - \eta H^\infty)^t \zeta(k-1-t)}_{e(k)}.
\end{aligned}$$

Now the rest of the proof bounds the magnitude of $e(k)$. From Lemma F.10 and Lemma F.13, we have

$$\begin{aligned}
\|H^\infty - H(k)\|_2 &\le \|H(0) - H^\infty\|_2 + \|H(0) - H(k)\|_2 \\
&= O\left(n \exp(-B^2/4) \sqrt{\frac{\log(n^2/\delta)}{m}} \right) + O(n(R_w + R_b)\exp(-B^2/2)).
\end{aligned}$$

Thus, we can bound $\zeta(k)$ as

$$\begin{aligned}
\|\zeta(k)\|_2 &\le \eta \|H^\infty - H(k)\|_2 \|f(k) - y\|_2 + \|\epsilon(k)\|_2 + \|\epsilon'(k)\|_2 \\
&= O\left(\eta n \left(\exp(-B^2/4) \sqrt{\frac{\log(n^2/\delta)}{m}} + (R_w + R_b)\exp(-B^2/2) \right) \right) \|f(k) - y\|_2 .
\end{aligned}$$

Notice that $\|H^\infty\|_2 \le \mathrm{Tr}(H^\infty) \le n$ since H^∞ is symmetric. By Theorem F.24, we pick $\eta = O(\lambda/n^2) \ll 1/\|H^\infty\|_2$ and, with probability at least $1 - \delta - e^{-\Omega(n)}$ over the random initialization, we have $\|f(k) - y\|_2 \le (1 - \eta\lambda/4)^{k/2} \|f(0) - y\|_2$.

Since we are using symmetric initialization, we have $(I - \eta H^\infty)^k f(0) = 0$.

Thus,

$$\begin{aligned}
\|e(k)\|_2 &= \left\|\sum_{t=0}^{k-1}(I-\eta H^\infty)^t\zeta(k-1-t)\right\|_2 \\
&\le \sum_{t=0}^{k-1}\|I-\eta H^\infty\|_2^t\,\|\zeta(k-1-t)\|_2 \\
&\le \sum_{t=0}^{k-1}(1-\eta\lambda)^t\eta n O\left(\exp(-B^2/4)\sqrt{\frac{\log(n^2/\delta)}{m}}+(R_w+R_b)\exp(-B^2/2)\right) \\
&\quad\cdot\|f(k-1-t)-y\|_2 \\
&\le \sum_{t=0}^{k-1}(1-\eta\lambda)^t\eta n O\left(\exp(-B^2/4)\sqrt{\frac{\log(n^2/\delta)}{m}}+(R_w+R_b)\exp(-B^2/2)\right) \\
&\quad\cdot(1-\eta\lambda/4)^{(k-1-t)/2}\|f(0)-y\|_2 \\
&\le k(1-\eta\lambda/4)^{(k-1)/2}\eta n O\left(\exp(-B^2/4)\sqrt{\frac{\log(n^2/\delta)}{m}}+(R_w+R_b)\exp(-B^2/2)\right) \\
&\quad\cdot\|f(0)-y\|_2 \\
&\le k(1-\eta\lambda/4)^{(k-1)/2}\eta n^{3/2} O\Bigg(\left(\exp(-B^2/4)\sqrt{\frac{\log(n^2/\delta)}{m}}+(R_w+R_b)\exp(-B^2/2)\right) \\
&\quad\cdot\left(\sqrt{1+(\exp(-B^2/2)+1/m)\log^3(2mn/\delta)}\right)\Bigg) \\
&= k(1-\eta\lambda/8)^{k-1}\eta n^{3/2} O\left(\exp(-B^2/4)\sqrt{\frac{\log(n^2/\delta)}{m}}+(R_w+R_b)\exp(-B^2/2)\right).
\end{aligned}$$

□

Lemma F.33. *Assume the parameter settings in Theorem F.24. Then with probability at least $1-\delta-e^{-\Omega(n)}$ over the random initialization, we have for all $k\ge 0$,*

$$\begin{aligned}
\|[W,b](k)-[W,b](0)\|_F \le \sqrt{y^\top(H^\infty)^{-1}y} &+ O\left(\frac{n}{\lambda}\left(\frac{\exp(-B^2/2)\log(n/\delta)}{m}\right)^{1/4}\right) \\
&+ O\left(\frac{n\sqrt{R\exp(-B^2/2)}}{\lambda}\right) \\
&+ \frac{n}{\lambda^2}\cdot O\left(\exp(-B^2/4)\sqrt{\frac{\log(n^2/\delta)}{m}}+R\exp(-B^2/2)\right)
\end{aligned}$$

where $R = R_w + R_b$.

Proof. Before we start, we assume all the events needed in Theorem F.24 succeed, which happens with probability at least $1 - \delta - e^{-\Omega(n)}$.

$$\begin{aligned}
&[W, \vec{b}](K) - [W, \vec{b}](0) \\
&= \sum_{k=0}^{K-1} [W, b](\vec{k}+1) - [W, \vec{b}](k) \\
&= -\sum_{k=0}^{K-1} Z(k)(u(k) - y) \\
&= \sum_{k=0}^{K-1} \eta Z(k)((I - \eta H^{\infty})^k y - e(k)) \\
&= \sum_{k=0}^{K-1} \eta Z(k)(I - \eta H^{\infty})^k y - \sum_{k=0}^{K-1} \eta Z(k) e(k) \\
&= \underbrace{\sum_{k=0}^{K-1} \eta Z(0)(I - \eta H^{\infty})^k y}_{T_1} + \underbrace{\sum_{k=0}^{K-1} \eta (Z(k) - Z(0))(I - \eta H^{\infty})^k y}_{T_2} - \underbrace{\sum_{k=0}^{K-1} \eta Z(k) e(k)}_{T_3}.
\end{aligned} \tag{F.9}$$

Now, by Lemma F.13, we have $\|Z(k) - Z(0)\|_F \le O(\sqrt{nR\exp(-B^2/2)})$ which implies

$$\begin{aligned}
\|T_2\|_2 &= \left\| \sum_{k=0}^{K-1} \eta (Z(k) - Z(0))(I - \eta H^{\infty})^k y \right\|_2 \\
&\le \sum_{k=0}^{K-1} \eta \cdot O(\sqrt{nR\exp(-B^2/2)}) \|I - \eta H^{\infty}\|_2^k \|y\|_2 \\
&\le \eta \cdot O(\sqrt{nR\exp(-B^2/2)}) \sum_{k=0}^{K-1} (1 - \eta\lambda)^k \sqrt{n} \\
&= O\left(\frac{n\sqrt{R\exp(-B^2/2)}}{\lambda} \right).
\end{aligned} \tag{F.10}$$

By $\|Z(k)\|_2 \le \|Z(k)\|_F \le \sqrt{2n}$, we get

$$\begin{aligned}
\|T_3\|_2 &= \left\| \sum_{k=0}^{K-1} \eta Z(k) e(k) \right\|_2 \\
&\le \sum_{k=0}^{K-1} \eta \sqrt{2n} \left(k(1-\eta\lambda/8)^{k-1} \eta n^{3/2} O\left(\exp(-B^2/4) \sqrt{\frac{\log(n^2/\delta)}{m}} + R\exp(-B^2/2) \right) \right) \\
&= \frac{n}{\lambda^2} \cdot O\left(\exp(-B^2/4) \sqrt{\frac{\log(n^2/\delta)}{m}} + R\exp(-B^2/2) \right). \qquad \text{(F.11)}
\end{aligned}$$

Define $T = \eta \sum_{k=0}^{K-1} (I - \eta H^\infty)^k$. By Lemma F.10, we know $\|H(0) - H^\infty\|_2 \le O(n\exp(-B^2/4)\sqrt{\frac{\log(n/\delta)}{m}})$ and this implies

$$\begin{aligned}
\|T_1\|_2^2 &= \left\| \sum_{k=0}^{K-1} \eta Z(0)(I - \eta H^\infty)^k y \right\|_2^2 \\
&= \|Z(0)Ty\|_2^2 \\
&= y^\top T Z(0)^\top Z(0) T y \\
&= y^\top T H(0) T y \\
&\le y^\top T H^\infty T y + \|H(0) - H^\infty\|_2 \|T\|_2^2 \|y\|_2^2 \\
&\le y^\top T H^\infty T y + O\left(n\exp(-B^2/4) \sqrt{\frac{\log(n/\delta)}{m}} \right) \left(\eta \sum_{k=0}^{K-1} (1-\eta\lambda)^k \right)^2 n \\
&= y^\top T H^\infty T y + O\left(\frac{n^2 \exp(-B^2/4)}{\lambda^2} \sqrt{\frac{\log(n/\delta)}{m}} \right).
\end{aligned}$$

Let $H^\infty = U\Sigma U^\top$ be the eigendecomposition. Then

$$T = U\left(\eta \sum_{k=0}^{K-1} (I - \eta\Sigma)^k \right) U^\top = U((I - (I-\eta\Sigma)^K)\Sigma^{-1})U^\top$$

$$\Rightarrow T H^\infty T = U((I - (I-\eta\Sigma)^K)\Sigma^{-1})^2 \Sigma U^\top = U(I - (I-\eta\Sigma)^K)^2 \Sigma^{-1} U^\top \preceq U\Sigma^{-1}U^\top = (H^\infty)^{-1}.$$

Thus,

$$\begin{aligned}
\|T_1\|_2^2 &= \left\| \sum_{k=0}^{K-1} \eta Z(0)(I - \eta H^\infty)^k y \right\|_2 \\
&\leq \sqrt{y^\top (H^\infty)^{-1} y + O\left(\frac{n^2 \exp(-B^2/4)}{\lambda^2} \sqrt{\frac{\log(n/\delta)}{m}} \right)} \\
&\leq \sqrt{y^\top (H^\infty)^{-1} y} + O\left(\frac{n}{\lambda} \left(\frac{\exp(-B^2/2)\log(n/\delta)}{m} \right)^{1/4} \right). \qquad \text{(F.12)}
\end{aligned}$$

Finally, plugging in the bounds in Equation (F.9), Equation (F.12), Equation (F.10), and Equation (F.11), we have

$$\begin{aligned}
&\|[W, b](K) - [W, b](0)\|_F \\
&= \left\| [W, \vec{b}](K) - [W, \vec{b}](0) \right\|_2 \\
&\leq \sqrt{y^\top (H^\infty)^{-1} y} + O\left(\frac{n}{\lambda} \left(\frac{\exp(-B^2/2)\log(n/\delta)}{m} \right)^{1/4} \right) \\
&\quad + O\left(\frac{n\sqrt{R \exp(-B^2/2)}}{\lambda} \right) \\
&\quad + \frac{n}{\lambda^2} \cdot O\left(\exp(-B^2/4) \sqrt{\frac{\log(n^2/\delta)}{m}} + R \exp(-B^2/2) \right).
\end{aligned}$$

□

F.9 The Benefit of Constant Initialization of Biases

In short, the benefit of constant initialization of biases lies in inducing sparsity in activation and thus reducing the per step training cost. This is the main motivation of our work on studying sparsity from a deep learning theory perspective. Since our convergence shows that sparsity doesn't change convergence rate, the total training cost is also reduced.

To address the width's dependence on B, our argument goes like follows. In practice, people set up neural network models by first picking a neural network of some pre-chosen size and then choose other hyper-parameters such as learning rate, initialization scale, etc. In our case, the hyper-parameter is the bias initialization. Thus, *the network width is picked before B.* Let's say we want to apply our theoretical result to guide our practice. Since we usually don't know the exact data separation and the minimum eigenvalue of the NTK, we don't have a good estimate on the exact width needed for the network to converge and generalize. We may pick a network with width that is much larger than needed (e.g. we pick a network of width $\Omega(n^{12})$ whereas only $\Omega(n^4)$ is needed; this is possible because the smallest eigenvalue of NTK can range from $[\Omega(1/n^2), O(1)]$). Also, it is an empirical observation that the neural networks used in practice are very overparameterized and there is always room for sparsification. If the network width is very large, then per step gradient descent is very costly since the cost scales linearly with width and can be improved to scale linearly with the number of active neurons if done smartly. If the bias is initialized to zero (as people usually do in practice), then the number of active neurons is $O(m)$. However, since we can sparsify the neural network activation by non-zero bias initialization, the number of active neurons can scale *sub-linearly* in m. Thus, if the neural network width we choose at the beginning is much larger than needed, then we are indeed able to obtain total training cost reduction by this initialization. The above is an informal description of the result proven in [SYZ21] and the message is *sparsity can help reduce the per step training cost.* If the network width is pre-chosen, then the lower bound on network width $m \geq \tilde{\Omega}(\lambda_0^{-4} n^4 \exp(B^2))$ in Theorem 3.1 can be translated into an upper bound on bias initialization: $B \leq \tilde{O}(\sqrt{\log \frac{\lambda_0^4 m}{n^4}})$ if $m \geq \tilde{\Omega}(\lambda_0^{-4} n^4)$. This would be a more appropriate interpretation of our result. Note that this is different from how Theorem 3.1 is presented: first pick B and then choose m; since m is picked later, m can always satisfy $B \leq \sqrt{0.5 \log m}$ and $m \geq \tilde{\Omega}(\lambda_0^{-4} n^4 \exp(B^2))$. Of course, we don't know the best (largest) possible B that works but as long as we can get some B to work, we can get computational gain from sparsity.

In summary, sparsity can reduce the per step training cost since we don't know the exact width needed for the network to converge and generalize. Our result should be interpreted as an upper bound on B since the width is always chosen before B in practice.

Appendix G: Omitted Materials from Chapter 18

G.1 Cryptographic Primitives

G.1.1 Public-key quantum money

Definition G.1 (Public Key Quantum Money)**.** A public-key (publicly-verifiable) quantum money consists of the following algorithms:

- $\mathsf{KeyGen}(1^\lambda)$: takes as input a security parameter λ, and generates a key pair $(\mathsf{sk}, \mathsf{pk})$.
- $\mathsf{GenNote}(\mathsf{sk})$: takes a secret key sk and generates a quantum banknote state $|\$\rangle$.
- $\mathsf{Ver}(\mathsf{pk}, |\$'\rangle)$: takes a public key pk, and a claimed money state $|\$'\rangle$, and outputs either 1 for accepting or 0 for rejecting.

A secure public-key quantum money should satisfy the following properties:

Verification Correctness: there exists a negligible function $\mathsf{negl}(\cdot)$ such that the following holds for any $\lambda \in \mathbb{N}$,

$$\Pr_{(\mathsf{sk},\mathsf{pk})\leftarrow\mathsf{KeyGen}(1^\lambda)}[\mathsf{Ver}(\mathsf{pk}, \mathsf{GenNote}(\mathsf{sk})) = \ 1] \geq 1 - \mathsf{negl}(\lambda)$$

Unclonable Security: Suppose a QPT adversary is given $q = \mathsf{poly}(\lambda)$ number of valid banknotes $\{\rho_i\}_{i\in[q]}$ and then generates $q' = q + 1$ banknotes $\{\rho'_j\}_{j\in[q']}$ where ρ'_j are potentially entangled, there exists a negligible function $\mathsf{negl}(\cdot)$, for all $\lambda \in \mathbb{N}$,

$$\Pr_{(\mathsf{sk},\mathsf{pk})\leftarrow\mathsf{KeyGen}(1^\lambda)}\left[\forall i \in [q'], \mathsf{Ver}(\mathsf{pk}, \rho'_j) = 1 : \{\rho'_j\} \leftarrow \mathcal{A}(1^\lambda, \{\rho_i\}\right] \leq \mathsf{negl}(\lambda)$$

Remark G.1. In rest of the chapter, q is set to be 1 for simplicity, and the scheme satisfies unclonable security if $\mathcal{A}$ cannot produce two banknotes that pass verification.

[AC12] shows that any public-key quantum money scheme that satisfies security when $q = 1$, can be generalized to a scheme that is secure when $q = \mathsf{poly}(\lambda)$, using quantum-secure digital signatures.

A non-perturb property is also required. That is, one can verify a quantum banknote polynomially many times and the banknote is still a valid banknote. Since Ver is almost a deterministic function, by Gentle Measurement Lemma (Lemma 18.1), the above definition implies the non-perturb property.

In some settings, instead of outputting $0/1$, Ver is required to output either $\perp$ which indicates the verification fails, or a serial number $s \in \mathcal{S}_\lambda$ if it passes the verification. In this case, the scheme should satisfy the following correctness (unique serial number property) and unclonable security [Zha19]:

Unique Serial Number: For a money state $|\$\rangle$, let $H_\infty(|\$\rangle) = -\log\min_s \Pr[\mathsf{Ver}(|\$\rangle) = s]$. We say a quantum scheme has unique serial number property, if $\mathbb{E}[H_\infty(|\$\rangle)]$ is negligible for all λ, $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $|\$\rangle$ is sampled from $\mathsf{GenNote}(\mathsf{sk})$.

Unclonable Security: Consider the following game with a challenger and an adversary,

1. The challenger runs $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $|\$\rangle \leftarrow \mathsf{GenNote}(\mathsf{sk})$, it then runs Ver to get a serial number s.
2. $\mathcal{A}$ is given the public key pk, the banknote $|\$\rangle$ and the serial number s.
3. $\mathcal{A}$ produces σ^* (which contains two separate registers, but they may be entangled) and denotes $\sigma_1 = \mathsf{Tr}_2[\sigma^*]$ and $\sigma_2 = \mathsf{Tr}_1[\sigma^*]$.
4. $\mathcal{A}$ wins if and only if $\mathsf{Ver}(\sigma_1) = \mathsf{Ver}(\sigma_2) = s$.

We say a public key quantum money scheme is secure, if for all QPT $\mathcal{A}$, it wins the above game with negligible probability in λ.

G.1.2 Obfuscation

Definition G.2 (Virtual Black-Box Obfuscation, [BGI+01])**.** An obfuscator $\mathcal{O}$ (with auxiliary input) for a collection of circuits $\mathcal{C} = \bigcup_{\lambda \in \mathbb{N}} \mathcal{C}_\lambda$ is a (worst-case) VBB obfuscator if it satisfies:

- **Functionality-Preserving:** For every $C \in \mathcal{C}$, every input x, $\Pr[\mathcal{O}(C)(x) = C(x)] = 1$.
- **Virtual Black-Box:** For every poly-size adversary $\mathcal{A}$, there exists a poly-size simulator $\mathcal{S}$, such that for every $\lambda \in \mathbb{N}$, auxiliary input $\mathsf{aux} \in \{0,1\}^{\mathsf{poly}(\lambda)}$, and every predicate $\pi : \mathcal{C}_\lambda \to \{0,1\}$, and every $\mathcal{C} \in \mathcal{C}_\lambda$:

  $$\left| \Pr_{\mathcal{A},\mathcal{O}}[\mathcal{A}(\mathcal{O}(C), \mathsf{aux}) = \pi(C)] - \Pr_{\mathcal{S}}[\mathcal{S}^C(1^\lambda, \mathsf{aux})) = \pi(C)] \right| \le \mathsf{negl}(\lambda)$$

  where the probability is over $C \leftarrow \mathcal{C}_\lambda$, and the randomness of the algorithms $\mathcal{O}, \mathcal{A}$ and $\mathcal{S}$.

G.2 Missing Details For Threshold Implementation

G.2.1 Proof of Theorem 18.6

Theorem G.1 (Theorem 18.6, restated)**.** *For any $\epsilon, \delta, \gamma, \mathcal{P}, D$, the algorithm of measurement $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma}$ that satisfies the followings:*

- *For all quantum state ρ, $\mathrm{Tr}[\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma-\epsilon} \cdot \rho] \ge \mathrm{Tr}[\mathsf{TI}_\gamma(\mathcal{P}_D) \cdot \rho] - \delta$.*
- *By symmetry, for all quantum state ρ, $\mathrm{Tr}[\mathsf{TI}_{\gamma-\epsilon}(\mathcal{P}_D) \cdot \rho] \ge \mathrm{Tr}[\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma} \cdot \rho] - \delta$.*
- *For all quantum state ρ, let ρ' be the collapsed state after applying $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma}$ on ρ. Then, $\mathrm{Tr}[\mathsf{TI}_{\gamma-2\epsilon}(P_D) \cdot \rho'] \ge 1 - 2\delta$.*
- *The expected running time is the same as $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}$.*

We give the following fact before proving the theorem.

Fact G.2. *Let D_0, D_1 be two real-valued probability distributions with shift distance $\Delta^{\epsilon}_{\mathsf{Shift}} = \delta$. Then, we have*

$$\Pr[D_0 \geq x - \epsilon] \geq \Pr[D_1 > x] - \delta, \quad \textit{and}$$
$$\Pr[D_1 \geq x - \epsilon] \geq \Pr[D_0 > x] - \delta$$

Proof. We prove the first inequality. By the definition of shift distance, we have

$$\Pr[D_0 \leq x - \epsilon] \leq \Pr[D_1 \leq x] + \delta.$$

Then, $\Pr[D_0 \geq x - \epsilon] = 1 - \Pr[D_0 \leq x - \epsilon] \geq 1 - \Pr[D_1 \leq x] - \delta = \Pr[D_1 \geq x] - \delta$. The second inequality can be proved in a symmetric way. □

Now, we prove the Theorem 18.6 in below.

Proof. By Theorem 18.5, we know that there exists an algorithm $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}$ that approximates the measurement of $\mathsf{ProjImp}(\mathsf{P}_D)$, i.e.,

$$\Delta^{\epsilon}_{\mathsf{Shift}}(\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}, \mathsf{ProjImp}(\mathcal{P}_D)) \leq \delta.$$

In particular, for any pure quantum state $|\psi\rangle$, let D_A be the distribution of $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}(|\psi\rangle)$ and D_P be the distribution of $\mathsf{ProjImp}(\mathcal{P}_D)(|\psi\rangle)$.

Then, by Fact G.2, we have

$$\Pr[D_A \geq \gamma - \epsilon] \geq \Pr[D_P \geq \gamma] - \delta,$$

Hence, by the definition of threshold implementation (Definition 18.9) and the construction of the algorithm $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma}$, we can get

$$\mathrm{Tr}\left[\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma-\epsilon} |\psi\rangle \langle\psi|\right] \geq \mathrm{Tr}\left[\mathsf{TI}_{\gamma}(\mathcal{P}_D) |\psi\rangle \langle\psi|\right] - \delta.$$

Note that mixed state is just a convex combination of pure states. Hence, by the linearity of trace, for any mixed state ρ, we have

$$\mathrm{Tr}\left[\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma-\epsilon} \cdot \rho\right] \geq \mathrm{Tr}\left[\mathsf{TI}_{\gamma}(\mathcal{P}_D) \cdot \rho\right] - \delta,$$

which proves the first bullet. The second bullet follows the same idea by symmetry.

For the third bullet, notice that the measurement algorithms $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma}$ and $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}$ do the same thing to the quantum state. So, ρ' is also the collapsed state after the measurement of $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}(\rho)$.

Since we assume that the outcome of $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma}(\rho)$ is 0, it implies the corresponding outcome of $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}(\rho)$ is at least γ.

By Theorem 18.5, $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}$ is (ϵ,δ)-almost projective, which means that if we apply $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}$ (again) to ρ', the outcome satisfies

$$\Pr[\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}(\rho') < \gamma - \epsilon] < \delta.$$

Theorem 18.5 also provides that the shift distance between $\mathsf{ProjImp}$ and API is small, which means

$$\Pr[\mathsf{ProjImp}(\mathcal{P}_D)(\rho') \leq \gamma - 2\epsilon] \leq \Pr[\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}(\rho') < \gamma - 2\epsilon + \epsilon] + \delta \leq 2\delta.$$

Hence,

$$\mathrm{Tr}[\mathsf{TI}_{\gamma-2\epsilon} \cdot \rho'] = 1 - \Pr[\mathsf{ProjImp}(\mathcal{P}_D)(\rho') \leq \gamma - 2\epsilon] \geq 1 - 2\delta.$$

The third bullet easily follows from the construction. $\square$

G.2.2 Proof of Lemma 18.7

Lemma G.3 (Lemma 18.7, restated)**.** *Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be two collections of projective measurements and D_1 and D_2 be any probability distributions defined on the index set of $\mathcal{P}_1$ and $\mathcal{P}_2$ respectively. For any $0 < \epsilon, \delta, \gamma < 1$, the algorithms $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_1,D_1,\gamma}$ and $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_2,D_2,\gamma}$ satisfy the followings:*

- *For any bipartite (possibly entangled, mixed) quantum state $\rho \in \mathscr{H}_{\mathcal{L}} \otimes \mathscr{H}_{\mathcal{R}}$,*

$$\mathrm{Tr}\left[\left(\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_1,D_1,\gamma-\epsilon} \otimes \mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_2,D_2,\gamma-\epsilon}\right)\rho\right] \geq \mathrm{Tr}\left[\left(\mathsf{TI}_\gamma(\mathcal{P}_{D_1}) \otimes \mathsf{TI}_\gamma(\mathcal{P}_{D_2})\right)\rho\right] - 2\delta.$$

- *For any (possibly entangled, mixed) quantum state ρ, let ρ' be the collapsed state after applying $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_1,D_1,\gamma} \otimes \mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_2,D_2,\gamma}$ on ρ (and normalized). Then,*

$$\operatorname{Tr}\left[\left(\mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_1}) \otimes \mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_2})\right)\rho'\right] \geq 1 - 4\delta.$$

Here $\mathcal{P}_{D_i}$ stands for the mixture of projective measurement $\mathcal{P}_i$ relative to D_i (see Definition 18.7).

Proof. We use the hybrid argument to show that $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_1,D_1,\gamma-\epsilon} \otimes \mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_2,D_2,\gamma-\epsilon}$ approximates $\mathsf{TI}_\gamma(\mathcal{P}_{D_1}) \otimes \mathsf{TI}_\gamma(\mathcal{P}_{D_2})$.

For brevity, let ATI_1 denote $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_1,D_1,\gamma-\epsilon}$ and ATI_2 denote $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_2,D_2,\gamma-\epsilon}$. Similarly, let TI_1 denote $\mathsf{TI}_\gamma(\mathcal{P}_{D_1})$ and TI_2 denote $\mathsf{TI}_\gamma(\mathcal{P}_{D_2})$.

We first show that,

$$\operatorname{Tr}[(\mathsf{TI}_1 \otimes \mathsf{TI}_2)\rho] \leq \operatorname{Tr}[(\mathsf{TI}_1 \otimes \mathsf{ATI}_2)\rho] + \delta. \tag{G.1}$$

Note that ρ is a bipartite quantum state in $\mathscr{H}_{\mathcal{L}} \otimes \mathscr{H}_{\mathcal{R}}$. So, we can consider $\mathsf{TI}_1 \otimes \mathsf{TI}_2$ as a measurement performed by two parties $\mathcal{L}$ and $\mathcal{R}$. In this way, we can write the trace as the probability that $\mathcal{L}$ gets outcome 0 and $\mathcal{R}$ gets outcome 0:

$$\operatorname{Tr}[(\mathsf{TI}_1 \otimes \mathsf{TI}_2)\rho] = \Pr[\mathcal{L} \leftarrow 0 \wedge \mathcal{R} \leftarrow 0].$$

We can see that from $\mathsf{TI}_1 \otimes \mathsf{TI}_2$ to $\mathsf{TI}_1 \otimes \mathsf{ATI}_2$, $\mathcal{L}$ performs the same measurement. Hence, we can condition on the event that $\mathcal{L}$ gets outcome 0 and let ρ_1 be the remaining mixed state that traced out the $\mathcal{L}$-part. Then, we get that

$$\begin{aligned}\Pr[\mathcal{L} \leftarrow 0 \wedge R \leftarrow 0] &= \Pr[\mathcal{L} \leftarrow 0] \cdot \Pr[R \leftarrow 0 | \mathcal{L} \leftarrow 0] \\ &= \Pr[\mathcal{L} \leftarrow 0] \cdot \operatorname{Tr}[\mathsf{TI}_2 \cdot \rho_1] \\ &\leq \Pr[\mathcal{L} \leftarrow 0] \cdot (\operatorname{Tr}[\mathsf{ATI}_2 \cdot \rho_1] + \delta) \\ &\leq \operatorname{Tr}[(\mathsf{TI}_1 \otimes \mathsf{ATI}_2)\rho] + \delta,\end{aligned}$$

where the first inequality follows from Theorem 18.6 and the last step follows from $\Pr[\mathcal{L} \leftarrow 0] \le 1$.

The next step is to show that:

$$\mathrm{Tr}[(\mathsf{TI}_1 \otimes \mathsf{ATI}_2)\rho] \le \mathrm{Tr}[(\mathsf{ATI}_1 \otimes \mathsf{ATI}_2)\rho] + \delta. \tag{G.2}$$

In this case, $\mathcal{R}$ performs the same measurement. We can condition on the event that $\mathcal{R}$ gets outcome 0 and let ρ_2 be the remaining mixed state traced out the $\mathcal{R}$-part.

Hence, by a similar argument, we get that

$$\begin{aligned}
\mathrm{Tr}[(\mathsf{TI}_1 \otimes \mathsf{ATI}_2)\rho] &= \Pr[\mathcal{L} \leftarrow 0 \wedge \mathcal{R} \leftarrow 0] \\
&= \Pr[\mathcal{R} \leftarrow 0] \cdot \Pr[\mathcal{L} \leftarrow 0 | \mathcal{R} \leftarrow 0] \\
&= \Pr[\mathcal{R} \leftarrow 0] \cdot \mathrm{Tr}[\mathsf{TI}_1 \cdot \rho_2] \\
&\le \Pr[\mathcal{R} \leftarrow 0] \cdot (\mathrm{Tr}[\mathsf{ATI}_1 \cdot \rho_2] + \delta) \\
&\le \mathrm{Tr}[(\mathsf{ATI}_1 \otimes \mathsf{ATI}_2)\rho] + \delta.
\end{aligned}$$

Combining the Eq. (G.1) and Eq. (G.2) proves the first bullet of the lemma:

$$\begin{aligned}
\mathrm{Tr}[(\mathsf{TI}_1 \otimes \mathsf{TI}_2)\rho] &\le \mathrm{Tr}[(\mathsf{TI}_1 \otimes \mathsf{ATI}_2)\rho] + \delta \\
&\le \mathrm{Tr}[(\mathsf{ATI}_1 \otimes \mathsf{ATI}_2)\rho] + 2\delta.
\end{aligned}$$

For the second part of the lemma, the trace can also be written as

$$\begin{aligned}
\mathrm{Tr}\left[\left(\mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_1}) \otimes \mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_2})\right)\rho'\right] &= \Pr[\mathcal{L} \leftarrow 0 \wedge \mathcal{R} \leftarrow 0] \\
&= \Pr[\mathcal{L} \leftarrow 0] \cdot \Pr[\mathcal{R} \leftarrow 0 | \mathcal{L} \leftarrow 0],
\end{aligned}$$

where $\mathcal{L}$ and $\mathcal{R}$ are now performing measurements on ρ'.

We first rewrite the term $\Pr[\mathcal{L} \leftarrow 0]$ as

$$\Pr[\mathcal{L} \leftarrow 0] = \mathrm{Tr}[(\mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_1}) \otimes \mathbf{I})\rho'].$$

We can see that this measure process is equivalent to the following process:

1. $\mathcal{R}$ first performs the measurement $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_2,D_2,\gamma}$ on the $\mathcal{R}$-part of ρ and gets a state ρ_1 such that $\mathrm{Tr}_{\mathcal{L}}[\rho_1] = \mathrm{Tr}_{\mathcal{L}}[\rho']$.
2. $\mathcal{L}$ measures $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_1,D_1,\gamma}$ on $\mathrm{Tr}_{\mathcal{R}}[\rho_1]$ and get the collapsed state ρ_2 such that $\rho_2 = \mathrm{Tr}_{\mathcal{R}}[\rho']$.
3. $\mathcal{L}$ measures $\mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_1})$ on ρ_2.

Hence, we have

$$\mathrm{Tr}[(\mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_1}) \otimes \mathbf{I})\rho'] = \mathrm{Tr}[\mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_1}) \cdot \rho_2],$$

By Theorem 18.6 (the third bullet),

$$\mathrm{Tr}[\mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_1}) \cdot \rho_2] \geq 1 - 2\delta.$$

Hence, we get that $\Pr[\mathcal{L} \leftarrow 0] \geq 1 - 2\delta$.

For the second term $\Pr[\mathcal{R} \leftarrow 0 | \mathcal{L} \leftarrow 0]$, it can be written as

$$\Pr[\mathcal{R} \leftarrow 0 | \mathcal{L} \leftarrow 0] = \mathrm{Tr}[(\mathbf{I} \otimes \mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_2})) \cdot \rho_3] = \mathrm{Tr}[\mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_2}) \cdot \rho_4],$$

where ρ_3 is the collapsed state conditioned on the outcome of $\mathcal{L}$ being 0 and $\rho_4 = \mathrm{Tr}_{\mathcal{L}}[\rho_3]$.

This measure process is equivalent to the followings:

1. $\mathcal{L}$ first performs two consecutive measurements $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_1,D_1,\gamma}$ and $\mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_1})$ on the $\mathcal{L}$-part of ρ, and gets the collapsed state ρ'' such that $\mathrm{Tr}_{\mathcal{R}}[\rho''] = \mathrm{Tr}_{\mathcal{R}}[\rho_3]$.
2. $\mathcal{R}$ measures $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_2,D_2,\gamma}$ on $\mathrm{Tr}_{\mathcal{L}}[\rho'']$ and gets ρ_3.
3. $\mathcal{R}$ measures $\mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_2})$ on ρ_4.

By Theorem 18.6 again, we have

$$\Pr[\mathcal{R} \leftarrow 0 | \mathcal{L} \leftarrow 0] = \mathrm{Tr}[\mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_2}) \cdot \rho_4] \geq 1 - 2\delta.$$

Therefore, we have

$$\mathrm{Tr}\left[\big(\mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_1}) \otimes \mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_2})\big)\rho'\right] \geq (1-2\delta)^2 \geq 1-4\delta,$$

which completes the proof of the second part of the lemma. □

Notice that Lemma 18.7 can be easily generalized to the case of q-partite state. We state the following corollary without proof:

Corollary G.4. *Let $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_q$ be q collections of projective measurements and D_i be any probability distributions defined on the index set of $\mathcal{P}_i$ for all $i \in [q]$. For any $0 < \epsilon, \delta, \gamma < 1$, for all $i \in [q]$, the algorithms $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_i, D_i, \gamma}$ satisfy the followings:*

- *For any q-partite (possibly entangled, mixed) quantum state $\rho \in \mathscr{H}_1 \otimes \cdots \otimes \mathscr{H}_q$,*

$$\mathrm{Tr}\left[\left(\bigotimes_{i=1}^{q} \mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_i, D_i, \gamma-\epsilon}\right)\rho\right] \geq \mathrm{Tr}\left[\left(\bigotimes_{i=1}^{q} \mathsf{TI}_{\gamma}(\mathcal{P}_{D_i})\right)\rho\right] - q\delta.$$

- *For any (possibly entangled, mixed) quantum state ρ, let ρ' be the collapsed state after applying $\bigotimes_{i=1}^{q} \mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_i, D_i, \gamma-\epsilon}$ on ρ (and normalized). Then,*

$$\mathrm{Tr}\left[\left(\bigotimes_{i=1}^{q} \mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_i})\right)\rho'\right] \geq 1 - 2q\delta.$$

G.3 Generalizing Learning Games

G.3.1 Generalized unlearnability

The γ-goodness test for quantum program (Definition 18.13) captures the intuition that a quantum program's behavior on classical inputs is γ-good comparing to the input-output behavior of f with respect to the input distribution D_f. For cryptographic primitives, as discussed in the introduction, achieving a particular cryptographic functionality does not necessarily mean to have the exact input-output behavior. As an example, to sign a message, there are usually more than one valid signatures and the intended functionality is preserved as long as any valid signature is provided.

For a randomized function f, we denote the input x of f as the real input taken by f as well as random coins used by f.

Definition G.3 (Predicate)**.** A classical predicate $E(P, y_1, \cdots, y_k, r)$ is a binary outcome function that runs a classical program P on a randomly sampled input x to get output z, and outputs 0/1 depending on whether $(z, y_1, \cdots, y_k, r) \in R$ for some binary relation defined by R. The randomness of sampling x, program P all depends on randomness r. $y_1, \cdots, y_k$ are auxiliary inputs that specify the relation.

Quantumly, it runs a quantum program on random classical input x and measure if $(z, y_1, \cdots, y_k, r) \in R$ in superposition, where z is the first register of the resulting state. In other words, it is a projective measurement indexed by r (also by $y_1, \cdots, y_k$).

We use $\mathsf{Samp}, \mathscr{F}$ to denote a cryptographic application. $\mathscr{F}$ denotes the intended functionality that this cryptographic application should achieve.

Definition G.4 (Cryptographic Application $\mathsf{Samp}, \mathscr{F}$)**.** Samp is a sampler that takes a security parameter λ and interacts with an adversary $\mathcal{A}$: $f \leftarrow (\mathcal{A} \Leftrightarrow \mathsf{Samp}(1^\lambda))$ where f is a classical circuit that contains some secret information s_f which is unknown to $\mathcal{A}$, and $\mathcal{A}$ can get some public information aux_f from the interaction.

$\mathscr{F} = \{F_\lambda\}$ and $F_\lambda(P, f, r)$ is a predicate which takes a program, a circuit f and randomness r. For all efficient $\mathcal{A}$, all f sampled by Samp, there exists a negligible function $\mathsf{negl}(\cdot)$ such that, $\Pr[F_\lambda(f, f, r) = 1] \geq 1 - \mathsf{negl}(\lambda)$.

Remark G.2. The sampling procedure Samp is defined in an interactive way. One example is predicate encryption. To generate a pair of keys, Samp needs to additionally take a description of a predicate function (which provided by $\mathcal{A}$). Therefore, we define the sampling procedure as an interaction between an algorithm $\mathcal{A}$ and

Samp. For most of the applications, it is safe to assume Samp that takes a security parameter and outputs a function f together with the public parameter aux_f, i.e., $(f, \mathsf{aux}_f) \leftarrow \mathsf{Samp}(1^\lambda)$.

This security of the cryptographic application is orthogonal to its correctness and unlearnability. The definition of security varies a lot when different applications are given. Some examples include CPA security for public key encryption schemes and signature unforgeability. However, the security should be easy to prove, when we implement a copy-protection/copy detection scheme using our construction. In this chapter, we only focus on its correctness and copy-protect security/copy-detect security/unlearnability/unremovability.

Definition G.5 (γ-Goodness Test with respect to f, E)**.** Let a quantum program for computing f be $(\rho, \{U_x\}_{x\in[N]})$.

- Define $\mathcal{P}_r = (P_r, Q_r)$ be the following projective measurement:
  - It first samples an input x using the randomness r; the sampling procedure is hardcoded in the predicate E (as defined in Definition G.3);
  - On input x, it runs U_x on the quantum state ρ;
  - It measures the output register by checking if the output of the quantum circuit satisfies the predicate $E(\cdot, f, r)$; output 1 if yes, and 0 otherwise.

  Let $\mathcal{P} = \{\mathcal{P}_r\}$ be a collection of projective measurements.
- D is the distribution of uniform randomness (over r).
- Let $\mathcal{P}_D = (P_D, Q_D)$ be the mixture of projective measurement defined in Definition 18.7.
- We say a quantum program is tested γ**-good** with respect to f, E if the threshold implementation $\mathsf{TI}_\gamma(\mathcal{P}_D)$ outputs 1.

Note that Definition 18.13 fits into this general definition, where the predicate E on a random input x (x is drawn depending on randomness r) and f, checks if the output is equal to $f(x)$.

We then generalize the learning game to the setting of cryptographic applications. Note that $\mathscr{E}$ may be not the same as $\mathscr{F}$. In the game below, an adversary tries to learn a more restricted functionality of f (described by $\mathscr{E}$).

Definition G.6 (Learning Game for $\mathsf{Samp}, \mathscr{E}$)**.** A learning game for a sampler Samp (which samples a function in $\mathcal{F}_\lambda$), a predicate $\mathscr{E} = \{E_\lambda\}$, and an adversary $\mathcal{A}$ is denoted as $\mathsf{LearningGame}^{\mathcal{A}}_{\mathsf{Samp},\mathscr{E},\gamma}(1^\lambda)$, which consists the following steps:

1. **Sampling Phase**: At the beginning of the game, $\mathcal{A}$ interacts with the challenger and samples $f \leftarrow (\mathcal{A} \Longleftrightarrow \mathsf{Samp}(1^\lambda))$.
2. **Query Phase**: $\mathcal{A}$ then gets oracle access to f;
3. **Output Phase**: Finally, $\mathcal{A}$ outputs a quantum program $(\rho, \{U_x\}_{x\in[N]})$.

The game outputs 1 if and only if the program is tested to be γ-good with respect to f, E_λ.

It is easy to see that Definition G.6 implies Definition 18.14. One example is digital signature. Samp picks a pair of signing key and verification key $(\mathsf{sk}, \mathsf{vk})$ and outputs a signing circuit $f = \mathsf{Sign}(\mathsf{sk}, \cdot)$ which hard-wires sk and appends vk with the circuit description. The predicate is defined as: sample m, r_s, r_v according to randomness r, run the program with input m and randomness r_s to obtain outcome z, decode sk, vk from the circuit f and the predicate is 1 if and only if $\mathsf{Ver}(\mathsf{vk}, m, z; r_v) = 1$. In other words, the predicate checks if the program outputs a valid signature on a random message.

Definition G.7 (Quantum Unlearnability of $(\mathsf{Samp}, \mathscr{F}), \mathscr{E}$)**.** $((\mathsf{Samp}, \mathscr{F}), \mathscr{E})$ is called γ-quantum-unlearnable if for any QPT adversary $\mathcal{A}$, there exists a negligible function

$\mathsf{negl}(\cdot)$ such that the following holds for all λ:

$$\Pr\left[b = 1,\, b \leftarrow \mathsf{LearningGame}^{\mathcal{A}}_{\mathsf{Samp},\mathscr{E},\gamma}(1^\lambda)\right] \leq \mathsf{negl}(\lambda)$$

G.3.2 Generalized copy protection

Definition G.8 (Quantum Copy Protection)**.** A quantum copy-protection scheme for $(\mathsf{Samp}, \mathscr{F}), \mathscr{E}$ consists of the following procedures:

$\mathsf{Setup}(1^\lambda) \to (\mathsf{sk})$: the setup algorithm takes in a security parameter λ in unary and generates a secret key sk.

$\mathsf{Generate}(\mathsf{sk}, f) \to (\rho_f, \{U_{f,x}\}_{x\in[N]})$: on input $f \in \mathcal{F}_\lambda$ and secret key sk, the vendor generates a quantum program $(\rho_f, \{U_{f,x}\}_{x\in[N]})$.

$\mathsf{Compute}(\rho_f, \{U_{f,x}\}_{x\in[N]}, x) \to y$: given a quantum program, a user can compute the function $f(x)$ on input x by applying $U_{f,x}$ on ρ and measuring the first register of the state.

Efficiency: Setup, $\mathsf{Compute}$ and $\mathsf{Generate}$ should run in $\mathsf{poly}(\lambda)$ time.

Correctness: For all $\lambda \in \mathbb{N}$, all efficient $\mathcal{A}$, every $f \leftarrow (\mathcal{A} \iff \mathsf{Samp}(1^\lambda))$, all $(\rho_f, \{U_{f,x}\}_{x\in[N]}) \leftarrow \mathsf{Generate}(\mathsf{sk}, f)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that,

unique output: for all $x \in [N]$, apply $U_{f,x}$ on ρ_f and measure the first register, with probability at least $1 - \mathsf{negl}(\lambda)$, the output is a fixed value $z_{f,x}$;

functionality preserving: $(\rho_f, \{U_{f,x}\}_{x\in[N]})$ are $(1 - \mathsf{negl}(\lambda))$-good with respect to f, E_λ with probability 1.

Security: It has γ-anti-piracy security defined below.

Note that the property "unique output" enables the copy-protected program can be evaluated polynomially many times.

Definition G.9 (γ-Anti-Piracy Security Game)**.** An anti-piracy security game for a sampler Samp, a predicate $\mathscr{E}$ and adversary $\mathcal{A}$ is denoted as $\mathsf{CopyProtectionGame}^{\mathcal{A}}_{\mathsf{Samp},\mathscr{E},\gamma}(1^\lambda)$, which consists of the following steps:

1. **Setup Phase**: At the beginning of the game, the challenger takes a security parameter λ and obtains secret key $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda)$.

2. **Sampling Phase**: $\mathcal{A}$ interacts with the challenger and samples $f \leftarrow (\mathcal{A} \Longleftrightarrow \mathsf{Samp}(1^\lambda))$.

3. **Query Phase**: $\mathcal{A}$ makes a single query to the challenger and obtains a quantum program for f: $(\rho_f, \{U_{f,x}\}_{x\in[N]}) \leftarrow \mathsf{Generate}(\mathsf{sk}, f)$.

4. **Output Phase**: Finally, $\mathcal{A}$ outputs a (possibly mixed and entangled) state σ over two registers R_1, R_2 and two sets of unitaries $(\{U_{R_1,x}\}_{x\in[N]}, \{U_{R_2,x}\}_{x\in[N]})$ They can be viewed as programs $\mathsf{P}_1 = (\sigma[R_1], \{U_{R_1,x}\}_{x\in[N]})$ and $\mathsf{P}_2 = (\sigma[R_2], \{U_{R_2,x}\}_{x\in[N]})$.

The game outputs 1 if and only if both programs $\mathsf{P}_1, \mathsf{P}_2$ are tested to be γ-good with respect to f, E_λ.

Similarly, we can define q-collusion resistant γ-anti-piracy security game

$$\mathsf{CopyProtectionGame}^{q,\mathcal{A}}_{\mathsf{Samp},\mathscr{E},\gamma}(1^\lambda)$$

in which the adversary $\mathcal{A}$ can make at most q queries in the query phases and is required to output $q+1$ programs $\{(\sigma[R_i], \{U_{R_i,x}\}_{x\in[N]})\}_{i\in[q+1]}$ such that each program is tested to be γ-good.

Definition G.10 (γ-Anti-Piracy-Security)**.** A copy-protection scheme for Samp and $\mathscr{E}$ has γ-anti-piracy security, if for any QPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that the following holds for all $\lambda \in \mathbb{N}$:

$$\Pr\left[b = 1, b \leftarrow \mathsf{CopyProtectionGame}^{\mathcal{A}}_{\mathsf{Samp},\mathscr{E},\gamma}(1^\lambda)\right] \leq \mathsf{negl}(\lambda) \tag{G.3}$$

G.3.3 Generalized copy detection

A copy detection scheme for $(\mathsf{Samp}, \mathscr{F}), \mathscr{E}$ is very similar to the copy-protection scheme, except it has an additional procedure Check which applies a projective measurement and checks if the quantum state is valid.

Definition G.11 (Quantum Copy Detection)**.** A quantum copy-detection scheme for $(\mathsf{Samp}, \mathscr{F}), \mathscr{E}$ consists of the following procedures:

$\mathsf{Setup}(1^\lambda)$, $\mathsf{Generate}(\mathsf{sk}, f)$ and $\mathsf{Compute}(\rho_f, \{U_{f,x}\}_{x\in[N]}, x)$ are the same as those in Definition G.8.

$\mathsf{Check}(\mathsf{pk}, \mathsf{aux}_f, \rho_f, \{U_{f,x}\}_{x\in[N]}) \to b, \rho'$: on input a public key pk, public information aux_f generated during Samp, a quantum program, it applies a binary projective measurement P_0, P_1 on ρ_f that depends on pk, aux_f, $\{U_{f,x}\}_{x\in[N]}$; it outputs the outcome b and the collapsed state ρ'.

Correctness ($\mathsf{Generate}$): The same as the security of Definition G.8.

Correctness (Check): For all $\lambda \in \mathbb{N}$, all efficient $\mathcal{A}$, every $f \leftarrow (\mathcal{A} \Longleftrightarrow \mathsf{Samp}(1^\lambda))$, all $(\rho_f, \{U_{f,x}\}_{x\in[N]}) \leftarrow \mathsf{Generate}(\mathsf{sk}, f)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that, $\mathsf{Check}(\mathsf{pk}, \mathsf{aux}_f, \rho_f, \{U_{f,x}\}_{x\in[N]})$ outputs 1 with probability at least $1 - \mathsf{negl}(\lambda)$.

Security: It has γ-copy-detection security defined below.

Definition G.12 (γ-Copy-Detection Security Game)**.** A copy-detection security game for a sampler Samp, a predicate $\mathscr{E}$ and adversary $\mathcal{A}$ is denoted as $\mathsf{CopyDetectionGame}^{\mathcal{A}}_{\mathsf{Samp},\mathscr{E},\gamma}(1^\lambda)$, which consists of the following steps:

1. **Setup Phase**: At the beginning of the game, the challenger takes a security parameter λ and obtains keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$.

2. **Sampling Phase**: $\mathcal{A}$ interacts with the challenger and samples a function f, which is denoted by $f \leftarrow (\mathcal{A} \Longleftrightarrow \mathsf{Samp}(1^\lambda))$. Let aux_f denote the public information $\mathcal{A}$ obtains during the interaction.

3. **Query Phase**: $\mathcal{A}$ makes a single query to the challenger and obtains a copy detection program for f: $(\rho_f, \{U_{f,x}\}_{x\in[N]}) \leftarrow \mathsf{Generate}(\mathsf{sk}, f)$.

4. **Output Phase**: Finally, $\mathcal{A}$ outputs a state σ over two registers R_1, R_2 and two sets of unitaries $(\{U_{R_1,x}\}_{x\in[N]}, \{U_{R_2,x}\}_{x\in[N]})$. They can be viewed as programs $\mathsf{P}_1 = (\sigma[R_1], \{U_{R_1,x}\}_{x\in[N]})$ and $\mathsf{P}_2 = (\sigma[R_2], \{U_{R_2,x}\}_{x\in[N]})$.

The game outputs 1 if and only if

- Apply Check on input $\mathsf{pk}, \mathsf{aux}_f, \mathsf{P}_i$ respectively and both outcomes are 1. Let P_i' be the collapsed program conditioned on outcomes are 1.

- *Both* programs $\mathsf{P}_1', \mathsf{P}_2'$ are both tested to be γ-good with respect to f, E_λ.

Similarly, we can define q-collusion resistant γ-copy-detection security game $\mathsf{CopyDetectionGame}^{\mathcal{A},q}_{\mathsf{Samp},\mathscr{E},\gamma}(1^\lambda)$, in which the adversary $\mathcal{A}$ can perform at most q query phases and output $q+1$ programs $P_i = (\sigma[R_i], \{U_{R_i,x}\}_{x\in[N]})$ for $i \in [q+1]$. The game outputs 1 if and only if for all $i \in [q+1]$, the outcome of applying Check on P_i is 1, and the collapsed program P_i' is tested to be γ-good.

Definition G.13 (γ-Copy-Detection-Security)**.** A copy detection scheme for Samp and $\mathscr{E}$ has γ-security, if for any QPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that the following holds for all $\lambda \in \mathbb{N}$:

$$\Pr\left[b = 1, b \leftarrow \mathsf{CopyDetectionGame}^{\mathcal{A}}_{\mathsf{Samp},\mathscr{E},\gamma}(1^\lambda)\right] \leq \mathsf{negl}(\lambda) \tag{G.4}$$

G.3.4 Watermarking primitives with public extraction

In this subsection, we give a unified definition that covers most of the definitions in the previous works about watermarking primitives, including [CHN+18, KW17, QWZ18, KW19, GKM+19]. We will give several concrete examples of watermarking schemes in Appendix G.3.5.

Definition G.14 (Watermarking Primitives for $(\mathsf{Samp}, \mathscr{F}), \mathscr{E}$)**.** A watermarking scheme for $(\mathsf{Samp}, \mathscr{F}), \mathscr{E}$ consists of the following classical algorithms:

$\mathsf{Setup}(1^\lambda)$: it takes as input a security parameter 1^λ and outputs keys $(\mathsf{xk}, \mathsf{mk})$. xk is the extracting key and mk is the marking key. We only consider publicly extractable watermarking scheme. Thus xk is always public.

$\mathsf{Samp}(1^\lambda)$: it takes a security parameter 1^λ,

$$f \leftarrow (\mathcal{A} \Longleftrightarrow \mathsf{Samp}(1^\lambda)).$$

We also denote aux_f as the public information $\mathcal{A}$ obtains during the interaction.

$\mathsf{Mark}(\mathsf{mk}, f, \tau)$: it takes a circuit f and a message $\tau \in \mathcal{M}_\lambda$, outputs a marked circuit $\widetilde{f}$.

$\mathsf{Extract}(\mathsf{xk}, \mathsf{aux}_f, f')$: it takes the public auxiliary information aux_f, a circuit and outputs a message in $\{\perp\} \cup \mathcal{M}_\lambda$.

Remark G.3. In some watermarking schemes, Setup also outputs a watermarking public parameter wpp and Samp takes this parameter to sample a function. Our construction works in this setting. For the sake of clarity, we use the above notion. $\mathsf{Extract}$ may also take an aux that specifies its restricted functionality that f' should achieve. We assume f' contains a piece of information aux as a comment.

It satisfies the following properties.

Definition G.15 (Correctness of Mark (Functionality Preserving))**.** For all λ, for every efficient algorithm $\mathcal{A}$, there exists a negligible function negl, for all $(\mathsf{xk}, \mathsf{mk}) \leftarrow \mathsf{Setup}(1^\lambda)$, and every $\tau \in \mathcal{M}_\lambda$,

$$\Pr\left[F_\lambda(\widetilde{f}, f, r) = 1 \; : \; \substack{f \leftarrow (\mathcal{A} \Longleftrightarrow \mathsf{Samp}(1^\lambda)) \\ \widetilde{f} \leftarrow \mathsf{Mark}(\mathsf{mk}, f, \tau)}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Definition G.16 (Correctness of $\mathsf{Extract}$)**.** For all λ, for every efficient algorithm $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$, for all $(\mathsf{xk}, \mathsf{mk}) \leftarrow \mathsf{Setup}(1^\lambda)$, and every $\tau \in \mathcal{M}_\lambda$, every aux,

$$\Pr\left[\tau \neq \mathsf{Extract}(\mathsf{xk}, \mathsf{aux}_f, \widetilde{f}||\mathsf{aux}) \; : \; \substack{f \leftarrow (\mathcal{A} \Longleftrightarrow \mathsf{Samp}(1^\lambda)) \\ \widetilde{f} \leftarrow \mathsf{Mark}(\mathsf{mk}, f, \tau)}\right] \leq \mathsf{negl}(\lambda),$$

where aux_f is the public information given to $\mathcal{A}$ and $\widetilde{f}||\mathsf{aux}$ is the program appended with aux.

Definition G.17 (Meaningfulness)**.** For all λ, for every efficient algorithm $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$, for every aux,

$$\Pr\left[\perp \neq \mathsf{Extract}(\mathsf{xk}, \mathsf{aux}_f, f||\mathsf{aux}) \; : \; \substack{(\mathsf{xk}, \mathsf{mk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ f \leftarrow (\mathcal{A} \Longleftrightarrow \mathsf{Samp}(1^\lambda))}\right] \leq \mathsf{negl}(\lambda).$$

where aux_f is the public information given to $\mathcal{A}$ and $f||\mathsf{aux}$ is the program appended with aux.

Definition G.18 (γ-Unremovability with respect to $\mathsf{Samp}, \mathscr{E}$)**.** Consider the following game, denoted as $\mathsf{WaterMarkingGame}^{\mathcal{A}}_{\mathsf{Samp}, \mathscr{E}, \gamma}$:

1. **Setup**: The challenger samples $(\mathsf{xk}, \mathsf{mk}) \leftarrow \mathsf{Setup}(1^\lambda)$. $\mathcal{A}$ then gets xk.
2. **Sampling Phase**: The challenger interacts with the algorithm $\mathcal{A}$ and samples $f \leftarrow (\mathcal{A} \Longleftrightarrow \mathsf{Samp}(1^\lambda))$.
3. **Query Phase**: $\mathcal{A}$ has classical access to $\mathsf{Mark}(\mathsf{mk}, f, \cdot)$ at any time. Define Q be the set of messages that $\mathcal{A}$ has queried on.
4. **Output Phase**: Finally, the algorithm outputs a circuit f^*.

The adversary wins the game if and only if

$$\mathsf{Extract}(\mathsf{xk}, \mathsf{aux}_f, f^*) \notin Q \ \wedge \ \Pr_r[E_\lambda(f^*, f, r) = 1] \geq \gamma$$

We say a watermarking scheme has γ-unremovability with respect to $\mathsf{Samp}, \mathscr{E}$, if for all QPT $\mathcal{A}$, it wins the above game with negligible probability in λ. We say it has q-collusion resistant γ-unremovability if the number of queries made in the query phase is at most q.

G.3.5 Examples of watermarking primitives

Let us look at how the definitions in [CHN+18, GKM+19] fit into our frameworks.

1. Watermarkable PRF in [CHN+18]:
   - $\mathsf{Setup}(1^\lambda) = (\mathsf{wpp}, \mathsf{xk}, \mathsf{mk})$;
   - $\mathsf{Samp}(1^\lambda, \mathsf{wpp})$ samples a PRF key k, $f = \mathsf{PRF}(k, \cdot)$, $\mathsf{aux}_f = \perp$.
   - $F_\lambda(\tilde{f}, f, r)$ is 0 if and only if it samples a random input x (according to r), and $\tilde{f}(x) = f(x)$.
   - Unremovability is defined by $E_\lambda = F_\lambda$. $\gamma = 1/2 + 1/\mathsf{poly}(\lambda)$.
2. Watermarkable signature in [GKM+19]:
   - $\mathsf{Setup}(1^\lambda) = (\mathsf{wpp}, \mathsf{xk}, \mathsf{mk})$;
   - $\mathsf{Samp}(1^\lambda, \mathsf{wpp})$ samples a pair of keys vk, sk and we interpret $f = \mathsf{Sign}(\mathsf{sk}, \cdot) || \mathsf{vk}$, $\mathsf{aux}_f = \mathsf{vk}$.
   - $F_\lambda(\tilde{f}, f, r)$ is 0 if and only if $\mathsf{Ver}(\mathsf{vk}, m, \tilde{f}(m)) = 1$, where vk is decoded from f and m is sampled by r.
   - Unremovability: $E_\lambda = F_\lambda$. γ is inverse polynomial.
3. Watermarkable public key encryption in [GKM+19]:

- $\mathsf{Setup}(1^\lambda) = (\mathsf{wpp}, \mathsf{xk}, \mathsf{mk})$;
- $\mathsf{Samp}(1^\lambda, \mathsf{wpp})$ is defined below:
  - It samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKEGen}(1^\lambda, \mathsf{wpp})$;
  - $f = \mathsf{Dec}(\mathsf{sk}, \cdot)||\mathsf{pk}$, $\mathsf{aux}_f = \mathsf{pk}$.
- $F_\lambda(\tilde{f}, f, r)$ is defined as:
  - Decode pk from f, sample m according to r;
  - Let $\mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, m)$;
  - It outputs 0 if and only if $\tilde{f}(\mathsf{ct}) = m$.
- Unremovability: $E_\lambda(\tilde{f}, f, r)$ defined as:
  - Decode pk from f, sample b according to r;
  - Decode $\mathsf{aux} = (m_0, m_1)$ from $\tilde{f}$; if $m_0 = m_1$, outputs 1;
  - Let $\mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, m_b)$;
  - It outputs 0 if and only if $\tilde{f}(\mathsf{ct}) = m_b$.

  And, $\gamma = 1/2 + 1/\mathsf{poly}(\lambda)$.

G.4 General Copy-Protection Scheme

In this section, we show the copy-protection scheme in Section 18.5 works for general unlearnable function families (or cryptographic applications, as defined in Definition G.7).

Theorem G.5. *Let $((\mathsf{Samp}, \mathscr{F}), \mathscr{E})$ be γ-quantum-unlearnable. Then the construction in Section 18.5 is a copy-protection scheme satisfies efficiency, correctness and $(\gamma - 1/\mathsf{poly})$-anti-piracy (see Definition G.10), for all polynomial poly.*

Efficiency and Correctness are straightforward. The proof of Theorem 18.8 works for the general case, by simply doing the followings:

1. $\mathsf{TI}, \mathsf{ATI}$ are now defined as the (approximated) projective measurement corresponding to the predicate E_λ;

2. In Lemma 18.10, 18.11 and 18.12, the randomness is taken over the general unlearnability game and copy-protection game.

G.5 General Quantum Copy-Detection

G.5.1 Construction

Now we construct a copy detection scheme for $\mathsf{Samp}, \mathscr{F}, \mathscr{E}$. Let QM and WM be a public key quantum money scheme and a publicly extractable watermarking scheme for $\mathsf{Samp}, \mathscr{F}, \mathscr{E}$, whose serial number space $\mathcal{S}_\lambda$ of QM is a subset of the message space $\mathcal{M}_\lambda$ of WM. We construct a copy detection scheme in Fig. G.1.

G.5.2 Efficiency and correctness

First, for all $\lambda \in \mathbb{N}$, all efficient $\mathcal{A}$, every $f \leftarrow (\mathcal{A} \Longleftrightarrow \mathsf{Samp}(1^\lambda))$, the program output is $(\rho_f, \{U_{f,x}\}_{x\in[N]})$, we have $\mathsf{Compute}(\rho_f, \{U_{f,x}\}_{x\in[N]}, x) = \tilde{f}(x)$, where $\tilde{f} = \mathsf{WM.Mark}(\mathsf{mk}, f, s)$ for some serial number s. From the correctness of WM, it satisfies **unique output** and **functionality preserving** (with respect to $\mathscr{F}$).

The correctness of Check comes from the correctness of $\mathsf{WM.Extract}$ and **unique serial number** property of QM. Check is a projection since $\mathsf{QM.Ver}$ is also a projection. Efficiency is straightforward.

G.5.3 Security

Theorem G.6. *Assume* QM *is a quantum money scheme and* WM *is a q-collusion resistant for* $\mathsf{Samp}, \mathscr{E}$ *with γ-unremovability, the above copy-detection scheme for* $\mathsf{Samp}, \mathscr{F}, \mathscr{E}$ *has q-collusion resistant γ-copy-detection-security.*

Proof. We prove the case for $q = 1$. Let $\mathcal{A}$ be a QPT algorithm that tries to break the security of the copy detection scheme. Let $(\sigma, U_{R_1}, U_{R_2})$ be the programs output

$\mathsf{Setup}(1^\lambda)$: it runs $\mathsf{WM.Setup}(1^\lambda)$ to get xk, mk, let $\mathsf{sk} = \mathsf{mk}$ and $\mathsf{pk} = \mathsf{xk}$.

$\mathsf{Generate}(\mathsf{sk}, f)$:

- it runs $\mathsf{QM.Gen}(1^\lambda)$ to get a money state $|\$\rangle$ and a serial number s (by applying $\mathsf{QM.Ver}$ to the banknote);
- let $\widetilde{f} = \mathsf{WM.Mark}(\mathsf{mk}, f, s)$ which is classical;
- it outputs the quantum state $\rho_f = (\widetilde{f}, |\$\rangle)$, and $\{U_{f,x}\}_{x\in[N]}$;
- let $\{U_{f,x}\}_{x\in[N]}$ describe the following unitary: on input a quantum state ρ, treat the first register as a classical function g, compute $g(x)$ in superposition.

$\mathsf{Check}(\mathsf{pk}, \mathsf{aux}_f, (\rho_f, \{U_{f,x}\}_{x\in[N]}))$:

- it parses and measures the first register, which is $(f', |\$'\rangle)$;
- it checks if $\mathsf{QM.Ver}(|\$'\rangle)$ is valid and it gets the serial number s';
- it then checks if $s' = \mathsf{WM.Extract}(\mathsf{pk} = \mathsf{xk}, \mathsf{aux}_f, f')$;
- if all the checks pass, it outputs 0; otherwise, it outputs 1.

Figure G.1: Quantum copy detection scheme.

by $\mathcal{A}$ which wins the game $\mathsf{CopyDetectionGame}^{\mathcal{A}}_{\mathsf{Samp},\mathscr{E},\gamma}$.

To win the game, the program $(\sigma, U_{R_1}, U_{R_2})$ should pass the following two tests:

1. Apply the projective measurement (defined by $\mathsf{Check}(\mathsf{pk}, \mathsf{aux}_f, \cdot)$) on both $\sigma[R_1]$ and $\sigma[R_2]$, and both outcomes are 0.

2. Let σ' be the state that passes step 1. Then both programs $(\sigma'[R_1], U_{R_1})$, $(\sigma'[R_2], U_{R_2})$ are tested to be γ-good with non-negligible probability.

In our construction, Check first measures the program registers. The resulting state is $\tilde{f}_1, \tilde{f}_2, \sigma$, where $\tilde{f}_1, \tilde{f}_2$ are supposed to be classical (marked) circuits that computes f and σ are (possibly entangled) states that are supposed to be quantum money for each of the program.

Next, Check applies $\mathsf{QM.Ver}$ on both registers of σ and computes serial numbers. Define S_b be the random variable of $\mathsf{QM.Ver}$ applying on $\sigma[R_b]$ representing the serial number of ρ_b. Define S be the random variable of $\mathsf{QM.Ver}(|\$\rangle)$ representing the serial number of the quantum money state in the $\mathsf{Generate}$ procedure.

Define E be the event that both $\mathsf{WM.Extract}(\mathsf{xk}, \mathsf{aux}_f, \tilde{f}_b) = S_b$ and at least one of S_1, S_2 is not equal to S. Define E' be the event that both S_1, S_2 are equal to S and both $\mathsf{WM.Extract}(\mathsf{xk}, \mathsf{aux}_f, \tilde{f}_b) = S_b$. If $\tilde{f}_1, \tilde{f}_2, \sigma$ passes the step 1, exactly one of E and E' happens.

In step 2, it simply tests if $\tilde{f}_1$ and $\tilde{f}_2$ are γ-good with respect to f, E_λ. Since $\tilde{f}_1, \tilde{f}_2$ are classical circuits, it is equivalent to check whether they work correctly on at least γ fraction of all inputs. If it passes step 2, we have for all $b \in \{1, 2\}$, $\Pr_r[E_\lambda(\tilde{f}_b, f, r) = 0] \geq \gamma$.

Therefore, the probability of $\mathcal{A}$ breaks the security game is indeed,

$$\begin{aligned}
&\Pr_{(\tilde{f}_1,\tilde{f}_2,\sigma)}\left[\forall b, \Pr_r[E_\lambda(\tilde{f}_b, f, r) = 0] \geq \gamma\right] \\
=& \Pr_{(\tilde{f}_1,\tilde{f}_2,\sigma)}\left[(E \vee E') \wedge \forall b, \Pr_r[E_\lambda(\tilde{f}_b, f, r) = 0] \geq \gamma\right] \\
\leq& \Pr_{(\tilde{f}_1,\tilde{f}_2,\sigma)}\left[E \wedge \forall b, \Pr_r[E_\lambda(\tilde{f}_b, f, r) = 0] \geq \gamma\right] + \Pr_{(\tilde{f}_1,\tilde{f}_2,\sigma)}[E']
\end{aligned}$$

Note that the probability is taken over the randomness of $\mathsf{CopyDetectionGame}^{\mathcal{A}}_{\mathsf{Samp},\mathscr{E},\gamma}$. Next we are going to show both probabilities are negligible, otherwise we can break the quantum money scheme or watermarking scheme.

Claim G.7. $\Pr_{(\tilde{f}_1,\tilde{f}_2,\sigma)}[E'] \leq \mathit{negl}(\lambda)$.

Proof. It corresponds to the security game of the quantum money scheme. Assume $\Pr[E']$ is non-negligible, we can construct an adversary $\mathcal{B}$ for the quantum money scheme with non-negligible advantage. Given a quantum money state $|\$\rangle$, the algorithm $\mathcal{B}$ does the following (it simulates the challenger for the copy-detection scheme):

- It first runs $\mathsf{WM.Setup}(1^\lambda)$ to get xk, mk and let $\mathsf{sk} = \mathsf{mk}$ and $\mathsf{pk} = \mathsf{xk}$.
- It interacts with $\mathcal{A}$ and samples f.
- Instead of sampling a new quantum money state, it uses the state $|\$\rangle$. Let $s = \mathsf{Ver}(|\$\rangle)$ and $\widetilde{f} \leftarrow \mathsf{WM.Mark}(\mathsf{mk}, f, s)$. It gives the instance $\rho_f = (\widetilde{f}, |\$\rangle)$.
- When $\mathcal{A}$ outputs $(\tilde{f}_1, \tilde{f}_2, \sigma)$, $\mathcal{B}$ outputs σ.

Thus $\Pr[E']$ is exact the probability that both verification gives s. $\square$

Claim G.8. $\Pr_{(\tilde{f}_1,\tilde{f}_2,\sigma)}\left[E \wedge \forall b, \Pr_r[E_\lambda(\tilde{f}_b, f, r) = 0]\right] \leq \mathit{negl}(\lambda)$.

Proof. It corresponds to the security game of the underlying watermarking scheme. Since if E happens, at least one of the circuit has different mark than s and it satisfies the correctness test F. The reduction is the following ($\mathcal{B}$ simulates the challenger for the copy-detection scheme):

- Given $\mathsf{xk}, \mathsf{aux}_f$ in the watermarking security game, $\mathcal{B}$ prepares a quantum money state $|\$\rangle$ with serial number s and gets the marked circuit $\widetilde{f}$ whose marking is s.
- It prepares $\rho_f = (\widetilde{f}, |\$\rangle)$ and feeds it to $\mathcal{A}$.
- When $\mathcal{A}$ outputs outputs $(\tilde{f}_1, \tilde{f}_2, \sigma)$, $\mathcal{B}$ outputs $\tilde{f}_b$ whose mark is not s, i.e, $\mathsf{Extract}(\mathsf{xk}, \mathsf{aux}_f, \tilde{f}_b) \neq s$.

When $\mathcal{A}$ succeeds, $\mathcal{B}$ breaks the security of the watermarking scheme. □

Thus, the probability of $\mathcal{A}$ breaks the game is negligible. □

It is natural to extend the proof to q-collusion resistance. We briefly sketch the proof for q-collusion resistance which is very similar to the case $q = 1$. Let $(\tilde{f}_1, \cdots, \tilde{f}_{q+1}, \sigma)$ be the output of the adversary. Let $s_1, \cdots, s_q$ be the serial numbers in the $\mathsf{Generate}$ procedure. Let $s'_1, \cdots, s'_{q+1}$ be the serial numbers corresponding to σ. If $\mathcal{A}$ succeeds, there are two cases:

1. $\{s'_i\}_{i\in[q+1]} \subseteq \{s_j\}_{j\in[q]}$: in this case, $\mathcal{A}$ successfully copies one of the money state. Thus, we can use $\mathcal{A}$ to construct an adversary for the quantum money scheme.
2. $\{s'_i\}_{i\in[q+1]} \subsetneq \{s_j\}_{j\in[q]}$: in this case, $\mathcal{A}$ successfully unmarks one of the marked program. Thus, we can use $\mathcal{A}$ to construct an adversary for the watermarking scheme.

Therefore, assuming the existence of q-collusion resistant quantum money scheme and watermarking scheme, the construction above is a q-collusion resistant copy-detection scheme.

Combining Theorem G.6 with the watermarking primitives (see examples in Appendix G.3.5), we can get the corresponding copy-detection schemes.

G.6 Public-key Quantum Money from Copy Detection

In this section, we show that we can use quantum copy detection and public-key encryption to construct a public-key quantum money scheme. This implication shows one more application of copy detection and further demonstrates the relationship between copy detection and public-key quantum money.

We give the following the construction of the public-key quantum money. Assume that we have an underlying public key encryption scheme called $\mathsf{PKE} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Enc})$ with message space $\mathcal{M}$, and an underlying copy detection scheme $\mathsf{CD} = (\mathsf{CD.Setup}, \mathsf{CD.Generate}, \mathsf{CD.Compute}, \mathsf{CD.Check})$.

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$:

- Take in security parameter λ
- Run $\mathsf{PKE.KeyGen}(1^\lambda) \to (\mathsf{PKE.pk}, \mathsf{PKE.sk})$ and $\mathsf{CD.Setup}(1^\lambda) \to (\mathsf{CD.pk}, \mathsf{CD.sk})$.
- Output $\mathsf{pk} = (\mathsf{PKE.pk}, \mathsf{CD.pk})$ and $\mathsf{sk} = (\mathsf{PKE.sk}, \mathsf{CD.sk})$.

$\mathsf{GenNote}(\mathsf{sk}) \to |\$\rangle$:

- Take in the secret key $\mathsf{sk} = (\mathsf{PKE.sk}, \mathsf{CD.sk})$.
- Run $\mathsf{CD.Generate}(\mathsf{CD.sk}, f = \mathsf{PKE.Dec}(\mathsf{PKE.sk}, \cdot)$ to generate a copy detection program $(\rho_f, \{U_{f,x}\}_{x\in[N]})$ for the function $f = \mathsf{PKE.Dec}(\mathsf{PKE.sk}, \cdot)$.
- Output $|\$\rangle = (\rho_f, \{U_{f,x}\}_{x\in[N]})$.

$\mathsf{Ver}(\mathsf{pk}, |\$'\rangle) \to 0/1$:

- Take in the public key $\mathsf{pk} = (\mathsf{PKE.pk}, \mathsf{CD.pk})$ and a claimed banknote state $|\$'\rangle$, i.e. a claimed copy detection program for $f = \mathsf{PKE.Dec}(\mathsf{PKE.sk}, \cdot)$.
- Parse the claimed banknote $|\$'\rangle$ as $(\mathsf{aux}_f, \rho'_f, \{U'_{f,x}\}_{x\in[N]})$.
- Run $\mathsf{CD.Check}(\mathsf{CD.pk}, \mathsf{aux}_f, \rho'_f, \{U'_{f,x}\}_{x\in[N]})) \to b$; if $b = 1$, output 1 (for reject).
- Test if the program $(\rho'_f, \{U'_{f,x}\}_{x\in[N]})$ is a γ-good program with respect to f, E_λ, using the public information in $\mathsf{pk} = (\mathsf{PKE.pk}, \mathsf{CD.pk})$; if yes, output 0; else output 1.

Figure G.2: Public-key Quantum Money Scheme from Copy Detection

G.6.1 Security analysis

We now show that the public-key quantum money construction has correctness and unclonable security, given a quantum copy detection scheme with correctness and γ-anti-piracy security. The proof is intuitive and we omit some details.

Verification Correctness By the computation correctness of the underlying copy detection scheme CD and decryption correctness of the underlying PKE, a valid banknote $|\$\rangle = (\rho_f, \{U_{f,x}\}_{x\in[N]})$ for $f = \mathsf{PKE.Dec}(\mathsf{PKE.sk}, \cdot)$ is supposed to pass Check and be a γ-good program with respect to f, E_λ with all but negligible probability. Therefore, verification correctness holds.

Unclonable Security We give a brief proof for the unclonable security of the quantum money scheme, whose security definition is given in Definition G.1.

Lemma G.9. *Assuming that the quantum copy-protection scheme* CD *has* γ*-anti-piracy, then public-key quantum money scheme has unclonable security.*

Proof. Suppose there is a QPT adversary $\mathcal{A}$ that breaks unclonable security, then we can construct a QPT adversary B that breaks γ-anti-piracy security for CD.

The quantum copy detection challenger interacts with B in a copy detection anti-piracy game: In the Setup phase, challenger runs the setup $\mathsf{CD.Setup}(1^\lambda)$ to generate the keys $(\mathsf{CD.pk}, \mathsf{CD.sk})$. In the Sampling phase, the challenger samples $f = \mathsf{PKE.Dec}(\mathsf{PKE.sk}, \cdot)$, where $(\mathsf{PKE.pk}, \mathsf{PKE.sk}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$; note that it gives $\mathsf{aux} = \mathsf{PKE.pk}$ to adversary B and $s_f = \mathsf{PKE.sk}$ is kept secret. B then gives $(\mathsf{CD.pk}, \mathsf{PKE.pk})$ to the quantum money adversary $\mathcal{A}$ as the public key. In the Query phase, copy detection challenger generates one copy of copy detection program $(\rho_f, \{U_{f,x}\}_{x\in[N]}) \leftarrow \mathsf{CD.Generate}(\mathsf{CD.sk}, f)$ and gives to B. Then B sends $(\rho_f, \{U_{f,x}\}_{x\in[N]})$ as a money state $|\$\rangle$ to $\mathcal{A}$. Finally, $\mathcal{A}$ output two claimed money states $\{|\$_i\rangle\} = \{\rho_i, U_i\}_{i\in[2]}$ and sends to B. B uses them as its pirate programs and passes to copy detection challenger. It is easy to see that if both claimed money states $\{|\$_i\rangle\}_{i\in[2]}$ produced by $\mathcal{A}$'s pass verification with non-negligible probability, then B wins the copy detection anti-piracy security game with non-negligible probability. □

Bibliography

[AA17] Yosi Atia and Dorit Aharonov. Fast-forwarding of hamiltonians and exponentially precise measurements. *Nature communications*, 8(1):1–9, 2017.

[AAB$^+$19] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, Oct 2019.

[AAKV01] Dorit Aharonov, Andris Ambainis, Julia Kempe, and Umesh Vazirani. Quantum walks on graphs. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 50–59, 2001. arXiv:quant-ph/0012090.

[Aar04] Scott Aaronson. Limitations of quantum advice and one-way communication. In *Proceedings. 19th IEEE Annual Conference on Computational Complexity, 2004.*, pages 320–332. IEEE, 2004.

[Aar06] Scott Aaronson. Oracles are subtle but not malicious. In *21st Annual IEEE Conference on Computational Complexity (CCC'06)*, pages 15–pp. IEEE, 2006.

[Aar09] Scott Aaronson. Quantum copy-protection and quantum money. In *2009 24th Annual IEEE Conference on Computational Complexity*, pages 229–242. IEEE, 2009.

[Aar15] Scott Aaronson. Read the fine print. *Nature Physics*, 11(4):291–293, 2015.

[Aar16] Scott Aaronson. The complexity of quantum states and transformations: from quantum money to black holes. *arXiv preprint arXiv:1607.05256*, 2016.

[Aar18] Scott Aaronson. Shadow tomography of quantum states. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, pages 325–338, New York, NY, USA, 2018. Association for Computing Machinery.

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, 2009.

[AB20] Kasra Alishahi and Milad Barzegar. Paving property for real stable polynomials and strongly rayleigh processes. *arXiv preprint arXiv:2006.13923*, 2020.

[Abb19] Amir Abboud. Fine-grained reductions and quantum speedups for dynamic programming. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[ABH17] Naman Agarwal, Brian Bullins, and Elad Hazan. Second-order stochastic optimization for machine learning in linear time. *The Journal of Machine Learning Research*, 18(1):4148–4187, 2017.

[ABN23] Anurag Anshu, Nikolas P Breuckmann, and Chinmay Nirkhe. Nlts hamiltonians from good quantum codes. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 1090–1096, 2023.

[ABY20] Richard Aoun, Marwa Banna, and Pierre Youssef. Matrix Poincaré inequalities and concentration. In *Advances in Mathematics*, volume 371. `https://arxiv.org/pdf/1910.13797.pdf`, 2020.

[AC09] Nir Ailon and Bernard Chazelle. The fast johnson–lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on computing*, 39(1):302–322, 2009.

[AC12] Scott Aaronson and Paul Christiano. Quantum money from hidden subspaces. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 41–60. ACM, 2012.

[AC23] Jason M Altschuler and Sinho Chewi. Faster high-accuracy log-concave sampling via algorithmic warm starts. *arXiv preprint arXiv:2302.10249*, 2023.

[ACG$^+$16] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on*

Computer and Communications Security, CCS '16, page 308–318, New York, NY, USA, 2016. Association for Computing Machinery.

[ACH+18] Scott Aaronson, Xinyi Chen, Elad Hazan, Satyen Kale, and Ashwin Nayak. Online learning of quantum states. *Advances in neural information processing systems*, 31, 2018.

[ACL+20] Scott Aaronson, Nai-Hui Chia, Han-Hsuan Lin, Chunhao Wang, and Ruizhe Zhang. On the quantum complexity of closest pair and related problems. In *35th Computational Complexity Conference (CCC 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[ACMM05] Amit Agarwal, Moses Charikar, Konstantin Makarychev, and Yury Makarychev. $o(\sqrt{\log n})$ approximation algorithms for min uncut, min 2cnf deletion, and directed cut problems. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 573–581, 2005.

[AD14a] Eric Allender and Bireswar Das. Zero knowledge and circuit minimization. In Erzsébet Csuhaj-Varjú, Martin Dietzfelbinger, and Zoltán Ésik, editors, *Mathematical Foundations of Computer Science 2014*, pages 25–32, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[AD14b] Megasthenis Asteris and Alexandros G Dimakis. Repairable fountain codes. *IEEE Journal on Selected Areas in Communications*, 32(5):1037–1047, 2014.

[ADBMS98] Pankaj K Agarwal, Mark De Berg, Jiri Matousek, and Otfried Schwarzkopf. Constructing levels in arrangements and higher order voronoi diagrams. *SIAM journal on computing*, 27(3):654–667, 1998.

[ADH$^+$19a] Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332, 2019.

[ADH$^+$19b] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *NeurIPS*, 2019.

[ADM$^+$18] Nima Anari, Constantinos Daskalakis, Wolfgang Maass, Christos Papadimitriou, Amin Saberi, and Santosh Vempala. Smoothed analysis of discrete tensor decomposition and assemblies of neurons. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[AdW17] Srinivasan Arunachalam and Ronald de Wolf. Guest column: A survey of quantum learning theory. *ACM SIGACT News*, 48(2):41–67, 2017.

[AEM92] Pankaj K Agarwal, David Eppstein, and Jirí Matousek. Dynamic half-space reporting, geometric optimization, and minimum spanning trees. In *Annual Symposium on Foundations of Computer Science (FOCS)*, volume 33, pages 80–80, 1992.

[AESW91] Pankaj K. Agarwal, Herbert Edelsbrunner, Otfried Schwarzkopf, and Emo Welzl. Euclidean minimum spanning trees and bichromatic closest pairs. *Discrete & Computational Geometry*, 6(3):407–422, 1991.

[AFS12] Andreas Argyriou, Rina Foygel, and Nathan Srebro. Sparse prediction with the k-support norm. *Advances in Neural Information Processing Systems (NeurIPS)*, 25, 2012.

[AG19] Joran van Apeldoorn and András Gilyén. Improvements in quantum SDP-solving with applications. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 99:1–99:15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. arXiv:1804.05058.

[AGDLHG05] Alán Aspuru-Guzik, Anthony D Dutoi, Peter J Love, and Martin Head-Gordon. Simulated quantum computation of molecular energies. *Science*, 309(5741):1704–1707, 2005.

[AGG^{+}20] Srinivasan Arunachalam, Alex B Grilo, Tom Gur, Igor C Oliveira, and Aarthi Sundaram. Quantum learning algorithms imply circuit lower bounds. *arXiv preprint arXiv:2012.01920*, 2020.

[AGGW17] Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. Quantum sdp-solvers: Better upper and lower bounds. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 403–414. IEEE, 2017.

[AGGW20] Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. Convex optimization using quantum oracles. *Quantum*, 4:220, 2020. arXiv:1809.00643.

[AGKM12] Sanjeev Arora, Rong Ge, Ravindran Kannan, and Ankur Moitra. Computing a nonnegative matrix factorization–provably. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing (STOC)*, pages 145–162. `https://arxiv.org/pdf/1111.0952.pdf`, 2012.

[AGKZ20] Ryan Amos, Marios Georgiou, Aggelos Kiayias, and Mark Zhandry. One-shot signatures and applications to hybrid quantum/classical authentication. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, pages 255–268. Association for Computing Machinery, 2020.

[AGSS12] Sanjeev Arora, Rong Ge, Sushant Sachdeva, and Grant Schoenebeck. Finding overlapping communities in social networks: toward a rigorous approach. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, pages 37–54, 2012.

[AHKZ20] Jonathan Allcock, Chang-Yu Hsieh, Iordanis Kerenidis, and Shengyu Zhang. Quantum algorithms for feedforward neural networks. *ACM Transactions on Quantum Computing*, 1(1):1–24, 2020.

[AHN$^+$21] Srinivasan Arunachalam, Vojtech Havlicek, Giacomo Nannicini, Kristan Temme, and Pawel Wocjan. Simpler (classical) and faster (quantum) algorithms for Gibbs partition functions. In *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 112–122. IEEE, 2021. arXiv:2009.11270.

[AHO98] F Alizadeh, JPA Haeberly, and ML Overton. Primal-dual interior-point methods for semidefinite programming: convergence rates, stability and numerical results. In *SIAM Journal on Optimization*, 1998.

[AHP20] Elad Aigner-Horev and Yury Person. On sparse random combinatorial matrices. *arXiv preprint arXiv:2010.07648*, 2020.

[AIL$^+$15] Alexandr Andoni, Piotr Indyk, TMM Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. Practical and optimal lsh for angular distance. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1225–1233. Curran Associates, 2015.

[AIR18] Alexandr Andoni, Piotr Indyk, and Ilya Razenshteyn. Approximate nearest neighbor search in high dimensions. *arXiv preprint arXiv:1806.09823*, 7, 2018.

[AJT19] Vedat Levi Alev, Fernando Granha Jeronimo, and Madhur Tulsiani. Approximating constraint satisfaction problems on high-dimensional expanders. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 180–201. IEEE, 2019.

[AK07] Sanjeev Arora and Satyen Kale. A combinatorial, primal-dual approach to semidefinite programs. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*, 2007.

[AK22] Eric R Anschuetz and Bobak T Kiani. Beyond barren plateaus: Quantum variational algorithms are swamped with traps. *arXiv preprint arXiv:2205.05786*, 2022.

[AKK$^+$08] Sanjeev Arora, Subhash A Khot, Alexandra Kolla, David Steurer, Madhur Tulsiani, and Nisheeth K Vishnoi. Unique games on expanding constraint graphs are easy. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 21–28, 2008.

[AKK$^+$20] Thomas D. Ahle, Michael Kapralov, Jakob Bæk Tejs Knudsen, Rasmus Pagh, Ameya Velingker, David P. Woodruff, and Amir Zandieh. Oblivious sketching of high-degree polynomial kernels. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 141–160, 2020.

[AL12] Radosław Adamczak and Rafał Latała. Tail and moment estimates for chaoses generated by symmetric random variables with logarithmically concave tails. *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, 48(4):1103 – 1136, 2012.

[AL13] Amir Abboud and Kevin Lewi. Exact weight subgraphs and the k-sum conjecture. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, 2013.

[AL20a] Vedat Levi Alev and Lap Chi Lau. Improved analysis of higher order random walks and applications. *arXiv preprint arXiv:2001.02827*, 2020.

[AL20b] Andris Ambainis and Nikita Larka. Quantum algorithms for computational geometry problems. In *15th Conference on the Theory of Quantum Computation, Communication and Cryptography*, 2020.

[AL22] Dong An and Lin Lin. Quantum linear system solver based on time-optimal adiabatic quantum computing and quantum approximate optimization algorithm. *ACM Transactions on Quantum Computing*, 3(2), mar 2022.

[ALG20] Nima Anari, Kuikui Liu, and Shayan Oveis Gharan. Spectral independence in high-dimensional expanders and applications to the hardcore model. *arXiv preprint arXiv:2001.00303*, 2020.

[ALGV19] Nima Anari, Kuikui Liu, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials ii: high-dimensional walks and an fpras for counting bases of a matroid. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1–12, 2019.

[ALL$^+$21] Dong An, Noah Linden, Jin-Peng Liu, Ashley Montanaro, Changpeng Shao, and Jiasu Wang. Quantum-accelerated multilevel Monte Carlo methods for stochastic differential equations in mathematical finance. *Quantum*, 5:481, 2021. arXiv:2012.06283.

[Alm19] Josh Alman. An illuminating algorithm for the light bulb problem. In *2nd Symposium on Simplicity in Algorithms (SOSA)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[ALM21] Radosław Adamczak, Rafał Latała, and Rafał Meller. Moments of gaussian chaoses in banach spaces. *Electronic Journal of Probability*, 26:1–36, 2021.

[ALO16] Zeyuan Allen Zhu, Yin Tat Lee, and Lorenzo Orecchia. Using optimization to obtain a width-independent, parallel, simpler, and faster positive SDP solver. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms(SODA)*, 2016.

[ALO$^+$21] Nima Anari, Kuikui Liu, Shayan Oveis Gharan, Cynthia Vinzant, and Thuy-Duong Vuong. Log-concave polynomials iv: approximate exchange, tight mixing times, and near-optimal sampling of forests. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 408–420, 2021.

[ALOV19] Nima Anari, Kuikui Liu, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials ii: high-dimensional walks and an fpras for counting bases of a matroid. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1–12, 2019.

[ALS$^+$18] Alexandr Andoni, Chengyu Lin, Ying Sheng, Peilin Zhong, and Ruiqi Zhong. Subspace embedding and linear regression with orlicz norm. In *International Conference on Machine Learning (ICML)*, pages 224–233. PMLR, 2018.

[ALW14] Amir Abboud, Kevin Lewi, and Ryan Williams. Losing weight by gaining edges. In *European Symposium on Algorithms (ESA)*, 2014.

[AM20] Srinivasan Arunachalam and Reevu Maity. Quantum boosting. In *International Conference on Machine Learning (ICML)*, pages 377–387. PMLR, 2020.

[Amb04] Andris Ambainis. Quantum search algorithms. *SIGACT News*, 35(2):22–35, 2004.

[Amb07] Andris Ambainis. Quantum walk algorithm for element distinctness. *SIAM Journal on Computing*, 37(1):210–239, 2007.

[Amb14] Andris Ambainis. On physical problems that are slightly more difficult than qma. In *2014 IEEE 29th Conference on Computational Complexity (CCC)*, pages 32–43, 2014.

[Ami19] Nima Amini. Spectrahedrality of hyperbolicity cones of multivariate matching polynomials. *Journal of Algebraic Combinatorics*, 50(2):165–190, 2019.

[AMOV18] Nima Anari, Tung Mai, Shayan Oveis Gharan, and Vijay V Vazirani. Nash social welfare for indivisible items under separable, piecewise-linear concave utilities. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2274–2290. SIAM, 2018.

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and system sciences*, 58(1):137–147, 1999.

[AMV21] Kyriakos Axiotis, Aleksander Madry, and Adrian Vladu. Faster sparse minimum cost flow by electrical flow localization. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, 2021.

[ANN$^+$17] Alexandr Andoni, Huy L Nguyen, Aleksandar Nikolov, Ilya Razenshteyn, and Erik Waingarten. Approximate near neighbors for general symmetric norms. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 902–913, 2017.

[ANN$^+$18] Alexandr Andoni, Assaf Naor, Aleksandar Nikolov, Ilya Razenshteyn, and Erik Waingarten. Hölder homeomorphisms and approximate nearest neighbors. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 159–169. IEEE, 2018.

[Ans00] Kurt M Anstreicher. The volumetric barrier for semidefinite programming. *Mathematics of Operations Research*, 2000.

[ANTZ21] Brandon Augustino, Giacomo Nannicini, Tamás Terlaky, and Luis F Zuluaga. Quantum interior point methods for semidefinite optimization. *arXiv preprint arXiv:2112.06025*, 2021.

[ANW14] Haim Avron, Huy L. Nguyen, and David P. Woodruff. Subspace embeddings for the polynomial kernel. In *NeurIPS*, 2014.

[AO14] Nima Anari and Shayan Oveis Gharan. The kadison-singer problem for strongly rayleigh measures and applications to asymmetric tsp. In *arXiv preprint*. `https://arxiv.org/pdf/1412.1143.pdf`, 2014.

[AO15] Nima Anari and Shayan Oveis Gharan. Effective-resistance-reducing flows, spectrally thin trees, and asymmetric tsp. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 20–39. IEEE, 2015.

[AO17] Nima Anari and Shayan Oveis Gharan. A generalization of permanent inequalities and applications in counting and optimization. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 384–396, 2017.

[AOR16] Nima Anari, Shayan Oveis Gharan, and Alireza Rezaei. Monte carlo markov chain algorithms for sampling strongly rayleigh distributions and determinantal point processes. In *Conference on Learning Theory*, pages 103–115. PMLR, 2016.

[AOSS16] Nima Anari, Shayan Oveis Gharan, Amin Saberi, and Mohit Singh. Nash social welfare, matrix permanent, and stable polynomials. *arXiv preprint arXiv:1609.07056*, 2016.

[AOSS18] Nima Anari, Shayan Oveis Gharan, Amin Saberi, and Nikhil Srivastava. Approximating the largest root and applications to interlacing families. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1015–1028. SIAM, 2018.

[AOV18] Nima Anari, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials, entropy, and a deterministic approximation algorithm for counting bases of matroids. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 35–46. IEEE, 2018.

[AP20] Prabhanjan Ananth and Rolando L. La Placa. Secure software leasing, 2020.

[AR15] Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing (STOC)*, pages 793–801, 2015.

[ARN17] Alexandr Andoni, Ilya Razenshteyn, and Negev Shekel Nosatzki. Lsh forest: Practical algorithms made theoretical. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 67–78. SIAM, 2017.

[ARV09] Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM)*, 2009.

[AS04] Scott Aaronson and Yaoyun Shi. Quantum lower bounds for the collision and the element distinctness problems. *Journal of the ACM (JACM)*, 51(4):595–605, 2004.

[AS16] Noga Alon and Joel H. Spencer. *The probabilistic method*. Wiley Series in Discrete Mathematics and Optimization. John Wiley & Sons, Inc., Hoboken, NJ, fourth edition, 2016.

[ASSN08] Abiodun Musa Aibinu, M. J. E. Salami, Amir Akramin Shafie, and Athaur Rahman Najeeb. Mri reconstruction using discrete fourier transform: A tutorial. *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 2:1852–1858, 2008.

[ASZ$^+$21] Amira Abbas, David Sutter, Christa Zoufal, Aurélien Lucchi, Alessio Figalli, and Stefan Woerner. The power of quantum neural networks. *Nature Computational Science*, 1(6):403–409, 2021.

[ATS03] Dorit Aharonov and Amnon Ta-Shma. Adiabatic quantum state generation and statistical zero knowledge. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 20–29, 2003. arXiv:quant-ph/0301023.

[AUY83] Alfred V Aho, Jeffrey D Ullman, and Mihalis Yannakakis. On notions of information transfer in vlsi circuits. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 133–139, 1983.

[AV95] David S Atkinson and Pravin M Vaidya. A cutting plane algorithm for convex programming that uses analytic centers. *Mathematical Programming*, 69(1-3):1–43, 1995.

[AW02] Rudolf Ahlswede and Andreas Winter. Strong converse for identification via quantum channels. *ITIT*, 48(3):569–579, 2002.

[AW08] Arash A Amini and Martin J Wainwright. High-dimensional analysis of semidefinite relaxations for sparse principal components. In *2008 IEEE International Symposium on Information Theory (ISIT)*, pages 2454–2458. IEEE, 2008.

[AW15] Josh Alman and Ryan Williams. Probabilistic polynomials and hamming nearest neighbors. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2015)*, pages 136–150. IEEE, 2015.

[AW21] Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 522–539. SIAM, 2021.

[AWY14] Amir Abboud, Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 218–230. SIAM, 2014.

[AWY15] Amir Abboud, Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2015)*, pages 218–230, 2015.

[AWY18] Josh Alman, Joshua R Wang, and Huacheng Yu. Cell-probe lower bounds from online communication complexity. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1003–1012, 2018.

[AY22] Srinivasan Arunachalam and Penghui Yao. Positive spectrahedra: Invariance principles and pseudorandom generators. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, page 208–221, New York, NY, USA, 2022. Association for Computing Machinery.

[AYBGM17] Yasin Abbasi-Yadkori, Peter Bartlett, Victor Gabillon, and Alan Malek. Hit-and-run for sampling and planning in non-convex spaces. In *Artificial Intelligence and Statistics*, pages 888–895. PMLR, 2017. arXiv:1610.08865.

[AZL17] Zeyuan Allen-Zhu and Yuanzhi Li. Follow the compressed leader: faster online learning of eigenvectors and faster mmwu. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

[AZL20] Zeyuan Allen-Zhu and Yuanzhi Li. Backward feature correction: How deep learning performs deep learning. *arXiv preprint arXiv:2001.04413*, 2020.

[AZL22] Zeyuan Allen-Zhu and Yuanzhi Li. Feature purification: How adversarial training performs robust deep learning. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 977–988. IEEE, 2022.

[AZLS19a] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *ICML*, 2019.

[AZLS19b] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. On the convergence rate of training recurrent neural networks. In *NeurIPS*, 2019.

[Ban10] Nikhil Bansal. Constructive algorithms for discrepancy minimization. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 3–10. IEEE, 2010.

[Ban19] Nikhil Bansal. On a generalization of iterated and randomized rounding. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2019.

[BB08] Julius Borcea and Petter Brändén. Applications of stable polynomials to mixed determinants: Johnson's conjectures, unimodality, and symmetrized fischer products. *Duke Mathematical Journal*, 143(2):205–223, 2008.

[BB09] Julius Borcea and Petter Brändén. The lee-yang and pólya-schur programs. ii. theory of stable polynomials and applications. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 62(12):1595–1631, 2009.

[BBB$^+$19] Frank Ban, Vijay Bhattiprolu, Karl Bringmann, Pavel Kolev, Euiwoong Lee, and David P Woodruff. A ptas for lp-low rank approximation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 747–766. SIAM, 2019.

[BBBV97] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, Oct 1997.

[BBC$^+$17] Jarosław Błasiok, Vladimir Braverman, Stephen R Chestnut, Robert Krauthgamer, and Lin F Yang. Streaming symmetric norms via measure concentration. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 716–729, 2017.

[BBF$^+$20] Kerstin Beer, Dmytro Bondarenko, Terry Farrelly, Tobias J Osborne, Robert Salzmann, Daniel Scheiermann, and Ramona Wolf. Training deep quantum neural networks. *Nature communications*, 11(1):1–6, 2020.

[BBK$^+$20] Mitali Bafna, Boaz Barak, Pravesh Kothari, Tselil Schramm, and David Steurer. Playing unique games on certified small-set expanders. *arXiv preprint arXiv:2006.09969*, 2020.

[BBL09] Julius Borcea, Petter Brändén, and Thomas Liggett. Negative dependence and the geometry of polynomials. *Journal of the American Mathematical Society*, 22(2):521–567, 2009.

[BBRR$^+$87] H.C.P. Berbee, C.G.E. Boender, A.H.G. Rinnooy Ran, C.L. Scheffer, Robert L. Smith, and Jan Telgen. Hit-and-run algorithms for the identification of nonredundant linear inequalities. *Mathematical Programming*, 37(2):184–207, 1987.

[BCC$^+$15] Dominic W Berry, Andrew M Childs, Richard Cleve, Robin Kothari, and Rolando D Somma. Simulating hamiltonian dynamics with a truncated taylor series. *Physical review letters*, 114(9):090502, 2015.

[BCC$^+$17] Dominic W. Berry, Andrew M. Childs, Richard Cleve, Robin Kothari, and Rolando D. Somma. Exponential improvement in precision for simulating sparse Hamiltonians. *Forum of Mathematics, Sigma*, 5, 2017.

[BCC$^+$21] Antonio Blanca, Pietro Caputo, Zongchen Chen, Daniel Parisi, Daniel Štefankovič, and Eric Vigoda. On mixing of markov chains: Coupling, spectral independence, and entropy factorization. *arXiv preprint arXiv:2103.07459*, 2021.

[BCK15] Dominic W. Berry, Andrew M. Childs, and Robin Kothari. Hamiltonian simulation with nearly optimal dependence on all parameters. *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, Oct 2015.

[BCMS19] Marcin Bownik, Pete Casazza, Adam W Marcus, and Darrin Speegle. Improved bounds in weaver and feichtinger conjectures. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 2019(749):267–293, 2019.

[BCMV14] Aditya Bhaskara, Moses Charikar, Ankur Moitra, and Aravindan Vijayaraghavan. Smoothed analysis of tensor decompositions. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing (STOC)*, pages 594–603, 2014.

[BCS97] Peter Bürgisser, Michael Clausen, and Mohammad A Shokrollahi. *Algebraic complexity theory*, volume 315. Springer Science & Business Media, 1997.

[BCWdW01] Harry Buhrman, Richard Cleve, John Watrous, and Ronald de Wolf. Quantum fingerprinting. *Phys. Rev. Lett.*, 87:167902, Sep 2001.

[BDG16] Nikhil Bansal, Daniel Dadush, and Shashwat Garg. An algorithm for komlós conjecture matching banaszczyk. In *57th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2016.

[BDGL18] Nikhil Bansal, Daniel Dadush, Shashwat Garg, and Shachar Lovett. The gram-schmidt walk: a cure for the banaszczyk blues. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 587–597, 2018.

[BDGM20] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Factoring and pairings are not necessary for io: Circular-secure lwe suffices. *IACR Cryptol. ePrint Arch*, 2020:1024, 2020.

[BDS16] Shalev Ben-David and Or Sattath. Quantum tokens for digital signatures. *arXiv preprint arXiv:1609.09047*, 2016.

[BdWD$^+$01] Harry Buhrman, Ronald de Wolf, Christoph Dürr, Mark Heiligman, Peter H"yer, Frédéric Magniez, and Miklos Santha. Quantum algorithms for element distinctness. In *Proceedings of the 16th Annual Conference on Computational Complexity (CCC 2001)*, pages 131–137, Washington, DC, USA, 2001. IEEE.

[BE06] Peter Borwein and Tamás Erdélyi. Nikolskii-type inequalities for shift invariant function spaces. *Proceedings of the American Mathematical Society*, 134(11):3243–3246, 2006.

[Bel58] Richard Bellman. On a routing problem. *Quarterly of applied mathematics*, 16(1):87–90, 1958.

[Ber24] Sergei Bernstein. On a modification of chebyshev's inequality and of the error formula of laplace. *Ann. Sci. Inst. Sav. Ukraine, Sect. Math*, 1(4):38–49, 1924.

[Bes98] Sergei N Bespamyatnikh. An optimal algorithm for closest-pair maintenance. *Discrete & Computational Geometry*, 19(2):175–195, 1998.

[BFS11] Brielin Brown, Steven T Flammia, and Norbert Schuch. Computational difficulty of computing the density of states. *Physical review letters*, 107(4):040501, 2011.

[BFV20] Adam Bouland, Bill Fefferman, and Umesh Vazirani. Computational Pseudorandomness, the Wormhole Growth Paradox, and Constraints on the AdS/CFT Duality (Abstract). In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, volume 151 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 63:1–63:2, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[BG17] Nikhil Bansal and Shashwat Garg. Algorithmic discrepancy beyond partial coloring. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2017.

[BGB$^+$18] Ryan Babbush, Craig Gidney, Dominic W Berry, Nathan Wiebe, Jarrod McClean, Alexandru Paler, Austin Fowler, and Hartmut Neven. Encoding electronic spectra in quantum circuits with linear t complexity. *Physical Review X*, 8(4):041015, 2018.

[BGI$^+$01] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im) possibility of obfuscating programs. In *Annual International Cryptology Conference*, pages 1–18. Springer, 2001.

[BGLS01] Heinz H Bauschke, Osman Güler, Adrian S Lewis, and Hristo S Sendov. Hyperbolic polynomials and convex analysis. *Canadian Journal of Mathematics*, 53(3):470–488, 2001.

[BGMZ18] James Bartusek, Jiaxin Guan, Fermi Ma, and Mark Zhandry. Preventing zeroizing attacks on ggh15. In *Proceedings of TCC 2018*, 2018.

[BGS13] Anne Broadbent, Gus Gutoski, and Douglas Stebila. Quantum one-time programs. In *Annual Cryptology Conference*, pages 344–360. Springer, 2013.

[BH12] Andries E Brouwer and Willem H Haemers. Distance-regular graphs. In *Spectra of graphs*, pages 177–185. Springer, 2012.

[Bha97] Rajendra Bhatia. Matrix analysis, volume 169 of. *Graduate texts in mathematics*, 1997.

[BHKL20] Mitali Bafna, Max Hopkins, Tali Kaufman, and Shachar Lovett. High dimensional expanders: Eigenstripping, pseudorandomness, and unique games. *arXiv e-prints*, pages arXiv–2011, 2020.

[BHKL21] Mitali Bafna, Max Hopkins, Tali Kaufman, and Shachar Lovett. Hypercontractivity on high dimensional expanders: a local-to-global approach for higher moments. *arXiv preprint arXiv:2111.09444*, 2021.

[BHMT02] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.

[BIS17] Arturs Backurs, Piotr Indyk, and Ludwig Schmidt. On the fine-grained complexity of empirical risk minimization: Kernel methods and neural networks. In *Advances in Neural Information Processing Systems*, pages 4308–4318, 2017.

[BIW19] Arturs Backurs, Piotr Indyk, and Tal Wagner. Space and time efficient kernel density estimation in high dimensions. In *NeurIPS*, pages 15773–15782, 2019.

[BIWX11] Arnab Bhattacharyya, Piotr Indyk, David P Woodruff, and Ning Xie. The complexity of linear dependence problems in vector spaces. In *Innovations in Computer Science (ICS)*, pages 496–508, 2011.

[BJL$^+$21] Anne Broadbent, Stacey Jeffery, Sébastien Lord, Supartha Podder, and Aarthi Sundaram. Secure software leasing without assumptions, 2021.

[BJLM13] Daniel J Bernstein, Stacey Jeffery, Tanja Lange, and Alexander Meurer. Quantum algorithms for the subset-sum problem. In *International Workshop on Post-Quantum Cryptography*, pages 16–33. Springer, 2013.

[BJM22] Nikhil Bansal, Haotian Jiang, and Raghu Meka. Resolving matrix spencer conjecture up to poly-logarithmic rank. *arXiv preprint arXiv:2208.11286*, 2022.

[BK99] Piotr Berman and Marek Karpinski. On some tighter inapproximability results. In *International Colloquium on Automata, Languages, and Programming*, pages 200–209. Springer, 1999.

[BKKT20] Sergey Bravyi, Alexander Kliesch, Robert Koenig, and Eugene Tang. Obstacles to variational quantum optimization from symmetry protection, Dec 2020.

[BKL+17] Fernando GSL Brandão, Amir Kalev, Tongyang Li, Cedric Yen-Yu Lin, Krysta M Svore, and Xiaodi Wu. Exponential quantum speed-ups for semidefinite programming with applications to quantum learning. *arXiv preprint arXiv:1710.02581*, 84, 2017.

[BKL+19] Fernando GSL Brandão, Amir Kalev, Tongyang Li, Cedric Yen-Yu Lin, Krysta M Svore, and Xiaodi Wu. Quantum sdp solvers: Large speed-ups, optimality, and applications to quantum learning. In *46th International Colloquium on Automata, Languages, and Programming (ICALP)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[BKS18] Boaz Barak, Pravesh K Kothari, and David Steurer. Small-set expansion in shortcode graph and the 2-to-2 conjecture. *arXiv preprint arXiv:1804.08662*, 2018.

[BL19] Anne Broadbent and Sébastien Lord. Uncloneable quantum encryption via random oracles. *IACR Cryptology ePrint Archive*, 2019:257, 2019.

[BLH18] Alberto Bernacchia, Mate Lengyel, and Guillaume Hennequin. Exact natural gradient in deep linear networks and its application to the nonlinear case. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2018.

[BLL$^+$21] Jan van den Brand, Yin Tat Lee, Yang P Liu, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Minimum cost flows, mdps, and ℓ_1-regression in nearly linear time for dense instances. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*. arXiv preprint arXiv:2101.05719, 2021.

[BLNR15] Alexandre Belloni, Tengyuan Liang, Hariharan Narayanan, and Alexander Rakhlin. Escaping the local minima via simulated annealing: Optimization of approximately convex functions. In *Conference on Learning Theory*, pages 240–265. PMLR, 2015. arXiv:1501.07242.

[BLNZ95] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, 16(5):1190–1208, 1995.

[BLSS20] Jan van den Brand, Yin Tat Lee, Aaron Sidford, and Zhao Song. Solving tall dense linear programs in nearly linear time. In *52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2020.

[BMN$^+$21] Ryan Babbush, Jarrod R McClean, Michael Newman, Craig Gidney, Sergio Boixo, and Hartmut Neven. Focus beyond quadratic speedups for error-corrected quantum advantage. *PRX Quantum*, 2(1):010103, 2021.

[BMO$^+$15] Boaz Barak, Ankur Moitra, Ryan O'Donnell, Prasad Raghavendra, Oded Regev, David Steurer, Luca Trevisan, Aravindan Vijayaraghavan, David Witmer, and John Wright. Beating the random assignment on constraint satisfaction problems of bounded degree. *CoRR*, abs/1505.03424, 2015.

[BMR$^+$20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[BNWN22] Aaron Bernstein, Danupon Nanongkai, and Christian Wulff-Nilsen. Negative-weight single-source shortest paths in near-linear time. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 600–611. IEEE, 2022.

[BOG08] Michael Ben-Or and Dan Gutfreund. Trading help for interaction in statistical zero-knowledge proofs. *Journal of Cryptology*, 16:95–116, 03 2008.

[BOM$^+$21] Kyle EC Booth, Bryan O'Gorman, Jeffrey Marshall, Stuart Hadfield, and Eleanor Rieffel. Quantum-accelerated constraint programming. *Quantum*, 5:550, 2021.

[Bou14] Jean Bourgain. An improved estimate in the restricted isometry problem. In *Geometric aspects of functional analysis*, pages 65–70. Springer, 2014.

[BP15] Nir Bitansky and Omer Paneth. On non-black-box simulation and the impossibility of approximate obfuscation. *SIAM Journal on Computing*, 44(5):1325–1383, 2015.

[BPC$^+$11] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[BPLC$^+$19] Carlos Bravo-Prieto, Ryan LaRose, M. Cerezo, Yigit Subasi, Lukasz Cincio, and Patrick J. Coles. Variational quantum linear solver, 2019.

[BPS19] Harry Buhrman, Subhasree Patro, and Florian Speelman. The quantum strong exponential-time hypothesis. *arXiv preprint arXiv:1911.05686*, 2019.

[BPS21] Harry Buhrman, Subhasree Patro, and Florian Speelman. A Framework of Quantum Strong Exponential-Time Hypotheses. In Markus Bläser and Benjamin Monmege, editors, *38th International Symposium on Theoretical Aspects of Computer Science (STACS 2021)*, volume 187 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 19:1–19:19, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[BPSW21] Jan van den Brand, Binghui Peng, Zhao Song, and Omri Weinstein. Training (overparametrized) neural networks in near-linear time. In *12th Innovations in Theoretical Computer Science Conference (ITCS)*, 2021.

[Brä10] Petter Brändén. Notes on hyperbolicity cones. *Verfügbar unter https://math. berkeley. edu/˜ bernd/branden. pdf*, 2010.

[Brä11] Petter Brändén. Obstructions to determinantal representability. *Advances in Mathematics*, 226(2):1202–1212, 2011.

[Brä14] Petter Brändén. Hyperbolicity cones of elementary symmetric polynomials are spectrahedral. *Optimization Letters*, 8(5):1773–1782, 2014.

[Brä18] Petter Brändén. Hyperbolic polynomials and the Kadison-Singer problem. In *arXiv preprint*. `https://arxiv.org/pdf/1809.03255`, 2018.

[Bra20] Jan van den Brand. A deterministic linear program solver in current matrix multiplication time. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2020.

[Bra21] Jan van den Brand. Unifying matrix data structures: Simplifying and speeding up iterative algorithms. In *Symposium on Simplicity in Algorithms (SOSA)*, pages 1–13. SIAM, 2021.

[BRB17] Aleksandar Botev, Hippolyt Ritter, and David Barber. Practical Gauss-Newton optimisation for deep learning. In *Proceedings of the 34th International Conference on Machine Learning*, pages 557–565, 2017.

[BRS93] Claude J.P. Bélisle, H. Edwin Romeijn, and Robert L. Smith. Hit-and-run algorithms for generating multivariate distributions. *Mathematics of Operations Research*, 18(2):255–266, 1993.

[BRSV17] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Proofs of useful work. *IACR Cryptology ePrint Archive*, 2017:203, 2017.

[BS76] Jon Louis Bentley and Michael Ian Shamos. Divide-and-conquer in multidimensional space. In *Proceedings of the 8th annual ACM Sym-*

posium on Theory of Computing (STOC 1976), pages 220–230. ACM, 1976.

[BS16] Boaz Barak and David Steurer. Proofs, beliefs, and algorithms through the lens of sum-of-squares. *Course notes: http://www. sumofsquares.org/public/index.ht* 2016.

[BS17] Fernando GSL Brandao and Krysta M Svore. Quantum speed-ups for solving semidefinite programs. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 415–426. IEEE, 2017.

[BSS12] Joshua Batson, Daniel A Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. *SIAM Journal on Computing*, 41(6):1704–1721, 2012.

[BT06] Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for np problems. *SIAM Journal on Computing*, 36(4):1119–1159, 2006.

[BTN01] Aharon Ben-Tal and Arkadi Nemirovski. *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. SIAM, 2001.

[Bub15] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[BV02] Dimitris Bertsimas and Santosh Vempala. Solving convex programs by random walks. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing (STOC)*, pages 109–115. ACM, 2002.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[BV10] Radim Belohlavek and Vilem Vychodil. Discovery of optimal factors in binary data via a novel method of matrix decomposition. *Journal of Computer and System Sciences*, 76(1):3–20, 2010.

[BVY14] Sam Burton, Cynthia Vinzant, and Yewon Youm. A real stable extension of the vamos matroid polynomial. *arXiv preprint arXiv:1411.2038*, 2014.

[BWP$^+$17] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

[BWZ16] Christos Boutsidis, David P Woodruff, and Peilin Zhong. Optimal principal component analysis in distributed and streaming models. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing (STOC)*, pages 236–249, 2016.

[BYJKS04] Ziv Bar-Yossef, Thathachar S Jayram, Ravi Kumar, and D Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68(4):702–732, 2004.

[BZ06] Ingemar Bengtsson and Karol Zyczkowski. *Geometry of Quantum States: An Introduction to Quantum Entanglement.* Cambridge University Press, 2006.

[Cam19] Earl Campbell. Random compiler for fast hamiltonian simulation. *Physical review letters*, 123(7):070503, 2019.

[Cam21] Earl T Campbell. Early fault-tolerant simulations of the hubbard model. *Quantum Science and Technology*, 7(1):015007, 2021.

[CB18] Lenaic Chizat and Francis Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. *Advances in neural information processing systems*, 31, 2018.

[CBKC21] Laura Clinton, Johannes Bausch, Joel Klassen, and Toby Cubitt. Phase estimation of local hamiltonians on nisq hardware, 2021.

[CC89] Andrew Chi-Chih. Lower bounds for algebraic computation trees with integer inputs. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS 1989)*, pages 308–313, 1989.

[CCAY$^+$18] Xiang Cheng, Niladri S Chatterji, Yasin Abbasi-Yadkori, Peter L Bartlett, and Michael I Jordan. Sharp convergence rates for langevin dynamics in the nonconvex setting. *arXiv preprint arXiv:1805.01648*, 2018.

[CCBJ18] Xiang Cheng, Niladri S. Chatterji, Peter L. Bartlett, and Michael I. Jordan. Underdamped langevin mcmc: A non-asymptotic analysis. In *Conference on learning theory*, pages 300–323. PMLR, 2018. arXiv:1707.03663.

[CCCW21] Shouvanik Chakrabarti, Chi-Ning Chou, Kai-Min Chung Chung, and Xiaodi Wu. Scalable verification of quantum supremacy based on circuit obfuscation. *Manuscript*, 2021.

[CCF04] Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theor. Comput. Sci.*, 312(1):3–15, 2004.

[CCGZ20] Zixiang Chen, Yuan Cao, Quanquan Gu, and Tong Zhang. A generalized neural tangent kernel analysis for two-layer neural networks. *Advances in Neural Information Processing Systems*, 33:13363–13373, 2020.

[CCH$^+$19] Shouvanik Chakrabarti, Andrew M. Childs, Shih-Han Hung, Tongyang Li, Chunhao Wang, and Xiaodi Wu. Quantum algorithm for estimating volumes of convex bodies, 2019. arXiv:1908.03903.

[CCH$^+$22] Nadiia Chepurko, Kenneth Clarkson, Lior Horesh, Honghao Lin, and David Woodruff. Quantum-inspired algorithms from randomized numerical linear algebra. In *International Conference on Machine Learning*, pages 3879–3900. PMLR, 2022.

[CCLW20] Shouvanik Chakrabarti, Andrew M. Childs, Tongyang Li, and Xiaodi Wu. Quantum algorithms and lower bounds for convex optimization. *Quantum*, 4:221, 2020. arXiv:1809.01731.

[CCZG21] Zixiang Chen, Yuan Cao, Difan Zou, and Quanquan Gu. How much over-parameterization is sufficient to learn deep ReLU networks? In *International Conference on Learning Representations (ICLR)*, 2021.

[CDG19] Yu Cheng, Ilias Diakonikolas, and Rong Ge. High-dimensional robust mean estimation in nearly-linear time. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2019.

[CDG$^+$20] Nicholas Carlini, Samuel Deng, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, Shuang Song, Abhradeep Thakurta, and Florian Tramer. An attack on instahide: Is private learning possible with instance encoding? *arXiv preprint arXiv:2011.05315*, 2020.

[CDGW19] Yu Cheng, Ilias Diakonikolas, Rong Ge, and David Woodruff. Faster algorithms for high-dimensional robust covariance estimation. In *Conference on Learning Theory (COLT)*, 2019.

[CDHS19] Yair Carmon, John C Duchi, Oliver Hinder, and Aaron Sidford. Lower bounds for finding stationary points i. *Mathematical Programming*, pages 1–50, 2019.

[CDST19] Yair Carmon, John C. Duchi, Aaron Sidford, and Kevin Tian. A rank-1 sketch for matrix multiplicative weights. In *Conference on Learning Theory (COLT)*, pages 589–623, 2019.

[CDWY20] Yuansi Chen, Raaz Dwivedi, Martin J. Wainwright, and Bin Yu. Fast mixing of Metropolized Hamiltonian Monte Carlo: Benefits of multi-step gradients. *Journal of Machine Learning Research*, 21(92):1–72, 2020. arXiv:1905.12247.

[CEL$^+$21] Sinho Chewi, Murat A Erdogdu, Mufan Bill Li, Ruoqi Shen, and Matthew Zhang. Analysis of langevin monte carlo from poincar\'e to log-sobolev. *arXiv preprint arXiv:2112.12662*, 2021.

[CFK$^+$15] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[CG18] Yu Cheng and Rong Ge. Non-convex matrix completion against a semi-random adversary. In *Conference On Learning Theory (COLT)*, 2018.

[CG19] Yuan Cao and Quanquan Gu. Generalization bounds of stochastic gradient descent for wide and deep neural networks. In *NeurIPS*, pages 10835–10845, 2019.

[CGH$^+$19] Tianle Cai, Ruiqi Gao, Jikai Hou, Siyu Chen, Dong Wang, Di He, Zhihua Zhang, and Liwei Wang. Gram-gauss-newton method: Learning overparameterized neural networks for regression problems. *arXiv preprint arXiv:1905.11675*, 2019.

[CGJ19] Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. The Power of Block-Encoded Matrix Powers: Improved Regression Techniques via Faster Hamiltonian Simulation. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 33:1–33:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[CGL$^+$20] Nai-Hui Chia, András Gilyén, Tongyang Li, Han-Hsuan Lin, Ewin Tang, and Chunhao Wang. Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning. In *Proceedings of the 52nd Annual ACM SIGACT symposium on theory of computing (STOC)*, pages 387–400, 2020.

[CGLZ20] Matthias Christandl, François Le Gall, Vladimir Lysikov, and Jeroen Zuiddam. Barriers for rectangular matrix multiplication. In *arXiv preprint*. `https://arxiv.org/pdf/2003.03019.pdf`, 2020.

[CGŠV21] Zongchen Chen, Andreas Galanis, Daniel Štefankovič, and Eric Vigoda. Rapid mixing for colorings via spectral independence. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1548–1557. SIAM, 2021.

[Cha00a] Timothy M Chan. Random sampling, halfspace range reporting, and construction of ($\leq k$)-levels in three dimensions. *SIAM Journal on Computing*, 30(2):561–575, 2000.

[Cha00b] Bernard Chazelle. *The discrepancy method.* Cambridge University Press, Cambridge, 2000. Randomness and complexity.

[Cha02] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing (STOC)*, pages 380–388, 2002.

[Cha12] Timothy M Chan. Optimal partition trees. *Discrete & Computational Geometry*, 47(4):661–690, 2012.

[Cha19] Timothy M Chan. Orthogonal range searching in moderate dimensions: kd trees and range trees strike back. *Discrete & Computational Geometry*, 61(4):899–922, 2019.

[CHC$^+$22] Matthias C Caro, Hsin-Yuan Huang, Marco Cerezo, Kunal Sharma, Andrew Sornborger, Lukasz Cincio, and Patrick J Coles. Generalization in quantum machine learning from few training data. *Nature communications*, 13(1):4919, 2022.

[Che52] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, pages 493–507, 1952.

[Che18] Lijie Chen. On the hardness of approximate and exact (bichromatic) maximum inner product. In *Proceedings of the 33rd Computational Complexity Conference (CCC 2018)*, pages 1–45, 2018.

[CHHK14] Parinya Chalermsook, Sandy Heydrich, Eugenia Holm, and Andreas Karrenbauer. Nearly tight approximability results for minimum biclique cover and partition. In *European Symposium on Algorithms (ESA)*, pages 235–246. Springer, 2014.

[Chi21] Andrew M. Childs. Lecture notes on quantum algorithms. Lecture notes at the University of Maryland, https://www.cs.umd.edu/ amchilds/qa/qa.pdf, 2021.

[CHL$^+$22] Xinyi Chen, Elad Hazan, Tongyang Li, Zhou Lu, Xinzhao Wang, and Rui Yang. Adaptive online learning of quantum states. *arXiv preprint arXiv:2206.00220*, 2022.

[CHN$^+$18] Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. Watermarking cryptographic capabilities. *SIAM Journal on Computing*, 47(6):2157–2202, 2018.

[CHO$^+$20] Lijie Chen, Shuichi Hirahara, Igor C Oliveira, Ján Pich, Ninad Rajgopal, and Rahul Santhanam. Beyond natural proofs: Hardness magnification and locality. *Leibniz International Proceedings in Informatics*, 151, 2020.

[CHS20] Nai-Hui Chia, Sean Hallgren, and Fang Song. On Basing One-way Permutations on NP-hard Problems under Quantum Reductions. *Quantum*, 4:312, August 2020.

[CIK17] Sunil Chandran, Davis Issac, and Andreas Karrenbauer. On the parameterized complexity of biclique cover and partition. In *11th International Symposium on Parameterized and Exact Computation (IPEC)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[CIKK16] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In *Proceedings of the 31st Conference on Computational Complexity*, CCC '16, Dagstuhl, DEU, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[CKL$^+$22] Li Chen, Rasmus Kyng, Yang P Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 612–623. IEEE, 2022.

[CKP13] Alessandro Cosentino, Robin Kothari, and Adam Paetznick. Dequantizing read-once quantum formulas. In *8th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2013)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.

[CKPS16] Xue Chen, Daniel M Kane, Eric Price, and Zhao Song. Fourier-sparse interpolation without a frequency gap. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 741–750. IEEE, 2016.

[CKS17] Andrew M Childs, Robin Kothari, and Rolando D Somma. Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. *SIAM Journal on Computing*, 46(6):1920–1950, 2017.

[CL12] Julia Chuzhoy and Shi Li. A polylogarithmic approximation algorithm for edge-disjoint paths with congestion 2. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 233–242, 2012.

[Cla88] Kenneth L Clarkson. A randomized algorithm for closest-point queries. *SIAM Journal on Computing*, 17(4):830–847, 1988.

[CLA$^+$21] Sinho Chewi, Chen Lu, Kwangjun Ahn, Xiang Cheng, Thibaut Le Gouic, and Philippe Rigollet. Optimal dimension dependence of the metropolis-adjusted langevin algorithm. In *Conference on Learning Theory*, pages 1260–1300. PMLR, 2021. arXiv:2012.12810.

[CLM20] Sitan Chen, Jerry Li, and Ankur Moitra. Learning structured distributions from untrusted batches: Faster and simpler. In *NeurIPS*. arXiv preprint arXiv:2002.10435, 2020.

[CLP$^+$21] Beidi Chen, Zichang Liu, Binghui Peng, Zhaozhuo Xu, Jonathan Lingjie Li, Tri Dao, Zhao Song, Anshumali Shrivastava, and Christopher Re.

MONGOOSE: A learnable LSH framework for efficient neural network training. In *Proceedings of the Nineth International Conference on Learning Representations (ICLR'2021)*, 2021.

[CLRS09] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT press, 2009.

[CLS19] Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*, 2019.

[CLSZ21] Sitan Chen, Xiaoxiao Li, Zhao Song, and Danyang Zhuo. On instahide, phase retrieval, and sparse matrix factorization. In *International Conference on Learning Representations*, 2021.

[CLV20] Zongchen Chen, Kuikui Liu, and Eric Vigoda. Rapid mixing of glauber dynamics up to uniqueness via contraction. *arXiv preprint arXiv:2004.09083*, 2020.

[CLV21] Zongchen Chen, Kuikui Liu, and Eric Vigoda. Optimal mixing of glauber dynamics: Entropy factorization via high-dimensional expansion. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1537–1550, 2021.

[CLW19] Yu Cao, Jianfeng Lu, and Lihan Wang. On explicit L^2-convergence rate estimate for underdamped Langevin dynamics, 2019. arXiv:1908.04746.

[CMD20] Pablo AM Casares and Miguel Angel Martin-Delgado. A quantum interior-point predictor–corrector algorithm for linear programming. *Journal of physics A: Mathematical and Theoretical*, 53(44):445305, 2020.

[CMJF$^+$20] Beidi Chen, Tharun Medini, Sameh Gobriel James Farwell, Charlie Tai, and Anshumali Shrivastava. SLIDE : In defense of smart algorithms over hardware acceleration for large-scale deep learning systems. In *MLSys'2020*, 2020.

[CMN$^+$18] Andrew M Childs, Dmitri Maslov, Yunseong Nam, Neil J Ross, and Yuan Su. Toward the first quantum simulation with quantum speedup. *Proceedings of the National Academy of Sciences*, 115(38):9456–9461, 2018.

[CMP20] Andrea Coladangelo, Christian Majenz, and Alexander Poremba. Quantum copy-protection of compute-and-compare programs in the quantum random oracle model, 2020.

[CMS11] Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(29):1069–1109, 2011.

[CN20] Yeshwanth Cherapanamjeri and Jelani Nelson. On adaptive distance estimation. In *Advances in Neural Information Processing Systems*, 2020.

[CN21] Yeshwanth Cherapanamjeri and Jelani Nelson. Terminal embeddings in sublinear time. In *FOCS*, 2021.

[CND$^+$22] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[COB19] Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in Neural Information Processing Systems*, 32, 2019.

[Coh16a] Michael Cohen. Improved spectral sparsification and Kadison-Singer for sums of higher-rank matrices. In *Banff International Research Station for Mathematical Innovation and Discovery*. `https://open.library.ubc.ca/cIRcle/collections/48630/items/1.0340957`, 2016.

[Coh16b] Michael B Cohen. Ramanujan graphs in polynomial time. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 276–281. IEEE, 2016.

[Cop82] Don Coppersmith. Rapid multiplication of rectangular matrices. *SIAM Journal on Computing*, 11(3):467–471, 1982.

[CP14] David Cattanéo and Simon Perdrix. The parameterized complexity of domination-type problems and application to linear codes. In *International Conference on Theory and Applications of Models of Computation*, pages 86–103. Springer, 2014.

[CP19a] Xue Chen and Eric Price. Active regression via linear-sample sparsification. In *Conference on Learning Theory (COLT)*, pages 663–695. PMLR, 2019.

[CP19b] Xue Chen and Eric Price. Estimating the frequency of a clustered signal. In *46th International Colloquium on Automata, Languages, and Programming (ICALP)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[CRAG18] Yudong Cao, Jhonathan Romero, and Alán Aspuru-Guzik. Potential of quantum computing for drug discovery. *IBM Journal of Research and Development*, 62(6):6–1, 2018.

[CRDH08] Yanhua Chen, Manjeet Rege, Ming Dong, and Jing Hua. Non-negative matrix factorization for semi-supervised data clustering. *Knowledge and Information Systems*, 17(3):355–379, 2008.

[CRO$^+$19] Yudong Cao, Jonathan Romero, Jonathan P Olson, Matthias Degroote, Peter D Johnson, Mária Kieferová, Ian D Kivlichan, Tim Menke, Borja Peropadre, Nicolas PD Sawaya, et al. Quantum chemistry in the age of quantum computing. *Chemical reviews*, 119(19):10856–10915, 2019.

[CRT06] Emmanuel J Candes, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 59(8):1207–1223, 2006.

[CS93] Ming-Hui Chen and Bruce Schmeiser. Performance of the Gibbs, hit-and-run, and Metropolis samplers. *Journal of Computational and Graphical Statistics*, 2(3):251–272, 1993.

[CS07] Maria Chudnovsky and Paul Seymour. The roots of the independence polynomial of a clawfree graph. *J. Combin. Theory Ser. B*, 97(3):350–357, 2007.

[CS09] Youngmin Cho and Lawrence Saul. Kernel methods for deep learning. *Advances in neural information processing systems*, 22, 2009.

[CT65] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

[CT06] Emmanuel J Candes and Terence Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE transactions on information theory*, 52(12):5406–5425, 2006.

[CT17] Timothy M Chan and Konstantinos Tsakalidis. Dynamic orthogonal range searching on the ram, revisited. *Leibniz International Proceedings in Informatics, LIPIcs*, 77:281–2813, 2017.

[CTV06] Kevin P Costello, Terence Tao, and Van Vu. Random symmetric matrices are almost surely nonsingular. *Duke Mathematical Journal*, 135(2):395–413, 2006.

[CV08] Kevin P Costello and Van H Vu. The rank of random graphs. *Random Structures & Algorithms*, 33(3):269–285, 2008.

[CV10] Kevin p Costello and Van Vu. On the rank of random sparse matrices. *Combinatorics, Probability and Computing*, 19(3):321–342, 2010.

[CVB20] Daan Camps and Roel Van Beeumen. Approximate quantum circuit synthesis using block encodings. *Physical Review A*, 102(5):052411, 2020.

[CW13] Kenneth L. Clarkson and David P. Woodruff. Low rank approximation and regression in input sparsity time. In *Symposium on Theory of Computing Conference (STOC)*, 2013.

[CW16] Timothy M Chan and Ryan Williams. Deterministic APSP, orthogonal vectors, and more: Quickly derandomizing razborov-smolensky. In *Proceedings of the 27th annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2016)*, pages 1246–1255. Society for Industrial and Applied Mathematics, 2016.

[CW19] Lijie Chen and Ryan Williams. An equivalence class for orthogonal vectors. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2019)*, pages 21–40. SIAM, 2019.

[CWB$^+$11] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537, 2011.

[Dan47] George B Dantzig. Maximization of a linear function of variables subject to linear inequalities. *Activity analysis of production and allocation*, 13:339–347, 1947.

[dBCvKO08] Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition.* Springer, 2008.

[DBd19] Radu-Alexandru Dragomir, Jérôme Bolte, and Alexandre d'Aspremont. Fast gradient methods for symmetric nonnegative matrix factorization. *arXiv preprint arXiv:1901.10791*, 2019.

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[DCR$^+$22] Alain Delgado, Pablo AM Casares, Roberto dos Reis, Modjtaba Shokrian Zini, Roberto Campos, Norge Cruz-Hernández, Arne-Christian Voigt, Angus Lowe, Soran Jahangiri, MA Martin-Delgado, et al. How to simulate key properties of lithium-ion batteries with a fault-tolerant quantum computer. *arXiv preprint arXiv:2204.11890*, 2022.

[DCWY18] Raaz Dwivedi, Yuansi Chen, Martin J. Wainwright, and Bin Yu. Log-concave sampling: Metropolis-Hastings algorithms are fast! In *Conference on learning theory*, pages 793–797. PMLR, 2018. arXiv:1801.02309.

[DD19] Yotam Dikstein and Irit Dinur. Agreement testing theorems on layered set systems. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1495–1524. IEEE, 2019.

[DDFH18] Yotam Dikstein, Irit Dinur, Yuval Filmus, and Prahladh Harsha. Boolean function analysis on high-dimensional expanders. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms* *and Techniques (APPROX/RANDOM 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[DDS$^+$09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[dEGJL07] Alexandre d'Aspremont, Laurent El Ghaoui, Michael I Jordan, and Gert RG Lanckriet. A direct formulation for sparse pca using semidefinite programming. *SIAM review*, 49(3):434–448, 2007.

[Del76] Philippe Delsarte. Association schemes and t-designs in regular semilattices. *Journal of Combinatorial Theory, Series A*, 20(2):230–243, 1976.

[DEL$^+$22] Irit Dinur, Shai Evra, Ron Livne, Alexander Lubotzky, and Shahar Mozes. Locally testable codes with constant rate, distance, and locality. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 357–374, 2022.

[dFDRC08] Ruairí de Fréin, Konstantinos Drakakis, Scott Rickard, and Andrzej Cichocki. Analysis of financial data using non-negative matrix factorization. *international mathematical forum*, 3(38):1853–1870, 2008.

[DG94] Jean-Pierre Dedieu and R. J. Gregorac. Corrigendum: "Obreschkoff's theorem revisited: what convex sets are contained in the set of hyperbolic polynomials?" [J. Pure Appl. Algebra **81** (1992), no. 3, 269–278; MR1179101 (93g:12001)] by Dedieu. *J. Pure Appl. Algebra*, 93(1):111–112, 1994.

[DH96] Christoph Durr and Peter Hoyer. A quantum algorithm for finding the minimum, 1996.

[DHK09] Varsha Dani, Thomas P. Hayes, and Sham M. Kakade. Structured logconcave sampling with a restricted gaussian oracle. In *Conference on Learning Theory*, pages 355–366, 2009.

[DHL19] Yihe Dong, Samuel Hopkins, and Jerry Li. Quantum entropy scoring for fast robust mean estimation and improved outlier detection. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6067–6077, 2019.

[DHS05] Chris Ding, Xiaofeng He, and Horst D Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the SIAM international conference on data mining (ICDM)*, pages 606–610. SIAM, 2005.

[DIIM04] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry (SoCG)*, pages 253–262, 2004.

[Dij59] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[DIRW20] Yihe Dong, Piotr Indyk, Ilya Razenshteyn, and Tal Wagner. Learning space partitions for nearest neighbor search. In *ICLR*. arXiv preprint arXiv:1901.08544, 2020.

[DJR21] Daniel Dadush, Haotian Jiang, and Victor Reis. A new framework for matrix discrepancy: Partial coloring bounds via mirror descent. *arXiv preprint arXiv:2111.03171*, 2021.

[DJS$^+$19] Huaian Diao, Rajesh Jayaram, Zhao Song, Wen Sun, and David Woodruff. Optimal sketching for kronecker product regression and low rank approximation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[DK17] Irit Dinur and Tali Kaufman. High dimensional expanders imply agreement expanders. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 974–985. IEEE, 2017.

[DK19] Arnak S. Dalalyan and Avetik Karagulyan. User-friendly guarantees for the Langevin Monte Carlo with inaccurate gradient. *Stochastic Processes and their Applications*, 129(12):5278–5311, 2019. arXiv:1710.00095.

[DKJ$^+$07] Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216, 2007.

[DKK$^+$16] I Diakonikolas, G Kamath, DM Kane, J Li, A Moitra, and A Stewart. Robust estimators in high dimensions without the computational intractability. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 655–664, 2016.

[DKK$^+$18a] Irit Dinur, Subhash Khot, Guy Kindler, Dor Minzer, and Muli Safra. On non-optimally expanding sets in grassmann graphs. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 940–951, 2018.

[DKK$^+$18b] Irit Dinur, Subhash Khot, Guy Kindler, Dor Minzer, and Muli Safra. Towards a proof of the 2-to-1 games conjecture? In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 376–389, 2018.

[DKS14] Martin Dyer, Ravi Kannan, and Leen Stougie. A simple randomised algorithm for convex optimisation: Application to two-stage stochastic programming. *Mathematical Programming*, 147:207–229, 2014.

[DKW05] Evgeny Dantsin, Vladik Kreinovich, and Alexander Wolpert. On quantum versions of record-breaking algorithms for SAT. *SIGACT News*, 36(4):103–108, December 2005.

[DLL$^+$19] Simon S Du, Jason D Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *ICML*, 2019.

[DLPM21] Michał Dereziński, Jonathan Lacotte, Mert Pilanci, and Michael W. Mahoney. Newton-less: Sparsification without trade-offs for the sketched newton update, 2021.

[DLT22] Yulong Dong, Lin Lin, and Yu Tong. Ground state preparation and energy estimation on early fault-tolerant quantum computers via quantum eigenvalue transformation of unitary matrices. *arXiv preprint arXiv:2204.05955*, 2022.

[DLY21] Sally Dong, Yin Tat Lee, and Guanghao Ye. A nearly-linear time algorithm for linear programs with small treewidth: A multiscale representation of robust central path. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2021.

[DM17] Alain Durmus and Eric Moulines. Nonasymptotic convergence analysis for the unadjusted Langevin algorithm. *The Annals of Applied Probability*, 27(3):1551–1587, 2017. arXiv:1507.05021.

[DM18] Irit Dinur and Pasin Manurangsi. Eth-hardness of approximating 2-csps and directed steiner network. In *9th Innovations in Theoretical*

Computer Science Conference (ITCS 2018). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[DMM19] Alain Durmus, Szymon Majewski, and Błażej Miasojedow. Analysis of Langevin Monte Carlo via convex optimization. *The Journal of Machine Learning Research*, 20(1):2666–2711, 2019. arXiv:1802.09188.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Conference on Theory of Cryptography*, TCC'06, page 265–284, Berlin, Heidelberg, 2006. Springer-Verlag.

[DMZS21] Shabnam Daghaghi, Nicholas Meisburger, Mengnan Zhao, and Anshumali Shrivastava. Accelerating slide deep learning on modern cpus: Vectorization, quantizations, memory optimizations, and more. *Proceedings of Machine Learning and Systems*, 3, 2021.

[DNTTJ18] Daniel Dadush, Aleksandar Nikolov, Kunal Talwar, and Nicole Tomczak-Jaegermann. Balancing vectors in any norm. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1–10. IEEE, 2018.

[Don06] D.L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[dP95] Gaspard Riche de Prony. Essai experimental et analytique: sur les lois de la dilatabilite des fluides elastique et sur celles de la force expansive de la vapeur de l'eau et de la vapeur de l'alkool, a differentes temperatures. *Journal Polytechnique ou Bulletin du Travail fait a l'Ecole Centrale des Travaux Publics*, 1795.

[DPMRF23] Giacomo De Palma, Milad Marvian, Cambyse Rouzé, and Daniel Stilck França. Limitations of variational quantum algorithms: a quantum optimal transport approach. *PRX Quantum*, 4(1):010309, 2023.

[DS95] Daniele Giorgio Degiorgi and Klaus Simon. A dynamic algorithm for line graph recognition. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 37–48. Springer, 1995.

[DS08] Samuel I Daitch and Daniel A Spielman. Faster approximate lossy generalized flow via interior point algorithms. In *Proceedings of the fortieth annual ACM symposium on Theory of computing (STOC)*, pages 451–460, 2008.

[DSL19] Roee David, Karthik C S., and Bundit Laekhanukit. On the complexity of closest pair via polar-pair of point-sets. *SIAM Journal on Discrete Mathematics*, 33(1):509–527, 2019.

[DSSW17] Huaian Diao, Zhao Song, Wen Sun, and David Woodruff. Sketching for kronecker product regression and p-splines. In *AISTATS*, 2017.

[DZPS19] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019.

[EGM$^+$20] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pages 2943–2952. PMLR, 2020.

[EH21] Murat A Erdogdu and Rasa Hosseinzadeh. On the convergence of langevin monte carlo: The interplay between tail growth and smoothness. In *Conference on Learning Theory*, pages 1776–1822. PMLR, 2021.

[EIS75] Shimon Even, Alon Itai, and Adi Shamir. "on the complexity of time table and multi-commodity flow problems". In *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)*, pages 184–193. IEEE, 1975.

[ER60] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[Erd45] Paul Erdös. On a lemma of littlewood and offord. *Bulletin of the American Mathematical Society*, 51(12):898–902, 1945.

[Eri95] Jeff Erickson. Lower bounds for linear satisfiability problems. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 388–395, 1995.

[ES18] Ronen Eldan and Mohit Singh. Efficient algorithms for discrepancy minimization in convex sets. *Random Structures & Algorithms*, 53(2):289–307, 2018.

[EU18] Alessandro Epasto and Eli Upfal. Efficient approximation for restricted biclique cover problems. *Algorithms*, 11(6):84, 2018.

[FAESS22] Alaa Fkirin, Gamal Attiya, Ayman El-Sayed, and Marwa A Shouman. Copyright protection of deep neural network models using digital watermarking: a comparative study. *Multimedia Tools and Applications*, 81(11):15961–15975, 2022.

[FC18] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018.

[Fel80] H. J. Fell. On the zeros of convex combinations of polynomials. *Pacific J. Math.*, 89(1):43–50, 1980.

[FG21] Spencer Frei and Quanquan Gu. Proxy convexity: A unified framework for the analysis of neural networks trained by gradient descent. *Advances in Neural Information Processing Systems*, 34:7937–7949, 2021.

[FGG14a] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[FGG14b] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm applied to a bounded occurrence constraint problem. *arXiv preprint arXiv:1412.6062*, 2014.

[FGG20a] Edward Farhi, David Gamarnik, and Sam Gutmann. The quantum approximate optimization algorithm needs to see the whole graph: A typical case. *arXiv preprint arXiv:2004.09002*, 2020.

[FGG20b] Edward Farhi, David Gamarnik, and Sam Gutmann. The quantum approximate optimization algorithm needs to see the whole graph: Worst case examples. *arXiv preprint arXiv:2005.08747*, 2020.

[FGH$^+$12] Edward Farhi, David Gosset, Avinatan Hassidim, Andrew Lutomirski, and Peter Shor. Quantum money from knots. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 276–289, New York, NY, USA, 2012. Association for Computing Machinery.

[FGL$^+$19] Fedor V Fomin, Petr A Golovach, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Approximation schemes for low-rank binary matrix approximation problems. *ACM Transactions on Algorithms (TALG)*, 16(1):1–39, 2019.

[FGYZ21] Weiming Feng, Heng Guo, Yitong Yin, and Chihao Zhang. Rapid mixing from spectral independence beyond the boolean domain. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1558–1577. SIAM, 2021.

[FHL08] Uriel Feige, MohammadTaghi Hajiaghayi, and James R Lee. Improved approximation algorithms for minimum weight vertex separators. *SIAM Journal on Computing*, 38(2):629–657, 2008.

[Fis05] Ernst Fischer. Über quadratische formen mit reellen koeffizienten. *Monatshefte für Mathematik und Physik*, 16(1):234–249, 1905.

[Fis04] Matthew P. A. Fisher. Duality in low dimensional quantum field theories. In D. Baeriswyl and L. Degiorgi, editors, *Strong interactions in low dimensions*, pages 419–438, Dordrecht, 2004. Springer Netherlands.

[FJLS20] Asaf Ferber, Vishesh Jain, Kyle Luh, and Wojciech Samotij. On the counting problem in inverse littlewood–offord theory. *Journal of the London Mathematical Society*, 2020.

[FKP19] Noah Fleming, Pravesh Kothari, and Toniann Pitassi. *Semialgebraic Proofs and Efficient Algorithm Design.* Foundations and Trends in Theoretical Computer Science, 2019.

[FKS20] Asaf Ferber, Matthew Kwan, and Lisa Sauermann. Singularity of sparse random matrices: simple proofs. *Combinatorics, Probability and Computing*, pages 1–8, 2020.

[FKT20] Vitaly Feldman, Tomer Koren, and Kunal Talwar. Private stochastic convex optimization: optimal rates in linear time. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 439–449, 2020.

[Fle13] Roger Fletcher. *Practical methods of optimization.* John Wiley & Sons, 2013.

[FLP16] Dimitris Fotakis, Michael Lampis, and Vangelis Th Paschos. Subexponential approximation schemes for csps: From dense to almost sparse. In *33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016).* Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[FMPS09] Herbert Fleischner, Egbert Mujuni, Daniël Paulusma, and Stefan Szeider. Covering graphs with few complete bipartite subgraphs. *Theoretical Computer Science*, 410(21-23):2045–2053, 2009.

[FYC23] Jiaojiao Fan, Bo Yuan, and Yongxin Chen. Improved dimension dependence of a proximal algorithm for sampling. *arXiv preprint arXiv:2302.10081*, 2023.

[Gam22] Jay Gambetta. Ibm quantum roadmap to build quantum-centric supercomputers, Aug 2022.

[Går51] Lars Gårding. Linear hyperbolic partial differential equations with constant coefficients. *Acta Mathematica*, 85:1–62, 1951.

[Går59] Lars Gårding. An inequality for hyperbolic polynomials. *Journal of Mathematics and Mechanics*, pages 957–965, 1959.

[Gav12] D. Gavinsky. Quantum money with classical verification. In *2012 IEEE 27th Conference on Computational Complexity*, pages 42–52, June 2012.

[GC05] Yuan Gao and George Church. Improving molecular cancer class discovery through sparse non-negative matrix factorization. *Bioinformatics*, 21(21):3970–3975, 2005.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, August 1986.

[GH16] Dan Garber and Elad Hazan. Sublinear time algorithms for approximate semidefinite programming. *Mathematical Programming*, 158(1-2):329–361, 2016.

[Gha15] Shayan Oveis Gharan. Proof of kadison-singer conjecture and the extensions, 2015.

[GIIS14] Anna C. Gilbert, Piotr Indyk, Mark A. Iwen, and Ludwig Schmidt. Recent developments in the sparse fourier transform: A compressed fourier transform for big data. *IEEE Signal Process. Mag.*, 31(5):91–100, 2014.

[Gil08] Michael B. Giles. Multilevel Monte Carlo path simulation. *Operations Research*, 56(3):607–617, 2008.

[Gil12] Nicolas Gillis. Sparse and unique nonnegative matrix factorization through data preprocessing. *The Journal of Machine Learning Research*, 13(1):3349–3386, 2012.

[Gil15] Michael B. Giles. Multilevel Monte Carlo methods. *Acta Numerica*, 24:259–328, 2015.

[GKM$^+$19] Rishab Goyal, Sam Kim, Nathan Manohar, Brent Waters, and David J Wu. Watermarking public-key cryptographic primitives. In *Annual International Cryptology Conference*, pages 367–398. Springer, 2019.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. One-time programs. In *Annual International Cryptology Conference*, pages 39–56. Springer, 2008.

[GKR13] Martin Grötschel, Sven O Krumke, and Jörg Rambau. *Online optimization of large scale systems.* Springer Science & Business Media, 2013.

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 612–621. IEEE, 2017.

[GL21] Leonid Gurvits and Jonathan Leake. Capacity lower bounds via productization. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 847–858, 2021.

[GLG22] Sevag Gharibian and François Le Gall. Dequantizing the quantum singular value transformation: Hardness and applications to quantum chemistry and the quantum pcp conjecture. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 19–32, 2022.

[GLL20] Rong Ge, Holden Lee, and Jianfeng Lu. Estimating normalizing constants for log-concave distributions: Algorithms and lower bounds. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 579–586, 2020. arXiv:1911.03043.

[GLL22] Tom Gur, Noam Lifshitz, and Siqi Liu. Hypercontractivity on high dimensional expanders. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 176–184, 2022.

[GLM08] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Physical review letters*, 100(16):160501, 2008.

[GLP21] Yu Gao, Yang P. Liu, and Richard Peng. Fully dynamic electrical flows: Sparse maxflow faster than goldberg-rao. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, 2021.

[GLSS18] Ankit Garg, Yin-Tat Lee, Zhao Song, and Nikhil Srivastava. A matrix expander chernoff bound. In *STOC*. `https://arxiv.org/pdf/1704.03864`, 2018.

[GMS86] M Gromov, V Milman, and G Schechtman. Asymptotic theory of finite dimensional normed spaces, volume 1200 of lectures notes in mathematics, 1986.

[GMS05] Anna C Gilbert, Shan Muthukrishnan, and Martin Strauss. Improved time bounds for near-optimal sparse fourier representations. In *Wavelets XI*, volume 5914, page 59141A. International Society for Optics and Photonics, 2005.

[Gol65] Sidney Golden. Lower bounds for the helmholtz function. *Physical Review*, 137(4B):B1127, 1965.

[GPPR04] Cyril Gavoille, David Peleg, Stéphane Pérennes, and Ran Raz. Distance labeling in graphs. *Journal of Algorithms*, 53(1):85–112, 2004.

[GPY20] Sevag Gharibian, Stephen Piddock, and Justin Yirka. Oracle Complexity Classes and Local Measurements on Physical Hamiltonians. In Christophe Paul and Markus Bläser, editors, *37th International Symposium on Theoretical Aspects of Computer Science (STACS 2020)*, volume 154 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:37, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[GR02] Lov Grover and Terry Rudolph. Creating superpositions that correspond to efficiently integrable probability distributions, 2002. arXiv:quant-ph/0208112.

[GRB+20] Jérôme F Gonthier, Maxwell D Radin, Corneliu Buda, Eric J Doskocil, Clena M Abuan, and Jhonathan Romero. Identifying challenges towards practical quantum advantage through resource estimation: the measurement roadblock in the variational quantum eigensolver. *arXiv preprint arXiv:2012.04001*, 2020.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 212–219, New York, NY, USA, 1996. Association for Computing Machinery.

[GS18] Sevag Gharibian and Jamie Sikora. Ground state connectivity of local hamiltonians. *ACM Trans. Comput. Theory*, 10(2), apr 2018.

[GS20] François Le Gall and Saeed Seddighin. Quantum meets fine-grained complexity: Sublinear time quantum algorithms for string problems. *arXiv preprint arXiv:2010.12122*, 2020.

[GSLW19] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 193–204, 2019.

[GST22] András Gilyén, Zhao Song, and Ewin Tang. An improved quantum-inspired algorithm for linear regression. *Quantum*, 6:754, 2022.

[GTC19] Yimin Ge, Jordi Tura, and J Ignacio Cirac. Faster ground state preparation and high-precision ground energy estimation with fewer qubits. *Journal of Mathematical Physics*, 60(2):022202, 2019.

[GU18] François Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the coppersmith-winograd tensor. In

Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2018.

[Gül97] Osman Güler. Hyperbolic polynomials and interior point methods for convex programming. *Mathematics of Operations Research*, 22(2):350–377, 1997.

[Gur04] Leonid Gurvits. Combinatorics hidden in hyperbolic polynomials and related topics. *arXiv preprint math/0402088*, 2004.

[Gur06] Leonid Gurvits. Hyperbolic polynomials approach to van der waerden/schrijver-valiant like conjectures: sharper bounds, simpler proofs and algorithmic applications. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 417–426, 2006.

[Gur07] Leonid Gurvits. Van der waerden/schrijver-valiant like conjectures and stable (aka hyperbolic) homogeneous polynomials: one theorem for all. *arXiv preprint arXiv:0711.3496*, 2007.

[GW94] Michel X Goemans and David P Williamson. .879-approximation algorithms for max cut and max 2sat. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing (STOC)*, pages 422–431, 1994.

[GWL$^+$22] Joshua J Goings, Alec White, Joonho Lee, Christofer S Tautermann, Matthias Degroote, Craig Gidney, Toru Shiozaki, Ryan Babbush, and Nicholas C Rubin. Reliably assessing the electronic structure of cytochrome p450 on today's classical computers and tomorrow's quantum computers. *arXiv preprint arXiv:2202.01244*, 2022.

[GY19] Sevag Gharibian and Justin Yirka. The complexity of simulating local measurements on quantum systems. *Quantum*, 3:189, 2019.

[GZ20] Marios Georgiou and Mark Zhandry. Unclonable decryption keys. Cryptology ePrint Archive, Report 2020/877, 2020. `https://eprint.iacr.org/2020/877`.

[Hal07] Thomas C Hales. The jordan curve theorem, formally and informally. *The American Mathematical Monthly*, 114(10):882–894, 2007.

[Ham21] Yassine Hamoudi. Quantum Sub-Gaussian Mean Estimator. In *29th Annual European Symposium on Algorithms*, volume 204 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 50:1–50:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. arXiv:2108.12172.

[HAR70] RA HARSHMAN. Foundations of the parafac procedure: Models and conditions for an" explanatory" multi-mode factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84, 1970.

[Has19] Matthew B Hastings. Classical and quantum bounded depth approximation algorithms. *arXiv preprint arXiv:1905.07047*, 2019.

[HBE22] Ye He, Krishnakumar Balasubramanian, and Murat A Erdogdu. Heavy-tailed sampling via transformed unadjusted langevin algorithm. *arXiv preprint arXiv:2201.08349*, 2022.

[HBR21] Hsin-Yuan Huang, Kishor Bharti, and Patrick Rebentrost. Near-term quantum algorithms for linear systems of equations with regression loss functions. *New Journal of Physics*, 23(11):113021, nov 2021.

[Hei01] Stefan Heinrich. Multilevel Monte Carlo methods. In *International Conference on Large-Scale Scientific Computing*, pages 58–67. Springer, 2001.

[Her15] Timon Hertli. *Improved Exponential Algorithms for SAT and ClSP*. PhD thesis, ETH Zurich, 2015.

[HH06] Eran Halperin and Elad Hazan. Haplofreq—estimating haplotype frequencies efficiently. *Journal of Computational Biology*, 13(2):481–500, 2006.

[HHL09] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.

[HIKP12a] Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. Nearly optimal sparse fourier transform. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 563–578, 2012.

[HIKP12b] Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. Simple and practical algorithm for sparse fourier transform. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms (SODA)*, pages 1183–1194. SIAM, 2012.

[Hil14] Roland Hildebrand. Canonical barriers on convex cones. *Mathematics of operations research*, 39(3):841–850, 2014.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[Hir18] Shuichi Hirahara. Non-black-box worst-case to average-case reductions within np. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 247–258. IEEE, 2018.

[HJO14] Trygve Helgaker, Poul Jorgensen, and Jeppe Olsen. *Molecular electronic-structure theory*. John Wiley & Sons, 2014.

[HJS$^+$22] Baihe Huang, Shunhua Jiang, Zhao Song, Runzhou Tao, and Ruizhe Zhang. Solving sdp faster: A robust ipm framework and efficient implementation. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 233–244. IEEE, 2022.

[HKNS15] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 21–30, 2015.

[HKT$^+$22] Hsin-Yuan Huang, Richard Kueng, Giacomo Torlai, Victor V Albert, and John Preskill. Provably efficient machine learning for quantum many-body problems. *Science*, 377(6613):eabk3333, 2022.

[HL16] Elad Hazan and Yuanzhi Li. An optimal algorithm for bandit convex optimization, 2016. arXiv:1603.04350.

[HL22] Max Hopkins and Ting-Chun Lin. Explicit lower bounds against $\omega(n)$-rounds of sum-of-squares. *arXiv preprint arXiv:2204.11469*, 2022.

[HLJ09] F Reese Harvey and H Blaine Lawson Jr. Hyperbolic polynomials and the dirichlet problem. *arXiv preprint arXiv:0912.5220*, 2009.

[HLSY21] Baihe Huang, Xiaoxiao Li, Zhao Song, and Xin Yang. Fl-ntk: A neural tangent kernel-based framework for federated learning convergence analysis. In *ICML*, 2021.

[HM19] Yassine Hamoudi and Frédéric Magniez. Quantum Chebyshev's inequality and applications. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming*, volume 132 of *Leibniz International Proceedings in Informatics*, pages 69:1–69:16, 2019. arXiv:1807.06456.

[HO14] Nicholas JA Harvey and Neil Olver. Pipage rounding, pessimistic estimators and matrix concentration. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 926–945. SIAM, 2014.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[Hor83] L Hormander. The analysis of linear partial differential operators ii. *Grundlehren*, 257, 1983.

[HOS18] Shuichi Hirahara, Igor C Oliveira, and Rahul Santhanam. Np-hardness of minimum circuit size problem for or-and-mod circuits. In *33rd Computational Complexity Conference (CCC 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[Hoy04] Patrik O Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(Nov):1457–1469, 2004.

[HR17] Ishay Haviv and Oded Regev. The restricted isometry property of subsampled fourier matrices. In *Geometric aspects of functional analysis*, pages 163–179. Springer, 2017.

[HRS21] Samuel B Hopkins, Prasad Raghavendra, and Abhishek Shetty. Matrix discrepancy from quantum communication. *arXiv preprint arXiv:2110.10099*, 2021.

[HRW98] Roger A Horn, Noah H Rhee, and So Wasin. Eigenvalue inequalities and equalities. *Linear Algebra and its Applications*, 270(1-3):29–44, 1998.

[HS07] Lisa Hellerstein and Rocco A Servedio. On PAC learning algorithms for rich Boolean function classes. *Theoretical Computer Science*, 384(1):66–76, 2007.

[HSC$^+$20] Yangsibo Huang, Zhao Song, Danqi Chen, Kai Li, and Sanjeev Arora. Texthide: Tackling data privacy in language understanding tasks. In *The Conference on Empirical Methods in Natural Language Processing (Findings of EMNLP)*, 2020.

[HSLA20a] Yangsibo Huang, Zhao Song, Kai Li, and Sanjeev Arora. Instahide challenge. `https://github.com/Hazelsuko07/InstaHide_Challenge`, 2020.

[HSLA20b] Yangsibo Huang, Zhao Song, Kai Li, and Sanjeev Arora. Instahide: Instance-hiding schemes for private distributed learning. In *International Conference on Machine Learning (ICML)*, pages 4507–4518, 2020.

[HST$^+$20] Baihe Huang, Zhao Song, Runzhou Tao, Ruizhe Zhang, and Danyang Zhuo. Instahide's sample complexity when mixing two private images. *arXiv preprint arXiv:2011.11877*, 2020.

[Hua18] Jiaoyang Huang. Invertibility of adjacency matrices for random d-regular graphs. *arXiv preprint arXiv:1807.06465*, 2018.

[HV07] J William Helton and Victor Vinnikov. Linear matrix inequality representation of sets. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 60(5):654–674, 2007.

[HW20] Aram W. Harrow and Annie Y. Wei. Adaptive quantum simulated annealing for Bayesian inference and estimating partition functions.

In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 193–212, 2020.

[HWO+19] Stuart Hadfield, Zhihui Wang, Bryan O'Gorman, Eleanor G Rieffel, Davide Venturelli, and Rupak Biswas. From the quantum approximate optimization algorithm to a quantum alternating operator ansatz. *Algorithms*, 12(2):34, 2019.

[HXZ+11] Zhaoshui He, Shengli Xie, Rafal Zdunek, Guoxu Zhou, and Andrzej Cichocki. Symmetric nonnegative matrix factorization: Algorithms and applications to probabilistic clustering. *IEEE Transactions on Neural Networks*, 22(12):2117–2131, 2011.

[HYF21] Robin Harper, Wenjun Yu, and Steven T Flammia. Fast estimation of sparse quantum noise. *PRX Quantum*, 2(1):010322, 2021.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 770–778, 2016.

[IK14] Piotr Indyk and Michael Kapralov. Sample-optimal Fourier sampling in any constant dimension. In *IEEE 55th Annual Symposium onFoundations of Computer Science (FOCS)*, pages 514–523. IEEE, 2014.

[IKP14] Piotr Indyk, Michael Kapralov, and Eric Price. (nearly) sample-optimal sparse fourier transform. In *Proceedings of the twenty-fifth* *annual ACM-SIAM symposium on Discrete algorithms*, pages 480–499. SIAM, 2014.

[IKV18] Russell Impagliazzo, Valentine Kabanets, and Ilya Volkovich. The power of natural properties as oracles. In *33rd Computational Complexity Conference (CCC 2018).* Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[Ila19] R. Ilango. AC0[p] lower bounds and np-hardness for variants of mcsp. *Electron. Colloquium Comput. Complex.*, 26:21, 2019.

[Ila20a] Rahul Ilango. Connecting Perebor Conjectures: Towards a Search to Decision Reduction for Minimizing Formulas. In Shubhangi Saraf, editor, *35th Computational Complexity Conference (CCC 2020)*, volume 169 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 31:1–31:35. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020.

[Ila20b] Rahul Ilango. Constant depth formula and partial function versions of mcsp are hard. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 424–433. IEEE, 2020.

[ILO20] Rahul Ilango, Bruno Loff, and Igor C. Oliveira. Np-hardness of circuit minimization for multi-output functions. In *35th Computational Complexity Conference (CCC 2020)*, CCC '20, Dagstuhl, DEU, 2020. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[IM98] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing (STOC)*, pages 604–613, 1998.

[Ind06] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM (JACM)*, 53(3):307–323, 2006.

[INT05] Toshiya Itoh, Tatsuya Nagatani, and Jun Tarui. Explicit construction of k-wise nearly random permutations by iterated Feistel transform. In *Workshop on Randomness and Computation*, 2005.

[IP01] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

[IRS21] Rahul Ilango, Hanlin Ren, and Rahul Santhanam. Hardness on any samplable distribution suffices: New characterizations of one-way functions by meta-complexity. *Electron. Colloquium Comput. Complex.*, 28:82, 2021.

[IRW17] Piotr Indyk, Ilya Razenshteyn, and Tal Wagner. Practical data-dependent metric compression with provable guarantees. *Advances in Neural Information Processing Systems*, 30, 2017.

[ITU92] ITU. Information technology - digital compression and coding of continuous - tone still images - requirements and guidelines. *CCITT, Recommendation*, 1992.

[IW97] Russell Impagliazzo and Avi Wigderson. P= BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 220–229, 1997.

[IW05] Piotr Indyk and David Woodruff. Optimal approximations of the frequency moments of data streams. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 202–208, 2005.

[Jai21] Vishesh Jain. Approximate spielman-teng theorems for the least singular value of random combinatorial matrices. *Israel Journal of Mathematics*, 242(1):461–500, 2021.

[Jef14] Stacey Jeffery. *Frameworks for Quantum Algorithms.* PhD thesis, University of Waterloo, 2014.

[Jen17] Frank Jensen. *Introduction to computational chemistry.* John Wiley & Sons, 2017.

[Jeř09] Emil Jeřábek. Approximate counting by hashing in bounded arithmetic. *Journal of Symbolic Logic*, 74(3):829–860, 2009.

[JGH18] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

[JGL10] Stephen P Jordan, David Gosset, and Peter J Love. Quantum-merlin-arthur–complete problems for stoquastic hamiltonians and markov matrices. *Physical Review A*, 81(3):032331, 2010.

[JJUW11] Rahul Jain, Zhengfeng Ji, Sarvagya Upadhyay, and John Watrous. QIP = PSPACE. *Journal of the ACM (JACM)*, 2011.

[JKDG08] Prateek Jain, Brian Kulis, Inderjit S Dhillon, and Kristen Grauman. Online metric learning and fast similarity search. In *NIPS*, volume 8, pages 761–768. Citeseer, 2008.

[JKG$^+$22] Peter D Johnson, Alexander A Kunitsa, Jérôme F Gonthier, Maxwell D Radin, Corneliu Buda, Eric J Doskocil, Clena M Abuan, and Jhonathan Romero. Reducing the cost of energy estimation in the variational quantum eigensolver algorithm with robust amplitude estimation. *arXiv preprint arXiv:2203.07275*, 2022.

[JKL97] Michael S Jacobson, André E Kézdy, and Jenő Lehel. Recognizing intersection graphs of linear uniform hypergraphs. *Graphs and Combinatorics*, 13(4):359–367, 1997.

[JKL$^+$20] Haotian Jiang, Tarun Kathuria, Yin Tat Lee, Swati Padmanabhan, and Zhao Song. A faster interior point method for semidefinite programming. In *61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2020.

[JL84] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

[JLL$^+$20] Arun Jambulapati, Yin Tat Lee, Jerry Li, Swati Padmanabhan, and Kevin Tian. Positive semidefinite programming: mixed, parallel, and width-independent. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*. ACM, 2020.

[JLLV20] He Jia, Aditi Laddha, Yin Tat Lee, and Santosh S Vempala. Reducing isotropy and volume to KLS: An $O^*(n^3\psi^2)$ volume algorithm. *arXiv preprint arXiv:2008.02146*, 2020.

[JLS18] Zhengfeng Ji, Yi-Kai Liu, and Fang Song. Pseudorandom quantum states. In *Annual International Cryptology Conference*, pages 126–152. Springer, 2018.

[JLS23] Yaonan Jin, Daogao Liu, and Zhao Song. Super-resolution and robust sparse continuous fourier transform in any constant dimension: Nearly linear time and sample complexity. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2023.

[JLSW20] Haotian Jiang, Yin Tat Lee, Zhao Song, and Sam Chiu-wai Wong. An improved cutting plane method for convex optimization, convex-concave games and its applications. In *STOC*, 2020.

[JLT20] Arun Jambulapati, Jerry Li, and Kevin Tian. Robust sub-gaussian principal component analysis and width-independent schatten packing. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020.

[JNV$^+$20] Zhengfeng Ji, Anand Natarajan, Thomas Vidick, John Wright, and Henry Yuen. MIP*=RE. *arXiv preprint arXiv:2001.04383*, 2020.

[Jor05] Stephen P. Jordan. Fast quantum algorithm for numerical gradient estimation. *Physical Review Letters*, 95(5):050501, 2005.

[JPR$^+$20] Siddhant Jayakumar, Razvan Pascanu, Jack Rae, Simon Osindero, and Erich Elsen. Top-kast: Top-k always sparse training. *Advances in Neural Information Processing Systems*, 33:20744–20754, 2020.

[JPV21] Vishesh Jain, Huy Tuan Pham, and Thuy Duong Vuong. Spectral independence, coupling with the stationary distribution, and the spectral gap of the glauber dynamics. *arXiv preprint arXiv:2105.01201*, 2021.

[JQST20] Fernando Granha Jeronimo, Dylan Quintana, Shashank Srivastava, and Madhur Tulsiani. Unique decoding of explicit epsilon-balanced codes near the gilbert-varshamov bound. *arXiv preprint arXiv:2011.05500*, 2020.

[JR23] Samuel Jaques and Arthur G. Rattew. Qram: A survey and critique, 2023.

[JŠ19] Tomas Juškevičius and Grazvydas Šemetulskis. Optimal littlewood-offord inequalities in groups. *Combinatorica*, 39(4):911–921, 2019.

[JST21] Fernando Granha Jeronimo, Shashank Srivastava, and Madhur Tulsiani. Near-linear time decoding of ta-shma's codes via splittable regularity. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1527–1536, 2021.

[JSWZ21] Shunhua Jiang, Zhao Song, Omri Weinstein, and Hengjie Zhang. Faster dynamic matrix inverse for faster lps. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*. arXiv preprint arXiv:2004.07470, 2021.

[JT19] Ziwei Ji and Matus Telgarsky. Polylogarithmic width suffices for gradient descent to achieve arbitrarily small test error with shallow relu networks. In *International Conference on Learning Representations*, 2019.

[JT20] Ziwei Ji and Matus Telgarsky. Polylogarithmic width suffices for gradient descent to achieve arbitrarily small test error with shallow relu networks. In *ICLR*, 2020.

[JY11] Rahul Jain and Penghui Yao. A parallel approximation algorithm for positive semidefinite programming. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, 2011.

[JY12] Rahul Jain and Penghui Yao. A parallel approximation algorithm for mixed packing and covering semidefinite programs. *CoRR*, abs/1201.6090, 2012.

[Kah64] Jean-Pierre Kahane. Sur les sommes vectorielles sigma plus minus un. *COMPTES RENDUS HEBDOMADAIRES DES SEANCES DE L ACADEMIE DES SCIENCES*, 259(16):2577, 1964.

[Kan18] Daniel M Kane. Quantum money from modular forms. *arXiv preprint arXiv:1809.05925*, 2018.

[Kap16] Michael Kapralov. Sparse fourier transform in any constant dimension with nearly-optimal sample complexity in sublinear time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 264–277, 2016.

[Kap17] Michael Kapralov. Sample efficient estimation and recovery in sparse FFT via isolation on average. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 651–662. IEEE Computer Society, 2017.

[Kar72] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[Kar84] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing (STOC)*, 1984.

[KB15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[KC00] Valentine Kabanets and Jin-Yi Cai. Circuit minimization problem. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 73–79, 2000.

[KDP12] Da Kuang, Chris Ding, and Haesun Park. Symmetric nonnegative matrix factorization for graph clustering. In *Proceedings of the SIAM International Conference on Data Mining (ICDM)*, pages 106–117. SIAM, 2012.

[KG12] Vassilis Kalofolias and Efstratios Gallopoulos. Computing symmetric nonnegative rank factorizations. *Linear algebra and its applications*, 436(2):421–435, 2012.

[Kha80] Leonid G Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.

[Khi23] A Khiintchine. Über dyadische bruche. *Math. Z*, 18:109, 1923.

[Kho02] Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 767–775, 2002.

[Kit95] Alexei Yu Kitaev. Quantum measurements and the Abelian stabilizer problem, 1995. arXiv:quant-ph/9511026.

[KKJ22] Rutuja Kshirsagar, Amara Katabarwa, and Peter D Johnson. On proving the robustness of algorithms for early fault-tolerant quantum computers. *arXiv preprint arXiv:2209.11322*, 2022.

[KKL20] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[KKO20] Anna R Karlin, Nathan Klein, and Shayan Oveis Gharan. An improved approximation algorithm for tsp in the half integral case. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 28–39, 2020.

[KKO21] Anna R Karlin, Nathan Klein, and Shayan Oveis Gharan. A (slightly) improved approximation algorithm for metric tsp. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 32–45, 2021.

[KKOZ21] Anna R Karlin, Nathan Klein, Shayan Oveis Gharan, and Xinzhi Zhang. An improved approximation algorithm for the minimum k-edge connected multi-subgraph problem. *arXiv preprint arXiv:2101.05921*, 2021.

[KKPJ21] Amara Katabarwa, Alex Kunitsa, Borja Peropadre, and Peter Johnson. Reducing runtime and error in vqe using deeper and noisier quantum circuits. *arXiv preprint arXiv:2110.10664*, 2021.

[Kle80] Victor Klee. On the complexity of d-dimensional Voronoi diagrams. *Archiv der Mathematik*, 34(1):75–80, 1980.

[KLM06] Ravi Kannan, László Lovász, and Ravi Montenegro. Blocking conductance and mixing in random walks. *Comb. Probab. Comput.*, 15(4):541–570, 2006.

[KLP19] Iordanis Kerenidis, Jonas Landman, and Anupam Prakash. Quantum algorithms for deep convolutional neural networks. *arXiv preprint arXiv:1911.01117*, 2019.

[KLP$^+$22] Isaac H Kim, Ye-Hua Liu, Sam Pallister, William Pol, Sam Roberts, and Eunseok Lee. Fault-tolerant resource estimate for quantum chemical simulations: Case study on li-ion battery electrolyte molecules. *Physical Review Research*, 4(2):023019, 2022.

[KLS20] Rasmus Kyng, Kyle Luh, and Zhao Song. Four deviations suffice for rank 1 matrices. In *Advances in Mathematics*. arXiv preprint arXiv:1901.06731, 2020.

[KM95] S Khuller and Y Matias. A simple randomized sieve algorithm for the closest-pair problem. *Information and Computation*, 118(1):34–37, 1995.

[KM01] Phillip Kaye and Michele Mosca. Quantum networks for generating arbitrary quantum states. In *International Conference on Quantum Information*, page PB28. Optical Society of America, 2001. arXiv:quant-ph/0407102.

[KM03] Kartik Krishnan and John E Mitchell. Properties of a cutting plane method for semidefinite programming. *submitted for publication*, 2003.

[KM16] Tali Kaufman and David Mass. High dimensional combinatorial random walks and colorful expansion. *arXiv preprint arXiv:1604.02947*, 2016.

[KM20a] CS Karthik and Pasin Manurangsi. On closest pair in euclidean metric: Monochromatic is as hard as bichromatic. *Combinatorica*, pages 1–35, 2020.

[KM20b] Tali Kaufman and David Mass. Local-to-global agreement expansion via the variance method. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[KMMS18] Subhash Khot, Dor Minzer, Dana Moshkovitz, and Muli Safra. Small set expansion in the johnson graph. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 25, page 78, 2018.

[KMR14] Felix Krahmer, Shahar Mendelson, and Holger Rauhut. Suprema of chaos processes and the restricted isometry property. *Communications on Pure and Applied Mathematics*, 67(11):1877–1904, 2014.

[KMS94] David Karger, Rajeev Motwani, and Madhu Sudan. Approximate graph coloring by semidefinite programming. In *Proceedings 35th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 1994.

[KMS17] Subhash Khot, Dor Minzer, and Muli Safra. On independent sets, 2-to-2 games, and grassmann graphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 576–589, 2017.

[KMY$^+$16] Jakub Konečný, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[KNPW11] Daniel M Kane, Jelani Nelson, Ely Porat, and David P Woodruff. Fast moment estimation in data streams in optimal space. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 745–754, 2011.

[KNW10] Daniel M Kane, Jelani Nelson, and David P Woodruff. On the exact space complexity of sketching and streaming small norms. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1161–1178. SIAM, 2010.

[KNY20] Fuyuki Kitagawa, Ryo Nishimaki, and Takashi Yamakawa. Secure software leasing from standard assumptions, 2020.

[KO20] Tali Kaufman and Izhar Oppenheim. High order random walks: Beyond spectral gap. *Combinatorica*, pages 1–37, 2020.

[KOS07] Emanuel Knill, Gerardo Ortiz, and Rolando D Somma. Optimal quantum measurements of expectation values of observables. *Physical Review A*, 75(1):012328, 2007.

[Kós08] Géza Kós. Two turán type inequalities. *Acta Mathematica Hungarica*, 119(3):219–226, 2008.

[KP08] Jingu Kim and Haesun Park. Sparse nonnegative matrix factorization for clustering. Technical report, Georgia Institute of Technology, 2008.

[KP17] Iordanis Kerenidis and Anupam Prakash. Quantum recommendation systems. In *8th Innovations in Theoretical Computer Science Conference (ITCS)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[KP20a] Iordanis Kerenidis and Anupam Prakash. Quantum gradient descent for linear systems and least squares. *Physical Review A*, 101(2):022316, 2020.

[KP20b] Iordanis Kerenidis and Anupam Prakash. A quantum interior point method for lps and sdps. *ACM Transactions on Quantum Computing*, 1(1):1–32, 2020.

[KPRW19] Ravi Kumar, Rina Panigrahy, Ali Rahimi, and David Woodruff. Faster algorithms for binary matrix factorization. In *International Conference on Machine Learning (ICML)*, pages 3551–3559, 2019.

[KPS21] Iordanis Kerenidis, Anupam Prakash, and Dániel Szilágyi. Quantum algorithms for second-order cone programming and support vector machines. *Quantum*, 5:427, 2021.

[KPV15] Mario Kummer, Daniel Plaumann, and Cynthia Vinzant. Hyperbolic polynomials, interlacers, and sums of squares. *Mathematical Programming*, 153(1):223–245, 2015.

[KR16] Apoorva Khare and Bala Rajaratnam. The khinchin–kahane inequality and banach space embeddings for abelian metric groups. *arXiv preprint math.PR/1610.03037*, 2016.

[Kre21] William Kretschmer. Quantum Pseudorandomness and Classical Complexity. In *16th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2021)*, volume 197 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:20, 2021.

[Kry95] N.V. Krylov. On the general notion of fully nonlinear second-order elliptic equations. *Transactions of the American Mathematical Society*, 347(3):857–895, 1995.

[KS59] Richard V Kadison and Isadore M Singer. Extensions of pure states. *American journal of mathematics*, 81(2):383–400, 1959.

[KS18] Rasmus Kyng and Zhao Song. A matrix chernoff bound for strongly rayleigh distributions and spectral sparsifiers from a few random spanning trees. In *FOCS*. `https://arxiv.org/pdf/1810.08345`, 2018.

[KS19] KR Khadiev and LI Safina. Quantum algorithm for shortest path search in directed acyclic graph. *Moscow University Computational Mathematics and Cybernetics*, 43(1):47–51, 2019.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[KSV02] A. Yu. Kitaev, A. H. Shen, and M. N. Vyalyi. *Classical and Quantum Computation*. American Mathematical Society, USA, 2002.

[KT06] Jon Kleinberg and Eva Tardos. *Algorithm Design*. Pearson Education India, 2006.

[KT19] Richard Kueng and Joel A Tropp. Binary component decomposition part ii: The asymmetric case. *arXiv preprint arXiv:1907.13602*, 2019.

[KT21a] Tali Kaufman and Ran J Tessler. Local to global high dimensional expansion and garland's method for general posets. *arXiv preprint arXiv:2101.12621*, 2021.

[KT21b] Richard Kueng and Joel A Tropp. Binary component decomposition part i: the positive-semidefinite case. *SIAM Journal on Mathematics of Data Science*, 3(2):544–572, 2021.

[KTE88] Leonid G Khachiyan, Sergei Pavlovich Tarasov, and I. I. Erlikh. The method of inscribed ellipsoids. *Soviet Math. Dokl*, 37(1):226–230, 1988.

[KV05] Subhash Khot and Nisheeth K Vishnoi. On the unique games conjecture. In *FOCS*, volume 5, page 3. Citeseer, 2005.

[KW17] Sam Kim and David J Wu. Watermarking cryptographic functionalities from standard lattice assumptions. In *Annual International Cryptology Conference*, pages 503–536. Springer, 2017.

[KW19] Sam Kim and David J Wu. Watermarking prfs from lattices: Stronger security via extractable prfs. In *Annual International Cryptology Conference*, pages 335–366. Springer, 2019.

[LAF$^+$09] Andrew Lutomirski, Scott Aaronson, Edward Farhi, David Gosset, Avinatan Hassidim, Jonathan Kelner, and Peter Shor. Breaking and making quantum money: toward a new quantum cryptographic protocol. *arXiv preprint arXiv:0912.3825*, 2009.

[Lat06] Rafał Latała. Estimates of moments and tails of gaussian chaoses. *The Annals of Probability*, 34(6):2315–2331, 2006.

[Lat20] Tor Lattimore. Improved regret for zeroth-order adversarial bandit convex optimisation. *Mathematical Statistics and Learning*, 2(3):311–334, 2020. arXiv:2006.00475.

[Lax57] Peter D Lax. Differential equations, difference equations and matrix theory. Technical report, New York Univ., New York. Atomic Energy Commission Computing and Applied, 1957.

[Lay22] David Layden. First-order trotter error from a second-order perspective. *Phys. Rev. Lett.*, 128:210501, May 2022.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LC17] Guang Hao Low and Isaac L Chuang. Optimal hamiltonian simulation by quantum signal processing. *Physical review letters*, 118(1):010501, 2017.

[LC19] Guang Hao Low and Isaac L Chuang. Hamiltonian simulation by qubitization. *Quantum*, 3:163, 2019.

[LCB10] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. att labs, 2010.

[LCC$^+$21] Shiwei Liu, Tianlong Chen, Xiaohan Chen, Zahra Atashgahi, Lu Yin, Huanyu Kou, Li Shen, Mykola Pechenizkiy, Zhangyang Wang, and Decebal Constantin Mocanu. Sparse training via boosting pruning plasticity with neuroregeneration. *Advances in Neural Information Processing Systems*, 34:9908–9922, 2021.

[LDFU13] Yichao Lu, Paramveer Dhillon, Dean P Foster, and Lyle Ungar. Faster ridge regression via the subsampled randomized hadamard transform.

In *Advances in neural information processing systems (NIPS)*, pages 369–377, 2013.

[LE20] Mufan Bill Li and Murat A Erdogdu. Riemannian langevin algorithm for solving semidefinite programs. *arXiv preprint arXiv:2010.11176*, 2020.

[Leh74] Philippe GH Lehot. An optimal algorithm to detect a line graph and output its root graph. *Journal of the ACM (JACM)*, 21(4):569–575, 1974.

[Leh11] Joseph Lehec. Moments of the gaussian chaos. In *Séminaire de Probabilités XLIII*, pages 327–340. Springer, 2011.

[LG14] François Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th international symposium on symbolic and algebraic computation (ISSAC)*, pages 296–303. ACM, 2014.

[LG21] Tor Lattimore and Andras Gyorgy. Improved regret for zeroth-order stochastic convex bandits. In *Conference on Learning Theory*, pages 2938–2964. PMLR, 2021.

[LH22] Ting-Chun Lin and Min-Hsiu Hsieh. c3-local testable codes from lossless expanders. *arXiv preprint arXiv:2201.11369*, 2022.

[LHP$^+$20] Jessica Lemieux, Bettina Heim, David Poulin, Krysta Svore, and Matthias Troyer. Efficient quantum walk circuits for metropolis-hastings algorithm. *Quantum*, 4:287, 2020.

[LHW17] Songtao Lu, Mingyi Hong, and Zhengdao Wang. A nonconvex splitting method for symmetric nonnegative matrix factorization: Convergence analysis and optimality. *IEEE Transactions on Signal Processing*, 65(12):3120–3135, 2017.

[Liu21] Kuikui Liu. From coupling to spectral independence and blackbox comparison with the down-up walk. *arXiv preprint arXiv:2103.11609*, 2021.

[LK22] Fangshuo Liao and Anastasios Kyrillidis. On the convergence of shallow neural network training with randomly masked neurons. *Transactions on Machine Learning Research*, 2022.

[LKAS$^+$21] Seth Lloyd, Bobak T Kiani, David RM Arvidsson-Shukur, Samuel Bosch, Giacomo De Palma, William M Kaminsky, Zi-Wen Liu, and Milad Marvian. Hamiltonian singular value transformation and inverse block encoding. *arXiv preprint arXiv:2104.01410*, 2021.

[LKK$^+$21] Jin-Peng Liu, Herman Øie Kolden, Hari K Krovi, Nuno F Loureiro, Konstantina Trivisa, and Andrew M Childs. Efficient quantum algorithm for dissipative nonlinear differential equations. *Proceedings of the National Academy of Sciences*, 118(35):e2026805118, 2021.

[LL18] Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *NeurIPS*, 2018.

[LLM21] Jerry Li, Allen Liu, and Ankur Moitra. Sparsification for sums of exponentials and its algorithmic applications. *arXiv preprint arXiv:2106.02774*, 2021.

[LLR95] Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

[LLV20] Aditi Laddha, Yin Tat Lee, and Santosh Vempala. Strong self-concordance and sampling. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1212–1222, 2020.

[LM15] Shachar Lovett and Raghu Meka. Constructive discrepancy minimization by walking on the edges. *SIAM Journal on Computing*, 44(5):1573–1582, 2015.

[LMM+21] Shiwei Liu, Decebal Constantin Mocanu, Amarsagar Reddy Ramapuram Matavalam, Yulong Pei, and Mykola Pechenizkiy. Sparse evolutionary deep learning with over one million artificial neurons on commodity hardware. *Neural Computing and Applications*, 33(7):2589–2604, 2021.

[LMR+11] Troy Lee, Rajat Mittal, Ben W. Reichardt, Robert Špalek, and Mario Szegedy. Quantum query complexity of state conversion. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 344–353. IEEE, 2011.

[LMR14] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. *Nature Physics*, 10(9):631–633, 2014.

[LMS11] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Slightly superexponential parameterized problems. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 760–776. SIAM, 2011.

[LNE+21] Sofiane Lounici, Mohamed Njeh, Orhan Ermis, Melek Önen, and Slim Trabelsi. Yes we can: Watermarking machine learning models beyond classification. In *2021 IEEE 34th Computer Security Foundations Symposium (CSF)*, pages 1–14. IEEE, 2021.

[LNNT16] Kasper Green Larsen, Jelani Nelson, Huy L Nguyên, and Mikkel Thorup. Heavy hitters via cluster-preserving clustering. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 61–70. IEEE, 2016.

[LNRW19] Jerry Li, Aleksandar Nikolov, Ilya Razenshteyn, and Erik Waingarten. On mean estimation for general norms with statistical queries. In *Conference on Learning Theory (COLT)*, pages 2158–2172. PMLR, 2019.

[LO43] John Edensor Littlewood and Albert Cyril Offord. On the number of real roots of a random algebraic equation (iii). *Rec. Math. [Mat. Sbornik] N.S.*, 12(3):277–286, 1943.

[LO94] Rafał Latała and Krzysztof Oleszkiewicz. On the best constant in the khinchin-kahane inequality. *Studia Mathematica*, 109(1):101–104, 1994.

[Lov77] L Lovász. Problem, beitrag zur graphentheorie und deren auwendungen, vorgstragen auf dem intern. koll, 1977.

[Lov99] László Lovász. Hit-and-run mixes fast. *Mathematical Programming*, 86(3):443–461, 1999.

[LP20a] Yin Tat Lee and Swati Padmanabhan. An $\widetilde{O}(m/\epsilon^{3.5})$-cost algorithm for semidefinite programs with diagonal constraints. In *Conference on Learning Theory (COLT)*, Proceedings of Machine Learning Research. PMLR, 2020.

[LP20b] Yanyi Liu and R. Pass. On one-way functions and kolmogorov complexity. *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1243–1254, 2020.

[LP21] Yanyi Liu and Rafael Pass. A note on one-way functions and sparse languages. *Electron. Colloquium Comput. Complex.*, 28:92, 2021.

[LPR05] Adrian Lewis, Pablo Parrilo, and Motakuri Ramana. The lax conjecture is true. *Proceedings of the American Mathematical Society*, 133(9):2495–2499, 2005.

[LRA93] Sue E Leurgans, Robert T Ross, and Rebecca B Abel. A decomposition for three-way arrays. *SIAM Journal on Matrix Analysis and Applications*, 14(4):1064–1083, 1993.

[LRR17] Avi Levy, Harishchandra Ramadas, and Thomas Rothvoss. Deterministic discrepancy minimization via the multiplicative weight update method. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 380–391. Springer, 2017.

[LS99] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[LS01] W.V. Li and Q.-M. Shao. Gaussian processes: Inequalities, small ball probabilities and applications. In *Stochastic Processes: Theory and Methods*, volume 19 of *Handbook of Statistics*, pages 533–597. Elsevier, 2001.

[LS14] Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in õ(sqrt(rank)) iterations and faster algorithms for maximum flow. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 424–433, 2014.

[LS15] Yin Tat Lee and He Sun. Constructing linear-sized spectral sparsification in almost-linear time. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 250–269, 2015.

[LS17] Yin Tat Lee and He Sun. An sdp-based algorithm for linear-sized spectral sparsification. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 678–687, 2017.

[LS19] Yin Tat Lee and Aaron Sidford. Solving linear programs with sqrt (rank) linear system solves. *arXiv preprint arXiv:1910.08033*, 2019.

[LSS$^+$20] Jason D Lee, Ruoqi Shen, Zhao Song, Mengdi Wang, and Zheng Yu. Generalized leverage score sampling for neural networks. In *NeurIPS*, 2020.

[LST20] Yin Tat Lee, Ruoqi Shen, and Kevin Tian. Logsmooth gradient concentration and tighter runtimes for metropolized Hamiltonian Monte Carlo. In *Conference on Learning Theory*, pages 2565–2597. PMLR, 2020. arXiv:2002.04121.

[LST21] Yin Tat Lee, Ruoqi Shen, and Kevin Tian. Lower bounds on Metropolized sampling methods for well-conditioned distributions. In *Advances in Neural Information Processing Systems*, volume 34, pages 18812–18824, 2021. arXiv:2106.05480.

[LSV18] Yin Tat Lee, Zhao Song, and Santosh S Vempala. Algorithmic theory of odes and sampling from well-conditioned logconcave densities. *arXiv preprint arXiv:1812.06243*, 2018.

[LSW15] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2015.

[LSZ19] Yin Tat Lee, Zhao Song, and Qiuyi Zhang. Solving empirical risk minimization in the current matrix multiplication time. In *Annual Conference on Learning Theory (COLT)*, 2019.

[LSZ20] Qipeng Liu, Amit Sahai, and Mark Zhandry. Quantum immune one-time memories, 2020.

[LT93] AG Levin and Regina Iosifovna Tyshkevich. Edge hypergraphs. *Diskretnaya Matematika*, 5(1):112–129, 1993.

[LT13] Michel Ledoux and Michel Talagrand. *Probability in Banach Spaces: isoperimetry and processes.* Springer Science & Business Media, 2013.

[LT20a] Lin Lin and Yu Tong. Near-optimal ground state preparation. *Quantum*, 4:372, 2020.

[LT20b] Lin Lin and Yu Tong. Optimal polynomial based quantum eigenstate filtering with application to solving quantum linear systems. *Quantum*, 4:361, 2020.

[LT22] Lin Lin and Yu Tong. Heisenberg-limited ground-state energy estimation for early fault-tolerant quantum computers. *PRX Quantum*, 3(1):010318, 2022.

[LTVM15] Dajie Liu, Stojan Trajanovski, and Piet Van Mieghem. Iligra: an efficient inverse line graph algorithm. *Journal of Mathematical Modelling and Algorithms in Operations Research*, 14(1):13–33, 2015.

[Lub02] Michael Luby. Lt codes. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pages 271–271. IEEE Computer Society, 2002.

[Lup58] Oleg B. Lupanov. On the synthesis of switching circuits. *Doklady Akademii Nauk SSSR*, 119(1):23–26, 1958.

[LV03] László Lovász and Santosh Vempala. Hit-and-run is fast and fun, 2003. Preprint, Microsoft Research, https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-2003-05.pdf.

[LV06] László Lovász and Santosh Vempala. Hit-and-run from a corner. *SIAM Journal on Computing*, 35(4):985–1005, 2006.

[LV07] László Lovász and Santosh Vempala. The geometry of logconcave functions and sampling algorithms. *Random Structures and Algorithms*, 30(3):307–358, 2007.

[LVAH12] Haibing Lu, Jaideep Vaidya, Vijayalakshmi Atluri, and Yuan Hong. Constraint-aware role mining via extended boolean matrix decomposition. *IEEE Transactions on Dependable and Secure Computing*, 9(5):655–669, 2012.

[LW17] Kasper Green Larsen and Ryan Williams. Faster online matrix-vector multiplication. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2182–2189. SIAM, 2017.

[LWME19] Xuechen Li, Yi Wu, Lester Mackey, and Murat A Erdogdu. Stochastic runge-kutta accelerates langevin monte carlo and beyond. *Advances in neural information processing systems*, 32, 2019.

[LXJ$^+$20] Zichang Liu, Zhaozhuo Xu, Alan Ji, Jonathan Li, Beidi Chen, and Anshumali Shrivastava. Climbing the wol: Training for cheaper inference. *arXiv preprint arXiv:2007.01230*, 2020.

[LXS$^+$19] Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8570–8581, 2019.

[LXW$^+$21] Xinjian Luo, Xiaokui Xiao, Yuncheng Wu, Juncheng Liu, and Beng Chin Ooi. A fusion-denoising attack on instahide with data augmentation. *arXiv preprint arXiv:2105.07754*, 2021.

[LYC14] Guang Hao Low, Theodore J. Yoder, and Isaac L. Chuang. Quantum inference on Bayesian networks. *Physical Review A*, 89(6):062315, 2014. arXiv:1402.7359.

[LYMP21] Shiwei Liu, Lu Yin, Decebal Constantin Mocanu, and Mykola Pechenizkiy. Do we actually need dense over-parameterization? in-time over-parameterization in sparse training. In *International Conference on Machine Learning*, pages 6989–7000. PMLR, 2021.

[LZ20] Lap Chi Lau and Hong Zhou. A spectral approach to network design. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 826–839, 2020.

[LZ22] Anthony Leverrier and Gilles Zémor. Quantum tanner codes. *arXiv preprint arXiv:2202.13641*, 2022.

[LZB20] Chaoyue Liu, Libin Zhu, and Misha Belkin. On the linearity of large non-linear models: when and why the tangent kernel is constant. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15954–15964. Curran Associates, Inc., 2020.

[LZB22] Chaoyue Liu, Libin Zhu, and Mikhail Belkin. Loss landscapes and optimization in over-parameterized non-linear systems and neural networks. *Applied and Computational Harmonic Analysis*, 59:85–116, 2022.

[LZUC22] Henrik R Larsson, Huanchen Zhai, Cyrus J Umrigar, and Garnet Kin Chan. The chromium dimer: closing a chapter of quantum chemistry. *arXiv preprint arXiv:2206.10738*, 2022.

[Mac05] David JC MacKay. Fountain codes. *IEE Proceedings-Communications*, 152(6):1062–1068, 2005.

[Mas79] William J Masek. Some np-complete set covering problems. *Unpublished Manuscript*, 1979.

[Mat92a] Jiří Matoušek. Efficient partition trees. *Discrete & Computational Geometry*, 8(3):315–334, 1992.

[Mat92b] Jiri Matousek. Reporting points in halfspaces. *Computational Geometry*, 2(3):169–186, 1992.

[Mat09] Jiri Matousek. *Geometric discrepancy: An illustrated guide*, volume 18. Springer Science & Business Media, 2009.

[MBS⁺18] Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature communications*, 9(1):1–6, 2018.

[MCC⁺19] Yi-An Ma, Niladri Chatterji, Xiang Cheng, Nicolas Flammarion, Peter Bartlett, and Michael I. Jordan. Is there an analog of nesterov acceleration for mcmc?, 2019.

[Mek14] Raghu Meka. Discrepancy and beating the union bound. In *Windows on theory, a research blog*. `https://windowsontheory.org/2014/02/07/discrepancy-and-beating-the-union-bound/`, 2014.

[MG15] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 2408–2417. JMLR.org, 2015.

[Min17] Stanislav Minsker. On some extensions of bernstein's inequality for self-adjoint operators. *Statistics & Probability Letters*, 127:111–119, 2017.

[Mir23] Piotr Mironowicz. Semi-definite programming and quantum information. *arXiv preprint arXiv:2306.16560*, 2023.

[MJC$^+$14] Lester Mackey, Michael I Jordan, Richard Y Chen, Brendan Farrell, and Joel A Tropp. Matrix concentration inequalities via the method of exchangeable pairs. *The Annals of Probability*, 42(3):906–945, 2014.

[MKZ$^+$09] Abdullah Mueen, Eamonn Keogh, Qiang Zhu, Sydney Cash, and Brandon Westover. Exact discovery of time series motifs. In *Proceedings of the 2009 SIAM international conference on data mining*, pages 473–484. SIAM, 2009.

[MLA$^+$22] Lars S Madsen, Fabian Laudenbach, Mohsen Falamarzi Askarani, Fabien Rortais, Trevor Vincent, Jacob FF Bulmer, Filippo M Miatto, Leonhard Neuhaus, Lukas G Helt, Matthew J Collins, et al. Quantum computational advantage with a programmable photonic processor. *Nature*, 606(7912):75–81, 2022.

[MM15] Pasin Manurangsi and Dana Moshkovitz. Approximating dense max 2-csps. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

[MMG$^+$08] Pauli Miettinen, Taneli Mielikäinen, Aristides Gionis, Gautam Das, and Heikki Mannila. The discrete basis problem. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 20(10):1348–1362, 2008.

[MMM19] Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Mean-field theory of two-layers neural networks: dimension-free bounds and kernel limit. In *Conference on Learning Theory*, pages 2388–2464. PMLR, 2019.

[MMM21] Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Learning with invariances in random features and kernel models. In Mikhail Belkin and Samory Kpotufe, editors, *Conference on Learning Theory, COLT 2021, 15-19 August 2021, Boulder, Colorado, USA*, volume 134 of *Proceedings of Machine Learning Research*, pages 3351–3418. PMLR, 2021.

[MMR$^+$17] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[MMR19] Konstantin Makarychev, Yury Makarychev, and Ilya Razenshteyn. Performance of johnson-lindenstrauss transform for k-means and k-medians clustering. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, pages 1027–1038, New York, NY, USA, 2019. Association for Computing Machinery.

[MMS$^+$18] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):1–12, 2018.

[MMS$^+$19] Sam McArdle, Alexander Mayorov, Xiao Shan, Simon Benjamin, and Xiao Yuan. Digital quantum simulation of molecular vibrations. *Chemical science*, 10(22):5725–5735, 2019.

[MMS20] Mateusz B Majka, Aleksandar Mijatović, and Lukasz Szpruch. Nonasymptotic bounds for sampling algorithms without log-concavity. *Annals of applied probability: an official journal of the Institute of Mathematical Statistics*, 30(4):1534–1581, 2020.

[MN20] Pauli Miettinen and Stefan Neumann. Recent developments in boolean matrix factorization. *arXiv preprint arXiv:2012.03127*, 2020.

[MNRS11] Frédéric Magniez, Ashwin Nayak, Jérémie Roland, and Miklos Santha. Search via quantum walk. *SIAM Journal on Computing*, 40(1):142–164, 2011.

[MNV17] Raghu Meka, Oanh Nguyen, and Van Vu. Anti-concentration for polynomials of independent random variables. In *Theory Of Computing*. arXiv preprint arXiv:1507.00829, 2017.

[Moi13] Ankur Moitra. An almost optimal algorithm for computing nonnegative rank. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1454–1464. SIAM, 2013.

[Moi15] Ankur Moitra. Super-resolution, extremal functions and the condition number of vandermonde matrices. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 821–830, 2015.

[Mon15] Ashley Montanaro. Quantum speedup of Monte Carlo methods. *Proceedings of the Royal Society A*, 471(2181):20150301, 2015. arXiv:1504.06987.

[MOSW22] Alexander Munteanu, Simon Omlor, Zhao Song, and David Woodruff. Bounding the width of neural networks via coupled initialization a worst case analysis. In *International Conference on Machine Learning*, pages 16083–16122. PMLR, 2022.

[MPS14] Andrew M McDonald, Massimiliano Pontil, and Dimitris Stamos. Spectral k-support norm regularization. *Advances in neural information processing systems*, 27, 2014.

[MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Cambridge university press, 1995.

[MRBAG16] Jarrod R McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18(2):023023, 2016.

[MRTC21] John M. Martyn, Zane M. Rossi, Andrew K. Tan, and Isaac L. Chuang. Grand unification of quantum algorithms. *PRX Quantum*, 2:040203, Dec 2021.

[MS13] J. Maldacena and L. Susskind. Cool horizons for entangled black holes. *Fortschritte der Physik*, 61(9):781–811, Aug 2013.

[MSS15a] Adam W. Marcus, Daniel A. Spielman, and Nikhil Srivastava. Interlacing families I: Bipartite Ramanujan graphs of all degrees. *Ann. of Math. (2)*, 182(1):307–325, 2015.

[MSS15b] Adam W. Marcus, Daniel A. Spielman, and Nikhil Srivastava. Interlacing families II: Mixed characteristic polynomials and the Kadison-Singer problem. *Ann. of Math. (2)*, 182(1):327–350, 2015.

[MSS18] Adam W Marcus, Daniel A Spielman, and Nikhil Srivastava. Interlacing families IV: Bipartite ramanujan graphs of all sizes. *SIAM Journal on Computing*, 47(6):2488–2509, 2018.

[MSVA16] Barsha Mitra, Shamik Sural, Jaideep Vaidya, and Vijayalakshmi Atluri. A survey of role mining. *ACM Computing Surveys (CSUR)*, 48(4):1–37, 2016.

[MT97] Yury Metelsky and Regina Tyshkevich. On line graphs of linear 3-uniform hypergraphs. *Journal of Graph Theory*, 25(4):243–251, 1997.

[MT00] Renato DC Monteiro and Takashi Tsuchiya. Polynomial convergence of primal-dual algorithms for the second-order cone program based on the mz-family of directions. *Mathematical programming*, 88(1):61–83, 2000.

[MT14] Tor Myklebust and Levent Tunçel. Interior-point algorithms for convex optimization based on primal-dual metrics. *arXiv preprint arXiv:1411.2129*, 2014.

[MW92] Yigal Meir and Ned S Wingreen. Landauer formula for the current through an interacting electron region. *Physical review letters*, 68(16):2512, 1992.

[MW05] Chris Marriott and John Watrous. Quantum Arthur–Merlin games. *Computational Complexity*, 14(2):122–152, 2005.

[MW17] Cody D Murray and R Ryan Williams. On the (non) np-hardness of computing circuit complexity. *Theory of Computing*, 13(1):1–22, 2017.

[MW19] Hesham Mostafa and Xin Wang. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *International Conference on Machine Learning*, pages 4646–4655. PMLR, 2019.

[MWR$^+$14] D. Marcos, P. Widmer, E. Rico, M. Hafezi, P. Rabl, U.-J. Wiese, and P. Zoller. Two-dimensional lattice gauge theories with superconducting quantum circuits. *Annals of Physics*, 351:634–654, Dec 2014.

[MZ10] Lingsheng Meng and Bing Zheng. The optimal perturbation bounds of the moore–penrose inverse under the frobenius norm. *Linear algebra and its applications*, 432(4):956–963, 2010.

[MZ13] Raghu Meka and David Zuckerman. Pseudorandom generators for polynomial threshold functions. *SIAM Journal on Computing*, 42(3):1275–1301, 2013.

[Nar16] Hariharan Narayanan. Randomized interior point methods for sampling and optimization. *The Annals of Applied Probability*, 26(1):597–641, 2016.

[NC10] Michael A Nielsen and Isaac Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

[NC11] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, USA, 10th edition, 2011.

[Nee12] Frank Neese. The orca program system. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 2(1):73–78, 2012.

[Nee18] Frank Neese. Software update: the orca program system, version 4.0. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 8(1):e1327, 2018.

[Nes88a] Yu Nesterov. Polynomial-time iterative methods in linear and quadratic programming. *Voprosy kibernetiki, Moscow*, pages 102–125, 1988.

[Nes88b] YY Nesterov. Polynomial methods in the linear and quadratic-programming. *Soviet Journal of Computer and Systems Sciences*, 26(5):98–101, 1988.

[Ngu13] Hoi H Nguyen. On the singularity of random combinatorial matrices. *SIAM Journal on Discrete Mathematics*, 27(1):447–458, 2013.

[Nis91] Noam Nisan. Lower bounds for non-commutative computation. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 410–418, 1991.

[NN89] Yurii Nesterov and Arkadi Nemirovski. Self-concordant functions and polynomial time methods in convex programming. preprint, central economic & mathematical institute, ussr acad. *Sci. Moscow, USSR*, 1989.

[NN92] Yurii Nesterov and Arkadi Nemirovski. Conic formulation of a convex programming problem and duality. *Optimization Methods and Software*, 1(2):95–115, 1992.

[NN94] Yurii Nesterov and Arkadi Nemirovski. *Interior-point polynomial algorithms in convex programming*, volume 13. Siam, 1994.

[NN13] Jelani Nelson and Huy L Nguyên. OSNAP: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2013.

[NP18] Simone Naldi and Daniel Plaumann. Symbolic computation in hyperbolic programming. *Journal of Algebra and Its Applications*, 17(10):1850192, 2018.

[NRR20] Assaf Naor, Shravas Rao, and Oded Regev. Concentration of markov chains with bounded moments. In *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, volume 56, pages 2270–2280. Institut Henri Poincaré, 2020.

[NS16] Aleksandar Nikolov and Mohit Singh. Maximizing determinants under partition constraints. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 192–201, 2016.

[NS19] Vasileios Nakos and Zhao Song. Stronger l2/l2 compressed sensing; without iterating. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 289–297, 2019.

[NSW19] Vasileios Nakos, Zhao Song, and Zhengyu Wang. (nearly) sample-optimal sparse fourier transform in any dimension; ripless and filterless. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1568–1577. IEEE, 2019.

[NTM01] Alexandros Nanopoulos, Yannis Theodoridis, and Yannis Manolopoulos. C2P: Clustering based on closest pairs. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB 2001)*, pages 331–340, 2001.

[NW99] Ashwin Nayak and Felix Wu. The quantum query complexity of approximating the median and related statistics. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 384–393, 1999. arXiv:quant-ph/9804066.

[OB$^+$19] OpenAI, :, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław “Psyho" Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d. O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning, 2019.

[OBD18] Davide Orsucci, Hans J. Briegel, and Vedran Dunjko. Faster quantum mixing for slowly evolving sequences of markov chains. *Quantum*, 2:105, 2018. arXiv:1503.01334.

[Oli09] Roberto Imbuzeiro Oliveira. Concentration of the adjacency matrix and of the laplacian in random graphs with independent edges. *arXiv preprint arXiv:0911.0600*, 2009.

[Oli19] Igor Carboni Oliveira. Advances in hardness magnification. `https://www.dcs.warwick.ac.uk/~igorcarb/documents/papers/magnification-note.pdf`, 2019.

[Ope23] OpenAI. Gpt-4 technical report, 2023.

[Opp11] Alan V. Oppenheim. Lecture notes: Fourier transform properties. https://ocw.aprende.org/resources/res-6-007-signals-and-systems-spring-2011/lecture-notes/MITRES_6_007S11_lec09.pdf, 2011.

[OPS19] Igor Carboni Oliveira, Ján Pich, and Rahul Santhanam. Hardness magnification near state-of-the-art lower bounds. In *34th Computational Complexity Conference (CCC 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[Orl77] James Orlin. Contentment in graph theory: Covering graphs with cliques. *Indagationes Mathematicae (Proceedings)*, 80(5):406–424, 1977.

[ORR13] Maris Ozols, Martin Roetteler, and Jérémie Roland. Quantum rejection sampling. *ACM Transactions on Computation Theory (TOCT)*, 5(3):1–33, 2013. arXiv:1103.2774.

[OS16] Igor C Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds and pseudorandomness. *arXiv preprint arXiv:1611.01190*, 2016.

[OS18] Igor Carboni Oliveira and Rahul Santhanam. Hardness magnification for natural problems. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 65–76. IEEE, 2018.

[OS20] Samet Oymak and Mahdi Soltanolkotabi. Toward moderate overparameterization: Global convergence guarantees for training shallow neural networks. *IEEE Journal on Selected Areas in Information Theory*, 1(1):84–105, 2020.

[Osg02] Brad Osgood. Lecture notes for ee 261 the fourier transform and its applications. https://see.stanford.edu/materials/lsoftaee261/book-fall-07.pdf, 2002.

[OSS$^+$19] Thomas E O'Brien, Bruno Senjean, Ramiro Sagastizabal, Xavier Bonet-Monroig, Alicja Dutkiewicz, Francesco Buda, Leonardo DiCarlo, and Lucas Visscher. Calculating energy derivatives for quantum chemistry on a quantum computer. *npj Quantum Information*, 5(1):1–12, 2019.

[OST19] Ryan O'Donnell, Rocco A Servedio, and Li-Yang Tan. Fooling polytopes. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 614–625, 2019.

[OWN$^+$97] Alan V Oppenheim, Alan S Willsky, Syed Hamid Nawab, Gloria Mata Hernández, et al. *Signals & systems*. Pearson Educación, 1997.

[Pag13] Rasmus Pagh. Compressed matrix multiplication. *ACM Transactions on Computation Theory (TOCT)*, 5(3):1–17, 2013.

[PAH$^+$18] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5):1333–1345, 2018.

[Pat10] Mihai Patrascu. Towards polynomial lower bounds for dynamic problems. In *Proceedings of the forty-second ACM symposium on Theory of computing (STOC)*, pages 603–610, 2010.

[Pel00] David Peleg. Proximity-preserving labeling schemes. *Journal of Graph Theory*, 33(3):167–176, 2000.

[PK21] Pavel Panteleev and Gleb Kalachev. Asymptotically good quantum and locally testable classical ldpc codes. *arXiv preprint arXiv:2111.03654*, 2021.

[PMS$^+$14] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, 5(1):1–7, 2014.

[Pol19] Yury Polyanskiy. Hypercontractivity of spherical averages in hamming space. *SIAM Journal on Discrete Mathematics*, 33(2):731–754, 2019.

[Pow07] Michael JD Powell. A view of algorithms for optimization without derivatives. *Mathematics Today-Bulletin of the Institute of Mathematics and its Applications*, 43(5):170–174, 2007.

[PP10] Ramamohan Paturi and Pavel Pudlák. On the complexity of circuit satisfiability. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC 2010)*, pages 241–250. ACM, 2010.

[PP12] Robert Peharz and Franz Pernkopf. Sparse nonnegative matrix factorization with ℓ_0-constraints. *Neurocomputing*, 80:38–46, 2012.

[PP14] Robin Pemantle and Yuval Peres. Concentration of lipschitz functionals of determinantal and other strong rayleigh measures. *Combinatorics, Probability and Computing*, 23(1):140–160, 2014.

[PPSZ05] Ramamohan Paturi, Pavel Pudlák, Michael E Saks, and Francis Zane. An improved exponential-time algorithm for k-SAT. *Journal of the ACM (JACM)*, 52(3):337–364, 2005.

[Pré71] András Prékopa. Logarithmic concave measures with applications to stochastic programming. *Acta Scientiarum Mathematicarum*, 32:301–316, 1971.

[Pré73] András Prékopa. On logarithmic concave measures and functions. *Acta Scientiarum Mathematicarum*, 34:335–343, 1973.

[Pre18] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

[Pri11] Eric Price. Efficient sketches for the set query problem. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 41–56. SIAM, 2011.

[PRT81] Svatopluk Poljak, Vojtěch Rödl, and Daniel TURZiK. Complexity of representation of graphs by set systems. *Discrete Applied Mathematics*, 3(4):301–312, 1981.

[PS15] Eric Price and Zhao Song. A robust sparse Fourier transform in the continuous setting. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 583–600. IEEE, 2015.

[PV21] Richard Peng and Santosh Vempala. Solving sparse linear systems faster than matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 504–521. SIAM, 2021.

[PW09] David Poulin and Pawel Wocjan. Preparing ground states of quantum many-body systems on a quantum computer. *Physical review letters*, 102(13):130503, 2009.

[PW17] Mert Pilanci and Martin J. Wainwright. Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence. *SIAM J. Optim.*, 27:205–245, 2017.

[QWZ18] Willy Quach, Daniel Wichs, and Giorgos Zirdelis. Watermarking prfs under standard assumptions: Public marking and security with extraction queries. In *Theory of Cryptography Conference*, pages 669–698. Springer, 2018.

[Rab76] Michael O Rabin. Probabilistic algorithms algorithms and complexity: New directions and recent results, 1976.

[Rab02] Majid Rabbani. Jpeg2000: Image compression fundamentals, standards and practice. *Journal of Electronic Imaging*, 11(2):286, 2002.

[Ral20] Patrick Rall. Quantum algorithms for estimating physical quantities using block encodings. *Physical Review A*, 102(2):022408, 2020.

[Ral21] Patrick Rall. Faster coherent quantum algorithms for phase, energy, and amplitude estimation. *Quantum*, 5:566, 2021.

[Raz17] Ilya Razenshteyn. *High-dimensional similarity search and sketching: algorithms and hardness*. PhD thesis, Massachusetts Institute of Technology, 2017.

[RDN$^+$22] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[Rei09] Ben W. Reichardt. Span programs and quantum query complexity: The general adversary bound is nearly tight for every boolean function. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 544–551. IEEE, 2009. arXiv:0904.2759.

[Ren88] James Renegar. A polynomial-time algorithm, based on newton's method, for linear programming. *Mathematical Programming*, 40(1-3):59–93, 1988.

[Ren01] James Renegar. *A Mathematical View of Interior-point Methods in Convex Optimization.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[Ren06] James Renegar. Hyperbolic programs, and their derivative relaxations. *Foundations of Computational Mathematics*, 6(1):59–79, 2006.

[Ren16] James Renegar. "Efficient" subgradient methods for general convex optimization. *SIAM Journal on Optimization*, 26(4):2649–2676, 2016.

[Ren19a] James Renegar. Accelerated first-order methods for hyperbolic programming. *Mathematical Programming*, 173(1-2):1–35, 2019.

[Ren19b] James Renegar. Personal communication, 2019.

[Rey89] George O Reynolds. *The New Physical Optics Notebook: Tutorials in Fourier Optics.* ERIC, 1989.

[RGM$^+$21] Julia E Rice, Tanvi P Gujarati, Mario Motta, Tyler Y Takeshita, Eunseok Lee, Joseph A Latone, and Jeannette M Garcia. Quantum computation of dominant products in lithium–sulfur batteries. *The Journal of Chemical Physics*, 154(13):134115, 2021.

[Ric07] Peter C. Richter. Quantum speedup of classical mixing processes. *Physical Review A*, 76(4):042306, 2007.

[RL16] Andrej Risteski and Yuanzhi Li. Algorithms and matching lower bounds for approximately-convex optimization. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

[RL20] Mohan Ravichandran and Jonathan Leake. Mixed determinants and the kadison–singer problem. *Mathematische Annalen*, 377(1):511–541, 2020.

[RML14] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Physical review letters*, 113(13):130503, 2014.

[Rot16] Thomas Rothvoss. Integer optimization and lattices. *University of Washington, Spring*, 2016.

[Rou73] Nicholas D Roussopoulos. A max$\{m, n\}$ algorithm for determining the graph h from its line graph g. *Information Processing Letters*, 2(4):108–112, 1973.

[RPG16] Siamak Ravanbakhsh, Barnabás Póczos, and Russell Greiner. Boolean matrix factorization and noisy completion via message passing. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning (ICML)*, pages 945–954, 2016.

[RR97] Alexander A Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 1(55):24–35, 1997.

[RR20] Victor Reis and Thomas Rothvoss. Vector balancing in lebesgue spaces. *arXiv preprint arXiv:2007.05634*, 2020.

[RRSW19] Prasad Raghavendra, Nick Ryder, Nikhil Srivastava, and Benjamin Weitz. Exponential lower bounds on spectrahedral representations of hyperbolicity cones. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2322–2332. SIAM, 2019.

[RRT12] Holger Rauhut, Justin Romberg, and Joel A Tropp. Restricted isometries for partial random circulant matrices. *Applied and Computational Harmonic Analysis*, 32(2):242–254, 2012.

[RS14] James Renegar and Mutiara Sondjaja. A polynomial-time affine-scaling method for semidefinite and hyperbolic programming. *arXiv preprint arXiv:1410.6734*, 2014.

[RS19] Roy Radian and Or Sattath. Semi-quantum money. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, AFT '19, pages 132–146. Association for Computing Machinery, 2019.

[RS21] Hanlin Ren and Rahul Santhanam. A relativization perspective on meta-complexity. *Electron. Colloquium Comput. Complex.*, 28:89, 2021.

[RSBG19] Abhishek Roy, Lingqing Shen, Krishnakumar Balasubramanian, and Saeed Ghadimi. Stochastic zeroth-order discretizations of Langevin diffusions for Bayesian inference, 2019. arXiv:1902.01373.

[RSL18] Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 10877–10887, 2018.

[RSS12] Sasha Rakhlin, Ohad Shamir, and Karthik Sridharan. Relax and randomize: From value to algorithms. *Advances in Neural Information Processing Systems*, 25, 2012.

[RSW16] Ilya Razenshteyn, Zhao Song, and David P Woodruff. Weighted low rank approximations with provable guarantees. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing (STOC)*, pages 250–263, 2016.

[RTD$^+$18] Theo Ryffel, Andrew Trask, Morten Dahl, Bobby Wagner, Jason Mancuso, Daniel Rueckert, and Jonathan Passerat-Palmbach. A generic framework for privacy preserving deep learning. *CoRR*, abs/1811.04017, 2018.

[Rud99] Mark Rudelson. Random vectors in the isotropic position. *Journal of Functional Analysis*, 164(1):60–72, 1999.

[RV07] Mark Rudelson and Roman Vershynin. Sampling from large matrices: An approach through geometric functional analysis. *Journal of the ACM (JACM)*, 54(4), 2007.

[RV08] Mark Rudelson and Roman Vershynin. On sparse reconstruction from fourier and gaussian measurements. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 61(8):1025–1045, 2008.

[RV09] Mark Rudelson and Roman Vershynin. Smallest singular value of a random rectangular matrix. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 62(12):1707–1739, 2009.

[RV13] Mark Rudelson and Roman Vershynin. Hanson-wright inequality and sub-gaussian concentration. *Electronic Communications in Probability*, 18:1–9, 2013.

[SAH$^+$20] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[San09] Rahul Santhanam. Circuit lower bounds for merlin–arthur classes. *SIAM Journal on Computing*, 39(3):1038–1061, 2009.

[Sar06] Tamás Sarlós. Improved approximation algorithms for large matrices via random projections. In *Proceedings of 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2006.

[Sau18] James Saunderson. A spectrahedral representation of the first derivative relaxation of the positive semidefinite cone. *Optimization Letters*, 12(7):1475–1486, 2018.

[SB03] Paris Smaragdis and Judith C Brown. Non-negative matrix factorization for polyphonic music transcription. In *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (IEEE Cat. No. 03TH8684)*, pages 177–180. IEEE, 2003.

[SB13] Rolando D Somma and Sergio Boixo. Spectral gap amplification. *SIAM Journal on Computing*, 42(2):593–610, 2013.

[SBB07] Rolando D. Somma, Sergio Boixo, and Howard Barnum. Quantum simulated annealing, 2007. arXiv:0712.1008.

[SBBK08] Rolando D. Somma, Sergio Boixo, Howard Barnum, and Emanuel Knill. Quantum simulations of classical annealing processes. *Physical Review Letters*, 101(13):130504, 2008. arXiv:0804.1571.

[SBM03] Jouni K Seppänen, Ella Bingham, and Heikki Mannila. A simple algorithm for topic identification in 0–1 data. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 423–434. Springer, 2003.

[Sch11] J. Schur. Bemerkungen zur theorie der beschränkten bilinearformen mit unendlich vielen veränderlichen. *Journal für die reine und angewandte Mathematik*, 140:1–28, 1911.

[Sch99] T Schoning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1999)*, pages 410–414. IEEE, 1999.

[SFGP21] Daniel Stilck França and Raul Garcia-Patron. Limitations of optimization algorithms on noisy quantum devices. *Nature Physics*, 17(11):1221–1227, 2021.

[SH75] M. I. Shamos and D. Hoey. Closest-point problems. In *Proceedings of the 16th Annual Symposium on Foundations of Computer Science (SFCS 1975)*, pages 151–162, 1975.

[ŠH06] Tomáš Šingliar and Miloš Hauskrecht. Noisy-or component analysis and its application to link analysis. *Journal of Machine Learning Research (JMLR)*, 7(Oct):2189–2213, 2006.

[Shi02] Yaoyun Shi. Both toffoli and controlled-not need little help to do universal quantum computation. *arXiv preprint quant-ph/0205115*, 2002.

[SHM$^+$16] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[Sho77] Naum Z Shor. Cut-off method with space extension in convex programming problems. *Cybernetics and systems analysis*, 13(1):94–96, 1977.

[Sho94] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

[SL11] Roman Sandler and Michael Lindenbaum. Nonnegative matrix factorization with earth mover's distance metric for image analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1590–1602, 2011.

[SL14] Anshumali Shrivastava and Ping Li. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). *Advances in Neural Information Processing Systems (NIPS)*, pages 2321–2329, 2014.

[SL15a] Anshumali Shrivastava and Ping Li. Asymmetric minwise hashing for indexing binary inner products and set containment. In *Proceedings of the 24th international conference on world wide web (WWW)*, pages 981–991, 2015.

[SL15b] Anshumali Shrivastava and Ping Li. Improved asymmetric locality sensitive hashing (alsh) for maximum inner product search (mips). In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 812–821, 2015.

[SL19] Ruoqi Shen and Yin Tat Lee. The randomized midpoint method for log-concave sampling. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 2100–2111, 2019. arXiv:1909.05503.

[SLJ$^+$15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[Smi84] Robert L. Smith. Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, 32(6):1296–1308, 1984.

[SMS18] Khot Subhash, Dor Minzer, and Muli Safra. Pseudorandom sets in grassmann graph have near-perfect expansion. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 592–601. IEEE, 2018.

[SO06] Mikkel N Schmidt and Rasmus K Olsson. Single-channel speech separation using sparse non-negative matrix factorization. In *Ninth International Conference on Spoken Language Processing*, 2006.

[SO12] Attila Szabo and Neil S Ostlund. *Modern quantum chemistry: introduction to advanced electronic structure theory.* Courier Corporation, 2012.

[Som19] Rolando D Somma. Quantum eigenvalue estimation via time series analysis. *New Journal of Physics*, 21(12):123025, 2019.

[Son19] Zhao Song. *Matrix Theory: Optimization, Concentration and Algorithms.* PhD thesis, The University of Texas at Austin, 2019.

[Spe85] Joel Spencer. Six standard deviations suffice. *Transactions of the American mathematical society*, 289(2):679–706, 1985.

[SRL12] Jacob T Seeley, Martin J Richard, and Peter J Love. The bravyi-kitaev transformation for quantum computation of electronic structure. *The Journal of chemical physics*, 137(22):224109, 2012.

[SS02] Michael Saks and Xiaodong Sun. Space lower bounds for distance approximation in the data stream model. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 360–369, 2002.

[SS15] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 1310–1321, New York, NY, USA, 2015. Association for Computing Machinery.

[SS17] Dominik Scheder and John P Steinberger. PPSZ for general k-SAT: Making Hertli's analysis simpler and 3-SAT faster. In *Proceedings of the 32nd Computational Complexity Conference (CCC 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms.* Cambridge university press, 2014.

[SSO19] Yiğit Subaşı, Rolando D Somma, and Davide Orsucci. Quantum algorithms for systems of linear equations inspired by adiabatic quantum computing. *Physical review letters*, 122(6):060504, 2019.

[SSS$^+$17] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[SSSSC11] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

[SST01] Kunihiko Sadakane, Norito Sugawara, and Takeshi Tokuyama. Quantum algorithms for intersection and proximity problems. In *Proceedings of the 12th International Symposium on Algorithms and Computation (ISAAC 2001)*, pages 148–159. Springer Berlin Heidelberg, 2001.

[SST05] PV Skums, SV Suzdal, and RI Tyshkevich. Edge intersection graphs of linear 3-uniform hypergraphs. *Electronic Notes in Discrete Mathematics*, 22:33–40, 2005.

[SSWZ22] Zhao Song, Baocheng Sun, Omri Weinstein, and Ruizhe Zhang. Sparse fourier transform over lattices: A unified approach to signal reconstruction. *arXiv preprint arXiv:2205.00658*, 2022.

[SSX21] Anshumali Shrivastava, Zhao Song, and Zhaozhuo Xu. Sublinear least-squares value iteration via locality sensitive hashing. *arXiv preprint arXiv:2105.08285*, 2021.

[Sta88] Richard P Stanley. Differential posets. *Journal of the American Mathematical Society*, 1(4):919–961, 1988.

[Ste66] P Stein. A note on the volume of a simplex. *The American Mathematical Monthly*, 73(3):299–301, 1966.

[Sus16] Leonard Susskind. Computational complexity and black hole horizons. *Fortschritte der Physik*, 64(1):24–43, 2016.

[SV17] Damian Straszak and Nisheeth K Vishnoi. Real stable polynomials and matroids: Optimization and counting. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 370–383, 2017.

[ŠVV09] Daniel Štefankovič, Santosh Vempala, and Eric Vigoda. Adaptive simulated annealing: A near-optimal connection between sampling and counting. *Journal of the ACM*, 56(3):1–36, 2009.

[SWL21] Zhenmei Shi, Junyi Wei, and Yingyu Liang. A theoretical analysis on feature learning in neural networks: Emergence from inputs and advantage over fixed features. In *International Conference on Learning Representations*, 2021.

[SWY$^+$19] Zhao Song, Ruosong Wang, Lin Yang, Hongyang Zhang, and Peilin Zhong. Efficient symmetric norm regression via linear sketching. *Advances in Neural Information Processing Systems*, 32, 2019.

[SWYZ21] Zhao Song, David P. Woodruff, Zheng Yu, and Lichen Zhang. Fast sketching of polynomial kernels of polynomial degree. In *ICML*, 2021.

[SWZ17] Zhao Song, David P Woodruff, and Peilin Zhong. Low rank approximation with entrywise ℓ_1-norm error. In *Proceedings of the 49th Annual Symposium on the Theory of Computing (STOC)*. ACM, 2017.

[SWZ19] Zhao Song, David P Woodruff, and Peilin Zhong. Relative error tensor low rank approximation. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2019.

[SXZ22] Zhao Song, Zhaozhuo Xu, and Lichen Zhang. Speeding up sparsification using inner product search data structures. *arXiv preprint arXiv:2204.03209*, 2022.

[SY19] Zhao Song and Xin Yang. Quadratic suffices for over-parametrization via matrix chernoff bound. In *arXiv preprint*. `https://arxiv.org/pdf/1906.03593.pdf`, 2019.

[SY21] Kazuhiro Seki and Seiji Yunoki. Quantum power method by a superposition of time-evolved states. *PRX Quantum*, 2(1):010333, 2021.

[Sys82] Maciej M Syslo. A labeling algorithm to recognize a line digraph and output its root graph. *Information Processing Letters*, 15(1):28–30, 1982.

[SYZ21] Zhao Song, Shuo Yang, and Ruizhe Zhang. Does preprocessing help training over-parameterized neural networks? In *Thirty-Fifth Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[SZ22] Zhao Song and Ruizhe Zhang. Hyperbolic concentration, anti-concentration, and discrepancy. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

[Sze04] Mario Szegedy. Quantum speed-up of Markov chain based algorithms. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 32–41. IEEE, 2004.

[Tan19] Ewin Tang. A quantum-inspired classical algorithm for recommendation systems. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 217–228, 2019.

[Tan21] Ewin Tang. Quantum principal component analysis only achieves an exponential speedup because of its state preparation assumptions. *Physical Review Letters*, 127(6):060503, 2021.

[Tao13] Terence Tao. Real stable polynomials and the kadison-singer problem. `https://terrytao.wordpress.com/2013/11/04/real-stable-polynomials-and-th`
2013.

[TAWL21] Yu Tong, Dong An, Nathan Wiebe, and Lin Lin. Fast inversion, preconditioned quantum linear system solvers, fast green's-function computation, and fast evaluation of matrix functions. *Physical Review A*, 104(3):032422, 2021.

[Tel22] Matus Telgarsky. Feature selection with gradient descent on two-layer networks in low-rotation regimes. *arXiv preprint arXiv:2208.02789*, 2022.

[Tho65] Colin J Thompson. Inequality with applications in statistical mechanics. *Journal of Mathematical Physics*, 6(11):1812–1813, 1965.

[TJ74] Nicole Tomczak-Jaegermann. The moduli of smoothness and convexity and the rademacher averages of the trace classes $s_p(1 \leq p < \infty)$. *Studia Mathematica*, 50(2):163–182, 1974.

[TMZE$^+$18] Norm M Tubman, Carlos Mejuto-Zaera, Jeffrey M Epstein, Diptarka Hait, Daniel S Levine, William Huggins, Zhang Jiang, Jarrod R McClean, Ryan Babbush, Martin Head-Gordon, et al. Postponing the orthogonality catastrophe: efficient state preparation for electronic structure simulations on quantum devices. *arXiv preprint arXiv:1809.05523*, 2018.

[TOG17] Csaba D Toth, Joseph O'Rourke, and Jacob E Goodman. *Handbook of discrete and computational geometry.* CRC press, 2017.

[Ton22] Yu Tong. Designing algorithms for estimating ground state properties on early fault-tolerant quantum computers. *Quantum Views*, 6:65, 2022.

[TOV$^+$11] Kristan Temme, Tobias J. Osborne, Karl G. Vollbrecht, David Poulin, and Frank Verstraete. Quantum Metropolis sampling. *Nature*, 471(7336):87–90, 2011. arXiv:0911.3635.

[Tra84] Boris A Trakhtenbrot. A survey of russian approaches to perebor (brute-force searches) algorithms. *Annals of the History of Computing*, 6(4):384–400, 1984.

[Tro12] Joel A Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of computational mathematics*, 12(4):389–434, 2012.

[Tro15] Joel A Tropp. An introduction to matrix concentration inequalities. *Foundations and Trends in Machine Learning*, 8(1-2):1–230, 2015.

[Tro18] Joel A Tropp. Second-order matrix concentration inequalities. *Applied and Computational Harmonic Analysis*, 44(3):700–736, 2018.

[TV07] Luca Trevisan and Salil Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007.

[TZ12] Mikkel Thorup and Yin Zhang. Tabulation-based 5-independent hashing with applications to linear probing and second moment estimation. *SIAM J. Comput.*, 41(2):293–331, 2012.

[vACGN22] Joran van Apeldoorn, Arjan Cornelissen, András Gilyén, and Giacomo Nannicini. Quantum tomography using state-preparation unitaries. *arXiv preprint arXiv:2207.08800*, 2022.

[VAG07] Jaideep Vaidya, Vijayalakshmi Atluri, and Qi Guo. The role mining problem: finding a minimal descriptive set of roles. In *Proceedings of the 12th ACM symposium on Access control models and technologies*, pages 175–184, 2007.

[Vai89a] Pravin M Vaidya. A new algorithm for minimizing convex functions over convex sets. In *30th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 338–343, 1989.

[Vai89b] Pravin M Vaidya. Speeding-up linear programming using fast matrix multiplication. In *30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 332–337. IEEE, 1989.

[Val84] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[Val12] Gregory Valiant. Finding correlations in subquadratic time, with applications to learning parities and juntas. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 11–20. IEEE, 2012.

[VBW98] Lieven Vandenberghe, Stephen Boyd, and Shao-Po Wu. Determinant maximization with linear matrix inequality constraints. *SIAM journal on matrix analysis and applications*, 19(2):499–533, 1998.

[vdBGJ+22] Jan van den Brand, Yu Gao, Arun Jambulapati, Yin Tat Lee, Yang P Liu, Richard Peng, and Aaron Sidford. Faster maxflow via improved dynamic spectral vertex sparsifiers. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 543–556, 2022.

[Ver20] Roman Vershynin. Concentration inequalities for random tensors. *Bernoulli*, 26(4):3139–3162, 2020.

[VGL+16] Arnaud Vandaele, Nicolas Gillis, Qi Lei, Kai Zhong, and Inderjit Dhillon. Efficient and non-convex coordinate descent for symmetric nonnegative matrix factorization. *IEEE Transactions on Signal Processing*, 64(21):5571–5584, 2016.

[VM10] Nilton Volpato and Arnaldo Moura. A fast quantum algorithm for the closest bichromatic pair problem, 2010.

[Voe11] David George Voelz. *Computational fourier optics: a MATLAB tutorial.* SPIE press Bellingham, Washington, 2011.

[Vu08] Van Vu. Random discrete matrices. In *Horizons of combinatorics*, pages 257–280. Springer, 2008.

[VW15] Virginia Vassilevska Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In *10th International Symposium on Parameterized and Exact Computation (IPEC 2015).* Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

[VW19] Santosh Vempala and Andre Wibisono. Rapid convergence of the unadjusted Langevin algorithm: Isoperimetry suffices. *Advances in Neural Information Processing Systems*, 32:8094–8106, 2019. arXiv:1903.08568.

[VZGG99] Joachim Von Zur Gathen and Jürgen Gerhard. *Modern computer algebra*. Cambridge university press, 1999.

[WA08] Pawel Wocjan and Anura Abeyesinghe. Speedup via quantum sampling. *Physical Review A*, 78(4):042336, 2008. arXiv:0804.4259.

[Wag11] David Wagner. Multivariate stable polynomials: theory and applications. *Bulletin of the American Mathematical Society*, 48(1):53–84, 2011.

[Wat94] Andrew B. Watson. Image compression using the discrete cosine transform. *Mathematica Journal*, 4:81–88, 1994.

[WB21] James D. Watson and Johannes Bausch. The complexity of approximating critical points of quantum phase transitions, 2021.

[WBC21a] Kianna Wan, Mario Berta, and Earl T Campbell. A randomized quantum algorithm for statistical phase estimation. *arXiv preprint arXiv:2110.12071*, 2021.

[WBC$^+$21b] Yulin Wu, Wan-Su Bao, Sirui Cao, Fusheng Chen, Ming-Cheng Chen, Xiawei Chen, Tung-Hsun Chung, Hui Deng, Yajie Du, Daojin Fan, et al. Strong quantum computational advantage using a superconducting quantum processor. *Physical review letters*, 127(18):180501, 2021.

[WBG20] James D. Watson, Johannes Bausch, and Sevag Gharibian. The complexity of translationally invariant problems beyond ground state energies, 2020.

[WCNA09] Pawel Wocjan, Chen-Fu Chiang, Daniel Nagaj, and Anura Abeyesinghe. Quantum algorithm for approximating partition functions. *Physical Review A*, 80(2):022340, 2009. arXiv:0811.0596.

[Wea04] Nik Weaver. The Kadison-Singer problem in discrepancy theory. *Discrete Math.*, 278(1-3):227–239, 2004.

[Wed73] Per-Åke Wedin. Perturbation theory for pseudo-inverses. *BIT Numerical Mathematics*, 13(2):217–232, 1973.

[WG15] Nathan Wiebe and Christopher Granade. Can small quantum systems learn?, 2015. arXiv:1512.03145.

[Whi92] Hassler Whitney. Congruent graphs and the connectivity of graphs. In *Hassler Whitney Collected Papers*, pages 61–79. Springer, 1992.

[Wib19] Andre Wibisono. Proximal langevin algorithm: Rapid convergence under isoperimetry. *arXiv preprint arXiv:1911.01469*, 2019.

[Wie83] Stephen Wiesner. Conjugate coding. *ACM Sigact News*, 15(1):78–88, 1983.

[Wil05] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2-3):357–365, 2005.

[Wil12] Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing (STOC)*, pages 887–898. ACM, 2012.

[Wil17] Ryan Williams. Pairwise comparison of bit vectors. Theoretical Computer Science Stack Exchange, 2017. `https://cstheory.stackexchange.com/q/37369`, visited 2020-07-21.

[Wil18] Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the ICM*, volume 3, pages 3431–3472. World Scientific, 2018.

[WJB03] Pawel Wocjan, Dominik Janzing, and Thomas Beth. Two QCMA-complete problems. *Quantum Info. Comput.*, 3(6):635–643, November 2003.

[WKJC21] Guoming Wang, Dax Enshan Koh, Peter D Johnson, and Yudong Cao. Minimizing estimation runtime on noisy quantum computers. *PRX Quantum*, 2(1):010346, 2021.

[WLW$^+$11] Fei Wang, Tao Li, Xin Wang, Shenghuo Zhu, and Chris Ding. Community discovery using nonnegative matrix factorization. *Data Mining and Knowledge Discovery*, 22(3):493–521, 2011.

[Woo49] Max A Woodbury. The stability of out-input matrices. *Chicago, IL*, 9, 1949.

[Woo50] Max A. Woodbury. *Inverting modified matrices.* Princeton University, Princeton, N. J., 1950. Statistical Research Group, Memo. Rep. no. 42,.

[Woo14] David P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1–2):1–157, 2014.

[WP11] Oren Weimann and David Peleg. A note on exact distance labeling. *Information processing letters*, 111(14):671–673, 2011.

[WRDR20] Zhihui Wang, Nicholas C Rubin, Jason M Dominy, and Eleanor G Rieffel. xy mixers: Analytical and numerical results for the quantum alternating operator ansatz. *Physical Review A*, 101(1):012320, 2020.

[WSC22] Keru Wu, Scott Schmidler, and Yuansi Chen. Minimax mixing time of the Metropolis-adjusted Langevin algorithm for log-concave sampling. *Journal of Machine Learning Research*, 23(270):1–63, 2022.

[WSJ22] Guoming Wang, Sukin Sim, and Peter D Johnson. State preparation boosters for early fault-tolerant quantum computation. *arXiv preprint arXiv:2202.06978*, 2022.

[WTFX07] Raymond Chi-Wing Wong, Yufei Tao, Ada Wai-Chee Fu, and Xiaokui Xiao. On efficient spatial matching. In *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB 2007)*, pages 579–590, 2007.

[WW20] Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious lwe sampling. *IACR Cryptol. ePrint Arch*, 2020:1042, 2020.

[WY14] Ryan Williams and Huacheng Yu. Finding orthogonal vectors in discrete structures. In *Proceedings of the 25th annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2014)*, pages 1867–1877. SIAM, 2014.

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under lwe. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 600–611. IEEE, 2017.

[WZ20] David P Woodruff and Amir Zandieh. Near input sparsity time kernel embeddings via adaptive sampling. In *ICML*, 2020.

[WZL$^+$22] Zongqi Wan, Zhijie Zhang, Tongyang Li, Jialin Zhang, and Xiaoming Sun. Quantum multi-armed bandits and stochastic linear bandits enjoy logarithmic regrets, 2022. arXiv:2205.14988.

[XLG03] Wei Xu, Xin Liu, and Yihong Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 267–273, 2003.

[XSS21] Zhaozhuo Xu, Zhao Song, and Anshumali Shrivastava. Breaking the linear iteration cost barrier for some well-known conditional gradient methods using maxip data-structures. *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021.

[YAG12] Man-Hong Yung and Alán Aspuru-Guzik. A quantum–quantum Metropolis algorithm. *Proceedings of the National Academy of Sciences*, 109(3):754–759, 2012. arXiv:1011.1468.

[Yao93] A Chi-Chih Yao. Quantum circuit complexity. In *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*, pages 352–361. IEEE, 1993.

[Ye20] Guanghao Ye. Fast algorithm for solving structured convex programs. Undergraduate thesis, University of Washington, 2020.

[YGL$^+$13] Xiaohui Yan, Jiafeng Guo, Shenghua Liu, Xueqi Cheng, and Yanfeng Wang. Learning topics in short texts by non-negative matrix factorization on term correlation matrix. In *proceedings of the SIAM International Conference on Data Mining (ICDM)*, pages 749–757. SIAM, 2013.

[YGS$^+$21] Zhewei Yao, Amir Gholami, Sheng Shen, Mustafa Mustafa, Kurt Keutzer, and Michael Mahoney. Adahessian: An adaptive second order optimizer for machine learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):10665–10673, May 2021.

[YHD$^+$12] Zhirong Yang, Tele Hao, Onur Dikmen, Xi Chen, and Erkki Oja. Clustering by nonnegative matrix factorization using graph random walk. *Advances in Neural Information Processing Systems (NeurIPS)*, 25:1079–1087, 2012.

[YN76] David B Yudin and Arkadi S Nemirovski. Evaluation of the information complexity of mathematical programming problems. *Ekonomika i Matematicheskie Metody*, 12:128–142, 1976.

[YTF$^+$19] Alp Yurtsever, Joel A. Tropp, Olivier Fercoq, Madeleine Udell, and Volkan Cevher. Scalable semidefinite programming, 2019.

[YZT23] Changhao Yi, Cunlu Zhou, and Jun Takahashi. Quantum phase estimation by compressed sensing. *arXiv preprint arXiv:2306.07008*, 2023.

[Zal98] Christof Zalka. Simulating quantum systems on a quantum computer. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 454(1969):313–322, 1998. arXiv:quant-ph/9603026.

[ZCDLP18] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

[ZCZG18] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. Stochastic gradient descent optimizes over-parameterized deep relu networks. In *arXiv preprint*. `https://arxiv.org/pdf/1811.08888`, 2018.

[ZCZG20] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. Gradient descent optimizes over-parameterized deep relu networks. *Machine learning*, 109(3):467–492, 2020.

[ZDZ$^+$21] Zhihui Zhu, Tianyu Ding, Jinxin Zhou, Xiao Li, Chong You, Jeremias Sulam, and Qing Qu. A geometric analysis of neural collapse with unconstrained features. *Advances in Neural Information Processing Systems*, 34:29820–29834, 2021.

[ZFRK10] Ruicong Zhi, Markus Flierl, Qiuqi Ruan, and W Bastiaan Kleijn. Graph-preserving sparse nonnegative matrix factorization with application to facial expression recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(1):38–52, 2010.

[ZG19] Difan Zou and Quanquan Gu. An improved analysis of training over-parameterized deep neural networks. In *NeurIPS*, pages 2053–2062, 2019.

[ZGJ$^+$18] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 159–172, 2018.

[Zha12] Mark Zhandry. How to construct quantum random functions. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 679–687. IEEE, 2012.

[Zha19] Mark Zhandry. Quantum lightning never strikes the same state twice. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 408–438. Springer, 2019.

[Zha20] Mark Zhandry. Schrödinger's pirate: How to trace a quantum decoder. Cryptology ePrint Archive, Report 2020/1191, 2020. `https://eprint.iacr.org/2020/1191`.

[ZLDZ07] Zhongyuan Zhang, Tao Li, Chris Ding, and Xiangsun Zhang. Binary matrix factorization with applications. In *Seventh IEEE international conference on data mining (ICDM)*, pages 391–400. IEEE, 2007.

[ZLL21] Chenyi Zhang, Jiaqi Leng, and Tongyang Li. Quantum algorithms for escaping from saddle points. *Quantum*, 5:529, 2021. arXiv:2007.10253.

[ZLLL18] Zhihui Zhu, Xiao Li, Kai Liu, and Qiuwei Li. Dropping symmetry for fast symmetric nonnegative matrix factorization. *Advances in Neural Information Processing Systems (NeurIPS)*, 31:5154–5164, 2018.

[ZMG19] Guodong Zhang, James Martens, and Roger B Grosse. Fast convergence of natural gradient descent for over-parameterized neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[Zou17] Qing Zou. The q-binomial inverse formula and a recurrence relation for the q-catalan–qi numbers. *J. Math. Anal*, 8(1):176–182, 2017.

[ZPD$^+$20] Yi Zhang, Orestis Plevrakis, Simon S Du, Xingguo Li, Zhao Song, and Sanjeev Arora. Over-parameterized adversarial training: An analysis overcoming the curse of dimensionality. In *NeurIPS*. arXiv preprint arXiv:2002.06668, 2020.

[ZS05] Ron Zass and Amnon Shashua. A unifying approach to hard and probabilistic clustering. In *Tenth IEEE International Conference on Computer Vision (ICCV)*, pages 294–301. IEEE, 2005.

[ZSM$^+$93] Zelda B. Zabinsky, Robert L. Smith, J. Fred McDonald, H. Edwin Romeijn, and David E. Kaufman. Improving hit-and-run for global optimization. *Journal of Global Optimization*, 3(2):171–192, 1993.

[Zve97] IE Zverovich. An analogue of the whithey theorem for edge graphs of multigraphs, and edge multigraphs. *Discrete Mathematics and Applications*, 7(3):287–294, 1997.

[ZWA13] Zhong-Yuan Zhang, Yong Wang, and Yong-Yeol Ahn. Overlapping community detection in complex networks using symmetric binary matrix factorization. *Physical Review E*, 87(6):062803, 2013.

[ZWD$^+$20a] Han-Sen Zhong, Hui Wang, Yu-Hao Deng, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Jian Qin, Dian Wu, Xing Ding, Yi Hu, et al. Quantum computational advantage using photons. *Science*, 370(6523):1460–1463, 2020.

[ZWD$^+$20b] Han-Sen Zhong, Hui Wang, Yu-Hao Deng, Ming-Cheng Chen, and Li-Chao Peng et al. Quantum computational advantage using photons. *Science*, 370(6523):1460–1463, 2020. arXiv:2012.01625.

[ZWJ22] Ruizhe Zhang, Guoming Wang, and Peter Johnson. Computing ground state properties with early fault-tolerant quantum computers. *Quantum*, 6:761, 2022.

[Zyg02] Antoni Zygmund. *Trigonometric series*, volume 1. Cambridge university press, 2002.

Vita

Ruizhe Zhang was born in Shandong, China, in April 1996. He received his Bachelor of Science degree in Computer Science from Fudan University in 2018. He was admitted to the Department of Computer Science at the University of Texas at Austin in August 2018, where he began his graduate studies. Ruizhe's primary research interests lie in theoretical computer science, quantum computing, and machine learning. He received the University Graduate Continuing Fellowship in 2022.

Address: 2501 Lake Austin Blvd
Austin, TX 78703

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.