FAST AND MEMORY EFFICIENT ALGORITHMS FOR STRUCTURED MATRIX SPECTRUM APPROXIMATION

by
Aditya Krishnan

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy

Baltimore, Maryland
September, 2022

Abstract

Approximating the singular values or eigenvalues of a matrix, i.e. spectrum approximation, is a fundamental task in data science and machine learning applications. While approximation of the top singular values has received considerable attention in numerical linear algebra, provably efficient algorithms for other spectrum approximation tasks such as spectral-sum estimation and spectrum density estimation are starting to emerge only recently. Two crucial components that have enabled efficient algorithms for spectrum approximation are access to randomness and structure in the underlying matrix. In this thesis, we study how randomization *and* the underlying structure of the matrix can be exploited to design fast and memory efficient algorithms for spectral sum-estimation and spectrum density estimation. In particular, we look at two classes of structure: *sparsity* and *graph structure*.

In the first part of this thesis, we show that sparsity can be exploited to give low-memory algorithms for spectral summarization tasks such as approximating some Schatten norms, the Estrada index and the logarithm of the determinant (log-det) of a sparse matrix. Surprisingly, we show that the space complexity of our algorithms are independent of the underlying dimension of the matrix. Complimenting our result for sparse matrices, we show that matrices that satisfy a certain smooth definition of sparsity, but potentially dense in the conventional sense, can be approximated in spectral-norm error by a truly sparse matrix. Our method is based on a simple sampling scheme that can be implemented in linear time. In the second part, we give the first truly sublinear time algorithm to approximate the spectral

density of the (normalized) adjacency matrix of an undirected, unweighted graph in earth-mover distance. In addition to our sublinear time result, we give theoretical guarantees for a variant of the widely-used Kernel Polynomial Method and propose a new moment matching based method for spectrum density estimation of Hermitian matrices.

First reader: Dr. Vladimir Braverman

Associate Professor

Department of Computer Science

Johns Hopkins University

Second reader: Dr. Michael Dinitz

Associate Professor

Department of Computer Science

Johns Hopkins University

Third reader: Dr. Edinah Koffi Gnang

Assistant Professor

Department of Applied Math and Statistics

Johns Hopkins University

Acknowledgements

First, I would like to thank my advisor Vladimir Braverman for giving me the freedom to work on many interesting research problems these past four years along with the opportunity to collaborate with incredible researchers.

I also want to thank my thesis readers – Michael Dinitz and Edinah Koffi Gnang. Mike has been an unofficial, secondary advisor during my time at Hopkins and constantly provided guidance on research and academic life in theoretical computer science. Edinah has been a pleasure to know on campus – I have greatly enjoyed discussing interesting linear algebra problems with him and very grateful for the energy and sincerity he brings to our discussions.

Outside of Hopkins, I owe many thanks to my collaborators Robert Krauthgamer, Roi Sinoff, Shay Sapir, Christopher Musco, Jingfeng Wu, Haoran Li, Soheil Kolouri and Edo Liberty. Robi has been one my research mentors throughout my PhD and been an incredibly inspiring collaborator – I owe many thanks to him for enabling a lot of the research included in this thesis. I am very grateful to have been able to collaborate with Shay on research and call him my friend – I have always been awed at the creativity he brings to solving problems and the patience and kindness he brings to research meetings. Chris has been an incredible research mentor to me over the last few years. I am very grateful he chose to believe in me and collaborate on a large, complicated project that took two years to complete. Chris's research interests and perspective on research problems has helped me develop an eye for interesting problems and his writing and talks have helped me greatly in developing

my technical communication skills. I also owe many thanks to Edo for being an incredible professional mentor, especially towards the end of my PhD. I am grateful for the patience and creativity he has brought to our collaboration and am excited to keep collaborating with him in the future.

I spent two of the four years of this journey during the COVID-19 pandemic. While I am incredibly grateful that I was not among the worst affected, the pandemic did take a significant toll. My friends and community here at Hopkins have been an incredible support system during this time. I am thankful for everyone who have been supportive during this time, especially – Ama, Aditya, Gabby, Alishah, Aarushi, Arka, Pratyush, Logan, Farnaz, Sepehr, and Ravi. I am incredibly grateful for their fun, levity and inclusivity. I am also thankful to have had an incredible network of friends and mentors whom I met early during my time at Hopkins – Jalaj, Nikita, Yasamin, Enayat, Niharika and Arun.

None of this would have been possible without the support of my incredible family - my loving and encouraging parents, my sister Anju and Vinay and Chetana, who have been like my elder siblings. They have supported me throughout this process and helped me have perspective on what is important.

Finally, I express immense gratitude to my loving partner Noshin Nova. Words cannot describe how important you are in my life – I am forever grateful that you have stood by by me steadfastly these last four years.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Numerical linear algebra is a core part of machine learning and data science. With the rise of big-data and the need for large scale machine learning, traditional methods to do numerical linear algebra are becoming increasingly inefficient and sometimes even prohibitive for most applications. This has led to a flurry of work in designing efficient algorithms – both in terms of memory and running time – for numerical linear algebra that are highly efficient but produce approximate solutions; trading-off accuracy for performance.

Randomization has emerged as a critical tool in developing these new methods. A classic example of this are sketching methods [50, 158] that have been used for dimensionality reduction, leading to fast algorithms for linear regression [34, 54, 110], clustering [38, 56], low-rank approximation [1, 34, 58], singular value decomposition [73] etc. Broadly, access to randomization enables us to develop algorithmic tools that lead to provably faster, lower-memory and smaller communication methods than their deterministic counterparts. In fact, the introduction of randomized techniques for numerical linear algebra has led to a field titled 'Randomized Numerical Linear Algebra' (RandNLA).

An important tool derived from randomization is *sampling*. Sampling methods are ubiquitous in algorithm design and numerical linear algebra specifically. Some examples include speeding up kernel regression [12, 116], seeding clustering algorithms [149], tracking statistics of a data stream [7, 32] and stochastic optimization algorithms such as stochastic gradient descent [16, 24, 161]; a crucial tool in enabling the training of large-scale neural networks.

In addition to randomization, structure in the underlying matrix can be greatly beneficial in designing efficient algorithms for numerical linear algebra. For instance, there exist a variety of fast algorithms for numerical linear algebra involving adjacency matrices or Laplacians of graphs like linear system solving [37, 140], cut and flow problems [128], sparsification [93] and spectrum approximation [42]. Similarly, fast iterative methods and efficient sketching methods have been proposed for a variety of problems when the underlying matrix is low rank; such as projection-cost preserving sketches for clustering [38, 117], block Krylov methods for principal component analysis [4, 60, 115] and oblivious subspace embeddings for matrices with quickly decaying spectra [36, 38]. Another example of structure in the underlying matrix is sparsity. A variety of sketching methods for numerical linear algebra have been proposed that can be applied in time proportional to the number of non-zero entries of the matrix, titled "input-sparsity time" sketching methods [34, 110].

In this thesis we will explore three problems in numerical linear algebra for which we design data-dependent sampling algorithms that exploit the underlying structure of the data to obtain provably faster and lower-memory algorithms than previous approaches.

1.1 Sampling for Spectrum Approximation of Structured Matrices

Many tasks in data science and machine learning require extracting information about the eigenvalues or singular values of a matrix [15, 49, 57, 63, 109, 130, 144]. Such tasks are broadly titled *spectrum approximation* and refer to an umbrella of problems that have motivated research at the intersection of numerical linear algebra and algorithm design.

Matrices that appear in applications of these tasks are highly structured [49, 57, 145] and so it is important to understand whether efficient algorithms can be designed that exploit this structure. In this thesis, we study how we can exploit the underlying structure of the matrix to design fast and memory-efficient algorithms for tasks in spectrum approximation. Our main approach is based on random sampling of the rows, columns and entries of the underlying matrix.

We briefly discuss spectrum approximation in the context of this thesis, the specific kinds of structured matrices we consider, and how we use sampling as a tool and its connection to efficient algorithm design in numerical linear algebra.

1.1.1 Spectrum Approximation

The task of spectrum approximation is to obtain an approximation to the n singular values of an $n \times n$ matrix, also known as the *spectrum*. In some contexts, such as is described in Chapter 4, the matrix is symmetric and the eigenvalues of the matrix are considered instead. Computing all the n singular values exactly can take $\Theta(n^\omega)$ time[1] for $\omega \approx 2.37286$

[1]The running time complexity here is the complexity of doing fast matrix multiplication, i.e., $O(n^\omega) = O(n^{2.37268})$.

and can require storing the entire matrix in working memory when traditional methods for singular value decomposition are used [14, 126]. The running time complexity and memory requirements of these methods can be prohibitively large even for a moderately sized matrix; for e.g., the matrices that appear in Ghorbani et al. [63] have $n \geq 10^5$ dimensions! Hence we often seek approximations to the spectrum that can be computed more efficiently but without losing critical information about it.

The most notable line of research on spectrum approximation considers approximating a small number of the largest magnitude singular values. This nature of approximation to the spectrum has been studied extensively and several methods such as partial SVD [73], rank-revealing QR factorization [46], randomized block Krylov methods [115, 134] and other iterative methods under the umbrella of principal component analysis and low-rank approximation have been proposed [3, 66, 136]. Many of these methods are nearly-linear time and optimized versions of these methods have been implemented in many machine learning and numerical linear algebra libraries [69, 134].

Instead of obtaining accurate approximations to a small number of the largest-magnitude eigenvalues, in this thesis we study methods to approximate the entire spectrum but only coarsely. The notions of approximation we consider capture information about the whole spectrum but potentially reveal less information about any specific singular value.

Concretely, we look at i) approximating spectral sums in Chapter 2 and, ii) approximation under earth-mover's distance in Chapter 4. While both of these have been studied less extensively than top singular value computation, designing fast and memory efficent algorithms for these spectrum approximation tasks has many applications in machine learning [49, 63, 109, 130] and scientific computing [15, 57, 157] and has recently received more attention in the literature [31, 48, 122, 145, 148, 162]. We discuss specific applications and the significance of our results within the respective chapters of each result.

1.1.2 Structure

There is a long history of research dedicated to designing more efficient algorithms for matrices that exhibit some underlying structure. For example, fast solving of linear systems of Laplacians [140], approximation of PSD matrices by sampling [52, 118] and sketching methods such as subspace embeddings [34, 110] and projection cost-preserving sketches [117] that exploit sparsity in the input matrix. In this thesis, we look at two notions of structure:

1. *Sparsity.* While real-world datasets are often enormous and exist in high dimensions, often most of the relevant information required to analyze the data is concentrated in a small fraction of the data points and dimensions. This structure in the data is often referred to as *sparsity*. Several algorithms have been proposed for tasks in numerical linear algebra that exploit sparsity in data such as speeding up dimensionality reduction [110], compressed sensing [26], low-rank approximation [40], regression [34], clustering [38], solving sparse linear systems [129], computation on sparse graphs etc. Sparsity can be viewed as a notion of intrinsic dimension of the data. The algorithmic challenge then is – how do we design algorithms to do linear algebra such that the complexity (both space complexity and running time complexity) of the algorithm depends only on this intrinsic dimension of the data and not on, the much larger, input dimension? We study this question in Chapter 2 and show several improvements for sparse matrices over dense matrices in a variety of spectrum approximation tasks.

   While some data is truly sparse, i.e., very few entries contain non-zero values, often data collected in real-world settings is noisy. In particular, the data is often separated into two components – signal and noise. The signal is often sparse and contains important information that needs to be analyzed or extracted, whereas, the noise is dense, unimportant and needs to be filtered out. This phenomenon is ubiquitous in machine

learning and numerical linear algebra and motivates many areas of research [26, 160].

In Chapter 3 we show how to efficiently filter out this noise, under certain structural assumptions, while retaining information about the spectrum of the original matrix.

2. *Graph Structure.* Graphs appear in many places in data science, several fields of research are dedicated to studying graph structured inputs – network science [13, 49], graph based neural networks [163], areas of computational physics and chemistry [71, 72, 157], computational geometry [29] etc. There has been a flurry of research in the last decade or so on faster numerical linear algebra for graphs, such as solving systems of Laplacians [129], spectral clustering [153, 156], metric embeddings and computing flows [128]. Often these faster methods crucially use the fact that the underlying data is graph structured; for example the Laplacian of a graph is symmetric and positive semidefinite (PSD). In Chapter 4, we exploit the underlying structure of Laplacians of graphs to obtain fast algorithms to approximate the spectrum of these matrices.

1.1.3 Sampling as a Tool

Sampling has been a core tool in designing algorithms for efficient numerical linear algebra. Random sampling is often used in the context of *matrix sketching*, wherein a small number of rows, columns or entries of the matrix are randomly selected in order to "compress" the matrix without losing critical information required to solve the problem. This approach has been used to give faster algorithms for a variety of problems including low-rank approximation, clustering, linear regression and spectral approximation. Another set of techniques based on sampling that has emerged in designing efficient algorithms for matrices is in the context of *stochastic iterative algorithms*. In particular, these techniques randomly select a small number of the rows, columns or entries of the matrix within some iterative computa-

tion involving the entire matrix, so as to speed-up computation or obtain memory savings. This approach has been been used to give faster algorithms for solving linear systems of matrices, approximate singular value decomposition, kernel regression, k-means clustering, principal component regression and been used extensively in the matrix streaming literature, leading to memory-efficient algorithms for low-rank approximation, estimating spectral-sums of matrix functions and many other problems.

The main algorithmic challenge in using random sampling is designing sampling distributions that are efficient to sample from, require few samples *and* lead to accurate solutions. This trade-off becomes especially pronounced since naive approaches such as sampling data uniformly can lead to (provably) bad approximations. We will see in this thesis, instances of sampling algorithms used in the context of matrix sketching in Chapter 3 and in the context of improving iterative alogrithms in Chapter 2 and 4.

1.2 Thesis Outline

This thesis is split into three technical chapters, each looking at a specific spectrum approximation problem. In Chapter 2 and Chapter 3 we consider matrices that have sparse structure and in Chapter 3 look at adjacency and Laplacian matrices of graphs.

We start by outlining our main results and give an overview of each chapter. The details of the problem definition and computational models have been deferred to the respective chapter of each problem.

In Chapter 2, we give low-memory algorithms for estimating spectral-sums of sparse matrices[2] for some important class of spectral-sums, including the Schatten p-norms for even

[2] A matrix with a constant number of non-zero entries in every row and column.

values of $p \geq 2$, the Estrada index and the logarithm of the determinant (log-det) of the matrix. We give a sampling algorithm that has space complexity completely independent of the matrix dimension n and only dependent on the sparsity of rows and columns. In particular, for a sparse matrix, our algorithm requires only a *constant* number of words of memory to accurately estimate these spectral-sums. These results improve over previous work in some computation models by giving the first dimension-independent memory bound for estimating these spectral-sums. Our algorithm is based on sampling rows from a carefully defined distribution in an iterative way; trading-off the space complexity of storing the sampled rows and the information revealed by them to estimate the spectral-sum.

Next, we show in Chapter 3 that matrices with rows and columns that have few heavy entries and (potentially) many entries with small magnitude can be well-approximated by a sparse matrix. In particular, we consider a generalization of the notion of sparsity of the rows and columns of the matrix, titled *numerical sparsity*; defined as the ratio between the ℓ_1-norm and ℓ_2-norm of the row (or column). We show that in fact a matrix that has numerically sparse rows and columns can be well approximated, under spectral-norm error, by a sparse matrix. We provide a simple sampling algorithm achieving this guarantee, running in linear time[3] that samples entries of the matrix from a carefully chosen distribution.

Finally, in Chapter 4 we consider spectral density estimation for any $n \times n$ normalized graph adjacency or Laplacian matrix of an undirected, unweighted graph. We show a $n \cdot \text{poly}(1/\epsilon)$ time algorithm for such matrices to obtain ϵ accuracy in Wasserstein-1 distance (earth-mover's distance). This running time is sublinear in the number of edges in the graph and the number of entries of the corresponding matrix. In fact, our method works more generally for all symmetric matrices with $O(1/\epsilon)$ matrix-vector multiplications with the matrix. It is based on a novel moment matching method that employs randomization

[3]When a rough estimate of the spectral-norm of the matrix is known.

to estimate the traces involving the matrix. The sublinear time result for graphs follows by combining the method for symmetric matrices along with random sampling of columns to avoid full matrix-vector multiplications. In addition to our moment matching method, we analyze a variant of the widely used kernel polynomial method and achieve matching guarantees.

1.3 Notation

Most notation is defined as needed in each chapter, but some notation is used throughout. $\mathbb{R}$ is used to denote the reals, $\mathbb{R}^n$ denotes the set of n-length vectors with real entries and $\mathbb{R}^{n\times d}$ denotes the set of $n \times d$ matrices with real entries. We use uppercase letters to denote matrices and lowercase letters to denote vectors and scalars. We let x_i denote the i-th entry of the vector x and A_{ij} denote the entry in the i-th row and j-th column of the matrix A. $\text{nnz}(A)$ denotes the number of non-zero entries in A. Where it is relevant we define the *sparsity* of a vector to be the number of non-zero entries and for matrices, define the sparsity to be the maximum number of non-zero entries in any of its columns or rows.

We let $^\top$ denote the transpose of a matrix or vector and denote the $n\times n$ identity matrix by I_n. For a vector $x \in \mathbb{R}^n$ and parameter $p > 0$ we define $\|x\|_p$ to be $\|x\|_p = (\sum_{i=1}^n |x_i|^p)^{1/p}$. We define $\|x\|_\infty$ to be equal to $\max_{i\in[n]} |x_i|$ and similarly $\|x\|_0$ denotes the number of non-zero entries in x.

For an $m \times n$ matrix A with $m \geq n$ we denote the n singular values of A by $\sigma_1(A) \geq \cdots \geq \sigma_n(A) \geq 0$. When it is clear from context, we omit (A). Similarly, when the matrix A is square and symmetric, we denote the n eigenvalues of the matrix by $\lambda_1(A) \geq \cdots \geq \lambda_n(A)$.

We let $\|A\|_F = \sqrt{\sum_{i,j} A_{ij}^2}$ be the Frobenius norm of the matrix A and when the matrix

is square, define $\mathrm{tr}(A) = \sum_i A_{ii}$ to be the trace of A. Moreover, we define $\|A\|_2$ to be the spectral norm and is given by

$$\|A\|_2 = \max_{\|x\|_2=1} \|Ax\|_2.$$

We often write $\tilde{O}(f)$ as a shorthand for $O(f \cdot \log^{O(1)}(n))$ where n is the dimension of the matrix being considered, and write $O_d(f)$ when the hidden constant might depend on the parameter d. $[t]$ denotes the set $\{1, \ldots, n\}$ for a parameter t.

Chapter 2

Spectral Sum Estimation for Sparse Matrices

2.1 Introduction

In several application domains, input matrices are often very sparse, meaning that only a small fraction of their entries are non-zero. In fact, in applications related to natural language processing (e.g. [61]), image recognition, medical imaging and computer vision (e.g. [67, 105]), the matrices are often doubly sparse, i.e., sparse in both rows and columns. Throughout, we define these matrices as *k-sparse*, meaning that every row and every column has at most k non-zero entries. The current work devises new algorithms to analyze the *spectrum* (singular values) of such sparse matrices, aiming to achieve efficiency (storage requirement in streaming model) that depends on *matrix sparsity* instead of *matrix dimension*.

We focus on fundamental functions of the spectrum, called the Schatten norms. Formally, the Schatten p-norm of a matrix $A \in \mathbb{R}^{m \times n}, m \geq n$ with singular values $\sigma_1 \geq \ldots \geq$

$\sigma_n \geq 0$ is defined for every $p \geq 1$ as

$$\|A\|_{S_p} := \left(\sum_{i=1}^{n} \sigma_i^p \right)^{1/p}.$$

This definition extends also to $0 < p < 1$, in which case it is not a norm, and also to $p = 0, \infty$ by taking the limit. Frequently used cases include $p = 0$, representing the rank of A, and $p = 1, 2, \infty$, commonly known as the trace/nuclear norm, Frobenius norm, and spectral/operator norm, respectively. Schatten norms are often used as surrogates for the spectrum, as explained in [90, 91, 123, 167], and some specific cases have applications in optimization, image processing, and differential privacy etc. [120, 164].

For a positive semidefinite (PSD) matrix A, the Schatten norms can be easily used to approximate other important spectral functions. One example is the Estrada index, which has applications in chemistry, physics, network theory and information theory (see survey by Gutman et al. [72]). Another example is the trace of matrix inverse, which is used for image restoration, counting triangles in graphs, to measure the uncertainty in data collections, and to bound the total variance of unbiased estimators (see e.g. [19, 30, 75, 162] for references). A third example is the logarithm of the determinant, used in many machine learning tasks, like Bayesian learning, kernel learning, Gaussian processes, tree mixture models, spatial statistics and Markov field models (see e.g. [74, 75, 146, 147] for references). Thus, our results for Schatten norm have further applications.

As the matrices in many real-world applications are often very large, storing the entire matrices in working memory can be impractical, and thus, as mentioned, analyzing them has become increasingly challenging. As a result, the data-stream model has emerged as a convenient model for how these data-sets are accessed in practice. In this model, the input matrix $A \in \mathbb{R}^{m \times n}$ is presented as a sequence of items/updates. In one common setting,

the *turnstile* model, each update has the form (i, j, δ) for $\delta \in \mathbb{Z}$ and represents an update $A_{ij} \leftarrow A_{ij} + \delta$. In another common setting, the *row-order* model, items (i, j, A_{ij}) arrive in a fixed order, sorted by location (i, j) lexicographically, providing directly the entry of A in that location. In both models, unspecified entries are 0 by convention, which is very effective for sparse matrices.

Row-order is a common access pattern for external memory algorithms. When the data is too large to fit into working memory and has be "streamed" into memory in some pattern, it is useful to assume that algorithms can make multiple, albeit few, passes over the input data. For a thorough discussion of such external memory algorithms, including motivation for the row-order model and for multiple passes over the data, see [64, 103, 152].

Designing small-space algorithms for estimating Schatten norms of an input matrix in the data-stream model is an important problem, and was investigated recently for various matrix classes and stream types [6, 23, 32, 99, 100, 101, 102]. However, all known algorithms require space that is polynomial in n, the matrix dimension, even if the matrix is highly sparse and the stream type is favorable, say row-order. A natural question then is:

> Does any streaming model admit algorithms for computing Schatten norms of a matrix presented as a stream, with storage requirement independent of the matrix dimension?

We *answer this question in the affirmative* for k-sparse matrices presented in row-order and all even integers p. Our algorithms extend to all integers $p \geq 1$ if the input matrix is PSD.

2.1.1 Main Results

Throughout, we write $\tilde{O}(f)$ as a shorthand for $O(f \cdot \log^{O(1)} n)$ where n is the dimension of the matrix, and write $O_d(f)$ when the hidden constant might depend on the parameter d. We assume that the entries of the matrix are integers bounded by $poly(n)$, and thus often count space in words, each having $O(\log n)$ bits. We denote by $\lceil p \rceil_4$ the smallest multiple of 4 that is greater than or equal to p, and similarly by $\lfloor p \rfloor_4$ the largest multiple of 4 that is smaller than or equal to p.

Upper and Lower Bounds for Row-Order Streams. Our main result is a new algorithm for approximating the Schatten p-norm (for even p) of a k-sparse matrix streamed in row-order, using $O(p)$ passes and $poly(k^p/\epsilon)$ space (independent of the matrix dimensions). This is stated in the next theorem, whose proof appears in Section 2.4.1.

Theorem 2.1.1. *There exists an algorithm that, given $p \in 2\mathbb{Z}_{\geq 2}$, $\epsilon > 0$ and a k-sparse matrix $A \in \mathbb{R}^{n \times n}$ streamed in row-order, makes $\lfloor p/4 \rfloor + 1$ passes over the stream using $O_p(\epsilon^{-2} k^{3p/2-3})$ words of space, and outputs $\bar{Y}(A)$ that $(1 \pm \epsilon)$-approximates $\|A\|_{S_p}^p$ with probability at least $2/3$.*

Theorem 2.1.1 provides a multi-pass algorithm whose space complexity depends only on the sparsity of the input matrix. A natural question is whether one can achieve a similar dependence also for *one-pass* algorithms, but our next theorem (proved in Section 2.6) shows that such algorithms require $\Omega((n^{1-4/\lfloor p \rfloor_4})$ bits of space, even for $O(1)$-sparse matrices.

It follows that multiple passes over the data are necessary for an algorithm for sparse matrices to have space complexity independent of the matrix dimensions.

Theorem 2.1.2. *For every $p \in 2\mathbb{Z}_{\geq 2}$ there is $\epsilon(p) > 0$ such that every algorithm that makes one pass over an $O_p(1)$-sparse matrix $A \in \mathbb{R}^{n \times n}$ streamed in row-order, and then outputs a*

Table 2.1: Bounds for Schatten norms (for even p) of k-sparse matrices in row-order streams. Upper bound space is counted in words. Lower bounds are for suitable $\epsilon(p) > 0$.

Which p	**Space Bound**	**Ref.**	**Comments**
$p > 4$	$\tilde{O}_{p,\epsilon}(k^{O(p)} n^{1-4/\lceil p \rceil_4})$	[23]	one-pass
	$O_{p,\epsilon}(k^{3p/2-3})$	Thm. 2.1.1	$\lfloor p/4 \rfloor + 1$ passes
	$O_{p,\epsilon}(k^{2ps} n^{1-1/s})$	Thm. 2.5.3	$\lfloor \frac{p}{2(s+1)} \rfloor + 1$ passes
	$\Omega(n^{1-4/\lfloor p \rfloor_4})$	Thm. 2.1.2	one-pass, $k = O(1)$
	$\Omega_t(k^{p/2-2})$	[23]	t passes, $k \leq n^{2/p}$
$p = 4$	$\tilde{O}_{p,\epsilon}(k)$	[23]	one-pass
	$O_p(\varepsilon^{-2})$	Thm. 2.7.2	one-pass, for all $k \leq n$

$(1 \pm \epsilon(p))$-approximation to $\|A\|_{S_p}^p$ with probability at least $2/3$, must use $\Omega(n^{1-4/\lfloor p \rfloor_4})$ bits of space.

We can further extend our primary algorithmic technique (from Theorem 2.1.1) in several different ways, and obtain improved algorithms for special families of matrices, algorithms in the more general turnstile model, and algorithms with a trade-off between the number of passes and the space requirement, as explaind later in this section. Table 2.1 summarizes our results for row-order streams, and compares them to bounds derived from previous work (when applicable).

Applications for Approximating Schatten Norms. We show in Section 2.8 two settings where, under certain simplifying conditions, our algorithms can be used to approximate other functions of the spectrum, and even weakly recover the entire spectrum. The basic idea is that it suffices to compute only a few Schatten norms, in which case our algorithms for k-sparse matrices in row-order streams can be used, and the overall algorithm will require only small space (depending on k).

The first setting considers spectral sums of PSD matrices. We use an idea from [17] to show that for a PSD input $A \in \mathbb{R}^{n\times n}$ whose eigenvalues lie in an interval $[\theta, 1)$, one can $(1 \pm 2\epsilon)$-approximate $\log\det(A)$ and $\mathrm{tr}(A^{-1})$ using the first $\frac{1}{\theta}\log\left(\frac{1}{\epsilon}\right)$ (integer) Schatten norms. We further show that given a Laplacian matrix whose eigenvalues lie in an interval $[0, \theta]$, one can $(1 \pm 2\epsilon)$-approximate the Estrada index using the first $(e\theta + 1)\log\frac{1}{\varepsilon}$ (integer) Schatten norms.

The second setting considers recovering the spectrum of a PSD matrix using a few Schatten norms of the matrix. We use an idea from [91] to approximate the spectrum of a PSD matrix whose eigenvalues lie in the interval $[0, 1]$, up to L_1-distance ϵn using the first $O(1/\epsilon)$ Schatten norms.

Experiments. We validated our row-order algorithm on real-world matrices representing academic collaboration network graphs. The experiments show that the space needed to approximate the Schatten 6-norm of these matrices is much smaller than the theoretical bound, and that the algorithm is efficient also for larger p values. In fact, the matrices in our experiments have $O(1)$-sparse in every row, but their columns are only sparse on average. We also experimented to check if the algorithm is robust to noise, and found that it is indeed effective also for nearly-sparse matrices. Our experiments validate that the storage requirement is independent of the matrix dimensions. See Section 4.6 for details.

2.1.2 Extensions of Main Results

Extension I: Fewer Passes. We show in Section 2.5 how to generalize our algorithmic technique to use fewer passes over the stream, albeit requiring more space. Our method attains the following pass-space trade-off. For any integer $s \geq 2$, our algorithm in Theo-

rem 2.5.3 makes $t(s) = \lfloor \frac{p}{2(s+1)} \rfloor + 1$ passes over the stream using $O_p\left(\epsilon^{-3} k^{2ps} n^{1-1/s}\right)$ words of space, and outputs a $(1 \pm \epsilon)$-approximation to $\|A\|_{S_p}^p$ for $p \in 2\mathbb{Z}_{\geq 2}$.

Extension II: Turnstile Streams. We design in Section 2.4.2 an algorithm for *turnstile* streams with an additional $\tilde{O}(\epsilon^{-O(p)} k^{3p/2-3} n^{1-2/p})$ factor in their space complexity compared to our algorithm for row-order streams. An additional $O(n^{1-2/p})$ factor is to be expected since the space complexity for estimating ℓ_p norms of vectors in turnstile streams is $\Omega(\frac{n^{1-2/p}}{t})$ if the algorithm is allowed to make t passes over the data. Our algorithm for turnstile streams makes $p+1$ passes over the stream. The algorithm of [99] for $O(1)$-sparse matrices in the turnstile model can obviously be extended to k-sparse matrices. Its space requirement is $k^{O(p)}$, and we believe that a straightforward extension of their analysis yields an exponent greater than $4.75p$

Extension III: Special Matrix Families. We give in Section 2.4.1 improved bounds for special families of k-sparse matrices that may be of potential interest. We show that for Laplacians of undirected graphs with degree at most $k \in \mathbb{N}$, one can $(1 \pm \epsilon)$-approximate the Schatten p-norm with space $O_p(\epsilon^{-2} k^{p/2-1})$ by making $p/2$ passes over a row-order stream. Additionally, for matrices whose non-zero entries lie in an interval $[\alpha, \beta]$ for $\alpha, \beta \in \mathbb{R}^+$, we can get nearly-tight upper bounds – our algorithm uses space $O_p(\epsilon^{-2} k^{p/2-1} (\beta/\alpha)^{p/2-2})$, which is nearly tight compared to the $\Omega(k^{p/2-2})$ multi-pass lower bound given in [23] where $\alpha = \beta = 1$.

Schatten 4-norm. We show in Section 2.7 a simple one-pass algorithm for $(1 \pm \epsilon)$-approximating the Schatten 4-norm of *any* matrix (not necessarily sparse) given in a row-order stream, using only $\tilde{O}_p(\epsilon^{-2})$ words of space. This improves a previous $\tilde{O}_p(\epsilon^{-2} k)$ bound from [23].

2.1.3 Technical Overview

Upper Bounds. We design an estimator that is inspired by the importance sampling framework and uses multiple passes over the data to implement the estimator. To the best of the our knowledge, this is the first algorithm for computing the Schatten p-norm in data streams that uses an adaptive sampling approach, i.e. the probability space of the algorithm's sampling in a given pass of the data is affected by the algorithm's decisions in the previous pass.

For an integer $p \in 2\mathbb{Z}_{\geq 1}$ and $q \coloneqq p/2$, the Schatten p-norm for a matrix $A \in \mathbb{R}^{n\times n}$, denoted by $\|A\|_{S_p}^p$, can be expressed as

$$\|A\|_{S_p}^p = \mathrm{tr}((AA^\top)^q) = \sum_{i_1,\ldots,i_q\in[n]} \langle a_{i_1}, a_{i_2}\rangle\langle a_{i_2}, a_{i_3}\rangle \ldots \langle a_{i_q}, a_{i_1}\rangle \tag{2.1}$$

where a_i is the i^{th} row of matrix A.

The Schatten p-norm can be interpreted using (2.1) as a sum over cycles of q inner-products (which we refer to informally as *cycles*) between rows of A. We assign each cycle in the above expression to one of the rows participating in that cycle. Hence, the Schatten p-norm can be expressed as a sum $\sum_{i=1}^n z_i$ where z_i is the cumulative weight of all the cycles assigned to row i.

Our algorithm starts by sampling a row $i \in [n]$ with probability proportional to the *heaviest* cycle assigned to row i (i.e., largest contribution to z_i). In the following $p/4$ stages, it samples one cycle assigned to i with probability proportional to the weight of the cycle. Since the rows *and* columns are sparse, each row cannot participate in "too many" cycles (because it is orthogonal to any row with a disjoint support). Specifically, we show that the number of cycles assigned to each row i is only a function of k and p. It follows that

sampling the first row with probability proportional to the heaviest contributing cycle is a good approximation (up to a factor depending only on k and p) to z_i, the actual contribution of row i to $\sum_{i\in[n]} z_i = \|A\|_{S_p}^p$.

The space complexity of sampling a row with probability proportional to its heaviest contributing cycle depends on the assigning process. A natural assigning is to assign every cycle to the row with largest l_2-norm participating in that cycle (breaking ties arbitrarily). Notice then that, by the Cauchy-Schwarz inequality, the heaviest contributing cycle to row i is simply $\|a_i\|_2^p$.

This estimator can be implemented in the row-order model easily by using weighted reservoir sampling [21, 151], as shown in Section 2.4.1. However, implementing it in turnstile streams is more challenging (see Section 2.4.2). Using approximate L_p-samplers presented in [112], we build an approximate cascaded $L_{p,2}$-norm[1] sampler, to sample rows i with probability proportional to $\|a_i\|_2^p$. Additionally, we use the Count-Sketch data structure to recover rows and sample cycles once we have sampled the first, "seed" row. This allows us to implement the estimator in turnstile data streams with an additional $\tilde{O}(\epsilon^{-O(p)} n^{1-2/p})$ factor in the space complexity attributed to the approximate cascaded $L_{p,2}$-norm sampler and an additional $O_p(k^{3p/2-3})$ factor that comes from approximating the sampling probabilities (compared to the row-order in which the sampling probabilities can be recovered exactly).

In Section 2.5 we generalize the design of the importance sampling estimator. Instead of assigning every cycle to a single row that appears in it, every cycle is mapped to s rows that participate in it, for parameter $s \in \mathbb{N}$. These s rows split the cycle into roughly $\frac{q}{s}$ segments such that each of these s rows participates in a segment where it is the heaviest' row (by l_2-norm). The algorithm samples s "seed" rows and then computes all the cycles (or alternatively samples one cycle) that are assigned to these s rows. Since the length of

[1]The $L_{p,2}$-norm of a matrix $A \in \mathbb{R}^{n\times m}$ for $p \geq 0$ is $\left(\sum_{i=1}^n \|a_i\|_2^p\right)^{1/p}$.

each of the segments reduces linearly with s, one can compute these cycles with fewer passes. However, the algorithm needs to sample more indices in order to ensure that each cycle has a sufficiently large probability of being "hit". This tension leads to a trade-off between passes and space.

Lower Bounds. We obtain an $\Omega(n^{1-4/\lfloor p \rfloor_4})$ bits lower bound for any algorithm that estimates the Schatten p-norm in one-pass of the stream for even p values. Our proof analyzes for even p values a construction presented in [99], which is based on a reduction from the Boolean Hidden Hypermatching problem. This lower bound holds even if the input matrix is promised to be $O(1)$-sparse.

2.1.4 Previous and Related Work

The bilinear sketching algorithm in [102] was the first non-trivial algorithm for Schatten p-norm estimation in turnstile streams. It requires only one-pass over the data and uses $O(\epsilon^{-2}n^{2-4/p})$ words of space.[2] Their algorithm uses $O(\varepsilon^{-2})$ independent $G_1AG_2^\top$ sketches, where $G_1, G_2 \in \mathbb{R}^{t\times n}$ are matrices with i.i.d. Gaussian entries and $t = O(n^{1-2/p})$.

Inspired by this sketch, [23] gave an almost quadratic improvement in the space complexity if the algorithm is allowed to make multiple passes over the data. Their estimator uses matrices $G_2, \ldots, G_p \in \mathbb{R}^{t\times n}$ with i.i.d. Gaussian entries and Gaussian vector $g_1 \in \mathbb{R}^n$ to output $g_1^\top AG_2^\top G_2 A \ldots G_p A g_1$. This estimate can be constructed in $p/2$ passes of the data and requires $O(\varepsilon^{-2})$ independent copies each using only $t = O(n^{1-\frac{1}{p-1}})$ space.

Restricting the input matrix to be $O(1)$-sparse allows for quadratic improvement in the space complexity of one-pass algorithms as shown in [99]. They show that sampling

[2] They also showed a lower bound of $\Omega(n^{2-4/p})$ for the dimension of bilinear sketching for approximating $\|A\|_{S_p}^p$ for all $p \geq 2$.

$O(n^{1-2/p})$ rows and storing them approximately using small space (since each row is sparse) is sufficient to $(1+\epsilon)$-approximate the Schatten p-norm by exploiting the fact that rows cannot "interact" with one another "too much" because of the sparsity restriction.

If we restrict the data stream to be row-order, then we can reduce the dependence on p in all the above algorithms by a factor of 2. As noted in [23], since $A^\top A = \sum_i a_i a_i^\top$ (where a_i is the i^{th} row of A) one can apply the above algorithms to $A^\top A$ instead of A by updating it with the outer product of every row with itself. Since $\|A^\top A\|_{S_{p/2}}^{p/2} = \|A\|_{S_p}^p$ (for even p values), the output is as desired and the dependence on p reduces by a factor of 2.

Lower Bounds. Every t-pass algorithm designed for turnstile streams requires $\Omega(n^{1-2/p}/t)$ bits, which follows by injecting the F_p-moment problem (see [68, 85]) into the diagonal elements. Li and Woodruff [99] showed that restricting the algorithm to a single pass over the turnstile stream, leads to a lower bound $\Omega(n^{1-\varepsilon})$ bits for every fixed $\varepsilon > 0$ and $p \notin 2\mathbb{Z}_{\geq 2}$, even if the input matrix is $O(1)$-sparse.[3] Later [23] proved that $\Omega(n^{1-\varepsilon})$ bits are required for $p \notin 2\mathbb{Z}_{\geq 2}$ even in row-order streams. In addition, they showed (Theorem 5.4 in Arxiv version) that t passes over row-order streams require space $\Omega(n^{1-4/p}/t)$ bits, however these matrices are actually $\Omega(n^{2/p})$-sparse (and not $O(1)$-sparse as may be understood from Table 2 therein). A simple adaptation of that result yields an $\Omega(k^{p/2-2}/t)$ space lower bound for k-sparse input matrices ($k \leq n^{2/p}$).

2.2 Notation and Preliminaries

The following useful fact comparing the lengths of the rows of A and its Schatten p-norm is proved in Appendix A.1.

[3] They also showed that for $p \in 2\mathbb{Z}_{\geq 2}$, single-pass algorithms require $\Omega(n^{1-2/p})$ bits even if all non-zeros in the input matrix are constants.

Fact 2.2.1. *Let matrix $A \in \mathbb{R}^{n\times n}$ have rows $\{a_i\}_{i\in[n]}$ and let $t \geq 1$. Then $\sum_{i\in[n]} \|a_i\|_2^{2t} \leq \|A\|_{S_{2t}}^{2t}$.*

Importance Sampling. Our main algorithmic technique is inspired by the importance sampling framework, as formulated by the following theorem, proved in Appendix A.2.

Theorem 2.2.2 (Importance Sampling)**.** *Let $z = \sum_{i\in[n]} z_i \geq 0$ be a sum of n reals. Let the random variable $\hat{Z}$ be an estimator computed by sampling a single index $i \in [n]$ according to the probability distribution given by $\{\tau_i\}_{i=1}^n$ and setting $\hat{Z} \leftarrow \frac{z_i}{\tau_i}$. If for some parameter $\lambda \geq 1$, each $\tau_i \geq \frac{|z_i|}{\lambda \cdot z}$, then*

$$\mathbf{E}[\hat{Z}] = z \quad \textit{and} \quad Var(\hat{Z}) \leq (\lambda z)^2.$$

Families of Matrices. We define two families of matrices that are of special interest.

- Let $\mathcal{L}_n \subseteq \mathbb{Z}^{n\times n}$ be the family of Laplacian matrices of undirected graphs $G([n], E)$ with positive edge-weights $\{w_{uv} > 0 : uv \in E\}$.
- Given positive constants $\alpha \leq \beta$, let $\mathcal{C}_{\alpha,\beta}^{m\times n} \subseteq \mathbb{R}^{m\times n}$ be the family of matrices C such that every entry $C_{i,j}$ is either zero or in the range $[\alpha, \beta]$. For the vector case (i.e. $n = 1$) we may write $\mathcal{C}_{\alpha,\beta}^{m}$.

2.3 An Estimator for Schatten p-Norm for $p \in 2\mathbb{Z}_{\geq 2}$

This section introduces our importance sampling estimator for Schatten p-norms. We begin in Section 2.3.1 with manipulating expression (2.1) for the Schatten p-norm by assigning every summand, i.e. a cycle of $p/2$ inner products, to its heaviest participating row, see

(2.4). We then use this new expression in Section 2.3.2 to give an importance sampling estimator. In Section 2.3.3 we prove several lemmas, referred to as projection lemmas, which are key to our analysis in Section 2.3.4.

2.3.1 Preliminaries

Fix a matrix $A \in \mathbb{R}^{n\times n}$ and $p \in 2\mathbb{Z}_{\geq 2}$. For a row a_i, we define the set of its *neighboring rows* $N(i) \coloneqq \{l \in [n] : \mathrm{supp}(a_i) \cap \mathrm{supp}(a_l) \neq \emptyset\}$. In addition, we denote the set of neighboring rows of a_j that have smaller length than row a_i

$$N_S^i(j) \coloneqq \{l \in N(j) : \ \|a_l\|_2 \leq \|a_i\|_2\}.$$

Building on this, we intorduce notation for certain "paths" of rows. Fixing some row indices $i, i_1 \in [n]$ and an integer $t \geq 2$, we then define

$$\Gamma(i_1, t) \coloneqq \left\{(i_1, \ldots, i_t) : \ i_2 \in N(i_1), \ldots, i_t \in N(i_{t-1})\right\},$$
$$\Gamma_S^i(i_1, t) \coloneqq \left\{(i_1, \ldots, i_t) : \ i_2 \in N_S^i(i_1), \ldots, i_t \in N_S^i(i_{t-1})\right\}.$$

We further define the weights of "paths" of inner products: given an integer $t \geq 2$ and indices $i_1, \ldots, i_t \in [n]$, let

$$\sigma(i_1, \ldots, i_t) \coloneqq \langle a_{i_1}, a_{i_2}\rangle\langle a_{i_2}, a_{i_3}\rangle \ldots \langle a_{i_{t-1}}, a_{i_t}\rangle.$$

Recall from (2.1) that the Schatten p-norm of $A \in \mathbb{R}^{n\times n}$ can be expressed in terms of the product of inner products of the rows of A. Using the above notation we manipulate it

as follows.

$$\|A\|_{S_p}^p = \operatorname{tr}\left((AA^\top)^q\right) = \sum_{i_1,\ldots,i_q\in[n]} \sigma(i_1,\ldots,i_q,i_1) \tag{2.2}$$

$$= \sum_{i_1} \sum_{\substack{(i_1,\ldots,i_{q-1})\\ \in\Gamma(i_1,q-1)}} \sum_{i_q\in N(i_1)} \sigma(i_1,\ldots,i_q,i_1) \tag{2.3}$$

$$= \sum_{i_1} \sum_{\substack{(i_1,\ldots,i_{q-1})\\ \in\Gamma_S^{i_1}(i_1,q-1)}} \sum_{i_q\in N_S^{i_1}(i_1)} c(i_1,\ldots,i_q)\sigma(i_1,\ldots,i_q,i_1) \tag{2.4}$$

where $1 \le c(i_1,\ldots,i_q) \le q$ is the number of times the sequence $(i_1,\ldots,i_q,i_1)$ or a cyclic shift of the sequence appears in Equation (2.3).

2.3.2 The Estimator

Our estimator is an importance sampling estimator for the quantity in (2.4). To define it, we need the following quantities:

$$\mathcal{S} \coloneqq \bigcup_{i\in[n]} \Gamma_S^i(i,q-1)$$

$$z_{(i_1,\ldots,i_{q-1})} \coloneqq \sum_{i_q\in N_S^{i_1}(i_1)} c(i_1,\ldots,i_q)\sigma(i_1,\ldots,i_q)\langle a_{i_q}, a_{i_1}\rangle \qquad \forall (i_1,\ldots,i_{q-1})\in\mathcal{S}$$

$$z \coloneqq \sum_{(i_1,\ldots,i_{q-1})\in\mathcal{S}} z_{(i_1,\ldots,i_{q-1})} = \|A\|_{S_p}^p \qquad \text{by Equation (2.4).}$$

Our importance sampling estimator, for the sum z, samples quantities $z_{(i_1,\ldots,i_{q-1})}$ indexed by $(i_1,\ldots,i_{q-1})\in\mathcal{S}$ in $q-1$ steps. In the first step, it samples row $i_1\in[n]$ with probability $\frac{\|a_{i_1}\|_2^p}{\sum_j \|a_j\|_2^p}$. In each step $2 \le t \le q-1$, conditioned on sampling i_{t-1} in step $t-1$ it samples

row $i_t \in N_S^{i_1}(i_{t-1})$ with probability

$$p_{i_{t-1}}^{i_1}(i_t) \coloneqq \frac{|\langle a_{i_{t-1}}, a_{i_t}\rangle|}{\sum_{l \in N_S^{i_1}(i_{t-1})} |\langle a_{i_{t-1}}, a_l\rangle|}.$$

Overall, a sequence $(i_1, \ldots, i_{q-1}) \in \mathcal{S}$ is sampled with probability

$$\tau_{(i_1,\ldots,i_{q-1})} = \frac{\|a_{i_1}\|_2^p}{\sum_j \|a_j\|_2^p} \prod_{t=2}^{q-1} p_{i_{t-1}}^{i_1}(i_t),$$

and the output estimator is

$$Y(A) \coloneqq \frac{1}{\tau_{(i_1,\ldots,i_{q-1})}} \cdot z_{(i_1,\ldots,i_{q-1})}.$$

2.3.3 Projection Lemmas

To analyze the estimator $Y(A)$, we need a few lemmas, which we call projection lemmas, for sparse matrices. We start with two lemmas for sparse matrices, followed by two lemmas for more specialized cases.

Lemma 2.3.1. *For every k-sparse matrix $B \in \mathbb{R}^{n\times k}$ with rows $b_1, \ldots, b_n$ and vector $x \in \mathbb{R}^k$ such that $\|x\|_2 \geq \|b_i\|_2$ for all $i \in [n]$, we have that*

$$\frac{\|Bx\|_1}{\|x\|_2^2} = \sum_{i=1}^{n} \frac{|\langle x, b_i\rangle|}{\|x\|_2^2} \leq k\sqrt{k}.$$

Proof. For a vector $y \in \mathbb{R}^k$ and $S \subseteq [k]$, let $y_{|S}$ to be the restriction of y onto its indices corresponding to set S.

For all $i \in [n]$, by the Cauchy-Schwarz inequality, $\langle x, b_i\rangle = \langle x_{|\operatorname{supp}(b_i)}, b_i\rangle \leq \|x_{|\operatorname{supp}(b_i)}\|_2 \|b_i\|_2$.

Hence,

$$\sum_{i=1}^{n} \frac{|\langle x, b_i \rangle|}{\|x\|_2^2} \leq \sum_{i=1}^{n} \frac{\|x_{|\operatorname{supp}(b_i)}\|_2 \|b_i\|_2}{\|x\|_2^2} \leq \sum_{i=1}^{n} \frac{\|x_{|\operatorname{supp}(b_i)}\|_2}{\|x\|_2} \leq \sum_{i=1}^{n} \frac{\|x_{|\operatorname{supp}(b_i)}\|_1}{\|x\|_2} \leq \frac{k\|x\|_1}{\|x\|_2},$$

where the last inequality follows from the sparsity of B (every column index is in $\operatorname{supp}(b_i)$ for at most k of the rows b_i). The lemma now follows by a simple application of the Cauchy-Schwarz inequality. □

We need another, similar, lemma in order to bound the variance.

Lemma 2.3.2. *For every k-sparse matrix $B \in \mathbb{R}^{n \times k}$ with rows $b_1, \ldots, b_n$ and a vector $x \in \mathbb{R}^k$ such that $\|x\|_2 \geq \|b_i\|_2$ for all $i \in [n]$, we have that*

$$\frac{\|Bx\|_2^2}{\|x\|_2^4} = \sum_{i=1}^{n} \frac{\langle x, b_i \rangle^2}{\|x\|_2^4} \leq k.$$

Proof. Following similar steps as that of Lemma 2.3.1,

$$\sum_{i=1}^{n} \frac{\langle x, b_i \rangle^2}{\|x\|_2^4} \leq \sum_{i=1}^{n} \frac{\|x_{|\operatorname{supp}(b_i)}\|_2^2}{\|x\|_2^2} \leq k,$$

where again the last inequality follows from the sparsity of B. □

The next two lemmas present bounds that improve over Lemma 2.3.1 in two special cases, when the k-sparse matrix is a graph Laplacian, and when all its non-zero entries come from a bounded range.

Lemma 2.3.3. *Let $G = ([n], E)$ be an undirected graph with positive edge weights $\{w_{uv}\}_{uv \in E}$. Let k be its maximum (unweighted) degree, and let $L(G) \in \mathbb{R}^{n \times n}$ be its Laplacian matrix with rows $l_1, \ldots, l_n$. Given $u \in [n]$, let the matrix B_u consist of all the rows l_v where $\|l_u\|_2 \geq \|l_v\|_2$,*

and interpret B_u also as a set of rows. Then,

$$\frac{\|B_u l_u\|_1}{\|l_u\|_2^2} = \sum_{l_v \in B_u} \frac{|\langle l_u, l_v \rangle|}{\|l_u\|_2^2} \leq 2k.$$

(Trivially, we can also omit from B_u rows where $\langle l_u, l_v \rangle = 0$.)

Proof. The main idea is that the additional matrix structure implies $\|l_u\|_1 \leq 2\|l_u\|_2$, which is better than what follows from the Cauchy-Schwarz inequality. Indeed, $\|l_u\|_2^2 = \big(- \sum_{t \in N(u)} w_{ut} \big)^2 + \sum_{t \in N(u)} w_{ut}^2 \geq \big(\sum_{t \in N(u)} w_{ut} \big)^2 = \big(\frac{1}{2} \|l_u\|_1 \big)^2$. Now using this inequality in the proof of Lemma 2.3.1, we have

$$\frac{\|B_u l_u\|_1}{\|l_u\|_2^2} \leq \frac{k\|l_u\|_1}{\|l_u\|_2} \leq 2k.$$

□

Lemma 2.3.4. *For positive constants $\alpha \leq \beta$ and a k-sparse matrix $B \in \mathcal{C}_{\alpha,\beta}^{n \times k}$ with rows $b_1, \ldots, b_n$ and a vector $x \in \mathcal{C}_{\alpha,\beta}^k$ such that $\|x\|_2 \geq \|b_i\|_2$ for all $i \in [n]$, we have that*

$$\frac{\|Bx\|_1}{\|x\|_2^2} = \sum_{i=1}^{n} \frac{|\langle x, b_i \rangle|}{\|x\|_2^2} \leq k\frac{\beta}{\alpha}.$$

Proof. By a direct calculation using the sparsity of B,

$$\sum_{i=1}^{n} \frac{|\langle x, b_i \rangle|}{\|x\|_2^2} \leq \sum_{j=1}^{k} \frac{|x_j| \cdot \beta k}{\alpha \|x\|_1} = k\frac{\beta}{\alpha}.$$

□

2.3.4 Analyzing the Estimator

We now prove that the importance sampling estimator $Y(A)$ given in Section 2.3.2 is an unbiased estimator with a small variance. In addition to analyzing the estimator for all k-sparse matrices, we provide in Theorem 2.3.6 improved bounds for two special families of k-sparse matrices: (i) Laplacians of undirected graphs and (ii) matrices whose non-zero entries lie in an interval $[\alpha, \beta]$ for parameters $0 < \alpha \leq \beta$.

Theorem 2.3.5. *For every $p \in 2\mathbb{Z}_{\geq 2}$ and a k-sparse matrix $A \in \mathbb{R}^{n \times n}$, the estimator $Y(A)$ given in Section 2.3.2 satisfies $\mathbf{E}[Y(A)] = \|A\|_{S_p}^p$ and $Var(Y(A)) \leq O_p(k^{\frac{3p}{2}-4})\|A\|_{S_p}^{2p}$.*

Proof. We will use the importance sampling framework of Theorem 2.2.2. In order to do so we must first argue that the values $\tau_{(i_1,\ldots,i_{q-1})}$ for $(i_1, \ldots, i_{q-1}) \in \mathcal{S}$ indeed form a probability distribution. It is easy to see that the probabilities of sampling the first row form a distribution over $[n]$. Similarly, for every $2 \leq t \leq q-1$, the values $p_{i_{t-1}}^{i_1}(\cdot)$ indeed form a probability distribution over the rows in $N_S^{i_1}(i_{t-1})$. The argument for $\tau_{(i_1,\ldots,i_{q-1})}$ follows by the law of total probability.

Per Theorem 2.2.2, it is sufficient to prove that for all $(i_1, \ldots, i_{q-1}) \in \mathcal{S}$,

$$\frac{1}{\tau_{(i_1,\ldots,i_{q-1})}} \cdot \left|z_{(i_1,\ldots,i_{q-1})}\right| \leq O_p(k^{\frac{3}{4}p-2})z \tag{2.5}$$

Fix a sequence of indices $(i_1, \ldots, i_{q-1}) \in \mathcal{S}$ and let $\zeta = \frac{\left|z_{(i_1,\ldots,i_{q-1})}\right|}{\tau_{(i_1,\ldots,i_{q-1})}}$. Inequality (2.5) can be shown as follows,

$$\zeta = \frac{\sum_j \|a_j\|_2^p}{\|a_{i_1}\|_2^p} \prod_{t=2}^{q-1} \frac{1}{p_{i_{t-1}}^{i_1}(i_t)} \left| \sum_{i_q \in N_S^{i_1}(i_1)} c(i_1, \ldots, i_q)\sigma(i_1, \ldots, i_q)\langle a_{i_q}, a_{i_1}\rangle \right|$$

$$\leq \frac{\sum_j \|a_j\|_2^p}{\|a_{i_1}\|_2^p} \frac{\prod_{t=2}^{q-1} \sum_{l \in N_S^{i_1}(i_{t-1})} |\langle a_{i_{t-1}}, a_l \rangle|}{|\sigma(i_1, \ldots, i_{q-1})|} \sum_{i_q \in N_S^{i_1}(i_1)} c(i_1, \ldots, i_q) \left| \sigma(i_1, \ldots, i_q) \langle a_{i_q}, a_{i_1} \rangle \right|$$

$$= \frac{\sum_j \|a_j\|_2^p}{\|a_{i_1}\|_2^p} \left(\prod_{t=2}^{q-1} \sum_{l \in N_S^{i_1}(i_{t-1})} |\langle a_{i_{t-1}}, a_l \rangle| \right) \sum_{i_q \in N_S^{i_1}(i_1)} c(i_1, \ldots, i_q) \left| \langle a_{i_{q-1}}, a_{i_q} \rangle \langle a_{i_q}, a_{i_1} \rangle \right|$$

By Young's Inequality for products of numbers and the bound on $c(i_1, \ldots, i_q)$,

$$\leq \frac{q}{2} \frac{\sum_j \|a_j\|_2^p}{\|a_{i_1}\|_2^p} \left(\prod_{t=2}^{q-1} \sum_{l \in N_S^{i_1}(i_{t-1})} |\langle a_{i_{t-1}}, a_l \rangle| \right) \left(\sum_{i_q \in N_S^{i_1}(i_1)} \langle a_{i_{q-1}}, a_{i_q} \rangle^2 + \langle a_{i_q}, a_{i_1} \rangle^2 \right)$$

$$= \frac{q}{2} \sum_j \|a_j\|_2^p \left(\frac{\prod_{t=2}^{q-1} \sum_{l \in N_S^{i_1}(i_{t-1})} |\langle a_{i_{t-1}}, a_l \rangle|}{\|a_{i_1}\|_2^{p-4}} \right) \left(\sum_{i_q \in N_S^{i_1}(i_1)} \frac{\langle a_{i_{q-1}}, a_{i_q} \rangle^2 + \langle a_{i_q}, a_{i_1} \rangle^2}{\|a_{i_1}\|_2^4} \right)$$

By applying Lemma 2.3.2 to the two inner-most summations and the fact that $\|a_{i_{q-1}}\|_2 \leq \|a_{i_1}\|_2$,

$$\leq qk \cdot \sum_j \|a_j\|_2^p \left(\frac{\prod_{t=2}^{q-1} \sum_{l \in N_S^{i_1}(i_{t-1})} |\langle a_{i_{t-1}}, a_l \rangle|}{\|a_{i_1}\|_2^{p-4}} \right)$$

By applying Lemma 2.3.1 and the fact that $\|a_{i_{t-1}}\|_2 \leq \|a_{i_1}\|_2$ for any $2 \leq t \leq q-1$,

$$\leq qk \sum_j \|a_j\|_2^p \left(\prod_{t=2}^{q-1} k\sqrt{k} \right) = qk^{\frac{3p}{4}-2} \sum_i \|a_i\|_2^p \leq \frac{pk^{\frac{3p}{4}-2}}{2} \|A\|_{S_p}^p$$

where the last inequality follows from Fact 2.2.1. □

Theorem 2.3.6. *For every $p \in 2\mathbb{Z}_{\geq 2}$ and a k-sparse Laplacian matrix $A \in \mathcal{L}_n$, the estimator $Y(A)$ given in Section 2.3.2 satisfies $Var(Y(A)) \leq O_p(k^{p/2-1})\|A\|_{S_p}^{2p}$. If instead the k-sparse matrix is $A \in \mathcal{C}_{\alpha,\beta}^{n \times n}$ for some $0 < \alpha \leq \beta$, then $Var(Y(A)) \leq O_p(k^{p/2-2} \left(\beta/\alpha\right)^{p/2-2})\|A\|_{S_p}^{2p}$.*

Proof. The bound for $\mathcal{L}_n$ (Laplacians) follows the above proof of Theorem 2.3.5 but bounding the summations using Lemma 2.3.3 instead of Lemma 2.3.1.

The bound for $\mathcal{C}^{n\times n}_{\alpha,\beta}$ uses a special case of the importance sampling lemma. Using the notation from Theorem 2.2.2, if $z_i > 0$ for all $i \in [n]$ then one can bound the variance by $\lambda(z)^2$. Using this, the proof follows the same argument as that of Theorem 2.3.5 but using Lemma 2.3.4 to bound the summations bounded by Lemmas 2.3.2 and 2.3.1. □

2.4 Implementing the Estimator: Row-Order and Turnstile Streams

In this section we show how to implement the importance sampling estimator defined in Section 2.3.2 in two different streaming models, row-order and turnstile streams. We start by stating two theorems that bound the space complexity of implementing the estimator in row-order streams. The first one is our main result from the Introduction, and applies to all k-sparse matrices. The second theorem considers special families of k-sparse matrices.

Theorem 2.1.1. *There exists an algorithm that, given $p \in 2\mathbb{Z}_{\geq 2}$, $\epsilon > 0$ and a k-sparse matrix $A \in \mathbb{R}^{n\times n}$ streamed in row-order, makes $\lfloor p/4 \rfloor + 1$ passes over the stream using $O_p(\epsilon^{-2}k^{3p/2-3})$ words of space, and outputs $\bar{Y}(A)$ that $(1 \pm \epsilon)$-approximates $\|A\|^p_{S_p}$ with probability at least $2/3$.*

Theorem 2.4.1. *There exists an algorithm that, given $p \in 2\mathbb{Z}_{\geq 2}$, $\epsilon > 0$, and a k-sparse matrix $A \in \mathcal{L}_n$ streamed in row-order, makes $\lfloor p/4 \rfloor + 1$ passes over the stream using $O_p(\epsilon^{-2}k^{p/2})$ words of space, and outputs $\bar{Y}(A)$ that $(1 \pm \epsilon)$-approximates $\|A\|^p_{S_p}$ with probability at least $2/3$. If instead the k-sparse matrix A is from $\mathcal{C}^{n\times n}_{\alpha,\beta}$ for $0 < \alpha \leq \beta$, then the space bound is $O_p(\epsilon^{-2}k^{p/2-1}\left(\beta/\alpha\right)^{p/2-2})$ words.*

We also show that the estimator defined in Section 2.3.2 can be implemented in turnstile streams in $p/2+3$ passes over the stream.

Theorem 2.4.2. *There exists an algorithm that, given $p \in 2\mathbb{Z}_{\geq 2}$, $\epsilon > 0$ and a k-sparse matrix $A \in \mathbb{R}^{n\times n}$ streamed in a turnstile fashion, makes $p/2+3$ passes over the stream using $O_p(k^{3p-6}n^{1-\frac{2}{p}}(\epsilon^{-1}\log n)^{O(p)})$ words of space, and outputs $\bar{Y}(A)$ that $(1\pm\epsilon)$-approximates $\|A\|_{S_p}^p$ with probability at least $2/3$.*

Outline. At a high level, the algorithms in all three theorems are similar, and compute multiple copies of the estimator defined in Section 2.3.2 in parallel and output their average (to reduce the variance). The algorithms differ in the number of copies, derived from Theorems 2.3.5 and 2.3.6. Here, and in Sections 2.4.1 and 2.4.2, we describe how to implement each estimator in $p/2$ stages, and in Section 2.4.3 we show how to reduce the number of stages to $\lfloor p/4\rfloor+1$. The first stage samples and stores a "seed" row which we will denote by a_{i_1}. Each subsequent stage $1<t<q$ stores two values: a row index i_t (and row a_{i_t} itself) and an interim estimate $Y_t \coloneqq \sigma(i_1,\ldots,i_t)$. The final stage q computes and outputs $\sum_{i_q\in N_S^{i_1}(i_1)} Y_{q-1}\cdot\langle a_{i_q},a_{i_1}\rangle c(i_1,\ldots,i_q)$, where $1\leq c(i_1,\ldots,i_q)\leq q$ is as defined in (2.4).

The estimator is relatively easy to implement in row-order streams using $p/2$ passes and $O_p(\epsilon^{-2}k^{3p/2-3})$ words of space as shown in Section 2.4.1. In turnstile streams however, the estimator is more difficult to implement. The technical roadblock is sampling the first, "seed" row $i_1\in[n]$ with probability proportional to $\frac{\|a_{i_1}\|_2^p}{\sum_j\|a_j\|_2^p}$. We use approximate samplers for turnstile streams to get around this roadblock. For a vector $x\in\mathbb{R}^n$ updated in a turnstile fashion, one can sample an index i with probability approximately $x_i^t/\|x\|_t^t$ for various $t\in[0,\infty)$. Such algorithms are called L_t-samplers and have been studied thoroughly, see e.g. [43]. Approximate samplers introduce a multiplicative (relative) error and an additive error in the sampling probability, which need to be accounted for when analyzing the algorithm

that uses the sampler.

Thus, in order to sample rows proportional to the quantities we want, we build two subroutines in the turnstile model:

1. Cascaded $L_{p,2}$-norm sampler for A, used to sample the seed row i_1 with probability approximately $\|a_{i_1}\|_2^p$. It runs in 2-*passes*, has relative error $O(\epsilon)$ and uses space $\tilde{O}_p(\epsilon^{-2}n^{1-2/p})$.

2. Compute inner products between a given row and its neighbors in space $\tilde{O}(k^2)$.

Using the two subroutines we can implement the estimator in Section 2.3.2 in $p+1$ passes of the stream in space $O_p(k^{3p-6}n^{1-2/p}(\epsilon^{-1}\log n)^{O(p)})$. The additional $\tilde{O}(n^{1-2/p})$ space complexity factor is introduced by the approximate $L_{p,2}$-sampler. We remark that this factor is actually unavoidable for algorithms that compute $\|A\|_{S_p}^p$ in the turnstile model, since there is an $\Omega(n^{1-2/p})$ lower bound for computing the l_p-norm of vectors in $\mathbb{R}^n$ (in turnstile streams), even if the algorithm is allowed multiple passes. The additional $O(k^{3p/2-3})$ factor in the space complexity for turnstile streams compared to row-order streams is due to the bias introduced in estimating the sampling probability of the first, "seed" row.

As mentioned earlier, a slightly improved version runs in $\lfloor p/4 \rfloor + 1$ and $p/2+3$ passes for row-order and turnstile streams respectively, with the same space complexities (up to constant factors). The idea is to build two parallel paths from the same seed row and eventually "stitch" the two into one cycle.

2.4.1 Row-Order Streams

In this section we show how to easily implement the estimator defined in Section 2.3.2 in $q = p/2$ passes over a row-order stream, i.e. a sligthly weaker version of Theorem 2.1.1.

As mentioned, in Section 2.4.3 we explain how to reduce the number of passes to $\lfloor p/4 \rfloor + 1$ using a small adjustment to the algorithm. Algorithm 1, computes multiple copies of the estimator in parallel using space $O(k)$ for each copy.

Algorithm 1 Algorithm for Schatten p-Norm of k-Sparse Matrices for $p \in 2\mathbb{Z}_{\geq 2}$ in Row-Order Streams

Input: $A \in \mathbb{R}^{n \times n}$ streamed in row-order, $p \in 2\mathbb{Z}_{\geq 2}$, $\epsilon > 0$, $m \in \mathbb{Z}^+$.

in parallel m **times do** ▷ Each copy is a "walk"
  $i_1, \ldots, i_q \leftarrow 0$, $Y_1, \ldots, Y_q \leftarrow 0$
  in pass 1 **do**
    sample *one* row $i_1 \in [n]$ with probability $\frac{\|a_{i_1}\|_2^p}{\sum_j \|a_j\|_2^p}$ ▷ Using Reservoir Sampling
    $Y_1 \leftarrow \frac{\sum_j \|a_j\|_2^p}{\|a_i\|_2^p}$

  in pass $2 \leq t \leq q-1$ **do**
    sample *one* row $i_t \in [n]$ with probability $p^{i_1}_{i_{t-1}}(i)$ ▷ As defined in Section 2.3.2
    $Y_t \leftarrow Y_{t-1} \cdot \frac{\langle a_{i_{t-1}}, a_{i_t} \rangle}{p^{i_1}_{i_{t-1}}(i)}$

  in pass q **do**
    compute $Y_q \leftarrow Y_{q-1} \sum_{i_q \in N_S^{i_1}(i_1)} \langle a_{i_{q-1}}, a_{i_q} \rangle \langle a_{i_q}, a_{i_1} \rangle c(i_1, \ldots, i_q)$

return average of the m copies of Y_q

Proof of Theorem 2.1.1 (version with $p/2$ passes). Algorithm 1 computes the estimator defined in Section 2.3.2 m times in parallel and outputs the average which we will denote by $\bar{Y}(A)$. Since the variance of the estimator is at most $C_p k^{\frac{3p}{2}-4}$ as per Theorem 2.3.5, by

setting $m = \frac{Ck^{\frac{3p}{2}-4}}{\epsilon^2}$ and the constant C appropriately, the guarantee on the estimate follows by an application of Chebyshev's Inequality to $\bar{Y}(A)$.

In pass t, each instance of the m parallel instances store the row a_{i_t} along with other estimates that can be stored in $O_p(1)$ words of space. Thus the total space complexity of the algorithm is $mk = O_p(\epsilon^{-2}k^{\frac{3p}{2}-3})$ words. □

The proof of Theorem 2.4.1 (version with $p/2$ passes) follows the above by adjusting m according to Theorem 2.3.6.

2.4.2 Turnstile Streams

Preliminaries for Approximate Sampling

We define approximate samplers which we will use in turnstile streams to implement our estimator. Approximate L_p samplers have been studied extensively, see e.g. [43].

Definition 2.4.3 (Approximate L_t Sampler)**.** Let $x \in \mathbb{R}^n$ be a vector and $t \geq 0$. An *approximate L_t sampler* with relative error ϵ, additive error Δ, and success probability $1-\delta$ is an algorithm that outputs each index $i \in [n]$ with probability

$$p_i \in (1 \pm \epsilon)\frac{|x_i|^t}{\|x\|_t^t} \pm \Delta,$$

and with probability δ the sampler is allowed to output FAIL.

If an approximate sampler has no relative error and its additive error is less than n^{-C}, for arbitrarily large constant $C > 0$, then it is referred to as an *exact* L_p-sampler.

Generalizing L_p-samplers, we define approximate $L_{p,q}$-samplers for matrices.

Definition 2.4.4 (Weak Approximate $L_{t,q}$ Sampler)**.** Let $t, q \geq 0$ be constants and $A \in \mathbb{R}^{n \times m}$ be a matrix with rows $a_1, \ldots, a_n$. An *approximate $L_{t,q}$ sampler* with relative error ϵ, additive error Δ, and success probability $1 - \delta$ is an algorithm that, conditioned on succeeding, outputs each index $i \in [n]$ with probability

$$p_i \in (1 \pm \epsilon) \frac{\|a_i\|_q^t}{\sum_{j \in [n]} \|a_j\|_q^t} \pm \Delta,$$

and on failing, which occurs with probability δ, outputs any index.

We draw the attention of the reader to the success condition of the $L_{p,q}$ sampler; unlike for L_p samplers, the above definition is a weaker guarantee but is sufficient for our purpose since we can absorb the probability of failure for the sampler into the failure probability of the Schatten p-norm algorithm.

We recall some properties of higher powers of Gaussian distributions which we will use later in the analysis of sampling subroutines that we build. First, we give the higher moments of mean zero Gaussian random variables.

Fact 2.4.5. *For $t \geq 0$, $r \in 2\mathbb{Z}_{\geq 1}$ and a random variable $X \sim \mathcal{N}(0, t^2)$, we have*

$$\mathbf{E}[|X|^r] = t^r (r-1)!!.$$

We state a concentration property for polynomial functions of independent Gaussian (and Rademacher) random variables called Hypercontractivity Inequalities. For an introduction to the theory of hypercontractivity, see e.g. Chapter 9 of [125].

Proposition 2.4.6 (Hypercontractivity Concentration Inequality, Theorem 1.9 [135])**.** *Consider a degree d polynomial $f(Y) = f(Y_1, \ldots, Y_n)$ of independent centered Gaussian or*

Rademacher random variables $Y_1, \ldots, Y_n$. *Denote the variance* $\sigma^2 = Var(f(Y))$, *then,*

$$\forall \lambda \geq 0, \quad \mathbf{Pr}[\left|f(Y) - \mathbf{E}[f(Y)]\right| \geq \lambda] \leq e^2 \exp\left(-\left(\frac{\lambda^2}{R \cdot \sigma^2}\right)^{\frac{1}{d}}\right)$$

where $R = R(d) > 0$ *depends only on* d.

Weak Sampler for Cascaded Norm $L_{p,2}$

Before giving our construction for approximate $L_{p,2}$ samplers in the turnstile model (Theorem 2.4.8), we recall some core results for L_p samplers that will be the algorithmic workhorse of our subroutine for $L_{p,2}$ sampling.

One can construct algorithms for approximate L_p samplers in various computational models. We look specifically at L_p samplers in the turnstile streaming model. The following algorithmic guarantees exist for approximate L_p samplers of vectors in turnstile streams.

Theorem 2.4.7 (Theorem 1.2 in [112])**.** *For* $\delta > 0$ *and* $p \in 2\mathbb{Z}^+$, *there exists an* 0*-relative-error* L_p*-sampler in turnstile streams, in* 2***-passes***, *with probability of outputting* FAIL *at most* n^{-C} *where* $C > 0$ *is an arbitrarily large constant. The algorithm uses* $O_p(n^{1-2/p} \log^{O(p)} n)$ *space.* [4]

For a given vector $x \in \mathbb{R}^n$ whose entries are streamed in a turnstile fashion, we will denote L_p-Sampler(x, δ) to be the output of the algorithm in Theorem 2.4.7 with failure probability at most δ. We will use this algorithm in turnstile streams for $p \geq 2$ to give an $O(\epsilon)$ relative error $L_{p,2}$ sampler and failure probability at most δ for any given $\delta > 0$. The algorithm is fairly simple and is described in Algorithm 2.

[4] The original theorem statement in the paper is for $p \in [0, 2]$ but it is well-known among experts that the result extends to $p > 2$.

Algorithm 2 Approximate $L_{p,2}$ Sampling Algorithm in Turnstile Streams

INPUT: $A \in \mathbb{R}^{n \times n}$ as a turnstile stream, $p \in \mathbb{Z}_{\geq 2}$, $\hat{\delta} \in (0,1), \epsilon > 0$.

1: Set $\hat{C}_p > 0$, $m \leftarrow \frac{\hat{C}_p \log^p n}{\epsilon^2}$ ▷ $\hat{C}_p$ depends only on p

2: construct $G \in \mathbb{R}^{n \times m}$, with i.i.d standard Gaussian entries ▷ drawn pseudorandomly

3: compute matrix $X \leftarrow \frac{1}{(p-1)!!} \cdot AG$

4: $(i, j) \leftarrow L_p$-SAMPLER$(x, \hat{\delta})$ ▷ where $x \in \mathbb{R}^{n^2}$ is the "flattened" version of X

5: **return** i if L_p-sampler didn't output FAIL otherwise return any index

The matrix X, defined on line 3 in the above algorithm, can be computed "on the fly" given updates to A in the stream.

We then give the following theorem for approximate $L_{p,2}$ sampling in turnstile streams by arguing for the vector x defined in Algorithm 2, the average of the p^{th} power of the entries corresponding to row i is tightly concentrated around $\|a_i\|_2^p$.

Theorem 2.4.8. *For every $\epsilon, C > 0, \delta \in (0,1)$ and $p \in 2\mathbb{Z}_{\geq 2}$, Algorithm 2 is an $O(\epsilon)$ relative error and $O(n^{-C})$ additive error $L_{p,2}$ weak sampler in turnstile streams with failure probability at most δ. The algorithm uses $O_p(n^{1-2/p}\epsilon^{-2}\log(\frac{1}{\delta})\log^{O(p)}(n))$ words of space.*

Proof. For a fixed $i \in [n]$, notice that $x_{i,1}, \ldots, x_{i,m}$ are independent and identically distributed as $\mathcal{N}\left(0, \frac{\|a_i\|_2^2}{((p-1)!!)^2}\right)$. Using Fact 2.4.5, $\mathbf{E}[x_{i,j}^p] = \|a_i\|_2^p$ for all $j \in [m]$ since p is even.

Let i^* be the output of Algorithm 2. From the guarantee for L_p-samplers by Theorem 2.4.7, conditioning on the L_p sampler succeeding, and setting the additive error sufficiently

low, the probability that $i^* = i$ is

$$\mathbf{Pr}[i^* = i] = \sum_{j=1}^{m} \frac{x_{i,j}^p}{\|x\|_p^p} \pm O\left(n^{-C}\right).$$

We will first show that, for a fixed $i \in [n]$, the quantity $\sum_{j=1}^{m} x_{i,j}^p$ is tightly concentrated around $m\|a_i\|_2^p$ with high probability *over the randomness of the Gaussian sketch*.

Set the polynomial $f : \mathbb{R}^m \to \mathbb{R}$ on the random variables $\{x_{i,j}\}_{j=1}^m$ to be $f(x_{i,1}, \ldots, x_{i,m}) = \sum_{j=1}^{m} x_{i,j}^m$. Since the random variables $\{x_{i,j}\}_{j=1}^m$ are independent,

$$Var(f(x_{i,1}, \ldots, x_{i,m})) = mVar(x_{i,*}^p) = m\|a_i\|_2^{2p} \frac{(2p-1)!! - ((p-1)!!)^2}{((p-1)!!)^2}$$

for even $p > 2$. Using this to apply the Hypercontractivity Concentration Inequality for Gaussian random variables given in Proposition 2.4.6 gives us,

$$\mathbf{Pr}[\left|\sum_{j=1}^{m} x_{i,j}^p - m\|a_i\|_2^p\right| \geq \epsilon m\|a_i\|_2^p] \leq e^2 \exp\left(-\left(\frac{\epsilon^2 m}{C_p}\right)^{\frac{1}{p}}\right)$$

where C_p is a constant only dependent on p.

By setting $\hat{C}_p$ in Algorithm 2 appropriately, we can apply the the union bound over all $i \in [n]$ to obtain,

$$\mathbf{Pr}[i^* = i] = \frac{(1 \pm O(\epsilon))\|a_i\|_2^p}{(1 \pm O(\epsilon))\sum_{l=1}^{n} \|a_l\|_2^p} \pm O(n^{-C}) \qquad \text{for all } i \in [n]$$

with probability at least $1 - \hat{\delta} - n^{-\hat{c}}$ (where $\hat{c}$ is dependent on $\hat{C}_p$). Setting $\hat{\delta}$ appropriately in Algorithm 2 gives us the theorem. □

Recovering Rows and Their Neighbors

We also give some subroutines to recover rows and their neighbors so that we can compute inner-products between rows, sample neighbors and compute the probabilities for the estimator. The algorithmic core for these subroutines will be sparse-recovery algorithms which can be implemented using the Count-Sketch data structure described below.

Theorem 2.4.9 (Count-Sketch [28])**.** *For all $w, n \in \mathbb{Z}^+$ and $\delta \in (0,1)$, there is a randomized linear function $M : \mathbb{R}^n \leftarrow \mathbb{R}^s$ with $S = O(w \log(n/\delta))$ and a recovery algorithm A satisfying the following. For input $x \in \mathbb{R}^n$, algorithm A reads Mx and outputs a vector $\tilde{x} \in \mathbb{R}^n$ such that*

$$\forall x \in \mathbb{R}^n, \quad \mathbf{Pr}[\|x - \tilde{x}\|_\infty \leq \frac{1}{\sqrt{w}} \min_{x' : \|x'\|_0 = w} \|x - x'\|_2] \geq 1 - \delta.$$

Denote the output of a Count-Sketch algorithm on vector $x \in \mathbb{R}^n$ with parameter $w \in \mathbb{Z}^+$ and failure probability $\delta \geq 0$ to be $\textsc{Count-Sketch}_w(x, \delta)$. Notice that if it is guaranteed that x is k-sparse, i.e. $\|x\|_0 \leq k$, then the output $\textsc{Count-Sketch}_k(x, \delta)$ recovers the vector x exactly with probability at least $1 - \delta$ because $\min_{\tilde{x} : \|\tilde{x}\|_0 = k} \|x - \tilde{x}\|_2 = 0$ for every k-sparse vector x.

Reverting to our setting of k-sparse matrices in turnstile streams, given a target index $i \in [n]$, it is clear how to recover row a_i using $\tilde{O}(k)$ space using the Count-Sketch algorithm stated. Given a row a_i, we can recover the neighboring rows $\{a_j : j \in N(i)\}$ by running $\textsc{Count-Sketch}_k(A_{*,j}, \tilde{\delta})$ for each $j \in \text{supp}(a_i)$ (where $A_{*,j}$ corresponds to the j^{th} column of A). Since each column and row is k-sparse, with $\tilde{O}(k^2)$ space, we can recover the neighbors of row a_i given access to a_i. In addition, by setting the failure probability to $\frac{\delta}{k+1}$ in the above calls to $\textsc{Count-Sketch}_k$, our recovery subroutine will succeed with probability at least $1 - \delta$.

Algorithm for Turnstile Streams

We are now ready to present the algorithm implementing the estimator stated in Section 2.3.2 for turnstile streams. We note that unlike in row-order streams, we cannot recover the probability of sampling the first row exactly in turnstile streams. Since the output probability of the samplers is approximate, it introduces some bias in the estimator which we will have to bound. Therefore, the proof of correctness for this algorithm slightly deviates from that given in Theorem 2.2.2 but uses the same underlying ideas.

Let us introduce notation for the subroutines we will need. Let $L_{p,2}$-SAMPLER(A, ϵ, δ) denote the output of the approximate $L_{p,2}$ sampler defined in Algorithm 2 with relative error ϵ, and failure probability δ. Additionally, we will need to estimate the cascaded norm $L_{p,2}$ of A in order to bias the quantity we sample in our importance sampling estimator. Denote by $L_{p,2}$-NORMESTIMATOR(A, ϵ, δ) the output of an algorithm for estimating the $L_{p,2}$-norm of A with relative error ϵ and failure probability δ, such as in Section 4 of [86].

We describe our algorithm for turnstile streams in Algorithm 3, which runs $p+1$ passes over the data, i.e. a sligthly weaker version of Theorem 2.4.2. As mentioned, the number of passes can be reduced to $\lfloor p/2 \rfloor + 3$ using the extra insight of Section 2.4.3.

Algorithm 3 Algorithm for Schatten p-norm of k-Sparse Matrices for $p \in 2\mathbb{Z}_{\geq 2}$ in Turnstile Streams

Input: $A \in \mathbb{R}^{n \times n}$ in a stream with turnstile updates, $p \in 2\mathbb{Z}_{\geq 2}$, $\epsilon > 0$, $m \in \mathbb{Z}^+$.

in parallel m **times do**
  $i_1, \ldots, i_q \leftarrow 0$, $Y_1, \ldots, Y_q \leftarrow 0$
  in stage 1 **do** ▷ takes 3 passes
    $i_1 \leftarrow L_{p,2}$-SAMPLER$(A, \frac{\epsilon}{k^{3p/4-2}}, \frac{1}{100})$
    $\tilde{a}_{i_1} \leftarrow$ COUNT-SKETCH$_k(a_{i_1}, \frac{1}{100})$
    $D_1 \leftarrow L_{p,2}$-NORMESTIMATOR$(A, \epsilon, \frac{1}{100})$
    $Y_1 \leftarrow \frac{D_1}{\|\tilde{a}_{i_1}\|_2^p}$

  in stage $2 \leq t \leq q-1$ **do** ▷ each stage takes 2 passes
    $\tilde{C}_{t-1} \leftarrow \{$COUNT-SKETCH$_k(A_{*,j}, \frac{1}{100kq}) : j \in \text{supp}(\tilde{a}_{i_{t-1}})\}$
    reconstruct rows $\tilde{R}_{t-1} \leftarrow \{r_j : \text{row } j \text{ has support in } \tilde{C}_{t-1} \text{ and } \|r_j\|_2 < \|\tilde{a}_{i_1}\|_2\}$.
    $D_t \leftarrow \sum_{j \in \tilde{R}_{t-1}} \langle \tilde{a}_{i_{t-1}}, r_j \rangle$
    sample row index $i_t \in \text{supp}(\tilde{R}_{t-1})$ with probability $\frac{\langle \tilde{a}_{i_{t-1}}, r_{i_t} \rangle}{D_t}$
    $\tilde{a}_{i_t} \leftarrow$ COUNT-SKETCH$_k(a_{i_t}, \frac{1}{100q})$
    $Y_t \leftarrow Y_{t-1} \cdot \frac{D_t}{\langle \tilde{a}_{i_{t-1}}, \tilde{a}_{i_t} \rangle} \cdot \langle \tilde{a}_{i_{t-1}}, \tilde{a}_{i_t} \rangle$

  in stage q **do**
    $\tilde{C}_{q-1} \leftarrow \{$COUNT-SKETCH$_k(A_{*,j}, \frac{1}{100k}) : j \in \text{supp}(\tilde{a}_{i_{q-1}})\}$
    reconstruct rows $\tilde{R}_{q-1} \leftarrow \{r_j : \text{row } j \text{ has support in } \tilde{C}_{q-1} \text{ and } \|r_j\|_2 < \|\tilde{a}_{i_1}\|_2\}$.
    compute

$$Y_q \leftarrow Y_{q-1} \sum_{r_j \in \tilde{R}_{q-1}} \langle \tilde{a}_{i_{q-1}}, r_j \rangle \langle r_j, \tilde{a}_{i_1} \rangle c(i_1, \ldots, i_{q-1}, j)$$

return average of the m copies of Y_q

Proof of Theorem 2.4.2 (version with $p+1$ passes). Recall that COUNT-SKETCH$_k$, from Section 2.4.2, will recover all the entries of a k-sparse vector exactly with high probability. By setting the failure probability of each call to COUNT-SKETCH$_k$ to be sufficiently low, we can apply a union bound over all executions and assume that the algorithm recovers all the rows

denoted by $\tilde{a}$ and r.

Let us assume that the L_p-sampler and Count-Sketch routines succeed and argue that taking the expectation over the randomness of the Gaussian sketch in the $L_{p,2}$-Sampler algorithm, the $L_{p,2}$-NormEstimator and the importance sampling estimator gives us that $|\mathbf{E}[\bar{Y}(A)] - \|A\|_{S_p}^p| \leq O_p(\epsilon)\|A\|_{S_p}^p$.

Recall that the algorithm invokes an $O\left(\frac{\epsilon}{k^{3p/4-2}}\right)$ relative error $L_{p,2}$-sampler in line 4. Since the additive error is less than n^{-C} for arbitrary $C \geq 0$, we can simply absorb it in the failure probability of the algorithm. We thus get,

$$\mathbf{E}[\bar{Y}(A)] = \sum_{\substack{(i_1,\ldots,i_{q-1}) \\ \in \mathcal{S}}} \left(1 \pm \frac{O(\epsilon)}{k^{3p/4-2}}\right) \frac{\|a_{i_1}\|_2^p}{\sum_j \|a_j\|_2^p} \frac{\mathbf{E}[D_1]}{\|a_{i_1}\|_2^p} \sum_{i_q \in N_S^{i_1}(i_1)} \sigma(i_1,\ldots,i_q,i_1)c(i_1,\ldots,i_q)$$

Since $L_{p,2}$-NormEstimator is an unbiased estimator for the $L_{p,2}$-norm, i.e. $\mathbf{E}[D_1] = \sum_j \|a_j\|_2^p$, we get

$$\left|\mathbf{E}[\bar{Y}(A)] - \|A\|_{S_p}^p\right| \leq \sum_{\substack{(i_1,\ldots,i_{q-1}) \\ \in \mathcal{S}}} \frac{O(\epsilon)}{k^{3p/4-2}} \left| \sum_{i_q \in N_S^{i_1}(i_1)} \sigma(i_1,\ldots,i_q,i_1)c(i_1,\ldots,i_q) \right|$$

We can upper bound the second term as we did in bounding the variance of the estimator in Theorem 2.2.2 to get $\left|\mathbf{E}[\bar{Y}(A)] - \|A\|_{S_p}^p\right| \leq O_p(\epsilon)\|A\|_{S_p}^p$

It is left to bound the variance of $\bar{Y}(A)$. Again, we assume that the L_p-Sampler and Count-Sketch routines succeed and recall that that for a sequence $(i_1,\ldots,i_{q-1}) \in \mathcal{S}$, we define $z_{(i_1,\ldots,i_{q-1})} = \sum_{i_q \in N_S^{i_1}(i_1)} \sigma(i_1,\ldots,i_q,i_1)c(i_1,\ldots,i_q)$. Given the guarantee of $L_{p,2}$ sampling in

Theorem 2.4.8, the variance of the estimate $\bar{Y}(A)$ is

$$Var(\bar{Y}(A)) \leq \frac{1}{m} \sum_{\substack{(i_1,\ldots,i_{q-1}) \\ \in \mathcal{S}}} (1 \pm \frac{O(\epsilon)}{k^{3p/4-2}}) \frac{1}{\sum_j \|a_j\|_2^p} \frac{\mathbf{E}[D_1^2]}{\|a_{i_1}\|_2^p} \prod_{t=2}^{q-1} \frac{1}{p_{i_{t-1}}^{i_1}(i_t)} \left(z_{(i_1,\ldots,i_{q-1})}\right)^2$$

By the accuracy guarantee of $L_{p,2}$-NormEstimator and Fact 2.2.1,

$$\leq \frac{1}{m} \sum_{\substack{(i_1,\ldots,i_{q-1}) \\ \in \mathcal{S}}} (1 \pm O(\epsilon)) \frac{\|A\|_{S_p}^p}{\|a_{i_1}\|_2^p} \prod_{t=2}^{q-1} \frac{1}{p_{i_{t-1}}^{i_1}(i_t)} \left(z_{(i_1,\ldots,i_{q-1})}\right)^2$$

Bounding this identically as we did in Theorem 2.2.2 and setting $m = \frac{Ck^{3p/2-4}}{\epsilon^2}$ give us $Var(\bar{Y}(A)) \leq C_p \epsilon \|A\|_{S_p}^{2p}$ where C_p is a constant dependent only on p.

The $L_{p,2}$-Sampler with $O\left(\frac{\epsilon}{k^{3p/4-2}}\right)$ relative error takes space $\tilde{O}_p(k^{\frac{3p}{2}-4} n^{1-\frac{2}{p}} (\epsilon^{-1} \log n)^{O(p)})$ and the $L_{p,2}$-NormEstimator takes space $\tilde{O}_p(n^{1-\frac{2}{p}} (\epsilon^{-1} \log n)^{O(p)})$. In addition, storing the rows recovered from Count-Sketch requires $\tilde{O}(k^2)$ space. Thus, the space complexity of repeating the estimator $m = \frac{Ck^{3p/2-4}}{\epsilon^2}$ times is $\tilde{O}_p(k^{3p-6} n^{1-\frac{2}{p}} (\epsilon^{-1} \log n)^{O(p)})$. We note that in stage 1, the sampler takes two passes, followed by another pass for Count-Sketch and the norm estimator. The subsequent stages requires two passes each giving a total of $3 + 2(q-1) = p + 1$ passes. □

2.4.3 Fewer Passes

As mentioned earlier, we can slightly modify the way we implement the estimator from Section 2.3.2 to reduce the number of passes that Algorithm 1 and Algorithm 3 make to $\lfloor \frac{p}{4} \rfloor + 1$ and $\frac{p}{2} + 3$, respectively. This is explained below and completes the proofs of Theorems 2.1.1, 2.4.1 and 2.4.2.

The idea is to sample each sequence $(i_1, \ldots, i_q) \in \mathcal{S}$ in a different way albeit with the

same probability. Assume for simplicity that $p \equiv 0 \pmod 4$. After sampling the first row $i_1 \in [n]$, we sample independently two "paths" of length $p/4 - 1$, each starting at i_1, with probabilities identical to the ones in the estimator. We then sum over the common neighbors of the endpoints of the two paths, using each of them to complete a cycle of length $p/2$. Formally, sample independently two sequences of rows $(i_1, l_1, \ldots, l_{q/2-1}), (i_1, j_1, \ldots, j_{q/2-1}) \in \Gamma_S^{i_1}(i_1, q/2-1)$. Denote by r the sequence of rows $(l_{q/2-1}, \ldots, l_1, i_1, j_1, \ldots, j_{q/2-1})$ then the following estimator is equivalent to the estimator described in Section 2.3.2 (slightly abusing the notation therein for concatenating two sequences of rows):

$$Y := \frac{1}{\tau_r} \sum_{\substack{m \in N_S^{i_1}(l_{q/2-1}) \\ \cap N_S^{i_1}(j_{q/2-1})}} c(r, i_q)\sigma(r)\langle a_{l_{q/2-1}}, a_m\rangle\langle a_{j_{q/2-1}}, a_m\rangle.$$

It is easy to verify that this estimator is unbiased, and that its variance can be bounded using the proof steps of Section 2.3.2. This estimator can be implemented algorithmically similarly to the description in Sections 2.4.1 and 2.4.2 using less passes over the stream. Specifically, the above approach decreases the number of "path" stages (i.e. all but the "seed" sampling stage) by a factor of (roughly) 2, and the space complexity remains the same up to constant factors. Therefore, we reduce the number of passes over the streams of Algorithm 1 and Algorithm 3 to $\lfloor \frac{p}{4} \rfloor + 1$ and $\frac{p}{2} + 3$, respectively. This concludes the proofs of Theorems 2.1.1, 2.4.1 and 2.4.2.

2.5 Pass-Space Trade-off

Very often streaming problems have a sharp transition in space complexity when comparing a single pass to multiple passes over the data. However, it turns out that for the Schatten p-norm of sparse matrices, the space dependence on the number of passes is smooth, allowing

one to pick the desired pass-space trade-off. Specifically, for any parameter $s \geq 2$, one can $(1 \pm \epsilon)$-approximate the Schatten p-norm in $\lfloor \frac{p}{2(s+1)} \rfloor + 1$ passes using $O_{p,s}(\epsilon^{-3} k^{2ps} n^{1-\frac{1}{s}})$ words of space.

Recall the Schatten p-norm formulation of (2.4). This in can be interpreted as partitioning the (contributing) length-q cycles according to their heaviest row, denoted here by i_1. Analogously, for any parameter $s \in [2, p-1]$, we split the cycle into $s+1$ segments of hop-distance roughly $\frac{q}{s+1}$, and further partition the cycles according to the heaviest row in each such segment. The idea is to "cover" a cycle by sampling s rows, where each sampled row is the heaviest among its segment. More precisely, each sample "covers" its segment, except for the heaviest row in the entire cycle that will "cover" two segments. Then, to evaluate the entire cycle we need $\lfloor \frac{q}{s+1} \rfloor + 1$ passes. The total space needed by the algorithm is $O_{p,s}(\epsilon^{-3} k^{2ps} n^{1-1/s})$ words of space, mostly as it computes multiple copies of the estimator (to reduce the variance), similarly to Section 2.4.

In the first subsection we focus on the case $s = 2$ and present a BFS-based algorithm, followed by a brief explanation how to improve the dependence on k by replacing the BFS with adaptive sampling as in the previous sections. In the second subsection we generalize the result to any $s \geq 2$.

2.5.1 The Basic Case $s = 2$ ($\lfloor \frac{p}{6} \rfloor + 1$ Passes)

As mentioned, (2.4) can be interpreted as considering only cycles that "start" from the heaviest row of the cycle (by "rotating" the cycle). We suggest a variation on this idea. Given a q-cycle "starting" at the heaviest row i, we identify the row j that is the heaviest among the rows at least $q/3$ cycle-hops away from i. In other words, if the cycle is $(i = i_1, \ldots, i_q)$, then j is the heaviest among (roughly) $i_{q/3}, \ldots, i_{2q/3}$. Therefore, our aim is to sample rows i

and j and then to connect four paths: two starting from i and two starting from j, each of hop-distance at most $q/3$. As we don't know in advance the hop-distance to row j, we store all possible options and only later decide which paths to stich together into a cycle.

Formally, we augment the notation of paths presented in Section 2.3. For indices $i, j, i_1 \in [n]$ and integers $t' \leq t'' \leq t$, define

$$\Gamma_S^{(i,j;t',t'')}(i_1,t) \coloneqq (i_1,\ldots,i_q) : \begin{cases} (i_1,\ldots,i_{t'}) \in \Gamma_S^i(i_1,t'), \\ (i_{t'},\ldots,i_{t''}) \in \Gamma_S^j(i_{t'},t''-t'+1), \text{ and,} \\ (i_{t''},\ldots,i_t) \in \Gamma_S^i(i''_t,t-t''+1) \end{cases}.$$

As we are actually interested in the special case where $t' = \lfloor \frac{q}{3} \rfloor + 1$ and $t'' = q - \lfloor \frac{q}{3} \rfloor$, we shall omit t', t'' from the superscript in this special case.

Recall that we focus on cycles in which $i_1 = i$, i.e. the heaviest row is the starting of the cycle. Furthermore, we want $j = i_l$ for some $l \in \{\lfloor \frac{q}{3} \rfloor + 2, \ldots, q - \lfloor \frac{q}{3} \rfloor\}$, i.e. j is part of the cycle, and is at least $\lfloor \frac{q}{3} \rfloor$ cycle-hops away from i. Accordingly, we can rewrite the Schatten p-norm as

$$\|A\|_{S_p}^p = \sum_{i,j} \sum_{\lfloor \frac{q}{3} \rfloor + 2 \leq l \leq q - \lfloor \frac{q}{3} \rfloor} \sum_{\substack{(i,i_2,\ldots,i_q) \\ \in \Gamma_S^{(i,j)}(i):\ i_l = j}} c(i,i_2,\ldots,i_q)\sigma(i,i_2,\ldots,i_q,i). \tag{2.6}$$

We are now ready to present our estimator and an algorithm implementing it. In the algorithm, instead of summing over all $i, j \in [n]$, we sample two multisets I, J and do a BFS of depth $\lfloor q/3 \rfloor$ from each $i \in I$ and $j \in J$, and eventually enumerate over all cycles involving these i, j as in (2.6).

Algorithm 4 Two-Set based Algorithm for Schatten p-Norm of k-Sparse Matrices for $p \in 2\mathbb{Z}_{\geq 2}$ in Row-Order Stream

Input: $A \in \mathbb{R}^{n\times n}$ streamed in row-order, $p \in 2\mathbb{Z}_{\geq 2}$, $\epsilon > 0$.

1: $r \leftarrow O(\epsilon^{-3} q^{5/2} k^{3p-6}\sqrt{n})$, $Y \leftarrow 0$.
2: **in parallel** $2r$ **times do**
3: **in pass** 1 **do**
4: sample a row $i \in [n]$ with probability $\tau_i = \frac{\|a_i\|_2^q}{\sum_m \|a_m\|_2^q}$ ▷ Using Reservoir Sampling
5:
6: **in pass** $2 \leq t \leq \lfloor q/3 \rfloor + 1$ **do**
7: store all rows of hop-distance at most $t-1$ from i that have l_2-norm smaller than row i
8:
9:
10: let multisets I and J contain the first and last r samples (from line 4), respectively
11: **for each** $(i,j) \in I \times J$ such that $\left(\frac{\epsilon}{qk^{2\lceil q/2 \rceil}}\right)^{3/p} \|a_i\|_2 \leq \|a_j\|_2 \leq \|a_i\|_2$ **do**

$$Y \mathrel{+}= \frac{1}{\tau_i \cdot \tau_j} \sum_{\lfloor \frac{q}{3} \rfloor + 2 \leq l \leq q - \lfloor \frac{q}{3} \rfloor} \sum_{\substack{(i,i_2,\ldots,i_q) \\ \in \Gamma_S^{(i,j)}(i):\ i_l = j}} c(i, i_2, \ldots, i_q)\sigma(i, i_2, \ldots, i_q, i)$$

12: **return** $\bar{Y} = \frac{1}{r^2} Y$

Theorem 2.5.1. *There exists an algorithm that, given $p \in 2\mathbb{Z}_{\geq 2}$, $\epsilon > 0$ and a k-sparse matrix $A \in \mathbb{R}^{n\times n}$ that is streamed in row-order, makes $\lfloor \frac{p}{6} \rfloor + 1$ passes over the stream using at most $O_p(\epsilon^{-3} k^{4p}\sqrt{n})$ words of space, and then outputs $\bar{Y}(A)$ that $(1 \pm 2\epsilon)$-approximates $\|A\|_{S_p}^p$ with probability at least $2/3$.*

Before the proof, we state the following theorem, which can be viewed as a variant of the Importance Sampling lemma (Theorem 2.2.2).

Lemma 2.5.2 (Two-Set Sampling)**.** *Let $z = \sum_{i,j\in[n]} z_{i,j} > 0$ for $n \geq 1$, and suppose the matrix defined by $\{z_{i,j}\}$ is Δ-sparse.*[5] *Let $I, J \in [n]$ be two random multisets of size r,*

[5] Δ can be viewed as an upper bound on the in-degrees and out-degrees of the directed graph defined by edge weights z_{ij} on vertex set $[n]$.

where each of their $2r$ *elements is chosen independently with replacement according to the distribution* $(\tau_l : l \in [n])$*. Consider the estimator*

$$Y = \frac{1}{r^2} \sum_{i \in I, j \in J} \frac{z_{i,j}}{\tau_i \cdot \tau_j}.$$

If $\lambda > 0$ *satisfies that for all* $i, j \in [n]$ *both* $\tau_i, \tau_j \geq \frac{1}{\lambda}\sqrt{\frac{|z_{i,j}|}{z}}$*, then*

$$\mathbf{E}[Y] = z \quad \textit{and} \quad Var(Y) \leq \left(\frac{\lambda^2}{r^2} + \frac{2\lambda\Delta}{r}\right) z \sum_{i,j \in [n]} |z_{i,j}|. \tag{2.7}$$

The proof of Lemma 2.5.2 is given in Appendix A.3. We now proceed to the proof of Theorem 2.5.1, remarking that $k^{O(p)}$ factor can be improved by using the Projection Lemmas, but for simplicity we use more straightforward arguments.

Proof of Theorem 2.5.1. First we remark that indeed in $\lfloor q/3 \rfloor + 1$ passes all the needed rows of a cycle are kept. For any cycle, row i needs to "cover" $\lfloor q/3 \rfloor + 1 + (q - (q - \lfloor q/3 \rfloor)) = 2\lfloor q/3 \rfloor + 1$ rows (including itself), which indeed happens as we do a BFS of size $\lfloor q/3 \rfloor$. Row j must cover at most $q - \lfloor q/3 \rfloor - (\lfloor q/3 \rfloor + 2) = q - 2\lfloor q/3 \rfloor - 2$ rows, including itself. As $\lfloor q/3 \rfloor + 1 \geq q - 2\lfloor q/3 \rfloor - 2$, we indeed again cover all possibly needed rows in the $\lfloor q/3 \rfloor + 1$ passes. We now go on to prove the approximation bounds. Let $\beta := \left(\frac{\epsilon}{qk^{p-2}}\right)^{3/p}$ and define for all $i, j \in [n]$

$$z_{i,j} := \begin{cases} \sum_{\lfloor \frac{q}{3} \rfloor + 2 \leq l \leq q - \lfloor \frac{q}{3} \rfloor} \sum_{\substack{(i,i_2,\ldots,i_q) \\ \in \Gamma_S^{(i,j)}(i):\ i_l = j}} c(i, i_2, \ldots, i_q)\sigma(i, i_2, \ldots, i_q, i) & \text{if } \|a_j\|_2 \leq \|a_i\|_2; \\ 0 & \text{otherwise.} \end{cases}$$

Then, by Equation (2.6), $z' := \sum_{i,j} z_{i,j} = \|A\|_{S_p}^p$. Since line 11 in the algorithm considers

only pairs (i,j) where $\frac{\|a_j\|_2}{\|a_i\|_2} \in [\beta, 1]$, we further define

$$z := \sum_{i,j:\ \frac{\|a_j\|_2}{\|a_i\|_2} \in [\beta,1]} z_{i,j}.$$

Let us show that the omitted terms do not contribute much to $z' = \|A\|_{S_p}^p$, and thus the error introduced by omitting them is small. For simplicity assume $q/3 \in \mathbb{N}$, then

$$\begin{aligned}
\left|z' - z\right| &\leq \sum_i \sum_{j:\ \frac{\|a_j\|_2}{\|a_i\|_2} \leq \beta} \left|z_{i,j}\right| \\
&\leq \sum_i \sum_{j:\ \frac{\|a_j\|_2}{\|a_i\|_2} \leq \beta} \sum_{\lfloor \frac{q}{3} \rfloor + 2 \leq l \leq q - \lfloor \frac{q}{3} \rfloor} \sum_{\substack{(i,i_2,\ldots,i_q) \\ \in \Gamma_S^{(i,j)}(i):\ i_l = j}} c(i,i_2,\ldots,i_q) \left|\sigma(i,i_2,\ldots,i_q,i)\right|
\end{aligned}$$

As $c(i,i_2,\ldots,i_q) \leq q$, and using the conditions on i and j we get

$$\leq q \sum_i \sum_{j:\ \frac{\|a_j\|_2}{\|a_i\|_2} \leq \beta} \sum_{\lfloor \frac{q}{3} \rfloor + 2 \leq l \leq q - \lfloor \frac{q}{3} \rfloor} \sum_{\substack{(i,i_2,\ldots,i_q) \\ \in \Gamma_S^{(i,j)}(i):\ i_l = j}} \|a_i\|_2^{2p/3} \|a_j\|_2^{p/3}$$

As each row has at most k^2 "neighboring" rows,

$$\leq k^{2(q-1)} q \beta^{p/3} \sum_i \|a_i\|_2^p = \epsilon \sum_i \|a_i\|_2^p.$$

Therefore, using Fact 2.2.1, we conclude

$$|z - \|A\|_{S_p}^p| \leq \epsilon \|A\|_{S_p}^p. \tag{2.8}$$

We proceed to show that the standard deviation of our estimator is bounded by ϵz, meaning that w.h.p $\bar{Y} \in (1 \pm \epsilon) z$, and together with (2.8) this yields $\bar{Y} \in (1 \pm 2\epsilon) \|A\|_{S_p}^p$. To

this end, we want to use Lemma 2.5.2 and thus wish to show that

$$\sum_{i,j}|z_{i,j}| \leq 2qk^{2\lceil q/2\rceil}z \tag{2.9}$$

and that $\lambda \coloneqq \sqrt{2qk^{p-4}\frac{n}{\beta^{2p/3}}} = \sqrt{2}q^{3/2}k^{3p/2-4}\frac{\sqrt{n}}{\epsilon}$ satisfies

$$\frac{|z_{i,j}|}{z} \leq \lambda^2\tau_j^2 \qquad \forall i,j \in [n]. \tag{2.10}$$

meaning that . We remark that (2.10) is indeed sufficient, as $\tau_j \leq \tau_i$, as otherwise $z_{i,j} = 0$ and the inequality trivially holds.

To prove (2.9), we use similar arguments as above, together with (2.8),

$$\begin{aligned}
\sum_{i,j}|z_{i,j}| &\leq q \cdot \sum_{i,j:\ \frac{\|a_j\|_2}{\|a_i\|_2}\in[\beta,1]} \sum_{\lfloor\frac{q}{3}\rfloor+2\leq l\leq q-\lfloor\frac{q}{3}\rfloor} \sum_{\substack{(i,i_2,\ldots,i_q)\\ \in\Gamma_S^{(i,j)}(i):\ i_l=j}} \|a_i\|_2^p \\
&\leq qk^{p-2}\sum_i \|a_i\|_2^p \\
&\leq 2qk^{p-2}z.
\end{aligned}$$

To prove (2.10), fix i,j such that $\frac{\|a_j\|}{\|a_i\|} \in [\beta,1]$, then by similar arguments, together with (2.8) and Fact 2.2.1,

$$\begin{aligned}
\frac{|z_{i,j}|}{z} &\leq \frac{1}{z}\sum_{\lfloor\frac{q}{3}\rfloor+2\leq l\leq q-\lfloor\frac{q}{3}\rfloor} \sum_{\substack{(i,i_2,\ldots,i_q)\\ \in\Gamma_S^{(i,j)}(i):\ i_l=j}} c(i,i_2,\ldots,i_q)|\sigma(i,i_2,\ldots,i_q,i)| \\
&\leq \frac{1}{z}\sum_{\lfloor\frac{q}{3}\rfloor+2\leq l\leq q-\lfloor\frac{q}{3}\rfloor} \sum_{\substack{(i,i_2,\ldots,i_q)\\ \in\Gamma_S^{(i,j)}(i):\ i_l=j}} q\|a_i\|_2^{2p/3}\|a_j\|_2^{p/3} \\
&\leq qk^{p-4}\frac{\|a_j\|_2^p}{\beta^{2p/3}z}
\end{aligned}$$

$$\leq 2qk^{p-4}\frac{\|a_j\|_2^p}{\beta^{2p/3}\|A\|_{S_p}^p}$$
$$\leq 2qk^{p-4}\frac{\|a_j\|_2^p}{\beta^{2p/3}\sum_m \|a_m\|_2^p}$$

using norm properties (by applying $\|v\|_q \leq n^{\frac{1}{q}-\frac{1}{p}}\|v\|_p$ to the vector $v = (\|a_1\|_2, \ldots, \|a_n\|_2)$),

$$\leq qk^{p-4}\frac{\|a_j\|_2^p}{\beta^{2p/3}(\sum_m \|a_m\|_2^q)^2/n}$$
$$\leq 2qk^{p-4}\frac{n}{\beta^{2p/3}} \cdot \tau_j^2.$$

We further note that for $z_{i,j}$ to be non-zero, row j must be at distance at most $\lceil q/2 \rceil$ from row i, and thus each row can participate in at most $k^{2\lceil q/2 \rceil}$ different non-zero $z_{i,j}$, i.e., $\Delta \leq k^{p/2-2}$. Combining all the above, we conclude that setting $r = O(\epsilon^{-2}\lambda\Delta) \cdot 2qk^{p-2} = O(\epsilon^{-3}q^{5/2}k^{3p-6}\sqrt{n})$ will give w.h.p a $(1 \pm 2\epsilon)$-approximation to the Schatten p-norm by Chebyshev's inequality.

As for each row in $I \cup J$ the algorithm stores neighborhoods of size $O\left((k^2)^{q/3}\right)$, and storing each row in the neighborhood takes $O(k)$ words, there is an extra factor of $k^{p/3+1}$. Thus the total space is $O(\epsilon^{-3}q^{5/2}k^{10p/3-5}\sqrt{n})$ words. □

Remark. As mentioned earlier, the BFS approach can be replaced with the adaptive sampling approach from previous sections. For the first r samples (in I), the algorithm adaptively samples two paths of hop-distance (roughly) $q/3$, similarly to Section 2.4.3. For each of the last r samples (in J), the algorithm chooses $\rho \in [q/3]$ uniformly at random (and independently of all other steps), and adaptively samples a path of hop-distance ρ and a path of hop-distance (roughly) $\frac{q}{3} - \rho$. It then tries to "stitch" these paths to create q-cycles. The bound on λ (i.e. (2.10)) increases by a factor of $q/3$ due to ρ (this can be viewed as replacing the BFS with multiple random paths), but as the algorithm does not keep the entire neigh-

borhoods, a $k^{p/3}$ factor is shaved from the space complexity. This, together with a tighter analysis, can improve the dependence on k in Theorem 2.5.1 to $k^{19p/8+O(1)}$.

2.5.2 General s (using $\lfloor \frac{p}{2(s+1)} \rfloor + 1$ Passes)

We generalize the algorithm from the previous subsection, such that given some integer $s \in [2, p-1]$, the algorithm samples in parallel in the first pass $r \cdot s$ rows for $r = O_{p,\epsilon,s}(k^{4p} n^{1-1/s})$, where each "seed" row i is sampled with probability $\tau_i = \frac{\|a_i\|_2^{p/s}}{\sum_m \|a_m\|_2^{p/s}}$. In the following passes it runs a BFS of depth (roughly) $\frac{q}{s+1}$, keeping all the shorter rows (in l_2-norm) in the neighborhood of each seed. The first r samples are denoted as multiset I, and the other samples are split into $s-1$ multisets of size r denoted as $J_1, \ldots, J_{s-1}$. The algorithm then considers s-tuples $(i, j_1, \ldots, j_{s-1})$ where $i \in I$ and every row $j_u \in J_u$ has l_2-norm in the range $(\beta', 1)$ relative to that of row i, for $\beta' \approx \left(\frac{\epsilon}{sqk^p}\right)^{(s+1)/p}$. The estimator is formed by looking at the eligible s-tuples, and for each such tuple adding the contributions of all the q-cycles obtained by "stitching" paths of hop-distance (roughly) $\frac{q}{s+1}$ passing through these seeds, as follows:

$$Y \mathrel{+}= \frac{1}{\tau_i \tau_{j_1} \cdots \tau_{j_{s-1}}} \sum_{\frac{q}{s+1} \le l_1 \le \frac{2q}{s+1}} \cdots \sum_{\frac{(s-1)q}{s+1} \le l_{s-1} \le \frac{s \cdot q}{s+1}} \sum_{\substack{(i,i_2,\ldots,i_q) \\ \in \Gamma_S^{(i,j_1,\ldots,j_{s-1})}(i): \\ i_{l_1}=j_1,\ldots,i_{l_{s-1}}=j_{s-1}}} c(i, i_2, \ldots, i_q) \sigma(i, i_2, \ldots, i_q, i).$$

The algorithm's final output is $\bar{Y} = \frac{1}{r^s} Y$.

Theorem 2.5.3. *There exists an algorithm that, given $p \in 2\mathbb{Z}_{\ge 2}$, $\epsilon > 0$, an integer $s \in [2, p-1]$ and a k-sparse matrix $A \in \mathbb{R}^{n \times n}$ streamed in row-order, makes $\lfloor \frac{p}{2(s+1)} \rfloor + 1$ passes over the stream using $O_p\left(\epsilon^{-3} k^{2ps} n^{1-\frac{1}{s}}\right)$ words of space, and outputs $\bar{Y}(A)$ that $(1 \pm 2\epsilon)$-approximates $\|A\|_{S_p}^p$ with probability at least $2/3$.*

Proof Sketch. The proof follows similar steps as the proof for $s = 2$. First, the error introduced by taking only certain cycles changes, as now we miss cycles in which at least one of the sampled j_u is smaller than β'. However their total contribution can be bounded by $(s-1)(\beta')^{p/(s+1)}qk^{p-2} < \epsilon$ relative to $\|A\|_{S_p}^p$. Next, an s-Set Sampling Lemma is proved using the same arguments as the Two-Set Sampling Lemma. It asserts that the estimator

$$Y = \frac{1}{r^s} \sum_{i \in I, j_1 \in J_1, \ldots, j_{s-1} \in J_{s-1}} \frac{z_{i,j_1,\ldots,j_{s-1}}}{\tau_i \tau_{j_1} \cdots \tau_{j_{s-1}}}$$

is unbiased, and that if $\lambda > 0$ satisfies that for every $i, j_1, \ldots, j_{s-1} \in [n]$, all $\tau_i, \tau_{j_1}, \ldots, \tau_{j_{s-1}} \geq \frac{1}{\lambda}\left(\frac{|z_{i,j_1,\ldots,j_{s-1}}|}{z}\right)^{1/s}$, then

$$\mathrm{Var}(Y) \leq O\left(\left(\Delta + \frac{\lambda}{r}\right)^s - \Delta^s\right) z \sum_{i,j_1,\ldots,j_{s-1} \in [n]} |z_{i,j_1,\ldots,j_{s-1}}|.$$

The proof for the inequality analogous to (2.9), which bounds the ratio between the absolute sum of $z_{i,j_1,\ldots,j_{s-1}}$ and z, is the same. To prove the bound λ (i.e. analogous to (2.10)), we need to bound the shortest j_u among rows $(j_1, \ldots, j_{s-1})$. To do so we first bound all "seeds" except j_u using row i, and then use the same arguments that result in $\lambda = \left(C_\epsilon q k^p \frac{n^{s-1}}{(\beta')^{2p/(s+1)}}\right)^{1/s}$ for a suitable constant C dependent on ϵ. Finally, now each i can have $(s-1)k^{2\lceil q/2 \rceil}$ different $(j_1, \ldots, j_{s-1})$, i.e. $\Delta \leq (s-1)k^{q+2}$. Picking $r = O\left(\epsilon^{-3} s \Delta^{s-1} \lambda\right)$ results in the desired approximation.

The space complexity analysis is as in the proof of Theorem 2.5.1, resulting in

$$O\left(\epsilon^{-3}(s-1)^s \cdot q^{2+1/s} \cdot k^{p(s/2+11/6+1/s)+2s-O(1)} \cdot n^{1-1/s}\right)$$

words of space. □

2.6 Lower Bound for One-Pass Algorithms in the Row-Order Model

We show a space lower bound of $\Omega(n^{1-4/\lfloor p \rfloor_4})$ bits for one-pass algorithms and even p values in the row-order model. Our main technical contribution is the analysis of even p values in a reduction presented in [99], based on the Boolean Hidden Hypermatching [25, 150]. Although this is not mentioned in [99], it can easily be verified from the proof of [99, Theorem 3] (stated below as Theorem 2.6.1) that this reduction applies also to the row-order model.[6] Our bound is closely related to the $\Omega(n^{1-1/\varepsilon})$ bits lower bound for $p \notin 2\mathbb{Z}$, proved in [23], and is also nearly tight with the upper bound from the same paper (see discussion at the end of this section).

We first recall the definitions presented in [99]. Let $D_{m,l}$ (for $0 \leq l \leq m$) be an $m \times m$ diagonal matrix with the first l diagonal elements equal to 1 and the remaining diagonal entries equal to 0, and let $\mathbf{1}_m$ be an m-dimensional vector full of 1s, thus $\mathbf{1}_m \mathbf{1}_m^\top$ is the $m \times m$ all-ones matrix. Define

$$M_{m,l}(\gamma) = \begin{pmatrix} \mathbf{1}_m \mathbf{1}_m^\top & 0 \\ \sqrt{\gamma} D_{m,l} & 0 \end{pmatrix},$$

where $\gamma > 0$ is a constant (which may depend on m).

Let $m \geq 2$ be an even integer, and let $p_m(l) \coloneqq \binom{m}{l}/2^{m-1}$ for $0 \leq l \leq m$. Let $\mathcal{E}(m)$ be the probability distribution defined on even integers $\{0, 2, \ldots, m\}$ with probability density function $p_m(l)$. Similarly, let $\mathcal{O}(m)$ be the distribution on odd integers $\{1, 3, \ldots, m-1\}$ with density function $p_m(l)$. We say that a function f on square matrices is *diagonally block-additive* if $f(X) = f(X_1) + \ldots + f(X_s)$ for any block diagonal matrix X with square blocks

[6] In fact, also Theorem 4 in [99] applies to row-order streams, providing a different proof for the $\Omega(n^{1-\varepsilon})$ lower bound for $p \notin 2\mathbb{Z}$ proved in [23].

$X_1, \ldots, X_s$. As noted in [99], $f(X) = \|X\|_{S_p}^p$ is diagonally block-additive.

We observe that the reduction presented in [99] is applicable also to row-order streams, and thus state below a slightly stronger version of Theorem 3 from that paper.

Theorem 2.6.1 (Theorem 3 in [99]). *Let t be an even integer and let f be a function of square matrices that is diagonally block-additive. If there exists $m = m(t)$ and $\gamma = \gamma(m) > 0$, such that the following "gap condition" holds:*

$$\mathbf{E}[l \sim \mathcal{E}(t)] f\left(M_{m,l}(\gamma)\right) - \mathbf{E}[l \sim \mathcal{O}(t)] f\left(M_{m,l}(\gamma)\right) \neq 0, \tag{2.11}$$

*then there exists a constant $\varepsilon = \varepsilon(t) > 0$ such that every **row-order** streaming algorithm that, given $X \in \mathbb{R}^{N \times N}$ (for sufficiently large N), approximates $f(X)$ within factor $1 \pm \varepsilon$ with constant error probability, must use $\Omega_t(N^{1-1/t})$ bits of space.*

We can now present our analysis for even p values.

Lemma 2.6.2. *Let $f(X) = \|X\|_{S_p}^p$, for $p \in 4\mathbb{Z}_{\geq 1}$. Then the gap condition (2.11) is satisfied, under the choice $m = t$ and $\gamma = 1$, if and only if $t \leq p/4$.*

Proof. As shown in the proof of Theorem 4 in [99], for $m = t$ and $\gamma = 1$, the non-zero singular values of a block $M_{t,l}(1)$ are as follows. For $l = 0$, the only non-zero singular value is t. For $0 < l < t$, the non-zero singular values are $r_1(l) = \sqrt{\frac{(t^2+1)+\sqrt{(t^2-1)^2+4lt}}{2}}$, $r_2(l) = \sqrt{\frac{(t^2+1)-\sqrt{(t^2-1)^2+4lt}}{2}}$ and 1 with multiplicity $l-1$. And for $l = t$, the non-zero singular values are $r_1(t) = \sqrt{\frac{(t^2+1)+\sqrt{(t^2-1)^2+4t^2}}{2}}$ and 1 with multiplicity $t-1$. Further note that that $r_2(t) = 0$. Using this, and recalling the distribution of the blocks, the left-hand side of the gap condition (2.11) is

$$\frac{1}{2^{t-1}} \left[t^p + \sum_{\text{even } l} \binom{t}{l} \left((l-1) + r_1^p(l) + r_2^p(l)\right) - \sum_{\text{odd } l} \binom{t}{l} \left((l-1) + r_1^p(l) + r_2^p(l)\right) \right] \tag{2.12}$$

and we can rewrite this as

$$\frac{1}{2^{t-1}}\left[t^p+\sum_{0<l\leq t}\binom{t}{l}(-1)^l(l-1)+\sum_{0<l\leq t}\binom{t}{l}(-1)^l\left(r_1^p(l)+r_2^p(l)\right)\right].$$

For the first sum, by Corollary 2 in [133], we know that

$$\sum_{l=0}^{t}(-1)^l\binom{t}{l}(l-1)=0$$

meaning that

$$\sum_{0<l\leq t}\binom{t}{l}(-1)^l(l-1)=1.$$

Let $q=p/2$ and $\zeta=r_1^p(l)+r_2^p(l)$. It holds that

$$\zeta=\left(\frac{(t^2+1)+\sqrt{(t^2-1)^2+4lt}}{2}\right)^q+\left(\frac{(t^2+1)-\sqrt{(t^2-1)^2+4lt}}{2}\right)^q$$

and using the binomial theorem,

$$=\frac{1}{2^q}\left[\sum_{i=0}^{q}\left(t^2+1\right)^{q-i}\left(\sqrt{(t^2-1)^2+4lt}\right)^i+\sum_{i=0}^{q}(-1)^i\left(t^2+1\right)^{q-i}\left(\sqrt{(t^2-1)^2+4lt}\right)^i\right].$$

We note that the alternating sum double the even values the zero out the odd values, thus the above can be rewritten as

$$=\frac{1}{2^{q-1}}\sum_{\text{even } i}\binom{q}{i}\left(t^2+1\right)^i\left((t^2-1)^2-4lt\right)^{\frac{q-i}{2}}.$$

and by applying it again, on the second multiplicative term,

$$=\frac{1}{2^{q-1}}\sum_{\text{even } i}\binom{q}{i}\left(t^2+1\right)^i\sum_{j=0}^{\frac{q-i}{2}}\binom{\frac{q-i}{2}}{j}\left(t^2-1\right)^{2j}\cdot(4t)^{\frac{q-i}{2}-j}\cdot l^{\frac{q-i}{2}-j}.$$

Combining the two insights results in (2.12), i.e.,

$$\frac{1}{2^{t-1}}\left[t^p+1+\sum_{l=1}^{t}(-1)^l\left(\frac{1}{2^{q-1}}\sum_{\text{even } i}\binom{q}{i}(t^2+1)^i\sum_{j=0}^{\frac{q-i}{2}}\binom{\frac{q-i}{2}}{j}(t^2-1)^{2j}\,(4tl)^{\frac{q-i}{2}-j}\,l^{\frac{q-i}{2}-j}\right)\right].$$

We further note that for $l=0$, the term in the inner parentheses is non-zero only when $\frac{q-i}{2}=j$. In this case we get, using the binomial theorem once more,

$$\frac{1}{2^{q-1}}\sum_{\text{even } i}\binom{q}{i}(t^2+1)^i(t^2-1)^{q-i}=\left(\frac{t^2+1+t^2-1}{2}\right)^q+\left(\frac{t^2+1-t^2+1}{2}\right)^q=1+t^p.$$

Therefore, we can rewrite (2.12) as

$$(2.12)=\frac{1}{2^{t-1}}\left(\sum_{l=0}^{t}(-1)^l\frac{1}{2^{q-1}}\sum_{\text{even } i}\binom{q}{i}(t+1)^i\sum_{j=0}^{\frac{q-i}{2}}\binom{\frac{q-i}{2}}{j}(t-1)^{2j}4^{\frac{q-i}{2}-j}l^{\frac{q-i}{2}-j}\right)$$

and using [99] observation,

$$=\frac{1}{2^{t-1}}(-1)^t t!\sum_{\text{even } i}\binom{q}{i}(t+1)^i\sum_{j=0}^{\frac{q-i}{2}}\binom{\frac{q-i}{2}}{j}(t-1)^{2j}4^{\frac{q-i}{2}-j}\left\{\begin{matrix}\frac{q-i}{2}\\ t\end{matrix}\right\}$$

where $\left\{\begin{matrix}\frac{q-i}{2}\\ t\end{matrix}\right\}$ are Stirling numbers of the second kind. As for a fixed t all terms are of the same sign, the sum vanishes only when $\left\{\begin{matrix}\frac{q-i}{2}\\ t\end{matrix}\right\}=0\ \forall i$, which happens when $t>q/2=p/4$. □

We remark that Lemma 2.6.2 extends to $p\equiv 2 \pmod 4$ when $t\leq(p-2)/4$, by replacing in the proof $q=p/2$ with $\tilde{q}=(p-2)/2$. The next theorem follows easily by combining Theorem 2.6.1 and Lemma 2.6.2.

Theorem 2.1.2. *For every $p\in 2\mathbb{Z}_{\geq 2}$ there is $\epsilon(p)>0$ such that every algorithm that makes*

one pass over an $O_p(1)$-sparse matrix $A \in \mathbb{R}^{n\times n}$ streamed in row-order, and then outputs a $(1\pm\epsilon(p))$-approximation to $\|A\|_{S_p}^p$ with probability at least $2/3$, must use $\Omega(n^{1-4/\lfloor p\rfloor_4})$ bits of space.

Proof. Let us first assume that $p \equiv 0 \pmod 4$. As shown in Lemma 2.6.2, the gap condition (2.11) holds for $f(X) = \|X\|_{S_p}^p$ and $t = p/4$, thus by Theorem 2.6.1 the space complexity is $\Omega(n^{1-1/t}) = \Omega(n^{1-4/p})$ bits. For $p \equiv 2 \pmod 4$ the claim holds for $t = (p-2)/4$, yielding an $\Omega(n^{1-4/(p-2)})$ bits lower bound. □

We note that for $p \equiv 0 \pmod 4$ the above matches up to logarithmic factors the upper bound for the row-order algorithm presented in [23], i.e. tight for matrices in which every row and column has $O(1)$ non-zero elements. For $p \equiv 2 \pmod 4$, there is a small gap: the lower bound is $\Omega(n^{1-4/(p-2)})$ while the upper bound obtained in [23] is $\tilde{O}_k(n^{1-4/(p+2)})$.

2.7 $O_\epsilon(1)$-Space Algorithm for Schatten 4-Norm of General Matrices

We present an $O(1/\epsilon^2)$-space algorithm for $(1+\epsilon)$-approximation of the Schatten 4-norm in the row-order model. As this result does not depend on the sparsity and is applicable to any matrix, it significantly improves the previously known row-order algorithm, presented in [23] that uses space $\tilde{O}_{p,\varepsilon}(k)$, and is also better than the result of Section 2.4.1.

The algorithm exploits the fact that $A^\top A = \sum_i a_i^\top a_i$ (i.e. summing over the outer product of every row with itself), to sketch the Frobenius norm $\sum_{j_1,j_2}((A^\top A)_{j_1,j_2})^2 = \|A^\top A\|_F^2 = \|A\|_{S_4}^4$. To do so, it uses two random 4-wise independent vectors, following an idea presented in [82] (extending the classic [5] result, see also [20, 22]), as follows.

Lemma 2.7.1 (Lemma 3.1 in [82]). *Consider random $h, g \in \{-1, 1\}^n$ where each vector is 4-wise independent (and independent of the other one). Let $v \in \mathbb{R}^{n^2}$ and $z_j = h_{j_1} g_{j_2}$ for $j \in [n]^2$, and define $\Upsilon = (\sum_{j \in [n]^2} z_j v_j)^2$. Then*

$$\mathbf{E}[\Upsilon] = \sum_{j \in [n]^2} v_j^2, \quad \textit{and} \quad \textit{Var}(\Upsilon) \leq 3(\mathbf{E}[\Upsilon])^2.$$

Algorithm 5 Algorithm for Schatten 4-Norm of General Matrices in Row-Order Streams

Input: $A \in \mathbb{R}^{n \times n}$ streamed in row-order, $\epsilon > 0$.

1: **in parallel** $m = \tilde{O}(1/\epsilon^2)$ **times do**
2: init: $Y \leftarrow 0$ and choose two random 4-wise independent vectors $h, g \in \{-1, 1\}^n$
3: upon receiving row a_i, update: $Y \mathrel{+}= \langle h, a_i \rangle \langle g, a_i \rangle$
4: let $\Upsilon \leftarrow Y^2$
5:
6: **return** average of the m copies of Υ

Theorem 2.7.2. *Suppose that $A \in \mathbb{R}^{n \times n}$ is a matrix given in one-pass row-order model. Algorithm 5 uses space $O(1/\epsilon^2)$ and outputs a $(1+\epsilon)$-approximation to $\|A\|_{S_4}^4$ with probability at least $2/3$.*

Proof. Consider one copy of the independent sketches. Using simple manipulations, we can write:

$$Y = \sum_i \left(\sum_{j_1} h_{j_1} A_{i,j_1} \right) \left(\sum_{j_2} g_{j_2} A_{i,j_2} \right) = \sum_{j_1, j_2} h_{j_1} g_{j_2} (A^\top A)_{j_1, j_2}$$

By looking at $A^\top A$ as vector of dimension n^2, it easily follows from 2.7.1 that $\mathbf{E}[\Upsilon] = \|A^\top A\|_F^2 = \|A\|_{S_4}^4$ and $\text{Var}(\Upsilon) \leq 3\|A\|_{S_4}^8$. Repeating the sketch $O(1/\epsilon^2)$ times independently, decreases the variance and gives the desired result (by Chebyshev's inequality). □

2.8 Applications

In this section we present two applications of our Schatten-norm algorithms to some common functions of the spectrum, by approximating these functions using low-degree polynomials. We employ the well-known idea that just as functions $f : \mathbb{R} \to \mathbb{R}$ can be approximated in a specific interval by polynomials arising from a Taylor expansion (or using Chebyshev polynomials), spectral functions can be approximated by matrix polynomials if the matrix eigenvalues lie in a bounded range. We just need to implement this method in a space-efficient streaming fashion. In some applications we require the input matrix to have a bounded spectrum. Unfortunately, there is no known streaming algorithm to estimate the spectrum of an input matrix.

2.8.1 Approximating Spectral Sums of Positive Definite Matrices

We demonstrate how our Schatten-norm estimators can be used to approximate commonly used spectral functions of sparse matrices presented as a data stream. We consider three different spectral functions, log-determinant, trace of matrix inverse and Estrada index of a Laplacian matrix, that all belong to the class of spectral sums, as defined below. These results apply to sparse matrices that are either positive definite (PD), positive semidefinite (PSD), or Laplacian. Throughout, $\log x$ denotes the natural logarithm of x.

Definition 2.8.1 (Spectral Sums [75])**.** Given a function $f : \mathbb{R} \to \mathbb{R}$ and a matrix $A \in \mathbb{R}^{n \times n}$ with real eigenvalues $\lambda_1, \ldots, \lambda_n$, a spectral sum is defined as

$$S_f(A) = \mathrm{tr}(f(A)) = \sum_{i=1}^{n} f(\lambda_i).$$

When $f(x) = \log x$, the sum is known as log-determinant, when $f(x) = 1/x$ it is known

as the trace of the matrix inverse, and when $f(x) = \exp(x)$ it is known as Estrada index.

Theorem 2.8.2. *For every spectral function S_f from the table below, there is an algorithm with the following guarantee. Given as input $\epsilon, \theta > 0$, and a k-sparse matrix $A \in \mathbb{R}^{n \times n}$ presented as a row-order stream and whose eigenvalues all lie in the interval I_f given in the table, the algorithm makes $\lfloor m_f/4 \rfloor + 1$ passes over the stream using W_f words of space and outputs an estimate $\rho(A)$ such that*

$$\Pr\left[\rho(A) \in (1 \pm 2\epsilon) S_f(A)\right] \geq 2/3.$$

S_f *Spectral Function*	I_f *Spectrum Interval*	m_f	W_f *Words of Space*
Log-Determinant	$[\theta, 2)$	$\lceil \frac{1}{\theta} \cdot \log \frac{1}{\epsilon} \rceil$	$O_{m_f}(\epsilon^{-2} k^{\frac{3m_f}{2}-3})$
Trace of Matrix Inverse	$[\theta, 2)$	$\lceil \frac{1}{\theta} \cdot \log \frac{1}{\epsilon} \rceil$	$O_{m_f}(\epsilon^{-2} k^{\frac{3m_f}{2}-3})$
Estrada Index of a Laplacian	$[0, \theta]$ [7]	$\lceil (e\theta + 1) \log \frac{1}{\varepsilon} - 1 \rceil$	$O_{m_f}(\epsilon^{-2} k^{\frac{m_f}{2}})$

At a high level, the proof follows that of Boutsidis et al. [17], who present a time-efficient algorithm for approximating the log-determinant of PD matrices. Besides extending their method to two other spectral sums, the main difference is that we need to adapt their argument to be space-efficient in the streaming model. More specifically, the log-determinant of a PD matrix is approximated in [17, Lemma 7] by a truncated version (i.e., only the first summands) of its Taylor expansion

$$\log\det(A) = -\sum_{p=1}^{\infty} \mathrm{tr}((I_n - A)^p)/p. \tag{2.13}$$

They then approximate the required matrix traces by multiplying the respective matrix by a Gaussian vector (from left and right), which can be implemented faster than matrix

powering, by starting with the vector and repeatedly multiply it by a matrix. While this is time-efficient, it is not space-efficient when the input matrix is sparse, in which case our streaming algorithm has better space complexity. One other difference to note is that our algorithm approximates each of the above-mentioned traces *separately*, and thus we need all the Taylor expansion coefficients to be non-negative, which indeed applies for these three spectral functions.[8]

Proof. We follow the proof framework of Lemma 8 in [17], achieving the desired relative error of the desired spectral function using a truncated version of its Taylor expansion, consisting m_f terms. The first relative error is due to the tail of the series, i.e. the terms that were not considered in the final estimate. For the log-determinant, the above-mentioned Lemma 8 shows that it suffices to $(1 \pm \epsilon)$-approximate the first $m_f = \lceil \frac{1}{\theta} \cdot \log \frac{1}{\epsilon} \rceil$ terms of its Taylor expansion (2.13) in order to obtain a $(1 \pm 2\epsilon)$-approximation of $\log \det(A)$. The same proof holds also for the Taylor expansion

$$\operatorname{tr}(A^{-1}) = \sum_{p=1}^{\infty} - \operatorname{tr}((I_n - A)^p)$$

and obtaining a $(1 \pm 2\epsilon)$-approximation of $\operatorname{tr}(A^{-1})$ (for the same value of m_f). To achieve this error bound for the Estrada index of a Laplacian, the number of values of its Taylor series (see e.g. [55, 72])

$$\operatorname{tr}(\exp(A)) = \sum_{p=0}^{\infty} \operatorname{tr}(A^p)/p! \tag{2.14}$$

that need to be approximated is $m_f = \lceil (e\theta + 1) \log \frac{1}{\varepsilon} - 1 \rceil$, as shown in Appendix A.4.

We are left to explain how to $(1 \pm \varepsilon)$-approximate the first m_f values of the Taylor expansion (causing the other relative error). Recall that if a matrix B is PSD then $\operatorname{tr}(B^p) =$

[8] Our method extends to alternating Taylor sums if the coefficients decrease by a constant factor, by bounding the approximation error difference of every two consecutive summands. One such an example is $\operatorname{tr}(\exp(-A))$.

$\sum \lambda_i^p = \|B\|_{S_p}^p$, where $\lambda_1, \ldots, \lambda_n \geq 0$ are its eigenvalues. Furthermore, for such matrices our algorithm works for every integer $p \geq 2$, and therefore this relative error is immediate from Theorems 2.1.1 and 2.4.1.

To conclude, we can compute the traces of B^p in parallel for all integer $p = 2, 3, \ldots, m_f$ using Algorithm 1, while for $p = 1$ one can compute $\|B\|_{S_1}^1 = \mathrm{tr}(B)$ by directly summing the main diagonal entries. These parallel executions take $\lfloor m/4 \rfloor + 1$ passes and the total space is at most $O_m(\epsilon^{-2} k^{3m/2-3})$ words of space for log-determinant and trace of matrix inverse, and $O_m(\epsilon^{-2} k^{m/2})$ words of space for the Estrada index of a Laplacian matrix. □

2.8.2 Approximating the Spectrum of PSD matrices

We present an application of our algorithm to (weakly) estimate the spectrum of a matrix, with eigenvalues bounded in $[0, 1]$ using approximations of a "few" Schatten norms of the matrix. This is based on the work of Cohen-Steiner et al. [42] on approximating the spectrum of a graph which is in turn based on insightful work by Wong and Valiant [91] on approximately recovering a distribution from its moments using the Moment Inverse method.

Fix a PSD matrix $A \in \mathbb{R}^{n \times n}$ with eigenvalues $1 \geq \lambda_1 \geq \ldots \geq \lambda_n$ and define the l-th moment of the spectrum to be $\frac{1}{n}\|A\|_{S_l}^l = \frac{1}{n}\sum_{i \in [n]} \lambda_i^l$. Cohen-Steiner et al. show that estimating $O(1/\epsilon)$ moments of A up to multiplicative error $O(\epsilon)$ is sufficient to estimate the spectrum of A within earth-mover distance $O(\epsilon)$. It is well-known that the the L_1 distance between two sorted vectors of length n is exactly n times the earth-mover distance between the corresponding point-mass distributions (uniform probability on each of the n indices). Hence, the recovery scheme of Cohen-Steiner et al. allows us to recover the spectrum within L_1 distance $O(\epsilon n)$ by estimating only $O\left(\frac{1}{\epsilon}\right)$ moments of the matrix A. Specifically, we get the following result,

Theorem 2.8.3 (Theorem 7 in [42]). *Given a constant $\epsilon > 0$, there exists a parameter $s = \frac{C}{\epsilon}$ (where $C > 0$ is an absolute constant) and an algorithm R such that, for a PSD matrix $A \in \mathbb{R}^{n \times n}$ with eigenvalues $\lambda = (\lambda_1, \ldots, \lambda_n) \in [0,1]^n$ and a vector $y \in \mathbb{R}^s$ with the property that $y_i = \|\lambda\|_i^i \pm \exp(-C'\epsilon)$ for all $i \in [s]$ and absolute constant $C' > 0$, R reads y and outputs a vector $\hat{\lambda}$ such that $\|\lambda - \hat{\lambda}\|_1 \leq \epsilon n$.*

For an error parameter $\epsilon > 0$ and parameter $s = \frac{C}{\epsilon}$ (where $C > 0$ is an absolute constant) as defined in the above theorem, given a k-sparse PSD matrix $A \in \mathbb{R}^{n \times n}$ that is streamed in row-order and whose eigenvalues are in the range $[0, 1]$, one can use Algorithm 1 to compute the vector $y \in \mathbb{R}^s$ with the desired guarantee using space $O(k^{3s/2-3} \exp(-C'\epsilon))$ for some absolute constant $C' > 0$ and using $\lfloor s/4 \rfloor + 1$ passes over the stream.

2.9 Experiments

In this section we present numerical experiments illustrating the performance of the row-order Schatten p-norm estimator described in Section 2.4.1. We simulate the row-order stream by reading the input matrix row by row. The results not only follow theoretical space bounds, showing that the algorithm is indeed independent of the matrix size, but are actually several orders of magnitude better. In addition, the experiments show that the algorithm is robust to noise, and these two results suggest that real-life behavior of the algorithm is significantly better than our theoretical bounds.

The inputs used are $\{0, 1\}^{n \times n}$ matrices, representing collaboration network graphs (nodes represent scientists and edges represent co-authoring a paper) from the e-print arXiv for scientific collaborations in five different areas in Physics. The data was obtained from the Stanford Large Network Dataset Collection [96] which was in-turn obtained from [97]. In or-

der to study the effect of sparsity, we "sparsify" each (of five) matrix by sampling 10 nonzero entries in each row uniformly at random (note that max column-sparsity can be larger than 10).

In the first experiment, we use the arXiv General Relativity and Quantum Cosmology collaboration network which has $n = 5242$ rows and columns; after "sparsifying" the matrix as mentioned, the max column-sparsity is 37 and the average sparsity is 6.1. We fix the value of p to be 6, and using our algorithm from Section 2.4.1, we vary number of estimators (walks) t and compute the *relative error* of the average of the t walks. We repeat this process 10 times for every value of t and plot the mean and standard deviation in Figure 2.1. In addition, we show in this figure the results of running the same experiment on a "noisy" version of the matrix, by adding to it an error matrix where 1/5 of the entries are drawn independently from $\mathcal{N}(0, 0.1^2)$[9].

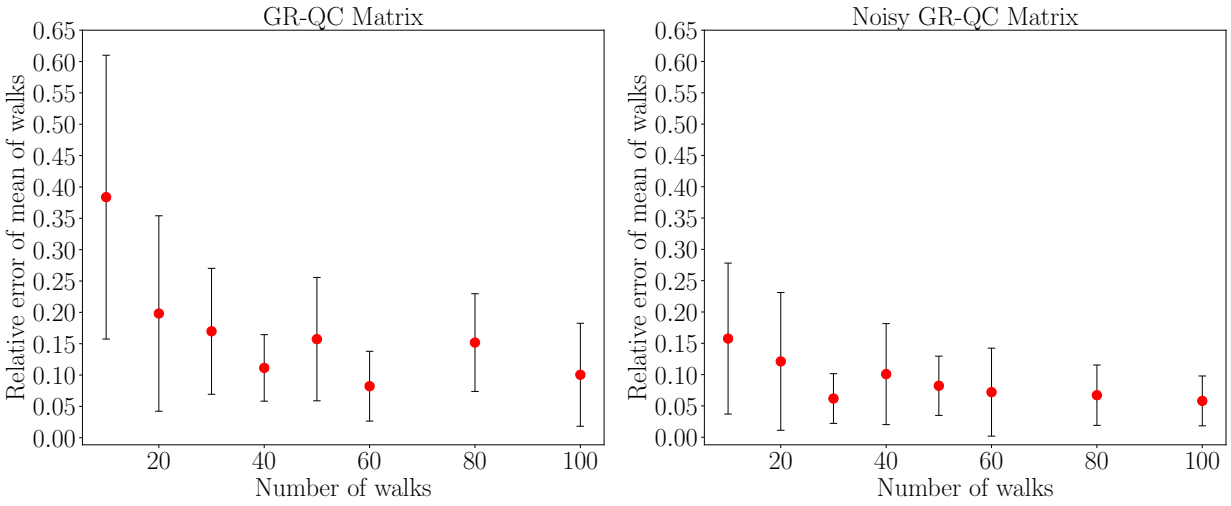

Figure 2.1: Relative error of Algorithm 1 for Schatten 6-norm of arXiv General Relativity and Quantum Cosmology Collaboration Network: Vary number of walks and plot relative error of the mean of the walks.

Recall that the number of independent walks (estimators) is ultimately the space re-

[9]This value assures the l_2-norm of the error in a row is "comparable" to the l_2-norm of the data: $(0.1)^2 \times 5242 \times 0.2 \approx 10 =$ max row-sparsity.

quired by Algorithm 1 (upto the space needed to store a row), as they are run the in parallel. Therefore, the left graph shows that the space actually needed to approximate the Schatten 6-norm of the selected input matrix is significantly smaller than the theoretical bound of Theorem 2.4.1, which is $O_p\varepsilon^{-2}k^(p/2) \approx 135000$. The other graph shows that the algorithm is robust to small random noise, i.e. it works also for nearly-sparse, where every row and column are dominated by a small amount of entries.

In the second experiment, we use all five collaboration networks – General Relativity and Quantum Cosmology ($n = 5242$), High Energy Physics - Phenomenology ($n = 9877$), High Energy Physics - Theory ($n = 12008$), Astro Physics ($n = 18772$) and Condensed Matter ($n = 23133$). For each matrix we compute walks (estimator from Section 2.4.1) until the mean of the walks is within 10% of the true Schatten 6-norm of the matrix. We repeat this process 10 times for each matrix and plot the median, the first and third quartile of the number of walks for the 10 trials in Figure 2.2.Since in the second and third experiments, most of the outputs of the 10 trials are concentrated around the median except for very few trials (one or two) which are very large outliers. Hence, we chose to output the first and third quartiles indicating the output of the majority of the trials.

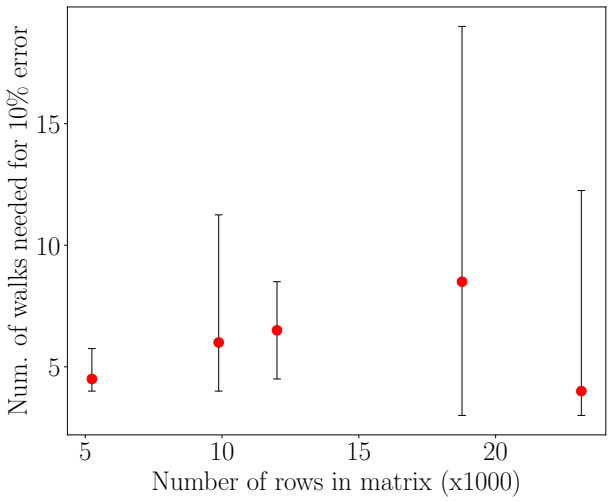

Figure 2.2: Number of walks to (1 ± 0.1)-approximate Schatten 6-norm of 5 different matrices from arXiv Physics Collaboration Network.

The above figure shows that indeed calculating the space needed to approximate the Schatten norms using our algorithm is independent of the matrix dimension. Again, as in Figure 2.1, it is easy to see that the number of estimators needed to approximate the Schatten 6-norm of the chosen matrices is several orders of scale better than the theoretical bound.

In our third experiment we compute the number of walks needed for the mean of the walks to be within 10% of the true Schatten p-Norm of the GR-QC matrix for different values of p. We vary the value of p and, for each value of p, compute the number of walks needed for 10 trials and plot the median, first and third quartile of the 10 trials in Figure 2.3.

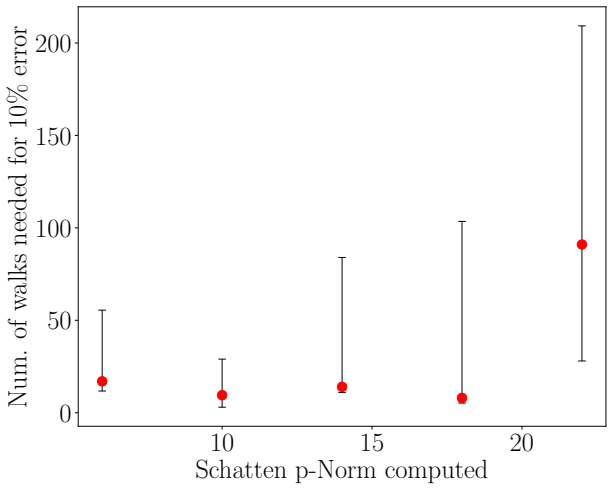

Figure 2.3: Number of walks to (1 ± 0.1)-approximate Schatten p-norm for arXiv General Relativity and Quantum Cosmology Collaboration Network (GR-QC) for different values of $p \in 2\mathbb{Z}^+$.

The last figure follows the previous figures, and shows that again the numerical results are much better than the theoretical bounds, in this case in the dependence on p. Although there is an expected increase in space as p grows, it is not rapid, and in particular is not exponential. This means, for example, that the space needed to approximate other spectral functions, as explained in Section 2.8, would be small, suggesting that our algorithm would be practical for such tasks.

Chapter 3

Sparsifying Numerically Sparse Matrices by Sampling

3.1 Introduction

In the previous chapter we focused on matrices that have very few non-zero entries, i.e. are sparse. While sparsity in the rows and/or columns of the matrix is a phenomenon for which many computational tasks on matrices admit faster algorithms, e.g., low-rank approximation [62, 79], regression problems [87] and semi-definite programming [8, 44], it is not a numerically smooth quantity. Specifically, for a vector $x \in \mathbb{R}^n$ to be k-sparse, at least $n-k$ entries of x must be 0. In practice, many entries could be small but non-zero, e.g. due to noise, and thus the vector would be considered dense.

A smooth analogue of sparsity for a matrix $A \in \mathbb{R}^{m \times n}$ can be defined as follows. First,

for a row (or column) vector $a \in \mathbb{R}^n$, define its *numerical sparsity* [70, 106] to be

$$\mathsf{ns}(a) \coloneqq \min\{k \geq 0 : \|a\|_1 \leq \sqrt{k}\|a\|_2\}. \tag{3.1}$$

This value is clearly at most the number of non-zeros in a, denoted $\|a\|_0$, but can be much smaller. Earlier work used variants of this quantity, referring to $\mathsf{ns}(a)$ as the ℓ_1/ℓ_2-sparsity of the vector [78, 80]. We further define the numerical sparsity of a matrix A, denoted $\mathsf{ns}(A)$, to be the maximum numerical sparsity of any of its rows and columns.

In order to take advantage of sparse matrices in various computational tasks, a natural goal is to approximate a matrix A with numerical sparsity $\mathsf{ns}(A)$ with another matrix $\tilde{A}$ of the same dimensions, that is k-sparse for $k = O(\mathsf{ns}(A))$ (i.e., every row and column is k-sparse). The seminal work of [1] introduced a framework for matrix sparsification via entrywise sampling for approximating the matrix A in spectral-norm. Specifically, they compute a sparse matrix $\tilde{A}$ by sampling and rescaling a small fraction of entries from A such that with high probability $\|A - \tilde{A}\|_2 \leq \epsilon\|A\|_2$ for some error parameter $\epsilon > 0$, where $\|\cdot\|_2$ denotes the spectral-norm. This motivates the following definition.

Definition 3.1.1. An *ϵ-spectral-norm approximation* for $A \in \mathbb{R}^{m\times n}$ is a matrix $\tilde{A} \in \mathbb{R}^{m\times n}$ satisfying

$$\|\tilde{A} - A\|_2 \leq \epsilon\|A\|_2. \tag{3.2}$$

When $\tilde{A}$ is obtained by sampling and rescaling entries from A, we call it an *ϵ-spectral-norm sparsifier*.

Before we continue, let us introduce necessary notations. Here and throughout, we denote the number of non-zero entries in a matrix A by $\mathrm{nnz}(A)$, the Frobenius-norm of A by $\|A\|_F$, the stable-rank of A by $\mathsf{sr}(A) \coloneqq \|A\|_F^2/\|A\|_2^2$, the i-th row and the j-th column of A by A_i and A^j, respectively, and the row-sparsity and column-sparsity of A by $\mathsf{rsp}(A) \coloneqq$

$\max_i \|A_i\|_0$ and $\mathsf{csp}(A) \coloneqq \max_j \|A^j\|_0$, respectively.

The framework of [1] can be used as a preprocessing step that "sparsifies" numerically sparse matrices in order to speed up downstream tasks. It thus motivated a line of work on sampling schemes [2, 9, 51, 65, 94, 95, 124], in which the output $\tilde{A}$ is an unbiased estimator of A, and the sampling distributions are simple functions of A and hence can be computed easily, say, in nearly $O(\mathrm{nnz}(A))$-time and with one or two passes over the matrix. Under these constraints, the goal is simply to minimize the sparsity of the ϵ-spectral-norm sparsifier $\tilde{A}$.

The latest work, by [2], provides a bound for a restricted class of "data matrices". Specifically, they look at matrices $A \in \mathbb{R}^{m\times n}$ such that $\min_i \|A_i\|_1 \geq \max_j \|A^j\|_1$, which can be a reasonable assumption when $m \ll n$. This restricted class does not include the class of square matrices, and hence does not include symmetric matrices such as covariance matrices. Hence, an important question is whether their results extend to a larger class of matrices. Our main result, described in the next section, resolves this concern in the affirmative.

3.1.1 Main Results

We generalize the sparsity bound of [2], which is the best currently known, to all matrices $A \in \mathbb{R}^{m\times n}$. Our main result is a sampling scheme to compute an ϵ-spectral-norm sparsifier for numerically sparse matrices A, as follows.

Theorem 3.1.2. *There is an algorithm that, given a matrix $A \in \mathbb{R}^{m\times n}$ and a parameter $\epsilon > 0$, where $m \geq n$, computes with high probability an ϵ-spectral-norm sparsifier $\tilde{A}$ for A with expected sparsity*

$$\mathbb{E}(\mathrm{nnz}(\tilde{A})) = O\left(\epsilon^{-2}\,\mathsf{ns}(A)\,\mathsf{sr}(A)\log m + \epsilon^{-1}\sqrt{\mathsf{ns}(A)\,\mathsf{sr}(A)n}\log m\right).$$

Moreover, it runs in $O(\mathrm{nnz}(A))$-time when a constant factor estimate of $\|A\|_2$ is given.[1]

We obtain this result by improving the main technique of [2]. Their sampling distribution arises from optimizing a concentration bound, called the matrix-Bernstein inequality, for the sum of matrices formed by sampling entries independently. Our distribution is obtained by the same approach, but arises from considering the columns and rows simultaneously.

In addition to the sampling scheme in Theorem 3.1.2, we analyze ℓ_1-sampling from every row (in Section 3.2.1).[2] This gives a worse bound than the above bound, roughly replacing the $\mathsf{sr}(A)$ term with n, but has the added advantage that the sampled matrix has uniform row-sparsity.

Lower Bound. Our next theorem complements our main result with a lower bound on the sparsity of *any* ϵ-spectral-norm approximation of a matrix A in terms of its numerical sparsity $\mathsf{ns}(A)$ and error parameter $\epsilon > 0$.[3]

Theorem 3.1.3. *Let $0 < \epsilon < \frac{1}{2}$ and $n, k \geq 1$ be parameters satisfying $k \leq O(\epsilon^2 n \log^2 \frac{1}{\epsilon})$. Then, there exists a matrix $A \in \mathbb{R}^{n \times n}$ such that $\mathsf{ns}(A) = \Theta(k \log^2 \frac{1}{\epsilon})$ and, for every matrix B satisfying $\|A - B\|_2 \leq \epsilon \|A\|_2$, the sparsity of every row and every column of B is at least $\Omega(\epsilon^{-2} k \log^{-2} \frac{1}{\epsilon}) = \tilde{\Omega}(\epsilon^{-2}) \cdot \mathsf{ns}(A)$.*

While the lower bound shows that the worst-case dependence on the parameters $\mathsf{ns}(A)$ and ϵ is optimal, it is based on a matrix with stable rank $\Omega(n)$. Settling the sample complexity when the stable rank is $o(n)$ is an interesting open question that we leave for future work.

[1] A constant factor estimate of $\|A\|_2$ can be computed in $\tilde{O}(\mathrm{nnz}(A))$-time by the power method.

[2] Sampling entry A_{ij} with probability proportional to $|A_{ij}|/\|A_i\|_1$

[3] We write $\tilde{O}(f)$ as a shorthand for $O(f \cdot \mathrm{polylog}(nm))$ where n and m are the dimensions of the matrix, and write $O_\epsilon(\cdot)$ when the hidden constant may depend on ϵ.

3.1.2 Comparison to Previous Work

The work of [1] initiated a long line of work on entrywise sampling schemes that approximate a matrix under spectral-norm [2, 9, 51, 65, 94, 95, 124]. Sampling entries independently has the advantage that the output matrix can be seen as a sum of independent random matrices whose spectral-norm can be bounded using known matrix concentration bounds. All previous work uses such matrix concentration bounds with the exception of [9] who bound the spectral-norm of the resulting matrix by analyzing the Rayleigh quotient of all possible vectors.

Natural distributions to sample entries are the ℓ_2 and ℓ_1 distributions, which correspond to sampling entry A_{ij} with probability proportional to $A_{ij}^2/\|A\|_F^2$ and $|A_{ij}|/\|A\|_1$ respectively.[4]

Prior work that use variants of the ℓ_2 sampling [1, 51, 94, 124] point out that sampling according to the ℓ_2 distribution causes small entries to "blow-up" when sampled. Some works, e.g. [51], get around this by zeroing-out small entries or by exceptional handling of small entries, e.g. [1], while others used distributions that combine the ℓ_1 and ℓ_2 distributions, e.g. [94]. All these works sample $\Omega(\epsilon^{-2} n \,\mathsf{sr}(A))$ entries in expectation to achieve an ϵ-spectral-norm approximation and our Theorem 3.1.2 provides an asymptotically better bound. For a full comparison see Table 3.1.

All these algorithms, including the algorithm of Theorem 3.1.2, sample a number of entries corresponding to $\mathsf{sr}(A)$, hence they must have an estimate of it, which requires estimating $\|A\|_2$. An exception is the bound in Theorem 3.2.2, which can be achieved without this estimate. In practice, however, and in previous work in this area, there is a sampling budget $s \geq 0$ and s samples are drawn according to the stated distribution, avoiding the

[4] Here and henceforth we denote by $\|A\|_1$ the entry-wise l_1 norm.

Table 3.1: Comparison between schemes for ϵ-spectral-norm sparsification. The first two entries in the third column present the ratio between the referenced sparsity and that of Theorem 3.1.2.

Expected Number of Samples	**Reference**	**Compared to Thm. 3.1.2**
$O(\epsilon^{-1} n \sqrt{\mathsf{ns}(A)\,\mathsf{sr}(A)})$	[9]	$\tilde{O}_\epsilon\Big(\min\Big(\frac{n}{\sqrt{\mathsf{ns}(A)\,\mathsf{sr}(A)}}, \sqrt{n}\Big)\Big)$
$O(\epsilon^{-2} n\,\mathsf{sr}(A) + n\,\mathrm{polylog}(n))$	[1]	$\tilde{O}_\epsilon\Big(\min\Big(\frac{n}{\mathsf{ns}(A)}, \sqrt{\frac{n\,\mathsf{sr}(A)}{\mathsf{ns}(A)}}\Big)\Big)$
$\tilde{O}(\epsilon^{-2} n\,\mathsf{sr}(A))$	[51, 94]	
$\tilde{O}(\epsilon^{-2}\,\mathsf{ns}(A)\,\mathsf{sr}(A)$ $+$ $\epsilon^{-1}\sqrt{\mathsf{ns}(A)\,\mathsf{sr}(A) n})$	[2]; Theorem 3.1.2	[2] is only for data matrices
$\tilde{O}(\epsilon^{-2} n\,\mathsf{ns}(A))$	Theorem 3.2.2	bounded row-sparsity
$\Omega(\epsilon^{-2} n\,\mathsf{ns}(A)\log^{-4}\frac{1}{\epsilon})$	Theorem 3.1.3	$\mathsf{sr}(A) = \Theta(n)$

need for this estimate. In this case, the algorithm of Theorem 3.1.2 can be implemented in two-passes over the data and in $O(\mathrm{nnz}(A))$ time.

3.1.3 Applications of Spectral-Norm Sparsification

We provide two useful applications of spectral-norm sparsification. More precisely, we use the sparsification to speed up two computational tasks on numerically sparse matrices: approximate matrix multiplication and approximate ridge regression. This adds to previous work, which showed applications to low-rank approximation [1], to semidefinite programming [9], and to PCA and sparse PCA [95]. These applications work in a black-box manner, and can thus employ our improved sparsification scheme.

Application I: Approximate Matrix Multiplication (AMM). Given matrices $A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{n\times p}$ and error parameter $\epsilon > 0$, the goal is to compute a matrix $C \in \mathbb{R}^{m\times p}$

such that $\|AB - C\| \leq \epsilon \|A\| \cdot \|B\|$, where the norm is usually either Frobenius-norm $\|\cdot\|_F$ or spectral-norm $\|\cdot\|_2$. In Section 3.3, we provide algorithms for both error regimes by combining our entrywise sampling scheme with previous AMM algorithms that sample a small number of columns of A and rows of B.

Theorem 3.1.4. *There exists an algorithm that, given matrices $A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{n\times p}$ parameter $0 < \epsilon < \frac{1}{2}$ and constant factor estimates of $\|A\|_2$ and $\|B\|_2$, computes a matrix $C \in \mathbb{R}^{m\times p}$ satisfying with high probability $\|AB - C\|_2 \leq \epsilon \|A\|_2 \|B\|_2$ in time*

$$O(\text{nnz}(A) + \text{nnz}(B)) + \tilde{O}(\epsilon^{-6} \sqrt{\mathsf{sr}(A)\,\mathsf{sr}(B)}\,\mathsf{ns}(A)\,\mathsf{ns}(B)).$$

Theorem 3.1.5. *There exists an algorithm that, given matrices $A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{n\times p}$ and parameter $0 < \epsilon < \frac{1}{2}$, computes a matrix $C \in \mathbb{R}^{m\times p}$ satisfying $\mathbb{E}\|AB - C\|_F \leq \epsilon \|A\|_F \|B\|_F$ in time*

$$O(\text{nnz}(A) + \text{nnz}(B) + \epsilon^{-6}\,\mathsf{ns}(A)\,\mathsf{ns}(B)).$$

Approximate Matrix Multiplication (AMM) is a fundamental problem in numerical linear algebra with a long line of formative work [33, 39, 53, 58, 108, 114, 166] and many others. These results fall into roughly three categories; sampling based methods, random projection based methods and a mixture of sampling and projection based methods. We focus on sampling based methods in our work.

There are two main error regimes considered in the literature: spectral-norm error and Frobenius-norm error. We focus on the results of [108] for spectral-norm error and [53] for Frobenius-norm error. Sampling based methods, including that of [53, 108], propose sampling schemes that are linear time or nearly-linear time: specifically, they write the product of two matrices as the sum of n outer products $AB = \sum_{i\in[n]} A^i B_i$, and then sample and compute each outer product $A^i B_i / p_i$ with probability $p_i \propto \|A^i\|_2 \|B_i\|_2$. Computing each of these

rank-1 outer products takes time bounded by $O(\mathsf{csp}(A)\,\mathsf{rsp}(B))$. This estimator is repeated sufficiently many times depending on the error regime under consideration.

Our entrywise-sampling scheme compounds well with this framework for approximate matrix multiplication by additionally sampling entries from the rows/columns sampled by the AMM algorithm. We essentially replace the $\mathsf{csp}(A)\,\mathsf{rsp}(B)$ term with $\mathsf{ns}(A)\,\mathsf{ns}(B)$, up to $\tilde{O}(poly(1/\epsilon))$ factors, for both Frobenius-norm and spectral-norm error regimes. It is plausible that the dependence on epsilon can be improved.

Application II: Approximate Ridge Regression. Given a matrix $A \in \mathbb{R}^{m\times n}$, a vector $b \in \mathbb{R}^m$ and a parameter $\lambda > 0$, the goal is to find a vector $x \in \mathbb{R}^n$ that minimizes $\|Ax - b\|_2^2 + \lambda\|x\|_2^2$. This problem is λ-strongly convex, has solution $x^* = (A^\top A + \lambda I)^{-1}A^\top b$ and condition number $\kappa_\lambda(A^\top A) \coloneqq \|A\|_2^2/\lambda$.

Given an initial vector $x_0 \in \mathbb{R}^n$ and a parameter $\epsilon > 0$, an ϵ-approximate solution to the ridge regression problem is a vector $\hat{x} \in \mathbb{R}^n$ satisfying $\|\hat{x} - x^*\|_{A^\top A+\lambda I} \leq \epsilon\|x_0 - x^*\|_{A^\top A+\lambda I}$, where we write $\|x\|_M \coloneqq x^\top Mx$ when M is a PSD matrix. We provide algorithms for approximate ridge regression by using our sparsification scheme as a preconditioner for known linear-system solvers in composition with a black-box acceleration framework by [59]. The following theorem is proved in Section 3.4.

Theorem 3.1.6. *There exists an algorithm that, given $A \in \mathbb{R}^{m\times n}, x_0 \in \mathbb{R}^n, \lambda > 0$ and $\epsilon > 0$, computes with high probability an ϵ-approximate solution to the ridge regression problem in time*

$$O_\epsilon(\mathrm{nnz}(A)) + \tilde{O}_\epsilon\left((\mathrm{nnz}(A))^{2/3}(\mathsf{ns}(A)\,\mathsf{sr}(A))^{1/3}\sqrt{\kappa_\lambda(A^\top A)}\right).$$

Moreover, when the input matrix A has uniform column (or row) norms, the running time in Theorem 3.1.6 can be reduced by a factor of roughly $(\mathsf{sr}(A)/n)^{1/6}$, for details see

Section 3.4.2.

Solving linear systems using preconditioning has a rich history that is beyond the scope of this work to summarize. Recently, the work of [70] designed algorithms with improved running times over popular methods using the Stochastic Variance Reduced Gradient Descent (SVRG) framework of [87]. They adapt it using efficient subroutines for numerically sparse matrices. They also suggested the idea of using spectral-norm sparsifiers as preconditioners for linear regression. While they considered the sparsification of [2] for computing the preconditioners, they required a stronger bound on the spectral-norm approximation than Theorem 3.1.6 does.

Our result is in general incomparable to that of [70]. In the case when the input has uniform column (or row) norms, our running time is roughly an $(\mathsf{ns}(A)/n)^{1/6}$-factor smaller than theirs, for details see Theorem 3.4.4 in Section 3.4.2.

Very recently, [27] have developed, independently of our work and as part of a suite of results on bilinear minimax problems, an algorithm for ridge regression with improved running time $\tilde{O}(\mathrm{nnz}(A) + \sqrt{\mathrm{nnz}(A)\,\mathsf{ns}(A)\,\mathsf{sr}(A)\kappa_\lambda(A^\top A)})$. Their approach is different and their techniques are more involved than ours.

3.2 Spectral-Norm Sparsification

In this section we state and prove our main results. We first prove the upper bound in Theorem 3.1.2. Then we analyze ℓ_1 sampling from the rows in Theorem 3.2.2, Section 3.2.1 that gives a slightly weaker bound but has the property that the resulting matrix has uniform row sparsity. In Section 3.2.2, we prove the lower bound in Theorem 3.1.3.

Theorem 3.1.2. *There is an algorithm that, given a matrix $A \in \mathbb{R}^{m\times n}$ and a parameter*

$\epsilon > 0$, where $m \geq n$, computes with high probability an ϵ-spectral-norm sparsifier $\tilde{A}$ for A with expected sparsity

$$\mathbb{E}(\mathrm{nnz}(\tilde{A})) = O\left(\epsilon^{-2}\,\mathsf{ns}(A)\,\mathsf{sr}(A)\log m + \epsilon^{-1}\sqrt{\mathsf{ns}(A)\,\mathsf{sr}(A)n}\log m\right).$$

Moreover, it runs in $O(\mathrm{nnz}(A))$-time when a constant factor estimate of $\|A\|_2$ is given.[5]

Before we prove Theorem 3.1.2, we start by stating a result on the concentration of sums of independent random matrices; the Matrix Bernstein Inequality.

Theorem 3.2.1 (Matrix Bernstein, Theorem 1.6 of [143])**.** *Consider a finite sequence $\{Z_k\}$ of independent, random $d_1 \times d_2$ real matrices, such that there is $R > 0$ satisfying $\mathbb{E}Z_k = 0$ and $\|Z_k\|_2 \leq R$ almost surely. Define*

$$\sigma^2 = \max\left\{\left\|\sum_k \mathbb{E}(Z_k Z_k^\top)\right\|_2, \left\|\sum_k \mathbb{E}(Z_k^\top Z_k)\right\|_2\right\}.$$

Then for all $t \geq 0$,

$$\mathbb{P}\left(\left\|\sum_k Z_k\right\|_2 \geq t\right) \leq (d_1 + d_2)\exp\left(\frac{-t^2/2}{\sigma^2 + Rt/3}\right).$$

Proof of Theorem 3.1.2. Let $\epsilon > 0$. Given a matrix A, define sampling probabilities as follows.

$$\begin{aligned}
p_{ij}^{(1)} &= \frac{|A_{ij}|}{\sum_{i'j'}|A_{i'j'}|} \\
p_{ij}^{(2)} &= \frac{\|A_i\|_1^2}{\sum_{i'}\|A_{i'}\|_1^2}\cdot\frac{|A_{ij}|}{\|A_i\|_1} \\
p_{ij}^{(3)} &= \frac{\|A^j\|_1^2}{\sum_{j'}\|A^{j'}\|_1^2}\cdot\frac{|A_{ij}|}{\|A^j\|_1} \\
p_{ij}^{*} &= \max_\alpha(p_{ij}^{(\alpha)}).
\end{aligned}$$

[5] A constant factor estimate of $\|A\|_2$ can be computed in $\tilde{O}(\mathrm{nnz}(A))$-time by the power method.

Observe that each $\alpha = 1, 2, 3$ yields a probability distribution because $\sum_{ij} p_{ij}^{(\alpha)} = 1$.

Let $s < mn$ be a parameter that we will choose later. Now sample each entry of A independently and scale it to get an unbiased estimator, i.e., compute $\tilde{A}$ by

$$\tilde{A}_{ij} = \begin{cases} \frac{A_{ij}}{p_{ij}} & \text{with prob. } p_{ij} = \min(1, s \cdot p_{ij}^*); \\ 0 & \text{otherwise.} \end{cases}$$

To bound the expected sparsity, observe that $p_{ij}^* \leq \sum_\alpha p_{ij}^{(\alpha)}$, and thus

$$\mathbb{E}[\text{nnz}(\tilde{A})] = \sum_{ij} p_{ij} \leq s \sum_{ij} \sum_{\alpha} p_{ij}^{(\alpha)} \leq 3s.$$

We show that each of the above distributions bounds one of the terms in matrix Bernstein bound. For each pair of indices (i, j) define a matrix Z_{ij} that has a single non-zero at the (i, j) entry, with value $\tilde{A}_{ij} - A_{ij}$. Its spectral-norm is $\|Z_{ij}\|_2 = |\tilde{A}_{ij} - A_{ij}|$. If $p_{ij} = 1$, this is 0. If $p_{ij} < 1$ then

$$\begin{aligned} |\tilde{A}_{ij} - A_{ij}| &\leq |A_{ij}| \max(1, \frac{1}{p_{ij}} - 1) \\ &\leq \frac{|A_{ij}|}{p_{ij}} \leq \frac{|A_{ij}|}{s p_{ij}^{(1)}} = \frac{1}{s} \sum_{i'j'} |A_{i'j'}| \\ &\leq \frac{\sqrt{\mathsf{ns}(A)}}{s} \sum_j \|A^j\|_2 \leq \frac{\sqrt{\mathsf{ns}(A) n}}{s} \|A\|_F =: R, \end{aligned}$$

where the last inequality follows from Cauchy-Schwarz inequality.

In order to bound σ^2, first notice that $\text{var}(\tilde{A}_{ij}) \leq \mathbb{E}(\tilde{A}_{ij}^2) = \frac{A_{ij}^2}{s p_{ij}^*}$. Now, since $Z_{ij} Z_{ij}^\top$ has a single non-zero entry at (i, i), and $Z_{ij}^\top Z_{ij}$ has a single non-zero entry at (j, j), both $\sum_{i,j} Z_{ij} Z_{ij}^\top$ and $\sum_{i,j} Z_{ij}^\top Z_{ij}$ are diagonal, where the (i, i) entry is $\sum_j (\tilde{A}_{ij} - A_{ij})^2$ in the former and the (j, j) entry is $\sum_i (\tilde{A}_{ij} - A_{ij})^2$ in the latter. Since these are diagonal matrices, their

spectral-norm equals their largest absolute entry, and thus

$$\begin{aligned}\Big\|\sum_{i,j}\mathbb{E}\big(Z_{ij}Z_{ij}^\top\big)\Big\|_2 &\le \max_i\Big(\sum_j\frac{A_{ij}^2}{sp_{ij}^*}\Big)\le\max_i\Big(\sum_j\frac{A_{ij}^2}{sp_{ij}^{(2)}}\Big)\\ &=\frac{1}{s}\max_i\Big(\sum_j\frac{|A_{ij}|\sum_{i'}\|A_{i'}\|_1^2}{\|A_i\|_1}\Big)=\frac{1}{s}\sum_{i'}\|A_{i'}\|_1^2\\ &\le\frac{1}{s}\sum_{i'}\mathsf{ns}(A)\|A_{i'}\|_2^2=\frac{\mathsf{ns}(A)}{s}\|A\|_F^2.\end{aligned}$$

The same bound can be shown for $\sum_{i,j}\mathbb{E}(Z_{ij}^\top Z_{ij})$ by using $p_{ij}^*\ge p_{ij}^{(3)}$, thus by the definition of σ^2, $\sigma^2\le\frac{\mathsf{ns}(A)}{s}\|A\|_F^2$. Finally, by the matrix-Bernstein bound,

$$\mathbb{P}\Big(\Big\|\sum_{i,j}Z_{ij}\Big\|_2\ge\epsilon\|A\|_2\Big)\le 2m\exp\left(-\frac{\epsilon^2\|A\|_2^2/2}{\frac{\mathsf{ns}(A)}{s}\|A\|_F^2+\epsilon\frac{\sqrt{\mathsf{ns}(A)n}}{s}\|A\|_F\|A\|_2/3}\right),$$

and since $\mathsf{sr}(A)=\frac{\|A\|_F^2}{\|A\|_2^2}$, by setting $s=O(\epsilon^{-2}\,\mathsf{ns}(A)\,\mathsf{sr}(A)\log m+\epsilon^{-1}\sqrt{\mathsf{ns}(A)\cdot n\cdot\mathsf{sr}(A)}\log m)$ we conclude that with high probability $\|\tilde{A}-A\|_2\le\epsilon\|A\|_2$, which completes the proof of Theorem 3.1.2. □

3.2.1 A Second Sampling Scheme

We analyze ℓ_1 row sampling, i.e. sampling entry (i,j) with probability $\frac{|A_{ij}|}{\|A_i\|_1}$, as was similarly done for numerically sparse matrices in [70], although they employed this sampling (i) in a different setting and (ii) on one row at a time. Here, we analyze how to employ this sampling on all the rows simultaneously for ϵ-spectral-norm sparsification. This sampling is inferior to the one in Theorem 3.1.2 in terms of $\mathrm{nnz}(\tilde{A})$, but has the additional property that the sparsity of every row is bounded. By applying this scheme to $A^\top$, we can alternatively obtain an ϵ-spectral-norm sparsifier where the sparsity of every column is bounded.

Theorem 3.2.2. *There is an algorithm that, given a matrix $A \in \mathbb{R}^{m\times n}$ and a parameter $\epsilon > 0$, computes in time $O(\mathrm{nnz}(A))$ with high probability an ϵ-spectral-norm sparsifier $\tilde{A}$ for A such that the sparsity of every row of $\tilde{A}$ is bounded by $O(\epsilon^{-2}\,\mathsf{ns}(A)\log(m+n))$.*

The algorithm is as follows. Given a matrix A and $\epsilon > 0$, define the sampling probabilities

$$\boxed{p_{ij} = \frac{|A_{ij}|}{\|A_i\|_1},}$$

and observe that for every i this induces probability distribution, i.e., $\sum_j p_{ij} = 1$. Let $s = O(\epsilon^{-2}\,\mathsf{ns}(A)\log(m+n))$. Now from each row of A sample s entries independently with replacement according to the above distribution, and scale it to get an unbiased estimator of that row; formally, for each row i and each $t = 1, \ldots, s$ draw a row vector

$$\boxed{Q_i^{(t)} = \begin{cases} \frac{A_{ij}}{p_{ij}} e_j^\top & \text{with prob. } p_{ij}, \end{cases}}$$

where $\{e_j\}_j$ is the standard basis of $\mathbb{R}^n$. Next, average the t samples for each row, and arrange these rows in a matrix $\tilde{A}$ that is an unbiased estimator for A; formally,

$$\boxed{\tilde{A} = \sum_{i=1}^{m} e_i \frac{1}{s} \sum_{t=1}^{s} Q_i^{(t)}.}$$

Clearly $\mathbb{E}(\tilde{A}) = A$ and every row of $\tilde{A}$ has at most s non-zeros. In order to bound the probability that $\tilde{A}$ is an ϵ-spectral-norm sparsifier of A, similarly to the proof of Theorem 3.1.2, we employ the matrix-Bernstein bound stated in Theorem 3.2.1.

Proof of Theorem 3.2.2. Given a matrix $A, \epsilon > 0$, let $k = \mathsf{ns}(A)$ and apply the algorithm of Theorem 3.2.2. Note that by the definition of $\mathsf{ns}(A)$ and by spectral-norm properties, the i-th row of A satisfies

$$\|A_i\|_1 \le \sqrt{k}\|A_i\|_2 \le \sqrt{k}\|A\|_2. \tag{3.3}$$

For each random draw, define a matrix $Z_{(it)}$ with exactly one non-zero row formed by placing $A_i - Q_i^{(t)}$ at the i-th row; formally, let $Z_{(it)} = e_i(A_i - Q_i^{(t)})$. Where it is clear from context we will omit the superscript from $Q_i^{(t)}$. The spectral-norm of $Z_{(it)}$ is

$$\|Z_{(it)}\|_2 = \|A_i - Q_i^{(t)}\|_2 \le \|A_i\|_2 + \|Q_i^{(t)}\|_2 = \|A_i\|_2 + \|A_i\|_1 \le 2\sqrt{k}\|A\|_2 =: R.$$

To bound σ^2, notice that $Z_{(it)}Z_{(it)}^\top$ has a single non-zero at the (i,i) entry with value $\|A_i - Q_i^{(t)}\|_2^2$, hence

$$\begin{aligned}\big\|\mathbb{E}\sum_{i,t} Z_{(it)}Z_{(it)}^\top\big\|_2 &= s\max_i \mathbb{E}\|A_i - Q_i\|_2^2 = s\max_i \mathbb{E}\|Q_i\|_2^2 - \|A_i\|_2^2 \\ &\le s\max_i \sum_j \|A_i\|_1 \cdot |A_{ij}| \le sk\|A\|_2^2.\end{aligned}$$

The other term $Z_{(it)}^\top Z_{(it)}$ satisfies $\mathbb{E}(Z_{(it)}^\top Z_{(it)}) = \mathbb{E}\big(Q_i^\top(Q_i - A_i)\big) = \mathbb{E}(Q_i^\top Q_i) - A_i^\top A_i$. The matrix $\mathbb{E}(Q_i^\top Q_i)$ is diagonal with value $|A_{ij}| \cdot \|A_i\|_1$ at the (j,j) entry, hence

$$\begin{aligned}\big\|\sum_{i,t}\mathbb{E}(Z_{(it)}^\top Z_{(it)})\big\|_2 &= s\big\|\sum_i(\mathbb{E}(Q_i^\top Q_i) - A_i^\top A_i)\big\|_2 \\ &= s\|\sum_i \mathbb{E}(Q_i^\top Q_i) - A^\top A\|_2 \\ &\le s\big(\big\|\sum_i \mathbb{E}(Q_i^\top Q_i)\big\|_2 + \|A^\top A\|_2\big) \\ &= s\big(\max_j \sum_i |A_{ij}| \cdot \|A_i\|_1 + \|A\|_2^2\big) \\ &\le s\sqrt{k}\big(\|A\|_2 \max_j \sum_i |A_{ij}| + \|A\|_2^2\big) \\ &= s\sqrt{k}\big(\|A\|_2 \max_j \|A^j\|_1 + \|A\|_2^2\big) \le 2s\cdot k\cdot\|A\|_2^2 =: \sigma^2.\end{aligned}$$

Now, by the matrix-Bernstein bound as stated in Theorem 3.2.1,

$$\mathbb{P}(\|A-\tilde{A}\|_2 \geq \epsilon\|A\|_2) = \mathbb{P}\big(\big\|\sum_{i,t} Z_{(it)}\big\|_2 \geq s\epsilon\|A\|_2\big)$$
$$\leq (m+n)\exp\Big(-\frac{s\epsilon^2\|A\|_2^2/2}{2k\|A\|_2^2 + \frac{2\epsilon}{3}\sqrt{k}\|A\|_2^2}\Big),$$

and by setting $s = O(\epsilon^{-2}k\log(m+n))$ we conclude that with high probability $\|\tilde{A}-A\|_2 \leq \epsilon\|A\|_2$. □

3.2.2 Lower Bounds

We provide a lower bound in Theorem 3.1.3 for spectral-norm sparsification, which almost matches the bound in Theorem 3.1.2 for a large range of ϵ and $\mathsf{ns}(A)$.

Theorem 3.1.3. *Let $0 < \epsilon < \frac{1}{2}$ and $n, k \geq 1$ be parameters satisfying $k \leq O(\epsilon^2 n \log^2 \frac{1}{\epsilon})$. Then, there exists a matrix $A \in \mathbb{R}^{n\times n}$ such that $\mathsf{ns}(A) = \Theta(k\log^2\frac{1}{\epsilon})$ and, for every matrix B satisfying $\|A-B\|_2 \leq \epsilon\|A\|_2$, the sparsity of every row and every column of B is at least $\Omega(\epsilon^{-2}k\log^{-2}\frac{1}{\epsilon}) = \tilde{\Omega}(\epsilon^{-2})\cdot\mathsf{ns}(A)$.*

Proof. We shall assume that k divides n, and that both are powers of 2, which can be obtained with changing the bounds by a constant factor. Let $m = \frac{n}{k}$, and notice it is a power of 2 as well.

Construct first a vector $a \in \mathbb{R}^m$ by concatenating blocks of length 2^i whose coordinates have value $2^{-(1+\alpha)i}$, for each $i \in \{0, ..., \log m - 1\}$, where $1 > \alpha \geq \Omega(\log^{-1} m)$ is a parameter that we will set later. The last remaining coordinate have value 0. Formally, the coordinates

of a are given by $a_j = 2^{-(1+\alpha)\lfloor \log j \rfloor}$, except the last one which is 0. Its ℓ_1 norm is

$$\|a\|_1 = \sum_{j=1}^{m} a_j = \sum_{i=0}^{\log m - 1} 2^i \cdot 2^{-(1+\alpha)i} = \frac{1 - 2^{-\alpha \log m}}{1 - 2^{-\alpha}} = \Theta(\alpha^{-1}).$$

A similar computation shows that $\|a\|_2 = \Theta(1)$, and thus $\mathsf{ns}(a) = \Theta(\alpha^{-2})$. Denote by $a_{\mathrm{tail}(c)}$ the vector a without its c largest entries, then its ℓ_2 norm is

$$\|a_{\mathrm{tail}(c)}\|_2^2 \geq \sum_{i=\lfloor \log c \rfloor + 1}^{\log m - 1} 2^i \cdot 2^{-2(1+\alpha)i} = \Omega(c^{-(1+2\alpha)}), \tag{3.4}$$

which almost matches the upper bound of Lemma 3 in [70].

Now, for $k = 1$ we construct a circulant matrix $A \in \mathbb{R}^{m \times m}$ by letting the vector a be its first row, and the j-th row is a cyclic shift of a with offset j. By well-known properties of circulant matrices, the t-th eigenvalue of A is given by $\lambda_t = \sum_j a_j (\omega_t)^j$ where $\omega_t = \exp\left(i \frac{2\pi t}{m}\right)$ and i is the imaginary unit, so $\|A\|_2 = \|a\|_1 = \Theta(\alpha^{-1})$. Consider $B \in \mathbb{R}^{m \times m}$ satisfying $\|A - B\|_2 \leq \epsilon \|A\|_2$, and suppose some row B_j of B has s non-zeros. Then using (3.4),

$$\|A - B\|_2 \geq \|A_j - B_j\|_2 \geq \|a_{\mathrm{tail}(s)}\|_2 = \Omega(s^{-(\frac{1}{2}+\alpha)}).$$

By the error bound $\|A - B\|_2 \leq \epsilon \|A\|_2$, we must have $s \geq (\Omega(\epsilon/\alpha))^{-\frac{2}{1+2\alpha}} \geq \Omega((\epsilon/\alpha)^{-\frac{2}{1+2\alpha}})$, which bounds from below the sparsity of every row, and similarly also of every column, of B.

To generalize this to larger numerical sparsity, consider as a first attempt constructing a vector $a' \in \mathbb{R}^n$ by concatenating k copies of a. Then clearly $\mathsf{ns}(a') = \Theta(k\, \mathsf{ns}(a))$. The circulant matrix of a' is equivalent to $A \otimes C$, where C is the all-ones matrix of dimension $k \times k$, and $\otimes$ is the Kronecker product. But this matrix has low rank, and thus might be easier to approximate. We thus construct a different matrix $A' = A \otimes H_k$, where H_k is

the $k \times k$ Hadamard matrix. Its numerical sparsity is the same as of the vector a', thus $\mathsf{ns}(A') = \Theta(k\,\mathsf{ns}(a))$. The eigenvalues of H_k are $\pm\sqrt{k}$. By properties of the Kronecker product, every eigenvalue of A' is the product of an eigenvalue of A with $\pm\sqrt{k}$, thus $\|A'\|_2 = \Theta(\sqrt{k}\|A\|_2) = \Theta(\sqrt{k}\alpha^{-1})$. We now apply the same argument we made for $k = 1$. Let $B' \in \mathbb{R}^{n\times n}$ be an ϵ-spectral-norm sparsifier of A'. If some row B'_j has s non-zeros then using (3.4),

$$\|A' - B'\|_2 \geq \|A'_j - B'_j\|_2 \geq \|a'_{\mathrm{tail}(s)}\|_2 = \Omega(\sqrt{k}(s/k)^{-(\frac{1}{2}+\alpha)}).$$

By the error bound $\|A' - B'\|_2 \leq \epsilon\|A'\|_2$, we must have $s \geq \Omega(k(\epsilon/\alpha)^{-\frac{2}{1+2\alpha}})$, which bounds the sparsity of every row and every column of B'.

We can set $\alpha = \log^{-1}\frac{1}{\epsilon} > \epsilon$. Note that this choice for α is in the range $[\log^{-1}\frac{n}{k}, 1]$, hence the construction hold. Now since $\frac{1}{1+2\alpha} \geq 1 - 2\alpha$, the lower bound on the sparsity of each row and each column of B' is $k(\epsilon/\alpha)^{-\frac{2}{1+2\alpha}} \geq k(\epsilon/\alpha)^{-2+4\alpha} \geq \Omega(k\epsilon^{-2}\log^{-2}\frac{1}{\epsilon})$. □

3.3 Application I: Approximate Matrix Multiplication

In this section, we show how to use ℓ_1 row/column sampling for fast approximate matrix multiplication (AMM). Given matrices $A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{n\times p}$ and error parameter $\epsilon > 0$, the goal is to compute a matrix $C \in \mathbb{R}^{m\times p}$ such that $\|AB - C\| \leq \epsilon\|A\| \cdot \|B\|$, where the norm is usually either the Frobenius-norm $\|\cdot\|_F$ or spectral-norm $\|\cdot\|_2$. We provide the first results on AMM for numerically sparse matrices with respect to both norms.

Theorem 3.1.4. *There exists an algorithm that, given matrices $A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{n\times p}$ parameter $0 < \epsilon < \frac{1}{2}$ and constant factor estimates of $\|A\|_2$ and $\|B\|_2$, computes a matrix*

$C \in \mathbb{R}^{m\times p}$ *satisfying with high probability* $\|AB - C\|_2 \leq \epsilon\|A\|_2\|B\|_2$ *in time*

$$O(\text{nnz}(A) + \text{nnz}(B)) + \tilde{O}(\epsilon^{-6}\sqrt{\mathsf{sr}(A)\,\mathsf{sr}(B)}\,\mathsf{ns}(A)\,\mathsf{ns}(B)).$$

Theorem 3.1.5. *There exists an algorithm that, given matrices* $A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{n\times p}$ *and parameter* $0 < \epsilon < \frac{1}{2}$*, computes a matrix* $C \in \mathbb{R}^{m\times p}$ *satisfying* $\mathbb{E}\|AB - C\|_F \leq \epsilon\|A\|_F\|B\|_F$ *in time*

$$O(\text{nnz}(A) + \text{nnz}(B) + \epsilon^{-6}\,\mathsf{ns}(A)\,\mathsf{ns}(B)).$$

The proofs of these theorems combine Theorem 3.2.2 with previous results on numerical sparsity and with previous results on AMM.

Lemma 3.3.1 (Lemma 4 of [70])**.** *Given a vector* $a \in \mathbb{R}^n$ *and a parameter* $\epsilon > 0$*, independently sampling* $(\epsilon^{-2}\,\mathsf{ns}(a))$ *entries according to the distribution* $\{p_i = \frac{|a_i|}{\|a\|_1}\}_i$ *and re-weighting the sampled coordinates by* $\frac{1}{p_i} \cdot \frac{1}{\epsilon^{-2}\,\mathsf{ns}(a)}$*, outputs a* $(\epsilon^{-2}\,\mathsf{ns}(a))$*-sparse vector* $a' \in \mathbb{R}^n$ *satisfying* $\mathbb{E}a' = a$ *and* $\mathbb{E}(\|a'\|_2^2) \leq (1+\epsilon^2)\|a\|_2^2$*.*

3.3.1 Proof of Theorem 3.1.4 (Spectral-Norm AMM)

In order to prove Theorem 3.1.4, we will use a result from [108]. Given matrices A, B, their product is $AB = \sum_i A^i B_i$. The algorithm in [108] samples corresponding pairs of columns from A and rows from B, hence the time it takes to compute an approximation of AB depends on the sparsity of these rows and columns.

Lemma 3.3.2 (Theorem 3.2 (ii) of [108].)**.** *There exists an algorithm that, given matrices* $A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{n\times p}$*, a parameter* $0 < \epsilon < 1/2$ *and constant factor estimates of* $\|A\|_2$ *and*

$\|B\|_2$, *computes in time*

$$O\Big(\operatorname{nnz}(A)+\operatorname{nnz}(B)+\epsilon^{-2}\,\mathsf{csp}(A)\,\mathsf{rsp}(B)\sqrt{\mathsf{sr}(A)\,\mathsf{sr}(B)}\log\left(\epsilon^{-1}\,\mathsf{sr}(A)\,\mathsf{sr}(B)\right)\Big)$$

a matrix C *that satisfies*

$$\mathbb{P}(\|C-AB\|_2\geq\epsilon\|A\|_2\|B\|_2)\leq\frac{1}{poly(\mathsf{sr}(A)\,\mathsf{sr}(B))}.$$

Proof of Theorem 3.1.4. Given $\epsilon>0$, our algorithm is as follows.

1. Apply the algorithm in Theorem 3.2.2 on A with parameter $\epsilon/4$ to compute a matrix A' satisfying $\|A'-A\|_2\leq\frac{\epsilon}{4}\|A\|_2$ and $\mathsf{csp}(A')\leq O(\epsilon^{-2}\,\mathsf{ns}(A)\log(m+n))$, and apply it on B with parameter $\epsilon/4$ to compute a matrix B' satisfying $\|B'-B\|_2\leq\frac{\epsilon}{4}\|B\|_2$ and $\mathsf{rsp}(B')\leq O(\epsilon^{-2}\,\mathsf{ns}(B)\log(n+p))$.
2. Apply the algorithm in Lemma 3.3.2 on A',B' with parameter $\epsilon/4$ to produce a matrix C. Output C.

The sampling in Theorem 3.2.2 satisfies the conditions for Lemma 3.3.1, hence $\mathbb{E}\|A'\|_F^2\leq\big(1+O(\frac{\epsilon^2}{\log(m+n)})\big)\|A\|_F^2$. Thus, with high probability, $\mathsf{sr}(A')\in(1\pm O(\epsilon))\,\mathsf{sr}(A)$, and similarly for B'. Ignoring the $\operatorname{nnz}(\cdot)$ terms, the time it takes for the algorithm from Lemma 3.3.2 on A',B' is

$$O\Big(\epsilon^{-6}\,\mathsf{ns}(A)\,\mathsf{ns}(B)\log(m+n)\log(n+p)\sqrt{\mathsf{sr}(A)\,\mathsf{sr}(B)}\log\left(\epsilon^{-1}\,\mathsf{sr}(A)\,\mathsf{sr}(B)\right)\Big),$$

hence the stated overall running time. The output C satisfies with high probability

$$\begin{aligned}\|AB-C\|_2&\leq\|(A-A')B\|_2+\|(A'(B-B')\|_2+\|A'B'-C\|_2\\&\leq\tfrac{\epsilon}{4}\|A\|_2\|B\|_2+\tfrac{\epsilon}{4}\|B\|_2(1+\tfrac{\epsilon}{4})\|A\|_2+\tfrac{\epsilon}{4}(1+\tfrac{\epsilon}{4})^2\|A\|_2\|B\|_2\leq\epsilon\|A\|_2\|B\|_2.\end{aligned}$$

□

3.3.2 Proof of Theorem 3.1.5 (Frobenius-Norm AMM)

We provide a sampling lemma for estimating outer products in the Frobenius-norm.

Lemma 3.3.3. *There exists an algorithm that, given vectors $a \in \mathbb{R}^n, b \in \mathbb{R}^m$ and parameter $0 < \epsilon < 1$, computes in time $O(\|a\|_0 + \|b\|_0)$ vectors $a', b' \in \mathbb{R}^n$ with sparsity $\epsilon^{-2}\,\mathsf{ns}(a)$ and $\epsilon^{-2}\,\mathsf{ns}(b)$, respectively, satisfying $\mathbb{E}(a'b'^\top) = ab^\top$ and $\mathbb{E}\|a'b'^\top - ab^\top\|_F^2 \leq \epsilon^2\|a\|_2^2\|b\|_2^2$.*

Proof. Given $0 < \epsilon < 1$, our algorithm is as follows.

1. Independently sample (with repetitions) $9\epsilon^{-2}\,\mathsf{ns}(a)$ entries from a according to the distribution $\{p_i^{(a)} = \frac{|a_i|}{\|a\|_1}\}_i$ and $9\epsilon^{-2}\,\mathsf{ns}(b)$ entries from b according to the distribution $\{p_i^{(b)} = \frac{|b_i|}{\|b\|_1}\}_i$.
2. Re-weight the sampled entries of a by $\frac{1}{p_i^{(a)}} \cdot \frac{1}{9\epsilon^{-2}\,\mathsf{ns}(a)}$ and similarly for b. Output the sampled vectors.

Denote the sampled vectors a' and b'. They satisfy the conditions of Lemma 3.3.1, hence they satisfy $\mathbb{E}(a'b'^\top) = ab^\top$ and $\mathbb{E}(\|a'\|_2^2) \leq (1 + \epsilon^2/3)\|a\|_2^2$ and similarly for b'. Thus,

$$\mathbb{E}\|a'b'^\top - ab^\top\|_F^2 = \mathbb{E}\|a'b'^\top\|_F^2 - \|ab^\top\|_F^2 = \mathbb{E}\|a'\|_2^2\|b'\|_2^2 - \|a\|_2^2\|b\|_2^2 \leq \epsilon^2\|a\|_2^2\|b\|_2^2.$$

□

In order to prove Theorem 3.1.5, we will use a result from [53]. The algorithm in [53] samples corresponding pairs of columns from A and rows from B, hence the time it takes to compute an approximation of AB depends on the sparsity of these rows and columns.

Lemma 3.3.4 (Lemma 4 of [53]). *There exists an algorithm that, given matrices $A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{n\times p}$ and parameter $0 < \epsilon < 1$, computes in time $O(\mathrm{nnz}(A) + \mathrm{nnz}(B) + \epsilon^{-2}\,\mathsf{csp}(A)\,\mathsf{rsp}(B))$ a matrix $C \in \mathbb{R}^{m\times p}$ satisfying $\mathbb{E}\|AB - C\|_F \le \epsilon\|A\|_F\|B\|_F$.*

Proof of Theorem 3.1.5. Let $0 < \epsilon < 1$. Recall that $AB = \sum_i A^i B_i$. Our algorithm is as follows.

1. Apply the algorithm in Lemma 3.3.3 on each pair of vectors A^i, B_i with parameter $\epsilon/3$ to obtain their sparse estimates $\hat{A}^i$ and $\hat{B}_i$.
2. Arrange the column vectors $\{\hat{A}^i\}$ in a matrix $\hat{A}$ and the row vectors $\{\hat{B}_i\}$ in a matrix $\hat{B}$.
3. Apply the algorithm in Lemma 3.3.4 on the matrices $\hat{A}$ and $\hat{B}$ with parameter $\epsilon/3$ to obtain their approximate product C. Output C.

The sparsity of the columns of $\hat{A}$ is bounded by $\epsilon^{-2}\,\mathsf{ns}(A)$ and the sparsity of the rows of $\hat{B}$ is bounded by $\epsilon^{-2}\,\mathsf{ns}(B)$. By the triangle inequality, Jensen inequality, Lemma 3.3.3 and Cauchy-Schwarz inequality,

$$\begin{aligned}\mathbb{E}\|AB - \hat{A}\hat{B}\|_F = \mathbb{E}\|\sum_i A^i B_i - \hat{A}^i\hat{B}_i\|_F &\le \sum_i \mathbb{E}\|A^i B_i - \hat{A}^i\hat{B}_i\|_F \\ &\le \sum_i \sqrt{\mathbb{E}\|A^i B_i - \hat{A}^i\hat{B}_i\|_F^2} \le \tfrac{\epsilon}{3}\sum_i \|A^i\|_2\|B_i\|_2 \le \tfrac{\epsilon}{3}\|A\|_F\|B\|_F.\end{aligned}$$

Additionally, by Jensen's inequality and Lemma 3.3.1,

$$\mathbb{E}\|\hat{A}\|_F \le \sqrt{\mathbb{E}\|\hat{A}\|_F^2} \le \sqrt{\sum_i (1+\tfrac{\epsilon^2}{9})\|A^i\|_2^2} \le (1+\tfrac{\epsilon}{3})\|A\|_F,$$

and similarly for $\hat{B}$. By the triangle inequality and Lemma 3.3.4,

$$\begin{aligned}\mathbb{E}\|C - AB\|_F &\leq \mathbb{E}(\|C - \hat{A}\hat{B}\|_F + \|\hat{A}\hat{B} - AB\|_F) \\ &\leq \tfrac{\epsilon}{3}(1+\tfrac{\epsilon}{3})^2\|A\|_F\|B\|_F + \tfrac{\epsilon}{3}\|A\|_F\|B\|_F \leq \epsilon\|A\|_F\|B\|_F.\end{aligned}$$

Except for the nnz$(\cdot)$ terms, the time it takes to compute the last step is $O(\epsilon^{-6}\,\mathsf{ns}(A)\,\mathsf{ns}(B))$, and the claimed running time follows. □

3.4 Application II: Preconditioning for Ridge Regression

Often, problem-specific preconditioners are used to reduce the condition number of the problem, since the time it takes for iterative methods to converge depends on the condition number. Specifically, for a matrix $M \in \mathbb{R}^{n\times n}$ and a linear-system $Mx = b$, any *invertible* matrix $P \in \mathbb{R}^{n\times n}$ has the property that the solution to the preconditioned linear-system $P^{-1}Mx = P^{-1}b$, is the same as that of the original problem. Using iterative methods to solve the preconditioned problem requires to apply $P^{-1}M$ to a vector in each iteration. In the case of ridge regression, $M = A^\top A + \lambda I$. Applying $(A^\top A + \lambda I)$ to a vector can be done in $O(\text{nnz}(A))$ time, and applying P^{-1} to a vector is equivalent to solving a linear-system in P, i.e. $\arg\min_x \|Px - y\|_2^2$ for some $y \in \mathbb{R}^n$. There is a trade-off between the number of iterations taken to converge for the preconditioned problem, and the time taken to (approximately) solve a linear-system in P. We show in this section how to use the sparsification scheme of Theorem 3.1.2 to construct a preconditioner for ridge-regression, and couple it with an acceleration framework by [59].

Theorem 3.1.6. *There exists an algorithm that, given $A \in \mathbb{R}^{m\times n}, x_0 \in \mathbb{R}^n, \lambda > 0$ and $\epsilon > 0$,*

computes with high probability an ϵ-approximate solution to the ridge regression problem in time

$$O_\epsilon(\text{nnz}(A)) + \tilde{O}_\epsilon\left((\text{nnz}(A))^{2/3}(\mathsf{ns}(A)\,\mathsf{sr}(A))^{1/3}\sqrt{\kappa_\lambda(A^\top A)}\right).$$

Since the term Ax is a linear combination of the columns of A, and the regularization term $\lambda\|x\|_2^2$ penalizes each coordinate of x equally, in practice, the columns of A are often pre-processed to have uniform norms before solving ridge-regression. For this case, in section 3.4.2, we show an improvement of roughly $(n/\,\mathsf{ns}(A))^{1/6}$ over Theorem 3.1.6.

We start by showing that given a matrix $A \in \mathbb{R}^{m\times n}$ and parameter $\lambda > 0$, if $P \in \mathbb{R}^{m\times n}$ is an ϵ-spectral-norm sparsifier for A, for small enough ϵ, the preconditioned problem has a constant condition number, hence requires only a constant number of iterations as described above. This was explored by [70], but they demanded ϵ to be $O(\frac{\lambda}{\|A\|_2^2})$, which is much smaller than necessary. In the next lemma we provide a tighter bound for ϵ.

Lemma 3.4.1. *Given matrix $A \in \mathbb{R}^{m\times n}$, parameters $\lambda > 0$ and $0 < \epsilon' < \frac{1}{2}$, then if a matrix $P \in \mathbb{R}^{m\times n}$ satisfies $\|A - P\|_2 < \epsilon\|A\|_2$ where $\epsilon = \frac{\sqrt{\lambda}\epsilon'}{\|A\|_2}$, then*

$$(1 - 2\epsilon')(A^\top A + \lambda I) \preceq P^\top P + \lambda I \preceq (1 + 2\epsilon')(A^\top A + \lambda I).$$

Setting $\epsilon' = 1/4$ yields that all the eigenvalues of $(P^\top P + \lambda I)^{-1}(A^\top A + \lambda I)$ are in the range $[\frac{2}{3}, 2]$. Using our sampling scheme in Theorem 3.1.2 with parameter ϵ as described here, denoting its output as P, provides a preconditioner for ridge regression with constant condition number. Hence solving this preconditioned problem, i.e, the linear-system $(P^\top P + \lambda I)^{-1}(A^\top A + \lambda I)x = (P^\top P + \lambda I)^{-1}b$ for some vector $b \in \mathbb{R}^n$, with any iterative method, takes $O_\epsilon(\text{nnz}(A) + T_P^\lambda)$ time, where T_P^λ is the time it takes to compute an approximate solution to $\arg\min_x \|(P^\top P + \lambda I)x - y\|_2^2$ for some vector $y \in \mathbb{R}^n$.

Proof of Lemma 3.4.1. For any $x \in \mathbb{R}^n$, by the Triangle inequality,

$$\|Px\|_2 \leq \|Ax\|_2 + \|(P-A)x\|_2 \leq \|Ax\|_2 + \sqrt{\lambda}\epsilon'\|x\|_2.$$

By squaring both sides and applying the AM-GM inequality,

$$\begin{aligned}\|Px\|_2^2 &\leq \|Ax\|_2^2 + \lambda\epsilon'^2\|x\|_2^2 + 2\|Ax\|_2\sqrt{\lambda}\epsilon'\|x\|_2 \\ &\leq \|Ax\|_2^2 + \lambda\epsilon'^2\|x\|_2^2 + \epsilon'\left(\|Ax\|_2^2 + \lambda\|x\|_2^2\right) \\ &= (1+\epsilon')\|Ax\|_2^2 + \lambda\epsilon'(1+\epsilon')\|x\|_2^2,\end{aligned}$$

and since $\epsilon' < 1$,

$$\|Px\|_2^2 + \lambda\|x\|_2^2 \leq (1+\epsilon')\|Ax\|_2^2 + \lambda(1+2\epsilon')\|x\|_2^2.$$

Hence $P^\top P + \lambda I \preceq (1+2\epsilon')(A^\top A + \lambda I)$.

Similarly, we get $\|Px\|_2 \geq \|Ax\|_2 - \sqrt{\lambda}\epsilon'\|x\|_2$, thus $\|Px\|_2^2 \geq (1-\epsilon')\|Ax\|_2^2 - \lambda\epsilon'(1-\epsilon')\|x\|_2^2$ and

$$\|Px\|_2^2 + \lambda\|x\|_2^2 \geq (1-\epsilon')\|Ax\|_2^2 + \lambda(1-2\epsilon')\|x\|_2^2.$$

□

3.4.1 Proof of Theorem 3.1.6

Solving the linear-system in $P^\top P + \lambda I$ can be done by the Conjugate Gradient (CG) method, and can be accelerated by the framework of [59], that, given an algorithm to compute an approximate solution to an Empirical Risk Minimization (ERM) problem, uses the algorithm to provide acceleration in a black-box manner. We restate the guarantees for these algorithms below.

Fact 3.4.2. *For a matrix $M \in \mathbb{R}^{m\times n}$, vector $y \in \mathbb{R}^m$ and parameters $\epsilon, \lambda > 0$, the Conjugate Gradient algorithm returns an ϵ-approximate solution to $\min_x \|Mx - y\|_2^2 + \lambda\|x\|_2^2$ in time $O(\text{nnz}(M)\sqrt{\kappa_\lambda(M)}\log(\frac{1}{\epsilon}))$, which we will denote by $T^\lambda_{\mathsf{CG}}(M, \epsilon)$.*

Lemma 3.4.3 (Acceleration. Theorem 1.1 of [59])**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a λ strongly convex function and for all $x_0 \in \mathbb{R}^n, c > 1, \lambda' > 0$, let $f_{min} = \min_{x\in\mathbb{R}^n}(f(x) + \frac{\lambda'}{2}\|x - x_0\|_2^2)$, assume we can compute $x_c \in \mathbb{R}^n$ in time T_c such that*

$$\mathbb{E}(f(x_c)) - f_{min} \leq \tfrac{1}{c}(f(x_0) - f_{min}),$$

then, given any $x_0 \in \mathbb{R}^n, c > 1, \lambda' \geq 2\lambda$, we can compute x_1 such that

$$\mathbb{E}(f(x_1)) - \min_x(f(x)) \leq \tfrac{1}{c}\big(f(x_0) - \min_x(f(x))\big)$$

in time $O\Big(T_{4(\frac{2\lambda'+\lambda}{\lambda})^{1.5}}\sqrt{\frac{\lambda'}{\lambda}}\log c\Big)$.

The measure of error in the above theorem coincides with the definition we gave for ϵ-approximation to ridge regression, since if $f(x) = \|Ax-b\|_2^2+\lambda\|x\|_2^2$ and $x^* = \text{argmin}_x f(X)$ then for any $x \in \mathbb{R}^n$, $(x - x^*)^T(A^TA + \lambda I)(x - x^*) = 2(f(x) - f(x^*))$. For a proof, see for example [122, Fact 39].

Note that the term $\frac{\lambda'}{2}\|x-x_0\|_2^2$ is not exactly of the same shape as the ridge term $\lambda'\|x\|_2^2$, but since

$$\|Ax - b\|_2^2 + \lambda\|x\|_2^2 + \lambda'\|x - x_0\|_2^2 = \|Ax - b\|_2^2 + (\lambda + \lambda')\|x\|_2^2 - 2\lambda' x_0^\top x + \lambda'\|x_0\|_2^2,$$

solving $\min_x(\|Ax-b\|_2^2+\lambda\|x\|_2^2+\lambda'\|x-x_0\|_2^2)$ is at most as hard as solving ridge regression with vector $A^Tb + \lambda' x_0$ and parameter $\lambda + \lambda'$. We are now ready to prove the result for preconditioned ridge-regression using our sparsifier as a preconditioner.

Proof of Theorem 3.1.6. We first explain how to compute an approximate solution for ridge regression with parameter $\lambda > 0$ and then apply the acceleration framework of Lemma 3.4.3 as a black-box.

Apply the sparsification scheme of Theorem 3.1.2 on A with parameter $\epsilon = \frac{\sqrt{\lambda}}{4\|A\|_2}$ as specified in Lemma 3.4.1 and denote its output by P. Solve the preconditioned linear-system $(P^\top P + \lambda I)^{-1}(A^\top A + \lambda I)x = (P^\top P + \lambda I)^{-1}b$ by any iterative method. As was described earlier, this takes $O_\epsilon(\text{nnz}(A) + T_P^\lambda)$ time. Use Conjugate gradients to solve each linear-system in $P^\top P + \lambda I$. It takes $O_\epsilon(\sqrt{\kappa_\lambda(P^\top P)}\,\text{nnz}(P))$ time. Since $\|P\|_2 \in (1 \pm \epsilon)\|A\|_2$ and $\kappa_\lambda(P^\top P) = \frac{\|P\|_2^2}{\lambda}$, by Theorem 3.1.2,

$$T_{\mathsf{CG}}^\lambda(P, \epsilon) = O_\epsilon\Big(\text{nnz}(P)\sqrt{\kappa_\lambda(P^\top P)}\Big) = O_\epsilon\Big(\frac{\|A\|_2^3}{\lambda^{1.5}}\,\mathsf{ns}(A)\,\mathsf{sr}(A)\log n + \frac{\|A\|_2^2}{\lambda}\sqrt{\mathsf{ns}(A)n \cdot \mathsf{sr}(A)}\log n\Big).$$

Applying the acceleration framework (Lemma 3.4.3) yields a running time of

$$\tilde{O}\Bigg(\Big(\text{nnz}(A) + \frac{\|A\|_2^3}{\lambda'^{1.5}}\,\mathsf{ns}(A)\,\mathsf{sr}(A) + \frac{\|A\|_2^2}{\lambda'}\sqrt{\mathsf{ns}(A)n \cdot \mathsf{sr}(A)}\Big)\sqrt{\frac{\lambda'}{\lambda}}\Bigg).$$

Set $\lambda' = \|A\|_2^2\big(\frac{\mathsf{ns}(A)\,\mathsf{sr}(A)}{\text{nnz}(A)}\big)^{2/3}$. If $n < \mathsf{ns}(A)\,\mathsf{sr}(A)\frac{\|A\|_2^2}{\lambda'} = (\mathsf{ns}(A)\,\mathsf{sr}(A))^{1/3}(\text{nnz}(A))^{2/3}$, which is a reasonable assumption in many cases (for example, if $\text{nnz}(A) > n^{3/2}$), then this choice for λ' balances the two major terms, resulting in the stated running time. □

3.4.2 Faster Algorithm for Inputs with Uniform Row Norms

The best running time, to our knowledge, for the ridge-regression problem on sparse matrices in general is using Stochastic Variance Reduced Gradient Descent (SVRG), originally introduced by [87], coupled with the acceleration framework of [59]. We utilize this method for solving the linear-system for $P^\top P + \lambda I$, where P is the preconditioner. This method is

fastest if the norms of the rows/columns of the input matrix A are uniform. We show the following theorem for solving ridge-regression on numerically sparse matrices with uniform row/column norms.

Theorem 3.4.4. *There exists an algorithm that, given a matrix $A \in \mathbb{R}^{m\times n}$ having uniform rows norms or uniform columns norms, a vector $x_0 \in R^n$ and parameters $\lambda > 0, \epsilon > 0$, computes an ϵ-approximate solution to the ridge regression problem in expected time*

$$O_\epsilon(\mathrm{nnz}(A)) + \tilde{O}_\epsilon\Big(\mathrm{nnz}(A)^{2/3}\sqrt{\mathsf{sr}(A)}\,\mathsf{ns}(A)^{1/3}n^{-1/6}\sqrt{\kappa_\lambda(A^\top A)}\Big).$$

Note that $(A^\top A + \lambda I)^{-1}A^\top = A^\top(AA^\top + \lambda I)^{-1}$. Hence, for any vector v, one can compute an ϵ-approximation for $(A^\top A + \lambda I)^{-1}A^\top v$ in time $O(\mathrm{nnz}(A)) + T^\lambda(A^\top, \epsilon)$. This doesn't change the condition number of the problem, i.e, $\kappa_\lambda(A^\top A) = \kappa_\lambda(AA^\top)$. Hence we only analyze the case where A is pre-processed such that the norms of the rows are uniform.

We provide a theorem from [122] that summarizes the running time of accelerated-SVRG.

Lemma 3.4.5 (Theorem 49 of [122])**.** *For a matrix M, vector $y \in \mathbb{R}^n$ and $\lambda, \epsilon > 0$, there exists an algorithm that computes with high probability an ϵ-approximate solution to $\min_x \|Mx - y\|_2^2 + \lambda\|x\|_2^2$ in time $T^\lambda(M, \epsilon)$ such that*

$$T^\lambda(M,\epsilon) \le O_\epsilon(\mathrm{nnz}(M)) + \tilde{O}_\epsilon\left(\sqrt{\mathrm{nnz}(M)\cdot\frac{\|M\|_F^2}{\lambda}\cdot\mathsf{rsp}(M)}\right).$$

Before we prove Theorem 3.4.4, note the following properties of the sampling in Theorem 3.1.2.

Lemma 3.4.6. *Given a matrix $A \in \mathbb{R}^{m\times n}$, parameter $\epsilon > 0$ and a random matrix $P \in \mathbb{R}^{m\times n}$ satisfying $\|P - A\|_2 \le \epsilon\|A\|_2$ and $\mathbb{E}P = A$, then the expected ℓ_2-norm of the i-th row and of*

the j-th column of P are bounded as

$$\mathbb{E}\|P_i\|_2^2 \leq \|A_i\|_2^2 + \epsilon^2\|A\|_2^2,$$

$$\mathbb{E}\|P^j\|_2^2 \leq \|A^j\|_2^2 + \epsilon^2\|A\|_2^2.$$

Proof. By properties of the spectral-norm, $\|P_i - A_i\|_2 \leq \|P - A\|_2 \leq \epsilon\|A\|_2$. Squaring this and taking the expectation yields $\mathbb{E}(\|P_i\|_2^2) - \|A_i\|_2^2 \leq \epsilon^2\|A\|_2^2$ as desired. The same holds for the columns. One can similarly get an high probability statement. □

Summing over all the rows or columns yields an immediate corollary,

Corollary 3.4.7. *The expected Frobenius-norm of P is bounded as $\mathbb{E}\|P\|_F^2 \leq \|A\|_F^2 + \epsilon^2 \min(n, m)\|A\|_2^2$.*

We are now ready to show the result for ridge-regression in the case that the norms of the rows of the input matrix A are uniform.

Proof of Theorem 3.4.4. We first explain how to compute an approximate solution for ridge regression with parameter $\lambda > 0$ and then apply the acceleration framework of Lemma 3.4.3 as a black-box.

Apply the sparsification scheme of Theorem 3.1.2 on A with parameter $\epsilon = \frac{\sqrt{\lambda}}{4\|A\|_2}$ as specified in Lemma 3.4.1 and denote its output by P. Solve the preconditioned linear-system $(P^\top P + \lambda I)^{-1}(A^\top A + \lambda I)x = (P^\top P + \lambda I)^{-1}b$ by any iterative method. As was described earlier, this takes $O_\epsilon(\text{nnz}(A) + T_P^\lambda)$ time. Use Accelerated-SVRG (Lemma 3.4.5) to solve each linear-system in $P^\top P + \lambda I$.

The bulk of the running time of the Accelerated-SVRG method is in applying vector-vector multiplication in each iteration, where one of the vectors is a row of P. The number of

iterations have dependence on $\mathsf{sr}(P)$, which by Corollary 3.4.7 is bounded by $O(\mathsf{sr}(A) + \frac{n}{\kappa_\lambda})$. The running time of each iteration is usually bounded by the maximum row sparsity, i.e, $\mathsf{rsp}(P)$. Instead, we can bound the expected running time with the expected row sparsity, denote as $s^*(P)$. The distribution for sampling each row is $p_i = \frac{\|P_i\|_2^2}{\|P\|_F^2}$ [122]. Hence, the expected running time will depend on $\sum_i p_i \|P_i\|_0$ instead of $\mathsf{rsp}(P)$. By Lemma 3.4.6 and the assumption that the norms of the rows of A are uniform,

$$s^*(P) = \sum_i p_i \|P_i\|_0 \leq \sum_i \frac{\|A_i\|_2^2 + \lambda}{\|P\|_F^2} \|P_i\|_0 \leq \mathrm{nnz}(P)\Big(\frac{1}{n} + \frac{\lambda}{\|P\|_F^2}\Big) \tag{3.5}$$

Now, by Lemma 3.4.5, equation 3.5 and corollary 3.4.7,

$$\begin{aligned}
T^\lambda(\mathsf{P}, \epsilon) &\leq O_\epsilon\Big(\mathrm{nnz}(P) + \sqrt{\mathrm{nnz}(P) s^*(P)\, \mathsf{sr}(P) \cdot \kappa_\lambda(P^\top P)}\Big) \\
&\leq O_\epsilon\Big(\mathrm{nnz}(P) + \mathrm{nnz}(P)\sqrt{\frac{\mathsf{sr}(P) \cdot \kappa_\lambda(P^\top P)}{n} + 1}\Big) \\
&\leq O_\epsilon\Big(\mathrm{nnz}(P) + \mathrm{nnz}(P)\sqrt{\frac{\mathsf{sr}(A) \cdot \kappa_\lambda(A^\top A)}{n}}\Big) \\
&\leq O_\epsilon(\mathrm{nnz}(P)) + \tilde{O}_\epsilon\Big(\frac{\kappa_\lambda(A^\top A)^{3/2}\, \mathsf{ns}(A)\, \mathsf{sr}(A)^{3/2}}{\sqrt{n}}\Big).
\end{aligned}$$

The last inequality is by plugging in $\mathrm{nnz}(P)$ for the second term. Applying the acceleration framework (Lemma 3.4.3) to the preconditioned problem (i.e, P is a $\frac{c}{\sqrt{\kappa_{\lambda'}(A^\top A)}}$-spectral-norm sparsifier of A), yields running time of

$$\tilde{O}_\epsilon\Big(\mathrm{nnz}(A) + \Big(\mathrm{nnz}(A) + \frac{\kappa_{\lambda'}(A^\top A)^{3/2}\, \mathsf{ns}(A)\, \mathsf{sr}(A)^{3/2}}{\sqrt{n}}\Big)\sqrt{\frac{\lambda'}{\lambda}}\Big)$$

Setting $\lambda' = \frac{\|A\|_2^2\, \mathsf{ns}(A)^{2/3}\, \mathsf{sr}(A)}{n^{1/3}\, \mathrm{nnz}(A)^{2/3}}$ results in the stated running time. □

Chapter 4

Fast Spectral Density Estimation

4.1 Introduction

As discussed earlier, computing a full eigendecomposition of a matrix $A \in \mathbb{R}^{n \times n}$ takes at least $O(n^\omega)$ time[1], which is prohibitively expensive for large matrices [14, 126]. So we are typically interested in extracting *partial information* about the spectrum. This can be done using iterative methods like the power or Lanczos methods, which access A via a small number of matrix-vector multiplications. Each multiplication takes at most $O(n^2)$ time to compute, and can be accelerated when A is sparse or structured, leading to fast algorithms.

However, the partial spectral information computed by most iterative methods is limited. Algorithms typically only obtain accurate approximations to the outlying, or *largest magnitude* eigenvalues of A, missing information about the *interior* of A's spectrum that may be critical in applications. For example, in network science, clusters of interior eigenvalues can indicate graph structures like repeated motifs [49]. In deep learning, information

[1]Here $\omega < 2.373$ is the fast matrix multiplication exponent.

on how the spectrum of a weight matrix differs from its random initialization can give hints about model convergence and generalization [109, 130], and Hessian eigenvalues are useful in optimization [63]. Coarse information about interior eigenvalues is also used to initialize parallel GPU based methods for full eigendecomposition [10, 98].

To address these needs and many other applications, there has been substantial interest in methods for estimating the full *spectral density* of a matrix A [157]. Concretely, assume that A is Hermitian with real eigenvalues $\lambda_1, \ldots, \lambda_n$. We view its spectrum as a probability density s:

$$\text{Spectral density:} \qquad s(x) = \frac{1}{n} \sum_{i=1}^{n} \delta(x - \lambda_i). \tag{4.1}$$

Here δ is the Dirac delta function. The goal is to find a probability density q that approximates s in some natural metric, like the Wasserstein distance. The density q can either be continuous (represented in some closed form) or discrete (represented as a list of approximate eigenvalues $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_n$). See Figure 4.1 for an illustration. Both sorts of approximation are useful in applications.

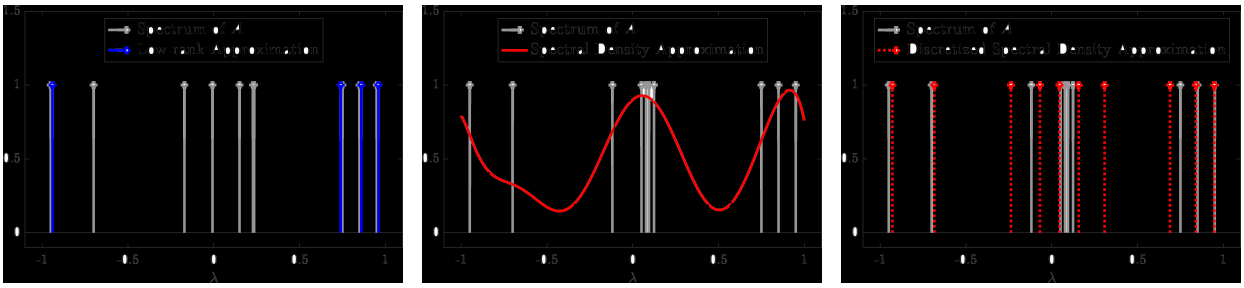

Figure 4.1: Different approximations for the spectrum of a matrix A with eigenvalues in $[-1, 1]$. A typical approximation computed using an iterative eigenvalue algorithm mostly preserves information about the largest magnitude eigenvalues. In contrast, the spectral density estimates in the two right figures coarsely approximate the entire distribution of A's eigenvalues, the first with a low-degree polynomial, and the second with a discrete distribution.

Methods for spectral density estimation that run in $o(n^\omega)$ time were first introduced

for applications in condensed matter physics and quantum chemistry [137, 139, 154]. Many are based on the combination of two important tools: 1) moment matching, and 2) stochastic trace estimation. Specifically, if we had access to moments of the distribution s, i.e. $\frac{1}{n}\sum_{i=1}^{n}\lambda_i$, $\frac{1}{n}\sum_{i=1}^{n}\lambda_i^2$, $\frac{1}{n}\sum_{i=1}^{n}\lambda_i^3$, etc., then we could find a good approximation q by finding a distribution that agrees with s on these moments. Moreover, these *spectral moments* can be computed via the matrix trace: note that $\text{tr}(A) = \sum_{i=1}^{n}\lambda_i$, $\text{tr}(A^2) = \sum_{i=1}^{n}\lambda_i^2$, $\text{tr}(A^3) = \sum_{i=1}^{n}\lambda_i^3$, etc. While we cannot hope to compute $\text{tr}(A^k)$ exactly in $o(n^\omega)$ time, thanks to stochastic trace estimators like Hutchinson's method, this trace can be approximated much more quickly [11, 81]. Such estimators are based on the observation that, for any matrix $B \in \mathbb{R}^{n\times n}$, $\text{tr}(B)$ can be well approximated by $\text{tr}(G^TBG)$ where $G \in \mathbb{R}^{n\times m}$ contains random sub-Gaussian entries and $m \ll n$. For any k degree polynomial g, $G^Tg(A)G$ can be computed with just $O(km)$ matrix-vector multiplications, so we can quickly approximate any low-degree moment of A's spectral density.

While this high-level approach and related techniques have been applied successfully to estimating the spectra of a wide variety of matrices [104, 157], theoretical guarantees have only appeared relatively recently. Perhaps surprisingly, it can be shown that many common methods provably run in *linear time* for any Hermitian matrix A. For instance, in work concurrent to ours, Chen, Trogdan, and Ubaru [31] show that for any $n \times n$ Hermitian matrix A with spectral density s, the popular Stochastic Lanczos Quadrature (SLQ) method provably computes an approximate spectral density q satisfying:

$$W_1(s, q) \leq \epsilon \tag{4.2}$$

using just $poly(1/\epsilon)$ matrix-vector multiplications with A. Above W_1 denotes the Wasserstein-1 distance, aka the "earth-movers distance".[2] We defer a formal definition of W_1 to Section

[2]We assume $\|A\|_2 \leq 1$ for simplicity of stating errror guarantees, noting that Wasserstein distance is not

4.2. The measure is convenient because, unlike many other measures of statistical distance, it allows a discrete distribution like the spectral density to be meaningfully compared to a possibly continuous approximation. For discrete approximations, the Wasserstein distance is related to a simple ℓ_1 metric. If we let $\Lambda = [\lambda_1, \ldots, \lambda_n]$ be a vector of A's eigenvalues and $\tilde{\Lambda} = [\tilde{\lambda}_1, \ldots, \tilde{\lambda}_n]$ be a vector of approximate eigenvalues, then $\|\Lambda - \tilde{\Lambda}\|_1 \leq n\epsilon$ if and only if $W_1(s, q) \leq \epsilon$ for the discrete spectral density q with eigenvalues in $\tilde{\Lambda}$.

As a step towards our main sublinear time result, in this work we show that similar bounds to [31] can also be proven for the popular kernel polynomial method (KPM) [157] and for a natural moment matching algorithm based on Chebyshev polynomials.

4.1.1 Our contributions

With linear time spectral density estimation algorithms in hand for all Hermitian matrices, a natural question is if we can go faster for specific classes of matrices. In particular, there has been growing interest in SDE algorithms for graph structured matrices like adjacency matrices and Laplacians [49]. A remarkable recent result by Cohen et al. [41] shows that, for normalized graph adajeceny matrices, it is possible to achieve guarantee (4.2) in $2^{O(1/\epsilon)}$ time, given appropriate query access to the target graph. Importantly, this runtime *does not depend on* n. However, given the exponential dependence on ϵ, the algorithm is impractical even for coarse spectral approximations.

Our main contribution is a method that obtains a *polynomial* dependence on ϵ, at the cost of a linear dependence on the matrix dimension n. Since A can have n^2 non-zero entries, the runtime is still *sublinear* in the problem size, but with a much more acceptable dependence on accuracy.

scale invariant. This assumption is without loss of generality since $\|A\|_2$ can always be scaled after computing the top eigenvector up to constant fact accuracy, which takes just $O(\log n)$ matrix-vector multiplications [115].

Theorem 4.1.1 (Sublinear time spectral density estimation for graphs.)**.** *Let $G = (V, E)$ be an unweighted, undirected n-vertex graph and let $A \in \mathbb{R}^{n \times n}$ be the normalized adjacency of G with spectral density s. Let $\epsilon, \delta \in (0, 1)$ be fixed values. Assume that we can 1) uniformly sample a random vertex in constant time, 2) uniformly sample a random neighbor of any vertex $i \in V$ in constant time, and 3) for a vertex i with degree d_i, read off all neighbors in $O(d_i)$ time.*[3] *Then there is a randomized algorithm with expected running time $O(npoly(\log(1/\delta)/\epsilon))$ which outputs a density function $q : [-1, 1] \to \mathbb{R}^+$ such that $W_1(q, s) \leq \epsilon$ with probability at least $1 - \delta$.*

Note that the normalized graph Laplacian $L = I - A$ has the same eigenvalues as A up to a shift and reflection, so Theorem 4.1.1 also yields a sublinear time result for normalized Laplacians, whose spectral densities are of interest in network science [49].

Robust spectral density estimation

Theorem 4.1.1 is proven in Section 4.5. A key component of the result is a sublinear time routine for computing coarse approximate matrix-vector products with any normalized graph adjacency matrix. To make use of such a routine, we need to develop an SDE algorithm that is *robust* to the use of an approximate matrix-vector oracle. This is one of the main contributions of our work, as previous methods assume exact matrix-vector products. Formally, we assume access to the oracle:

Definition 4.1.2. An *ϵ_{MV}-approximate matrix-vector multiplication oracle* for $A \in \mathbb{R}^{n \times n}$ and error parameter $\epsilon_{\mathsf{MV}} \in (0, 1)$ is an algorithm that, given any vector $y \in \mathbb{R}^n$, outputs a vector z such that $\|z - Ay\|_2 \leq \epsilon_{\mathsf{MV}} \|A\|_2 \|y\|_2$. We will denote a call to such an oracle for by $\mathrm{AMV}(A, y, \epsilon_{\mathsf{MV}})$.

[3] A standard adjacency list representation of the graph would support these operations. As discussed in Section 4.5, assumption (3) can be eliminated at the cost of an extra $\log n$ in the runtime as long as we know vertex degrees.

In Section 4.4.2 we prove the following for any Hermitian matrix A (e.g., real symmetric) under the assumption that $\|A\|_2 \leq 1$, i.e., that A's eigenvalues lie in $[-1, 1]$:

Theorem 4.1.3 (Robust spectral density estimation)**.** *Let $A \in \mathbb{R}^{n \times n}$ be a Hermitian matrix with spectral density s and $\|A\|_2 \leq 1$. Let C, C', C'' be fixed positive constants. For any $\epsilon, \delta \in (0, 1)$ and $\epsilon_{MV} = C''\epsilon^{-3}\ln(1/\epsilon)$, there is an algorithm (Algorithm 6, with Algorithm 8 used as a subroutine to approximate moments) which makes $T = C\ell/\epsilon$ calls to an ϵ_{MV}-approximate matrix-vector oracle for A, where $\ell = \max\left(1, \frac{C'}{n}\epsilon^{-2}\log^2(\frac{1}{\epsilon\delta})\log^2(\frac{1}{\epsilon})\right)$, and in $poly(1/\epsilon)$ additional runtime, outputs a probability density function $q : [-1, 1] \to \mathbb{R}^{\geq 0}$ such that $W_1(s, q) \leq \epsilon$ with probability $1 - \delta$.*

The requirement for the approximate matrix-vector oracle in Theorem 4.1.3 is relatively weak: we only need accuracy ϵ_{MV} that is polynomial in the final accuracy ϵ. Importantly, there is no dependence on $1/n$, which allows for the theorem to be combined with coarse AMV methods, including the one developed in Section 4.5 for normalized adjacency matrices. Based on random sampling, that method returns an ϵ-approximate matrix-vector multiply in $O(n/\epsilon^2)$ time. This immediately yields our result for graphs given by Theorem 4.1.1. We hope that Theorem 4.1.3 will find broader applications, since spectral density estimation is often applied to matrices where we only have inexact access to A. For example, A might be a Hessian matrix that we can multiply by approximately using stochastic approximation [127, 165], or the inverse of some other matrix, which we can multiply by approximately using an iterative solver.

We note that the result in Theorem 4.1.3 actually *improves* as n increases. Intuitively, when A is larger, each matrix-vector product returns more information about the spectral density s, so we can estimate it more easily. We also remark that the density function q returned by Algorithm 6 is in the form of an $O(1/\epsilon^3)$ dimensional vector, with the i-th entry corresponding to probability mass placed on the i-th point of an evenly spaced grid

on $[-1, 1]$. Alternatively, a simple rounding scheme that runs in $O(n + poly(1/\epsilon))$ time can extract from q a vector of approximate eigenvalues $\tilde{\Lambda} = [\tilde{\lambda}_1, \ldots, \tilde{\lambda}_n]$ satisfying $\|\Lambda - \tilde{\Lambda}\|_1 \leq n\epsilon$, which, as discussed, is ϵ close to the spectral density s in Wasserstein distance (see Theorem 4.8.1).

Our approach for density estimation is based on a moment matching method that approximates *Chebyshev polynomial* moments instead of the standard moments. I.e. we approximate $\text{tr}(T_0(A))$, ..., $\text{tr}(T_N(A))$ where $T_0, \ldots, T_N$ are the Chebyshev polynomials of the first kind and then return a distribution whose Chebyshev moments closely match our approximations. By leveraging Jackson's theorem on polynomial approximation of Lipschitz functions [84], we show how to bound the Wasserstein distance between two distributions in terms of the magnitude of the differences between their first $N = O(1/\epsilon)$ Chebyshev moments (see Lemma 4.3.1). Unlike results for standard moments [92], the bound shows a near-linear relationship between Wasserstein distance and difference in the Chebyshev moments. Ultimately this allows us to obtain a polynomial dependence on ϵ in the number of approximate matrix-vector multiplications needed in Theorem 4.1.3.

Along the way to proving that theorem, in Section 4.4.1 we first establish the follow result that is compatible with exact matrix-vector multiplications:

Theorem 4.1.4 (Linear time spectral density estimation)**.** *Let $A \in \mathbb{R}^{n \times n}$ be a Hermitian matrix with spectral density s and $\|A\|_2 \leq 1$. Let C, C' be fixed positive constants. For any $\epsilon, \delta \in (0, 1)$, there is an algorithm (Algorithm 6, with Algorithm 7 used as a subroutine to approximate moments) which computes $T = C\ell/\epsilon$ matrix-vector multiplications with A where $\ell = \max\left(1, \frac{C'}{n}\epsilon^{-2}\log^2(\frac{1}{\epsilon\delta})\log^2(\frac{1}{\epsilon})\right)$, and in $poly(1/\epsilon)$ additional runtime, outputs a probability density function $q : [-1, 1] \to \mathbb{R}^{\geq 0}$ such that $W_1(s, q) \leq \epsilon$ with probability $1 - \delta$.*

As in Theorem 4.1.3, the theorem improves as n increases, requiring just $T = O(1/\epsilon)$

matrix vector multiplies when $n = \Omega(1/\epsilon^2)$. The runtime of Theorem 4.1.4 is dominated by the cost of the matrix-vector multiplications, which take $O(T \cdot n^2)$ time to compute for a dense matrix, and $O(T \cdot \text{nnz}(A))$ time for a sparse matrix with $\text{nnz}(A)$ non-zero entries, so the algorithm runs in linear time when ϵ, δ are considered constant.

Given Theorem 4.1.4, we prove Theorem 4.1.3 by showing that the error introduced by approximate matrix-vector multiplications does not hinder our ability to estimate the Chebyshev polynomial moments. We do so by drawing on stability results for the three-term recurrence relation defining these polynomials [35, 119].

Remark. The number of matrix-vector multiplies $N\ell = N \cdot \max(1, \frac{C'}{n}\epsilon^{-2}\log^2(\frac{1}{\epsilon\delta})\log^2(\frac{1}{\epsilon}))$ in Theorems 4.1.3 and 4.1.4 can be improved by up to a $\log^2(1/\epsilon)$ factor in the regime when n is small, specifically $n \leq C'\epsilon^{-2}\log^2(1/(\epsilon\delta))$. This is discussed further in Section 4.4.

Spectral density estimation via the kernel polynomial method

In addition to the Chebyshev moment matching method used to give Theorem 4.1.4 and Theorem 4.1.3, we prove that a version of the popular kernel polynomial method (KPM) can be used to obtain a spectral density estimate with similar running times, albeit with slightly worse dependence on the accuracy parameter ϵ.[4] Along with the Stochastic Lanczos Quadrature method, the kernel polynomial method is one of two dominant spectrum estimation algorithms used in practice.

Given sufficiently accurate approximations to the Chebyshev polynomial moments, the KPM method outputs a density function q in the form of a $O(1/\epsilon)$ degree polynomial multiplied by a simple closed form function. This is described in Algorithm 11 in Section 4.7.2 and

[4] We believe that the extra $O(\epsilon^{-2})$ factor in the number of matrix-vector multiplications (or calls to an approximate matrix-vector oracle in the robust setting) may be an artifact of our analysis and can be further improved to match the approximate Chebyshev moment matching bounds.

should be thought of as analagous to Algorithm 6. Specifically, we can obtain Theorem 4.1.4 and Theorem 4.1.3 with $\ell = \max(1, \frac{C'}{n}\epsilon^{-4}\log^2(\frac{1}{\epsilon\delta}))$ and $\epsilon_{\mathsf{MV}} = C''\epsilon^{-4}$ (in the robust setting), by using Algorithm 11 instead of Algorithm 6. Our proof in the KPM case is again based on Jackson's work on polynomial approximations for Lipschitz functions: we take advantage of the fact that Jackson constructs approximations that are both *linear* and *preserve positivity* [83].

4.1.2 Related work

As mentioned, most closely related to our sublinear time result on graphs is the result of Cohen et al. [41]. They prove a result which matches the guarantee of Theorem 4.1.1, but with runtime of $2^{O(1/\epsilon)}$ – i.e., with *no dependence* on n. In comparison, our result depends linearly on n, but only polynomially on $1/\epsilon$. An interesting open question is if a $poly(1/\epsilon)$ time algorithm is possible but we conjecture that the trade-off between the dependence on n and the accuracy ϵ is inherent. Our bound in Lemma 4.3.1 on the Wasserstein-1 distance between two distributions can be seen as analagous to Proposition 1 from [92], which is the basis of the result in [41]. They bound the Wasserstein-1 distance between two distributions in terms of the differences in the standard moments of the distributions. The bound requires an exponentially small dependence on $1/\epsilon$, i.e. $2^{-O(1/\epsilon)}$, in the difference between the standard moments while the bound from Lemma 4.3.1 only requires an $O(\epsilon/\ln(1/\epsilon))$ difference in the Chebyshev moments.

As discussed, algorithms for spectral density estimation have been studied since the early 90s [137, 139, 154] but only analyzed recently. In addition to the work of Chen, Trogdon, and Ubaru that was discussed [31], [121] provides an algorithm for computing an approximate histogram for the spectrum of matrix. That result can be shown to yield an ϵ error approximation to the spectral density in the Wasserstein-1 distance with roughly

$O(1/\epsilon^5)$ matrix-vector multiplications. This compares to the improved $O(1/\epsilon)$ matrix-vector multiplications required by our Theorem 4.1.4.

Matrix-vector query algorithms. Our work fits into a broader line of work on proving upper and lower bounds on the *matrix-vector query complexity* of linear algebraic problems, from top eigenvector, to matrix inversion, to rank estimation [18, 47, 111, 138, 141]. The goal in this model is to minimize the total number of matrix-vector multiplications with A, recognizing that such multiplications either 1) dominate runtime cost or 2) are the only way to access A when it is an implicit matrix. The matrix-vector query model generalizes both classical Krylov subspace methods, as well as randomized sketching methods [159]. Studying other basic linear algebra problem when matrix-vector multiplication queries are only assumed to be approximate (as in Definition 4.1.2) is an interesting future direction.

4.1.3 Paper Roadmap

We describe notation and preliminaries on polynomial approximation in Section 4.2. We use these tools in Section 4.3 to prove that a good approximation to the first $O(1/\epsilon)$ Chebyshev polynomial moments of the spectral density can be used to extract a good approximation in Wasserstein-1 distance. This result is the basis for our result on robust spectral density estimation stated in Theorem 4.1.4 and linear time spectral density estimation stated in Theorem 4.1.3, which are proven in Section 4.4. Finally, we give a randomized algorithm to implement an approximate matrix-vector multplication oracle for adjacency matrices in Section 4.5 and prove our main result, Theorem 4.1.1. In Section 4.7 we describe and analyze the kernel polynomial method, showing that it too can be used to obtain a spectral density estimate given approximations to the first $O(1/\epsilon)$ Chebyshev polynomial moments. In Section 4.6, we empirically investigate the potential of combining approximate matrix-vector multiplications with our moment matching method, the kernel polynomial method,

and the stochastic Lanczos quadrature method studied in [31]. We show that all three can achieve accurate SDE estimates in sublinear time for a variety of graph Laplacians.

4.2 Preliminaries

Throughout we assume that $A \in \mathbb{R}^{n\times n}$ is Hermitian with eigendecomposition $A = U\Lambda U^*$, where $UU^* = U^*U = I_{n\times n}$. We assume that A's eigenvalues satisfy $-1 \le \lambda_n \le \cdots \le \lambda_1 \le 1$. In many applications A is real symmetric. We denote A's spectral density by s, which is defined in (4.1). Our goal is to approximate s in the Wasserstein-1 metric with another distribution q supported on $[-1, 1]$. Specifically, as per the dual formulation given by the Kantorovich-Rubinstein theorem [88], for s, q supported on $[-1, 1]$ the metric is equal to:

$$W_1(s, q) = \sup_{\substack{f:\mathbb{R}\to\mathbb{R} \\ |f(x)-f(y)|\le|x-y| \ \forall x,y}} \left\{ \int_{-1}^{1} f(x)\left(s(x) - q(x)\right) dx \right\}. \tag{4.3}$$

In words, s and q are close in Wasserstein-1 distance if their difference has small inner product with all 1-Lipschitz functions f. Alternatively, $W_1(s, q)$ is equal to the cost of "changing" one distribution to another, where the cost of moving one unit of mass from x to y is $|x - y|$: this is the "earthmover's" formulation common in computer science. Note that (4.3) can be applied to arbitrary functions s, q, even if they are *not distributions*, and we will occasionally do so.

Functions and inner products. We introduce notation for functions used throughout the paper. Let $\mathcal{F}([-1, 1], \mathbb{R})$ denote the space of real-valued functions on $[-1, 1]$. For $g, h \in \mathcal{F}([-1, 1], \mathbb{R})$, let $\langle g, h\rangle$ denote $\langle g, h\rangle \coloneqq \int_{-1}^{1} g(x)h(x)dx$. For $f \in \mathcal{F}([-1, 1], \mathbb{R})$, we define $\|f\|_2 \coloneqq \sqrt{\langle f, f\rangle}$ and let $\|f\|_\infty$ denote the max-norm $\|f\|_\infty = \max_{x\in[-1,1]} |f(x)|$. We let $\|f\|_1$

denote $\|f\|_1 = \int_{-1}^{1} |f(x)|dx$.

Let $\mathcal{F}(\mathbb{Z}, \mathbb{R})$ be the space of real-valued functions on the integers, $\mathbb{Z}$. For $f, g \in \mathcal{F}(\mathbb{Z}, \mathbb{R})$ let $(f * g)$ denote the discrete convolution: $(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$. Let $\mathcal{F}(\mathbb{N}, \mathbb{R})$ be the space of real-valued functions on the natural numbers, $\mathbb{N}$. For functions in $\mathcal{F}(\mathbb{Z}, \mathbb{R})$ or $\mathcal{F}(\mathbb{N}, \mathbb{R})$ we typically used square brackets instead of parentheses.

For two functions f, g let $h = fg$ (or $h = f \cdot g$) and $j = f/g$ denote the pointwise product and quotient respectively. I.e. $h(x) = f(x)g(x)$ and $j(x) = f(x)/g(x)$ for all x.

Chebyshev polynomials. Our approach is based on approximating Chebyshev polynomial moments of A's spectral density, and we will use basic properties of these polynomials, the k^{th} of which we denote T_k. The Chebyshev polynomial of the first kind can be defined via the recurrence:

$$T_0(x) = 1 \qquad T_1(x) = x$$
$$T_k(x) = 2x \cdot T_{k-1}(x) - T_{k-2}(x) \qquad \text{for } k \geq 2.$$

We will use the well known fact that the Chebyshev polynomials of the first kind are bounded between $[-1, 1]$, i.e. $\max_{x \in [-1,1]} |T_k(x)| \leq 1$.

Let $w(x) \coloneqq \frac{1}{\sqrt{1-x^2}}$. It is well known that $\langle T_0, w \cdot T_0 \rangle = \pi$, $\langle T_k, w \cdot T_k \rangle = \pi/2$ for $k > 0$, and

$$\langle T_i, w \cdot T_j \rangle = 0 \qquad \text{for } i \neq j.$$

In other words, the Chebyshev polynomials are orthogonal on $[-1, 1]$ under the weight function w. The first k Chebyshev polynomials form an orthogonal basis for the degree k poly-

nomials under this weight function. We let $\bar{T}_k$ denote the *normalized* Chebyshev polynomial $\bar{T}_k \coloneqq T_k / \sqrt{\langle T_k, w \cdot T_k \rangle}$.

Definition 4.2.1 (Chebyshev Series)**.** The *Chebyshev expansion or series* for a function $f \in \mathcal{F}([-1,1], \mathbb{R})$ is given by

$$\sum_{k=0}^{\infty} \langle f, w \cdot \bar{T}_k \rangle \cdot \bar{T}_k.$$

We call $\sum_{k=0}^{N} \langle f, w \cdot \bar{T}_k \rangle \cdot \bar{T}_k$ the truncated Chebyshev expansion or series of degree N.

Other notation. Let $[n]$ denote $1, \ldots, n$. For a scalar function $f : \mathbb{R} \to \mathbb{R}$ and $n \times n$ matrix A with eigendecomposition $U \Lambda U^*$, we let $f(A)$ denote the matrix function $U f(\Lambda) U^*$. Here $f(\Lambda)$ is understood to mean f applied entrywise to the diagonal matrix Λ containing A's eigenvalues. Note that $\mathrm{tr}(f(A)) = \sum_{i=1}^{n} f(\lambda_i)$. When $f(x)$ is a degree q polynomial, $c_0 + c_1 x + \ldots, c_q x^q$, then we can check that $f(A)$ exactly equals $c_0 I + c_1 A + \ldots, c_q A^q$, where I is then $n \times n$ identity matrix. So $f(A)y$ can be computed for any vector y using q matrix-vector multiplications with A.

4.3 Approximate Chebyshev Moment Matching

In this section we show that the spectral density s of a Hermitian matrix A with eigenvalues in $[-1, 1]$ can be well approximated given access to *approximations* of the first $N = O(1/\epsilon)$ normalized Chebyshev polynomial moments of s, i.e., to approximations of $\mathrm{tr}(\bar{T}_1(A)), \ldots, \mathrm{tr}(\bar{T}_N(A))$. We state our result in Algorithm 6 and analyze it in Section 4.3.1. We show later, in Section 4.4, a method to approximate these moments using a stochastic trace estimator, implemented with either exact or approximate matrix vector multiplications with A.

Given approximations $\tilde{\tau}_1, \ldots, \tilde{\tau}_N$ to the first N normalized Chebyshev moments of A,

a natural approach is to find a probability density $q : [-1, 1] \to \mathbb{R}^+$ such that the first N normalized Chebyshev moments of q, i.e., $\langle \bar{T}_1, q \rangle, \ldots, \langle \bar{T}_N, q \rangle$, closely approximate $\tilde{\tau}_1, \ldots, \tilde{\tau}_N$. In order for this approximate moment matching approach to return a good spectral density estimate, it requires that: *for any density function q, if the first N Chebyshev moments of q closely approximate those of s, then q must be close to s in Wasserstein distance.* To that end, we prove the following lemma:

Lemma 4.3.1. *Let $N \in 4\mathbb{N}^+$ be a degree parameter and p, q be distributions on $[-1, 1]$.*

$$W_1(p, q) \leq \frac{36}{N} + 2 \sum_{k=1}^{N} \frac{|\langle \bar{T}_k, p \rangle - \langle \bar{T}_k, q \rangle|}{k}.$$

Lemma 4.3.1 shows that if the first N normalized Chebyshev moments of two distributions are identical, then the Wasserstein distance between the distributions is at most $O(1/N)$. When the moments between the distributions differ, the contribution of the difference between the k-th moments to the Wasserstein distance is scaled by $O(1/k)$. In particular, the lemma shows that deviation in the lower moments between distributions contributes more to the Wasserstein distance.

To prove Lemma 4.3.1, we will use two well-known results on approximating Lipschitz functions by polynomials. The first is proven in [84]. and concerns uniform approximation of Lipschitz continuous functions by a Chebyshev series:

Fact 4.3.2. *Let $f \in \mathcal{F}([-1, 1], \mathbb{R})$ be a Lipschitz continuous function with Lipschitz constant $\lambda > 0$. Then, for every $N \in 4\mathbb{N}^+$, there exists $N + 1$ constants $\hat{b}_N[0] > \cdots > \hat{b}_N[N] \geq 0$ such that the polynomial $\bar{f}_N = \sum_{k=0}^{N} \frac{\hat{b}_N[k]}{\hat{b}_N[0]} \langle f, w \cdot \bar{T}_k \rangle \bar{T}_k$ has the property that $\max_{x \in [-1,1]} |f(x) - \bar{f}_N(x)| \leq 18\lambda/N$.*

The coefficients of the polynomial in Fact 4.3.2 are not explicitly stated since we only require the existence of such a polynomial in order to prove Lemma 4.3.1. We defer the

reader to Appendix 4.7.1 for an explicit construction of the polynomial[5] and Appendix B.1.6 for a proof of Fact 4.3.2.

Next, we state a well-known fact that the magnitude of the inner-product of a Lipschitz function f with the k-th Chebyshev polynomial (for $k \geq 1$) under the Chebyshev weight function $w = 1/\sqrt{1-x^2}$ is bounded by $O(1/k)$, i.e., $|\langle f, w \cdot \bar{T}_k \rangle| \leq O(1/k)$. Our proof is given in Appendix B.3 and is a simple adaptation of the proof of Theorem 4.2 in [142].

Fact 4.3.3. *Let $f \in \mathcal{F}([-1,1], \mathbb{R})$ be a Lipschitz continuous function with Lipschitz constant $\lambda > 0$. Then, for any $k \geq 1$, we have that $|\langle f, w \cdot \bar{T}_k \rangle| = |\int_{-1}^{1} f(x)\bar{T}_k(x)w(x)dx| \leq 2\lambda/k$.*

With Fact 4.3.2 and 4.3.3 in place, we are now ready to prove Lemma 4.3.1

Proof of Lemma 4.3.1. Recall that the dual formulation of the Wasserstein-1 distance due to Kantorovich-Rubinstein gives us that $W_1(p,q) = \sup_{f \in \text{lip}_1} \int_{-1}^{1} f(x)(p(x) - q(x))dx$ where lip_1 denotes the set of 1-Lipschitz functions on $[-1,1]$. Let $f \in \text{lip}_1$ be an arbitrary 1-Lipschitz function and let $\{\hat{b}_N[k]\}_{k=0}^{N}$ and $\bar{f}_N$ be the coefficients and polynomial respectively from Fact 4.3.2 for function f. We can then bound $W_1(p,q)$ using the triangle inquality as

$$W_1(p,q) \leq \underbrace{\int_{-1}^{1} |f(x) - \bar{f}_N(x)|(p(x) - q(x))dx}_{t_1} + \underbrace{\int_{-1}^{1} \bar{f}_N(p(x) - q(x))dx}_{t_2}.$$

Using the fact that f is Lipschitz and the bound from Fact 4.3.2, along with the fact that p and q are distributions, we have that $t_1 \leq 36/N$.

It is left to bound t_2. We expand t_2 using the Chebyshev series expansion of $\bar{f}_N$ and

[5]The construction of the polynomial $\bar{f}_N$ in Fact 4.3.2 and its uniform approximation to f forms the basis of our alternate approach, the Kernel Polynomial Method, which is discussed in-depth in Appendix 4.7.1.

note that $\langle g/w, w \cdot \bar{T}_k \rangle = \langle g, \bar{T}_k \rangle$ for any function $g \in \mathcal{F}([-1,1], \mathbb{R})$, giving us

$$t_2 = \int_{-1}^{1} \bar{f}_N(x) w(x) \cdot \frac{p(x) - q(x)}{w(x)} dx = \int_{-1}^{1} \bar{f}_N(x) w(x) \cdot \sum_{k=0}^{\infty} \langle p - q, \bar{T}_k \rangle \bar{T}_k(x) dx$$
$$= \int_{-1}^{1} \left(w(x) \sum_{k=0}^{N} \frac{\hat{b}_N[k]}{\hat{b}_N[0]} \langle f, w \cdot \bar{T}_k \rangle \bar{T}_k(x) \right) \left(\sum_{k=0}^{\infty} \langle p - q, \bar{T}_k \rangle \bar{T}_k(x) \right) dx.$$

By the orthogonality of the Chebyshev polynomials under the weight function w and the fact that $\langle \bar{T}_k, \bar{T}_k \rangle = 1$ for all $k \in [N]$, we can bound the magnitude of t_2 as

$$|t_2| \leq \sum_{k=1}^{N} |\langle f, w \cdot \bar{T}_k \rangle| \cdot |\langle \bar{T}_k, p \rangle - \langle \bar{T}_k, q \rangle|$$

since we have that $0 \leq \hat{b}_N[k]/\hat{b}_N[0] \leq 1$ and $|\int_{-1}^{1} \bar{T}_k(p(x) - q(x))dx| = |\langle \bar{T}_k, p \rangle - \langle \bar{T}_k, q \rangle|$ for each $k \in [N]$. Additionally, since p and q are distributions we have that $\langle \bar{T}_0, s \rangle = \langle \bar{T}_0, z \rangle = 1/\sqrt{\pi}$. We then use the bound from Fact 4.3.3 on $|\langle f, w \cdot \bar{T}_k \rangle|$ for each $k \in [N]$. Putting this together gives us that $|t_2| \leq \sum_{k=1}^{N} 2|\langle \bar{T}_k, p \rangle - \langle \bar{T}_k, q \rangle|/k$.

Putting together the bound on t_1 and t_2 gives us the bound on $W_1(p, q)$. □

4.3.1 Moment Matching Algorithm

With Lemma 4.3.1 in place, our next step is develop a method to find a distribution q with Chebyshev moments closely matching a given set of target moments. In order to search for a distribution, we consider an evenly-spaced grid of the interval $[-1, 1]$. Specifically, let $d \in \mathbb{N}^+$ be a discretization parameter and let $X_d = [-1, -1 + \frac{2}{d}, \ldots, 1 - \frac{2}{d}, 1]$ be a $(d+1)$-length evenly-spaced grid of the interval $[-1, 1]$. Our goal is to output a distribution supported on X_d for an appropriately chosen value of d. Any such distribution can be described by a vector in $\mathbb{R}^d_{\geq 0}$ such that the i-th entry corresponds to the probability mass placed at point

$-1 + 2i/d$ on the grid. Where it is clear from the context, we will denote the distribution and its probability mass vector interchangeably.

In order to compute the first N normalized Chebyshev moments of functions on the grid X_d, we define two matrices $\mathcal{T}_N^d, \widehat{\mathcal{T}}_N^d \in \mathbb{R}^{N \times d}$ such that for $k \in [N]$ and $i \in [d]$,

$$(\mathcal{T}_N^d)_{k,i} = \bar{T}_k(-1 + 2i/d) \qquad \text{and} \qquad (\widehat{\mathcal{T}}_N^d)_{k,i} = \frac{\bar{T}_k(-1 + 2id)}{k}.$$

The matrix $\mathcal{T}_N^d$ corresponds to a "discretization" of the continuous operator that computes the first N normalized Chebyshev moments of a continuous function on $[-1, 1]$. In particular, for a distribution q supported on X_d, we have that $\langle q, \bar{T}_k \rangle = \sum_{i=0}^{d} q_i \bar{T}_k(-1+2i/d) = (\mathcal{T}_N^d q)_k$. Notice that the matrix $\mathcal{T}_N^d$ does not contain the row for $\bar{T}_0$; since we are working with distributions we know that $\bar{T}_0(q) = 1/\sqrt{\pi} \cdot \int_{-1}^{1} q dx = 1/\sqrt{\pi}$ for any distribution q on $[-1, 1]$. The matrix $\widehat{\mathcal{T}}_N^d$ is the matrix $\mathcal{T}_N^d$ with the k-th row scaled by $1/k$. With this notation in place, we state the approximate moment matching algorithm in full in Algorithm 6.

Algorithm 6 Approximate Chebyshev Moment Matching

Input: Symmetric $A \in \mathbb{R}^{n \times n}$, degree parameter $N \in 4\mathbb{N}^+$, algorithm $\mathcal{M}(A)$ that computes moment approximations $\tilde{\tau}_1, \ldots, \tilde{\tau}_N$ with the guarantee that $|\tilde{\tau}_k - \frac{1}{n}\operatorname{tr}(\bar{T}_k(A))| \leq (N \ln(eN))^{-1}$ for all k.

Output: A vector q corresponding to a discrete density function on $[-1, 1]$.

1: For $k = 1, \ldots, N$ use $\mathcal{M}$ to compute $\tilde{\tau}_1, \ldots, \tilde{\tau}_N$ and set $z = [\tilde{\tau}_1/1, \tilde{\tau}_2/2, \ldots, \tilde{\tau}_N/N]$.

2: Set $d = \lceil N^3/2 \rceil$ and compute matrix $\widehat{\mathcal{T}}_N^d \in \mathbb{R}^{N \times d}$. ▷ $(\widehat{\mathcal{T}}_N^d)_{k,i} = \bar{T}_k(-1 + \frac{2i}{d})/k$.

3: Minimize $\|\widehat{\mathcal{T}}_N^d q - z\|_1$ subject to $q^\top \vec{1} = 1$ and $q \geq 0$.

4: Return q.

Note that the optimization problem in Line 3 of Algorithm 6 can easily be written as a linear program in $O(d + N)$ variables and constraints and hence can be solved efficiently in $poly(N, d) = poly(1/\epsilon)$ time[6]. Since this method is independent of the matrix dimension n,

[6] Additionally, note that the optimization problem has a convex objective and constraints – in particular, the set of distributions supported on X_d is a convex set. The objective function $\|\widehat{\mathcal{T}}_N^d q - z\|_1$ is not differentiable, but has subgradients. Hence, this program can be solved efficiently in $poly(1/\epsilon)$ time using a projected

it is a lower order term in the running time stated in Theorems 4.1.4 and 4.1.3, as we will discuss in Section 4.4.

We show that when $N = O(1/\epsilon)$, Algorithm 6 returns a distribution satisfying $W(s, q) \leq \epsilon$.

Lemma 4.3.4. *Let $\epsilon \in [0, 1]$ and let $N \geq 18/\epsilon$. Then the distribution $q : [-1, 1] \to \mathbb{R}^+$ returned by Algorithm 6 satisfies $W_1(q, s) \leq 3\epsilon$.*

Proof. We start by giving some notation – for a distribution $y : [-1, 1] \to \mathbb{R}^+$, we denote $\vec{\tau}_y \coloneqq [\langle \bar{T}_1, y\rangle, \ldots, \langle \bar{T}_N, y\rangle]$ to be the vector of the first N normalized Chebyshev moments of y. For an integer $k \in \mathbb{N}^+$, we denote $\vec{k}$ to be the vector in $\mathbb{R}^k$ given by $\vec{k} \coloneqq [1, \ldots, k]$ and for a vector $y \in \mathbb{R}^k$ write $y/\vec{k}$ to denote the vector $y/\vec{k} \coloneqq [y_1/1, \ldots, y_k/k]$. Notice then that we have $\vec{\tau}_q = \mathcal{T}_N^d q$ and $\vec{\tau}_q/\vec{N} = \widehat{\mathcal{T}}_N^d q$.

We start by bounding the scaled differences in the first N normalized Chebyshev moments of q and s in order to use Lemma 4.3.1 on q and s.

$$\|\vec{\tau}_q/\vec{N} - \vec{\tau}_s/\vec{N}\|_1 \leq \|\vec{\tau}_q/\vec{N} - z\|_1 + \|z - \vec{\tau}_s/\vec{N}\|_1 \leq \|\vec{\tau}_q/\vec{N} - z\|_1 + \frac{1}{N}. \tag{4.4}$$

The first inequality follows by applying the triangle inequality and in the second inequality we used the fact that $\|z - \vec{\tau}_s/\vec{N}\|_1 = \sum_{k=1}^N |\tilde{\tau}_k - (\vec{\tau}_s)_k|/k \leq H_n \cdot (N \ln(eN))^{-1} \leq 1/N$.

Next we show that there exists a distribution q' supported on X_d such that $\|\vec{\tau}_{q'}/\vec{N} - z\| \leq 1/N$. To this end, consider the following distribution q^* on X_d:

$$q^*(x) = \frac{1}{n}\sum_{i=1}^{n} \delta(x - \operatorname*{argmin}_{p \in X_d} |p - \lambda_i|).$$

subgradient method. This requires an oracle that projects onto the the probability simplex supported on the grid X_d – an algorithm that runs in $O(d \log d)$ time has been given in multiple papers, see [155] for more details.

In words, q^* is the distribution corresponding to moving the mass from each λ_i to its nearest point on the grid X_d. Notice that we have $W_1(s, q^*) \leq 1/d$ due to the earthmover distance interpretation of the Wasserstein-1 distance.

Applying the triangle inequality and the guarantee from the moment approximations, we get that $\|\vec{\tau}_{q^*}/\vec{N} - z\|_1 \leq 1/N + \|\vec{\tau}_{q^*}/\vec{N} - \vec{\tau}_s/\vec{N}\|_1$. It is left then to bound $\|\vec{\tau}_{q^*}/\vec{N} - \vec{\tau}_s/\vec{N}\|_1$. To this end, we state the following well-known fact about the derivatives of Chebyshev polynomials.

Fact 4.3.5. *For $k \geq 1$, $\frac{d\bar{T}_k(x)}{dx} = kU_{k-1}(x)$.*

We then have using the definition of q^* that, for any $1 \leq k \leq N$,

$$|\langle \bar{T}_k, s - q^* \rangle| = \left| \frac{1}{n} \sum_{i=1}^{n} \bar{T}_k(\lambda_i) - \bar{T}_k(\operatorname*{argmin}_{p \in X_d} |p - \lambda_i|) \right| \leq \frac{1}{n} \sum_{i=1}^{n} \left| \bar{T}_k(\lambda_i) - \bar{T}_k(\operatorname*{argmin}_{p \in X_d} |p - \lambda_i|) \right|$$
$$\leq \frac{\sqrt{2}}{n\sqrt{\pi}} \sum_{i=1}^{n} \max_{x \in [-1,1]} \left| \frac{dT_k(x)}{dx} \right| \cdot |\lambda_i - \operatorname*{argmin}_{p \in X_d} |p - \lambda_i|| \leq \frac{\sqrt{2}k^2}{d\sqrt{\pi}}$$

where in the last inequality we used the fact that $\max_{x \in [-1,1]} |U_{k-1}(x)| \leq k$. It follows then that

$$\|\vec{\tau}_{q^*}/\vec{N} - \vec{\tau}_s/\vec{N}\|_1 = \sum_{k=1}^{N} \frac{|(\vec{\tau}_{q^*})_k - (\vec{\tau}_s)_k|}{k} \leq \frac{N(N+1)}{d\sqrt{2\pi}} \leq \frac{1}{N}$$

by taking the sum over all k and noting that $d \geq N^3/2$. Putting these bounds together gives us that $\|\vec{\tau}_{q^*}/\vec{N} - z\|_1 \leq 2/N$.

Since $\|\vec{\tau}_q/\vec{N} - z\|_1 \leq \|\vec{\tau}_{q^*}/\vec{N} - z\|_1$ from Line 3 of Algorithm 6, we plug this into (4.4) to get that $\|\vec{\tau}_q/\vec{N} - \vec{\tau}_s/\vec{N}\|_1 \leq 3/N$. We can then use Lemma 4.3.1 with distributions q and s along with the fact that $\|\vec{\tau}_q/\vec{N} - \vec{\tau}_s/\vec{N}\|_1 = \sum_{k=1}^{N} |(\vec{\tau}_s)_k - (\vec{\tau}_q)_k|/k \leq 3/N$ to give us the result since $N > 18/\epsilon$. □

Remark. Note that Algorithm 6 can easily be adapted when the minimization problem in Line 3 is solved approximately – as is the case if projected subgradient descent methods are used. In particular, a constant factor approximation to the minimal loss increases the Wasserstein distance bound in Lemma 4.3.4 by an $O(1)$ factor.

4.4 Efficient Chebyshev Moment Approximation

With Lemma 4.3.4 in place, we are ready to prove our main results. To do so, we need to show how to efficiently approximate the first N Chebyshev moments of a matrix A's spectral density s, as required by Algorithm 6. Recall that the k^{th} normalized Chebyshev moment of s is equal to $\langle s, \bar{T}_k \rangle = \frac{1}{n} \operatorname{tr}(\bar{T}_k(A))$. We will prove that this trace can be approximated using Hutchinson's stochastic trace estimator, implemented with either exact or approximate matrix-vector multiplications with A.

This estimator requires repeatedly computing $\bar{T}_k(A)g$ for a random vector g, which is done using the standard three-term (forward) recurrence for the Chebyshev polynomials and requires a total of k matrix-vector multiplications with A. We analyze the basic approach in Section 4.4.1, which yields Theorem 4.1.4. Then in Section 4.4.2, we argue that the approach is stable even when implemented with approximate matrix-vector multiplication, which yields Theorem 4.1.3.

4.4.1 Exact Matrix-Vector Multiplications

Hutchinson's estimator is a widely used estimator to efficiently compute accurate estimates of $\operatorname{tr}(R)$ for any square matrix $R \in \mathbb{R}^{n \times n}$. Each instance of the estimator computes the quadratic form $g^\top R g$ for a random vector $g \in \{-1, 1\}^n$ whose entries are Rademacher

random variables. This an unbiased estimator for $\mathrm{tr}(R)$ with variance $\leq 2\|R\|_F^2$, and its error has been analyzed in several earlier results [11, 131]. We apply a standard high-probability bound from [111, 132]:

Lemma 4.4.1 (Lemma 2, [111])**.**[7] *Let $R \in \mathbb{R}^{n\times n}$, $\delta \in (0, 1/2]$, $l \in \mathbb{N}$. Let $g^{(1)}, \ldots, g^{(\ell)} \in \{-1, 1\}^{n\times n}$ be ℓ random vectors with i.i.d $\{-1, +1\}$ random entries. For a fixed constant C, with probability at least $1 - \delta$,*

$$|\mathrm{tr}(R) - \frac{1}{\ell}\sum_{i=1}^{l}(g^{(i)})^\top R g^{(i)}| \leq \frac{C\log(1/\delta)}{\sqrt{\ell}}\|R\|_F.$$

For a polynomial $p \in \mathcal{F}([-1, 1], \mathbb{R})$ with degree k, applying Hutchinson's estimator to $R = p(A)$ requires computing $p(A)g$, which can always be done with k matrix-vector multiplies with A. If $p(x)$ admits a recursive construction, like the Chebyshev polynomials, then this recurrence can be used. Specifically, for the Chebyshev polynomials, we have:

$$\begin{aligned} &T_0(A)g = g \qquad T_1(A)g = Ag \\ &T_k(A)g = 2A \cdot T_{k-1}(A)g - T_{k-2}(A)g \qquad \text{for } k \geq 2. \end{aligned} \tag{4.5}$$

A moment estimation algorithm based on Hutchinson's estimator is stated as Algorithm 7.

Remark. In total, Algorithm 7 requires $N \cdot \ell$ matrix multiplications with A since for each i $T_1(A)g^{(i)}, \ldots, T_N(A)g^{(i)}$ can but computed using the same N steps of the (4.5) recurrence. It requires $O(n\ell N)$ additional runtime to compute and sum all inner products of the form $(g^{(i)})^T T_k(A)g^{(i)}$.

[7] In [111] the lemma is stated with an assumption that $\ell > O(1/\delta)$. However, it is easy to see that the same claim holds without this assumption, albeit with a quadratically worse $\log(1/\delta)$ dependence. The proof follows from same application of the Hanson-Wright inequality used in that work.

Algorithm 7 Hutchinson Moment Estimator

Input: Symmetric $A \in \mathbb{R}^{n\times n}$ with $\|A\|_2 \leq 1$, degree $N \in 4\mathbb{N}^+$, number of repetitions $\ell \in \mathbb{N}^+$.

Output: Approximation $\tilde{\tau}_k$ to moment $\frac{1}{n}\operatorname{tr}(\bar{T}_k(A))$ for all $k \in 1, \ldots, N$.

1: Draw $g^{(1)}, \ldots, g^{(l)} \sim \text{Uniform}(\{-1, 1\}^n)$.

2: For $k = 1, \ldots, N$, $\tilde{\tau}_k \leftarrow \frac{\sqrt{2/\pi}}{\ell n}\sum_{i=1}^{l}(g^{(i)})^\top T_k(A)g^{(i)}$. ▷ Computed using recurrence in (4.5)

3: Return $\tilde{\tau}_1, \ldots, \tilde{\tau}_N$.

Our main bound on the accuracy of Algorithm 7 follows:

Lemma 4.4.2. *If Algorithm 7 is run with $\ell = \max\left(1, C \cdot \log^2(N/\delta)/(n\Delta^2)\right)$, where C is a fixed positive constant, then with probability $1-\delta$ the approximate moments returned satisfy $|\tilde{\tau}_k - \frac{1}{n}\operatorname{tr}(\bar{T}_k(A))| \leq \Delta$ for all $k = 1, \ldots, N$.*

Proof. Fix $k \in \{1, \ldots, N\}$. Note that $\frac{1}{n}\operatorname{tr}(\bar{T}_k(A)) = \frac{\sqrt{2/\pi}}{n}\operatorname{tr}(T_k(A))$. Let C be the constant from Lemma 4.4.1. If $\ell = \max\left(1, C^2 \cdot \log^2(N/\delta)/(n\Delta^2)\right)$, then by that lemma we have that with probability at least $1 - \delta/N$:

$$|\tilde{\tau}_k - \frac{\sqrt{2/\pi}}{n}\operatorname{tr}(T_k(A))| \leq \frac{1}{n}\frac{C\log(N/\delta)}{\sqrt{\ell}}\|T_k(A)\|_F \leq \frac{C\sqrt{2/\pi}}{\sqrt{n}}\sqrt{\frac{\log(N/\delta)}{\ell}} \leq \Delta.$$

The second to last inequality follows from the fact that $\|T_k(A)\|_2 \leq 1$ and thus $\|T_k(A)\|_F \leq \sqrt{n}$. Applying a union bound over all $k \in 1, \ldots, N$ gives the claim. □

Theorem 4.1.4 immediately follows as a corollary of Lemma 4.4.2 and Lemma 4.3.4.

Proof of Theorem 4.1.4. We implement Algorithm 6 with Algorithm 7 used as a subroutine to approximate the Chebyshev polynomial moments, which requires setting $\Delta = \frac{1}{N\ln(eN)}$. By Lemma 4.4.2, we conclude that we need to set $\ell = \max\left(1, CN^2\log^2(N/\delta)\log^2(eN)/n\right)$. Then, by Lemma 4.3.4, setting $N = O(1/\epsilon)$ ensures that Algorithm 6 returns a distribution q which is ϵ close to A's spectral density s in Wasserstein distance. □

4.4.2 Approximate Matrix-Vector Multiplications

Algorithm 7 assumes access to an oracle for computing exact matrix-vector multiplies with A. In this section, we show that the method continues to work well even when each term in Hutchinson's estimator, $g^\top T_k(A)g$, is computed using an approximate matrix-vector multiplication oracle for A (see Definition 4.1.2). As discussed in Section 4.1.1, the robustness of the estimator allows the approximate moment matching method to be applied in many settings where A can only be access implicitly. It also forms the basis of our sublinear time algorithm for computing the spectral density of a normalized graph adjacency or Laplacian matrix, which are presented in the Section 4.5.

To show that approximate matrix-vector multiplications suffice, we leverage well understood stability properties of the three-term forward recurrence for Chebyshev polynomials of the first kind [35, 119]. These properties allows us to analyze the cumulative error when $T_k(A)g$ is computed via this recurrence. Specifically, we analyze the following algorithm:

Algorithm 8 Hutchinson Moment Estimator w/ Approximate Multiplications

Input: Symmetric $A \in \mathbb{R}^{n\times n}$ with $\|A\|_2 \leq 1$, degree $N \in 4\mathbb{N}^+$, number of repetitions $\ell \in \mathbb{N}^+$, ϵ_{MV}-approximate matrix vector multiplication oracle AMV for A (see Definition 4.1.2).

Output: Approximation $\tilde{\tau}_k$ to moment $\frac{1}{n}\operatorname{tr}(\bar{T}_k(A))$ for all $k \in 1, \ldots, N$.

for $i = 1, \ldots, \ell$ iterations **do**
  Draw $g \sim \text{Uniform}(\{-1, 1\}^n)$.
  $\tilde{v}_0 \leftarrow g$, $\tilde{v}_1 \leftarrow \text{AMV}(A, g, \epsilon_{\mathsf{MV}})$.
  $\tilde{\tau}_{1,i} \leftarrow g^T\tilde{v}_1$
  for $k = 2$ to N **do**
    $\tilde{v}_k \leftarrow 2 \cdot \text{AMV}(A, \tilde{v}_{k-1}, \epsilon_{\mathsf{MV}}) - \tilde{v}_{k-2}$.
    $\tilde{\tau}_{k,i} \leftarrow g^T\tilde{v}_k$
For $k = 1, \ldots, N$, $\tilde{\tau}_k \leftarrow \frac{1}{\ell}\sum_{i=1}^{\ell}\tilde{\tau}_{k,i}$.
Return $\tilde{\tau}_1, \ldots, \tilde{\tau}_N$.

Algorithm 8 assumes access to an approximate matrix-vector multiplication oracle for

A with error ϵ_{MV} (recall Definition 4.1.2). Since $\|A\|_2 \leq 1$, for any vector y, we have that:

$$\|\mathrm{AMV}(A, y, \epsilon_{\mathsf{MV}}) - Ay\|_2 \leq \epsilon_{\mathsf{MV}} \|y\|_2. \tag{4.6}$$

The algorithm uses this oracle to apply the recurrence from (4.5), approximately computing each $T_k(A)g$ for $k = 1, \ldots, N$, which in turn allows us to approximately compute $g^\top T_k(A)g$. Note that when $\epsilon_{\mathsf{MV}} = 0$, Algorithm 8 is exactly equivalent to Algorithm 7.

Notation. Analyzing this approach requires accounting for error accumulates across iterations. To do so, we introduce some basic notation. Let v_k denote the true value of $T_k(A)g$, and let $\tilde{v}_k$ denote our computed approximation. We initialize the recurrence with $\tilde{v}_{-1} = \vec{0}$ and $\tilde{v}_0 = v_0 = g$. For $k = 0, \ldots, N-1$, let $w_k = \mathrm{AMV}(A, \tilde{v}_k, \epsilon_{\mathsf{MV}})$ and note that $\|w_k - A\tilde{v}_k\|_2 \leq \epsilon_{\mathsf{MV}} \|\tilde{v}_k\|_2$. In iteration k of the recurrence, we compute $\tilde{v}_{k+1}$ by applying the recurrence:

$$\tilde{v}_{k+1} \coloneqq 2w_k - \tilde{v}_{k-1}.$$

For each $i \in 0, \ldots, N$ we denote:

- $\delta_k \coloneqq v_k - \tilde{v}_k$, with $\delta_0 = \vec{0}$. This is the *accumulated error* up to iteration k.
- $\xi_{k+1} \coloneqq A\tilde{v}_k - w_k$, with $\xi_0 = 0$. $2\xi_{k+1}$ is the *new error* introduced in iteration k due to approximate matrix-vector multiplication.

As in Clenshaw's classic work [35], it can be shown that δ_k *itself evolves according to a simple recurrence*, which ultimately lets us show that it can be expressed as a summation involving Chebyshev polynomials of the *second* kind, which are easily bounded. Specifically, we have:

Fact 4.4.3. *$\delta_1 = \xi_1$ and for $2 \leq k \leq N$, $\delta_k = 2A\delta_{k-1} - \delta_{k-2} + 2\xi_k$.*

Proof. The claim for δ_1 is direct since $v_0 = \tilde{v}_0$: we have $\delta_1 = v_1 - \tilde{v}_1 = Av_0 - w_0$. For $2 \leq k \leq N$, we prove the claim by writing the difference $\delta_k = v_k - \tilde{v}_k = v_k - 2(A\tilde{v}_{k-1} + \xi_k) + \tilde{v}_{k-2}$. We can then replace $v_k = 2Av_{k-1} - v_{k-2}$ and substitute in $(v_{k-1} - \tilde{v}_{k-1}) = \delta_{k-1}$ and $(v_{k-2} - \tilde{v}_{k-2}) = \delta_{k-2}$. □

The Chebyshev polynomials of the second kind are defined via the following recurrence:

Definition 4.4.4 (Chebyshev Polynomials of the Second Kind)**.** For $k \in \mathbb{N}^{\geq 0}$ the k-th Chebyshev polynomial of the second kind $U_k(x)$ is given by

$$U_0(x) = 1 \qquad U_1(x) = 2x$$
$$U_k(x) = 2x \cdot U_{k-1}(x) - U_{k-2}(x) \qquad \text{for } k \geq 2.$$

We also define $U_{-1}(x) = 0$, which is consistent with the recurrence.

Using these polynomials, we can characterize the accumulated error δ_k in terms of the error introduced in each of the prior iterations.

Lemma 4.4.5. *For $k = 1, \ldots, N$, we have*

$$\delta_k = U_{k-1}(A)\xi_1 + 2\sum_{i=2}^{k} U_{k-i}(A)\xi_i. \tag{4.7}$$

Proof. We prove the lemma by induction on $j \leq k$. For $j = 0$, the lemma is trivial since $\delta_0 = 0$ by definition and $U_{-1}(A) = 0$. For $j = 1$, $\delta_1 = \xi_1 = U_0(A)\xi_1$. By Fact 4.4.3, for $2 \leq j < k$, we have:

$$\delta_j = 2\xi_j + \underbrace{2A\delta_{j-1} - \delta_{j-2}}_{z_1}. \tag{4.8}$$

We can apply the inductive hypothesis on z_1 and recombine terms using Definition 4.4.4 to get:

$$\begin{aligned} z_1 &= 2A \cdot \left(U_{j-2}(A)\xi_1 + 2\sum_{i=2}^{j-1} U_{j-1-i}(A)\xi_i \right) - U_{j-3}(A)\xi_1 - 2\sum_{i=2}^{j-2} U_{j-2-i}(A)\xi_i \\ &= U_{j-1}(A)\xi_1 + U_1(A)\cdot 2\xi_{j-1} + \sum_{i=2}^{j-2} \left(2AU_{j-1-i}(A) - U_{j-2-i}(A)\right)\cdot 2\xi_i \\ &= U_{j-1}(A)\xi_1 + \sum_{i=2}^{j-1} U_{j-i}(A)\cdot 2\xi_i \end{aligned}$$

Noting that plugging into (4.8) and noting that $2\xi_j = 2U_0(A)\xi_j$ completes the proof. □

Our goal is to use Lemma 4.4.5 to establish that δ_k is small because each ξ_i is small. It is well known that the Chebyshev polynomials of the second kind satisfy the following bounds for any $k \in \mathbb{N}$:

$$|U_k(x)| \leq k+1 \qquad \text{for} \qquad x \in [-1,1]. \tag{4.9}$$

This is the upper bound we need to proceed. Specifically, we will show that each estimator using Algorithm 8, $g^\top \tilde{v}_k$, well approximates Hutchinson's estimator $g^\top T_k(A)g = g^\top v_k$.

Claim 4.4.6. *For quantities $v_k, \tilde{v}_k$ and $0 \leq \epsilon_{MV} \leq 1/2k^2$, we have*

$$\left| g^\top T_k(A)g - g^\top \tilde{v}_k \right| \leq 2\,\epsilon_{MV}\cdot(k+1)^2\|g\|_2^2.$$

Proof. By the definition of δ_k, we have $|g^\top T_k(A)g - g^\top \tilde{v}_k| = |g^\top \delta_k|$. By Cauchy-Schwarz we can bound $|g^\top \delta_k| \leq \|g\|_2\|\delta_k\|_2$. We are left to bound $\|\delta_k\|_2$. Applying Lemma 4.4.5 and

triangle inequality, we have

$$\|\delta_k\|_2 \leq \|U_{k-1}(A)\|_2\|\xi_1\|_2 + \sum_{i=2}^{k} 2\|U_{k-i}(A)\|_2\|\xi_i\|_2$$

Then applying (4.9) and the fact that $\|A\|_2 \leq 1$, we have $\|U_{k-i}(A)\|_2 \leq (k-i+1)$. Hence,

$$\|\delta_k\|_2 \leq k\|\xi_1\|_2 + \sum_{i=2}^{k} 2(k-i+1)\|\xi_i\|_2 \leq \sum_{i=1}^{k} 2(k-i+1)\|\xi_i\|_2.$$

Using that $\xi_i \leq \epsilon_{\mathsf{MV}} \|\tilde{v}_{i-1}\|_2$, and that $\|T_i(A)\|_2 \leq 1$ for all i and thus $\|v_i\|_2 \leq \|g\|_2$, we have:

$$\begin{aligned}\|\delta_k\|_2 &\leq \sum_{i=1}^{k} 2(k-i+1)\,\epsilon_{\mathsf{MV}}\,\|\tilde{v}_{i-1}\|_2 \leq 2\,\epsilon_{\mathsf{MV}} \sum_{i=1}^{k}(k-i+1)(\|v_{i-1}\|_2 + \|\delta_{i-1}\|_2) \\ &\leq \epsilon_{\mathsf{MV}}\, k(k+1)\left(\|g\|_2 + \max_{i<k}\|\delta_i\|_2\right).\end{aligned}$$

Inducting on δ_j for $j \leq k$ gives us $\|\delta_k\|_2 \leq 2\,\epsilon_{\mathsf{MV}}(k+1)^2\|g\|_2$, which completes the proof. □

Lemma 4.4.7. *If Algorithm 8 is run with* $\ell = \max\left(1, C \cdot \log^2(N/\delta)/(n\Delta^2)\right)$ *and* $\epsilon_{MV} = \Delta/4N^2$*, where* C *is a fixed positive constant, then with probability* $1-\delta$ *the approximate moments returned satisfy* $|\tilde{\tau}_k - \frac{1}{n}\operatorname{tr}(\bar{T}_k(A))| \leq \Delta$ *for all* $k = 1, \ldots, N$.

Proof. Fix $k \in \{1, \ldots, N\}$. Let $g^{(1)}, \ldots, g^{(\ell)}$ be the random vectors drawn in the outer for-loop of Algorithm 8. Let $\{\tilde{v}_k^{(i)}\}_{i\in[\ell]}$ be the ℓ vectors computed by the inner for-loop and let $\{\delta_k^{(i)} \coloneqq \tilde{v}_k^{(i)} - T_k(A)g^{(i)}\}_{i\in[\ell]}$ be the ℓ error vectors. Recalling that $\frac{1}{n}\operatorname{tr}(\bar{T}_k(A)) = \frac{\sqrt{2/\pi}}{n}\operatorname{tr}(T_k(A))$, we have:

$$|\tilde{\tau}_k - \frac{\sqrt{2/\pi}}{n}\operatorname{tr}(T_k(A))| \leq \frac{\sqrt{2/\pi}}{n\ell}\sum_{i=1}^{\ell}\left|(g^{(i)})^\top\delta_k^{(i)}\right| + |\frac{\sqrt{2/\pi}}{n\ell}\sum_{i=1}^{\ell}(g^{(i)})^\top T_k(A)g^{(i)} - \frac{1}{n}\operatorname{tr}(T_k(A))|$$

Applying Claim 4.4.6 and Lemma 4.4.1, with probability at least $1 - \delta/N$, we thus have

$$|\tilde{\tau}_k - \frac{1}{n}\operatorname{tr}(\bar{T}_k(A))| \leq 2(k+1)^2\,\epsilon_{\mathsf{MV}} \cdot \frac{\sqrt{2/\pi}}{n\ell}\sum_{i=1}^{\ell}\|g^{(i)}\|_2^2 + \Delta/2 \leq \Delta/2 + \Delta/2.$$

The last inequality follows from the fact that $\|g^{(i)}\|_2^2 = n$ for all $i \in [\ell]$, and the choice of $\epsilon_{\mathsf{MV}} = \Delta/4N^2$. Applying a union bound over all $k = 1, \ldots, N$ gives the claim. □

Theorem 4.1.3 immediately follows.

Proof of Theorem 4.1.3. We implement Algorithm 6 with Algorithm 8 used as a subroutine to approximate the Chebyshev polynomial moments, which requires setting $\Delta = \frac{1}{N\ln(eN)}$. By Lemma 4.4.7, we conclude that we need to set $\ell = \max\left(1, CN^2\log^2(N/\delta)\log^2(eN)/n\right)$ and $\epsilon_{\mathsf{MV}} = 1/(4N^3\ln(eN))$. Then, by Lemma 4.3.4, setting $N = O(1/\epsilon)$ ensures that Algorithm 6 returns a distribution q which is ϵ close to A's spectral density s in Wasserstein distance. □

Improving the number of matrix-vector multiplications. We currently require the error bound in Algorithm 6 for estimating the Chebyshev moments to be the same for each of the N moments, i.e., parameter $\Delta = (N\ln(eN))^{-1}$. We note that the number of matrix-vector multiplications can be improved slightly in Theorems 4.1.3 and 4.1.4, potentially by a factor of $\log^2(1/\epsilon)$ for small n. This can be achieved by requiring a different error bound for estimating each moment. Specifically, we modify the requirement in Algorithm 6 for the estimate $\tilde{\tau}_k$ of the k-th normalized Chebyshev moment $\frac{1}{n}\operatorname{tr}(\bar{T}_k(A))$ to have error $|\tilde{\tau}_k - \frac{1}{n}\operatorname{tr}(\bar{T}_k(A))| \leq (k/N^5)^{1/4}$. Plugging this into Lemma 4.4.2, we require at most $\sum_{k=1}^{N}\max(1, CN^{2.5}\log^2(N/\delta)/(n\sqrt{k}))$ matrix-vector multiplications to estimate the N moments, where C is a fixed constant. For comparison to the bounds in Theorems 4.1.4 and 4.1.3, the above bound decreases linearly in n until $n \geq CN^2\log^2(N/\delta)$ and for very large n is bounded by $O(1/N)$. In the regime where n is small, e.g., when $n \leq CN^2\log^2(N/\delta)$, the

bounds from the theorems give $O(N^3 \log^2(N/\delta) \log^2(eN)/n)$ matrix-vector multiplications, whereas the above bound simplifies to at most $O(N^3 \log^2(N/\delta)/n)$ multiplications, saving a $O(\log^2(N)) = O(\log^2(1/\epsilon))$ factor. Lemma 4.4.7 can be adapted identically to give the same bound in the approximate matrix-vector multiplication case. To give intuition for the Wasserstein error of the resulting density, if the density estimate q output by Algorithm 6 satisfied the requirement that $|\langle q, \bar{T}_k \rangle - \frac{1}{n} \operatorname{tr}(\bar{T}_k(A))| \leq (k/N^5)^{1/4}$ for $k \in 1, \ldots, N$, then we have by Lemma 4.3.1 that $W_1(s, q) \leq 36/N + (2/N^{5/4}) \cdot \sum_{k=1}^{N} k^{-3/4} \leq 36/N + 8/N = O(1/N)$. This intuition can be used to adapt the proof of Lemma 4.3.4 to show that Algorithm 6 with moment guarantees as mentioned output a density q such that $W_1(s, q) \leq O(1/N)$.

4.5 Sublinear Time Methods for Graphs

With the proof of Theorem 4.1.3 in place, we are now ready to state our sublinear time result for adjacency matrices of graphs. The significance of Theorem 4.1.3 is that it allows for the approximate Chebyshev moment matching method in Algorithm 6 to be combined with any randomized algorithm for approximating matrix-vector multiplications with A. In this section we prove Theorem 4.1.1 by showing that for the normalized adjacency matrix of any undirected, un-weighted graph, such an algorithm can actually be implemented in *sublinear time*, leading to a sublinear time spectral density estimation (SDE) algorithm for computing graph spectra from these matrices.

Computational Model. Let $A \in \mathbb{R}^{n \times n}$ be the adjacency matrix for an unweighted, n-vertex graph $G = (V, E)$ and let $\bar{A} = D^{-1/2} A D^{-1/2}$ be the symmetric normalized adjacency matrix, where D is an $n \times n$ diagonal matrix containing the degree of each vertex in V. For a node i, let $\mathcal{N}(i) = \{j : (j, i) \in E\}$ denote the set of i's neighboring vertices. We assume a computational model where we can 1) uniformly sample a random vertex in constant

time, 2) uniformly sample a random neighbor of any vertex i in constant time, and 3) for a vertex i with degree d_i, read off all neighbors of i in $O(d_i)$ time. A standard adjacency list representation of the graph would allow us to perform these operations but weaker access models would also suffice.[8]

Using this model for accessing the adjacency matrix, we show that, for any $\epsilon_{\mathsf{MV}} \in (0, 1)$ and failure probability $\delta \in (0, 1)$, an ϵ_{MV}-approximate matrix-vector multiplication oracle for $\bar{A}$ can be implemented in $O(n\,\epsilon_{\mathsf{MV}}{}^{-2}\log(1/\delta))$ time. Via Theorem 4.1.3, this immediately yields an algorithm for computing an SDE that is ϵ close in Wasserstein-1 distance to $\bar{A}$'s spectral density in roughly $\tilde{O}(n/\epsilon^7)$ time for sufficiently large n, and at most $\tilde{O}(n/\epsilon^9)$ time, for fixed δ where the $\tilde{O}(\cdot)$ hides factors of $poly(\log(1/\epsilon))$. Our main result is stated as Theorem 4.1.1 in Section 4.1.1.

The same algorithm can be used to approximate the spectral density of the normalized Laplacian of G by a simple shift and scaling. Specifically, $\bar{A}$ can be obtained from the normalized Laplacian $\bar{L}$ via $\bar{A} = I - \bar{L}$, and the spectral density of $\bar{L}$, $s_{\bar{L}}(x)$ satisfies $s_{\bar{L}}(1 - x) = s_{\bar{A}}(x)$, where $s_{\bar{A}}$ is the spectral density of $\bar{A}$. So if we obtain an ϵ-approximate SDE q for $\bar{A}$ by Theorem 4.1.1, then the function p satisfying $p(1 - x) = q(x)$ is an ϵ-approximate SDE for $s_{\bar{L}}$. We thus have:

Corollary 4.5.1. *Given the the normalized adjacency matrix of G, there exists an algorithm that takes $O\left(npoly\left(\frac{\log(1/\delta)}{\epsilon}\right)\right)$ expected time and outputs a density function q that is ϵ close to the spectral density of the normalized Laplacian of G with probability at least $1 - \delta$.*

[8] E.g., random crawl access to a network [89]. We also note that, if desired, assumption 3) can be removed entirely with a small logarithmic runtime overhead, as long as we know the degree of i. Specifically, 3) can be implemented with $O(d_i \log n)$ calls to 2): we simply randomly sample neighbors until all d_i are found. A standard analysis of the coupon collector problem [Section 3.6, 113] shows that that the expected number of samples will be $O(d_i \log d_i) \leq O(d_i \log n)$.

4.5.1 Approximate Matrix-Vector Multiplication for Adjacency Matrices

We implement an approximate matrix-vector multiplication oracle for $\bar{A}$ in Algorithm 9, which is inspired by a randomized matrix-multiplication method of [53]. Throughout this section, let $\bar{A}^i$ denote the i^{th} column of $\bar{A}$. Given a sampling budget $t \in \mathbb{N}$, the algorithm samples t indices from $1, \ldots, n$ independently and with replacement – i.e., the same index might be sample multiple times. For each index it samples, the algorithm decides to accept or reject the column corresponding to that index with some probability. To approximate $\bar{A}y$, the algorithm outputs the multiplication of the accepted columns, rescaled appropriately, with the corresponding elements of y.

Algorithm 9 AMV Multiplication Oracle for Normalized Adjacency Matrices

Input: Normalized adjacency matrix $\bar{A} \in \mathbb{R}^{n \times n}$, degrees $[d_1, \ldots, d_n]$, $y \in \mathbb{R}^n$, and parameter $t \in \mathbb{N}$.

Output: A vector $z \in \mathbb{R}^n$ that approximates $\bar{A}y$.

Initialize $z \leftarrow \vec{0}$.
for t iterations **do**
    Sample a node j uniformly at random from $\{1, \ldots, n\}$.
    Sample a neighbor $i \in \mathcal{N}(j)$ uniformly at random.
    Sample x uniformly at random from $[0, 1]$.
    if $x \leq \frac{1}{d_i}$ **then**
        $w \leftarrow \frac{1}{p_i} \cdot y_i \bar{A}^i$ where $p_i = \frac{1}{nd_i} \sum_{j \in \mathcal{N}(i)} \frac{1}{d_j}$.
    else
        $w \leftarrow \vec{0}$.
    $z \leftarrow z + w$.
return $\frac{1}{t} z$

The following lemma bounds the expected squared error of Algorithm 9's:

Lemma 4.5.2. *Let $z \in \mathbb{R}^n$ be the output of Algorithm 9 with sampling budget t. We have:*

$$\mathbf{E}[\|\bar{A}y - z\|_2^2] = \frac{n}{t}\|y\|_2^2 - \frac{1}{t}\|\bar{A}y\|_2^2$$

Proof. Let b denote $b = \bar{A}y$. Consider a single iteration of the main loop in Algorithm 9, which generates a vector w that is added to z. Let X_i be an indicator random variable that is 1 if w is set to a scaling of $\bar{A}^i$ on that iteration, and 0 otherwise. $X_i = 1$ if and only if 1) a neighbor of i is sampled at Line 3 of the algorithm, 2) i is sampled at Line 4 of the algorithm, and 3) the uniform random variable x satisfies $x < 1/d_i$. So, we see that $\mathbf{Pr}[X_i = 1]$ is exactly equal to $p_i = \frac{1}{nd_i}\sum_{j\in\mathcal{N}(i)}\frac{1}{d_j}$. It follows that, by the time we reach Line 11, w is an unbiased estimator for b. I.e., $\mathbf{E}[w] = b$. Of course, this also implies that $\mathbf{E}[z] = b$.

Our goal is to show that $\mathbf{E}[\|b - z\|^2] = \frac{n}{t}\|y\|_2^2 - \frac{1}{t}\|b\|_2^2$. Since the random vector $b - z$ has mean zero and is the average of t i.i.d. copies of the mean zero random vector $b - w$, it suffices that show:

$$\mathbf{E}[\|b - w\|_2^2] = n\|y\|_2^2 - \|b\|_2^2. \tag{4.10}$$

By linearity of expectation and the fact that $\mathbf{E}[w] = b$, we have

$$\mathbf{E}[\|b - w\|_2^2] = \|b\|_2^2 + \mathbf{E}[\|w\|_2^2] - 2\langle\mathbf{E}[w], b\rangle = \mathbf{E}[\|w\|_2^2] - \|b\|_2^2.$$

So to prove (4.10), we need to show that $\mathbf{E}[\|w\|_2^2] = n\|y\|_2^2$. We expand w in terms of the indicator random variables $X_1, \ldots, X_n$. Notice that since we only sample one column in each iteration, the random variable $X_iX_j = 0$ for all $i \neq j$. Thus, we have:

$$\begin{aligned}\mathbf{E}[\|w\|_2^2] &= \sum_{k=1}^{n}\mathbf{E}\left[\sum_{i,j\in[n]}\frac{X_iX_j}{p_ip_j}(\bar{A}^iy_i)_k(\bar{A}^jy_j)_k\right] = \sum_{k=1}^{n}\mathbf{E}\left[\sum_{i=1}^{n}\frac{X_i^2}{p_i^2}\cdot(\bar{A}^iy_i)_k^2\right] \\ &= \sum_{i=1}^{n}\frac{1}{p_i}\cdot\|\bar{A}^iy_i\|_2^2 = \sum_{i=1}^{n}ny_i^2 = n\|y\|_2^2\end{aligned}$$

In the last equalities we used the fact that $\mathbf{E}[X_i^2] = p_i$ and that, for a normalized graph adjacency matrix, $\|\bar{A}^i\|_2^2 = \sum_{j \in \mathcal{N}(i)} \frac{1}{d_i d_j} = np_i$. This proves (4.10), from which we conclude the lemma. □

Using Lemma 4.5.2, we show that there is an ϵ_{MV}-approximate matrix-vector oracle for $\bar{A}$ based on Algorithm 9 with success probability at least $1-\delta$ that runs in $O(n\,{\epsilon_{\mathsf{MV}}}^{-2}\log^2(\frac{1}{\delta}))$ time.

Proposition 4.5.3. *Let $\bar{A} \in \mathbb{R}^{n\times n}$ be the symmetric normalized adjacency matrix of an n-vertex graph G and let $\epsilon_{MV}, \delta \in (0,1)$ be fixed constants. There is an algorithm that, given a vector $y \in \mathbb{R}^n$, and access to G as described above, takes $O(n\,{\epsilon_{MV}}^{-2}\log(\frac{1}{\delta}))$ expected time and outputs a vector $z \in \mathbb{R}^n$ such that $\|z - \bar{A}y\|_2 \le \epsilon_{MV}\|y\|_2$ with probability at least $1-\delta$.*

Proof. By Lemma 4.5.2, we have that $\mathbf{E}[\|\bar{A}y - z\|_2^2] \le \frac{n}{t}\|y\|_2^2$. Fix $t = 48n\,{\epsilon_{\mathsf{MV}}}^{-2}$. Then, by Lemma 4.5.2 and Markov's inequality, we have that when Algorithm 9 is called on $\bar{A}$ with parameter t,

$$\mathbf{Pr}[\|\bar{A}y - z\|_2 > \frac{\epsilon_{\mathsf{MV}}}{4}\|y\|_2] \le \frac{16n\|y\|_2^2}{t\,{\epsilon_{\mathsf{MV}}}^2\,\|y\|_2^2} \le \frac{1}{4}. \tag{4.11}$$

In order improve our success probability from $3/4$ to $1-\delta$, we use the standard trick of repeating the above process $r = c\log(\frac{1}{\delta})$ times for a constant c to be fixed later. Let $z_1, \ldots, z_r \in \mathbb{R}^n$ be the output of running Algorithm 9 r times with parameter t. We can return as our estimate for $\bar{A}y$ the first z_i such that $\|z_i - z_j\|_2 \le \frac{\epsilon_{\mathsf{MV}}}{2}\|y\|_2$ for at least $r/2+1$ vectors z_j from $z_1, \ldots, z_n$.

To see why this works, note that a Chernoff bound can be used to claim that with probability $> 1-\delta$, at least $r/2+1$ vectors z_j from $z_1, \ldots, z_r$ have that $\|z_j - \bar{A}y\|_2 \le \frac{\epsilon_{\mathsf{MV}}}{4}\|y\|_2$.

By a triangle inequality we have that for all such z_j and z_k,

$$\|z_j - z_k\|_2 \leq \|z_j - \bar{A}y\|_2 + \|z_k - \bar{A}y\|_2 \leq \frac{\epsilon_{\mathsf{MV}}}{2}\|y\|_2.$$

Thus, the z_i we picked must satisfy that $\|z_i - \bar{A}y\| \leq \frac{3\,\epsilon_{\mathsf{MV}}}{4}\|y\|_2$ by the triangle inequality.

All that remains is to bound the expected runtime of Algorithm 9, which we will run r separate times. To do so, note that all index sampling can be done in just $O(t)$ time, since sampling a random vertex and a random neighbor of the vertex are assumed to be $O(1)$ time operations. The costly part of the algorithm is computing the sampled column w at each iteration. In the case that $w = \vec{0}$, this cost is of course zero. However, when $w = \frac{1}{p_i}\bar{A}^i y_i$ for some i, computing the column and adding it to z takes $O(d_i)$ time, which can be large in the worst case. Nevertheless, we show that it is small in expectation. This may seem a bit surprising: while nodes with high degree are more likely to be sampled by Line 4 in Algorithm 9, they are rejected with higher probability in Line 6. Formally, let $\mathrm{nnz}(w)$ denote the number of non-zero entries in w. We have:

$$\mathbf{E}\left[\mathrm{nnz}(w)\right] = \sum_{i=1}^{n} \mathrm{nnz}(\bar{A}^i) \cdot p_i = \sum_{i=1}^{n} \sum_{j \in \mathcal{N}(i)} \frac{d_i}{n \cdot d_i d_j} = \frac{1}{n} \sum_{i=1}^{n} \sum_{j \in \mathcal{N}(i)} \frac{1}{d_j} = 1.$$

The final equality follows from expanding the double sum: since node j has exactly d_j neighbors, $\frac{1}{d_j}$ appears exactly d_j times in the sum. So $\sum_{i=1}^{n} \sum_{j \in \mathcal{N}(i)} \frac{1}{d_j} = n$.

We run Algorithm 9 with $t = O(n/\,{\epsilon_{\mathsf{MV}}}^2)$ iterations, so it follows that the expected total sparsity of all w's constructed equals $O(n/\,{\epsilon_{\mathsf{MV}}}^2)$, which dominates the expected running time of our method.

□

Proof of Theorem 4.1.1. The accuracy and running time claim follows from combining the

ϵ_{MV}-approximate vector multiplication oracle described in Proposition 4.5.3 with Algorithm 6, which is analyzed in Theorem 4.1.3. □

As discussed in the introduction, Cohen et al. [41] prove a result which matches the guarantee of Theorem 4.1.1, but with runtime of $2^{O(1/\epsilon)}$ – i.e., with *no dependence* on n. In comparison, our result depends linearly on n, but only polynomially on $1/\epsilon$. In either case, the result is quite surprising, as the runtime is *sublinear* in the input size: A could have up to $O(n^2)$ non-zero entries.

4.6 Experiments

We support our theoretical results by implementing our Chebyshev moment matching method (Algorithm 6). When using exact matrix-vector multiplications, the kernel polynomial method (KPM) of Algorithm 11 and the stochastic Lanczos quadrature method (SLQ) studied in [31] have both been confirmed to work well empirically. So, one set of experiments is aimed at comparing these methods to the moment matching method (MM) implemented with exact matrix-vector multiplications. A second set of experiments evaluates the performance of the MM and KPM methods when implemented with approximate matrix-vector multiplies. Specifically, we use our sublinear time randomized method for multiplication by graph adjacency matrices from Section 4.5.

We consider the normalized adjacency matrix of three graphs, two of which we construct and one which we obtain from a publicly available dataset for sparse matrices:

- `cliquePlusRandBipartite` is a graph with 10000 vertices, partitioned into two disconnected components. The first component is a clique with 5000 nodes and the second is a bipartite graph with 2500 vertices in each partition, constructed by sampling each

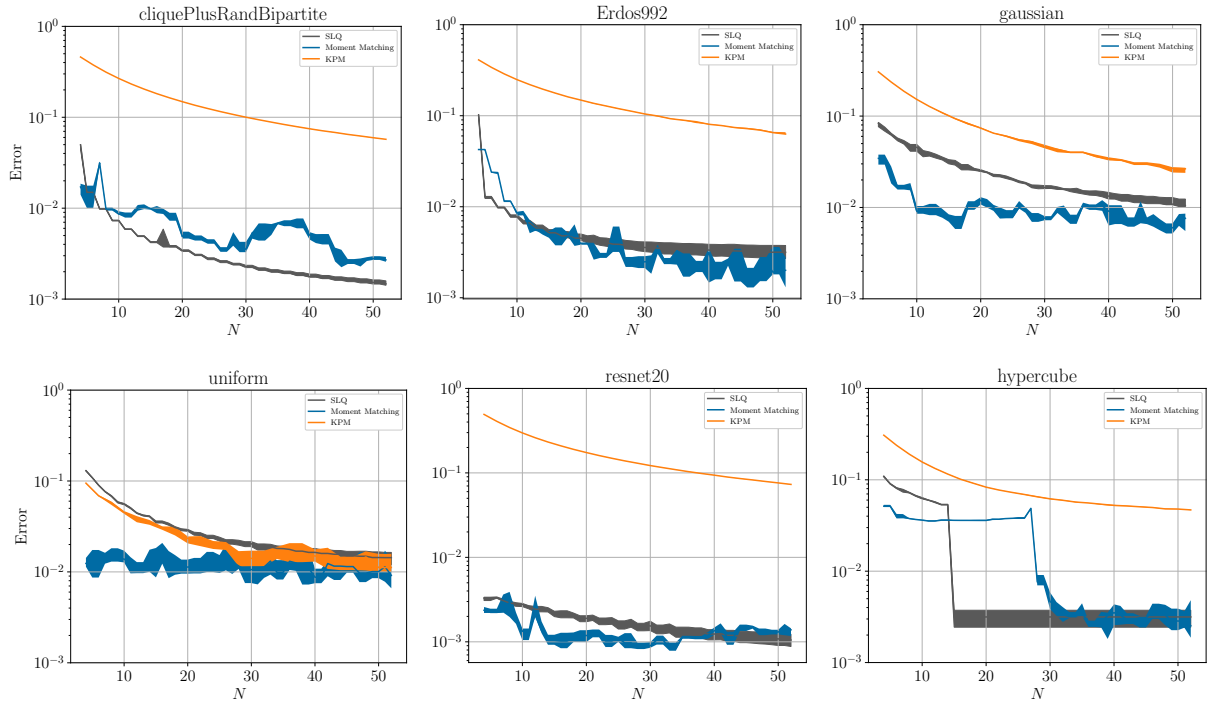

Figure 4.2: Wasserstein error of density estimate resulting from approximate Chebyshev moment matching method (MM), the Jackson damped kernel polynomial method (KPM) and Stochastic Lanczos Quadrature (SLQ) method. For MM and KPM, Hutchinson's estimator is used to estimate the Chebyshev moments. The x-axis corresponds to the number of moments computed for MM and KPM, and the number of Lanczos iterations used for SLQ. All methods use 5 (random) starting vectors except for resnet20 and hypercube that use 1 starting vector, so the x-axis is directly proportional to the number of matrix-vector multiplications used by each method. Each experiment is repeated 10 times; the solid line represents the median error of the 10 trials and the shaded regions represent the first and third quartiles.

of the 2500^2 possible edges independently with probability 0.05. This graph has a normalized adjacency matrix with ~ 5000 eigenvalues at 0, two eigenvalues at 1, one at -1 and the rest of its eigenvalues are roughly evenly spread out between -0.5 and 0.5.

- `hypercube` is a 16384 vertex boolean hypercube graph on 14 bit strings.[9] Its normalized adjacency matrix has eigenvalues at $-1, -\frac{6}{7}, \frac{-5}{7}, \ldots, 0, \ldots, \frac{6}{7}, 1$. The multiplicity of the

[9] A boolean hypercube contains a vertex for each distinct b bit string, and an edge between two vertices if the corresponding strings differ on exactly 1 bit.

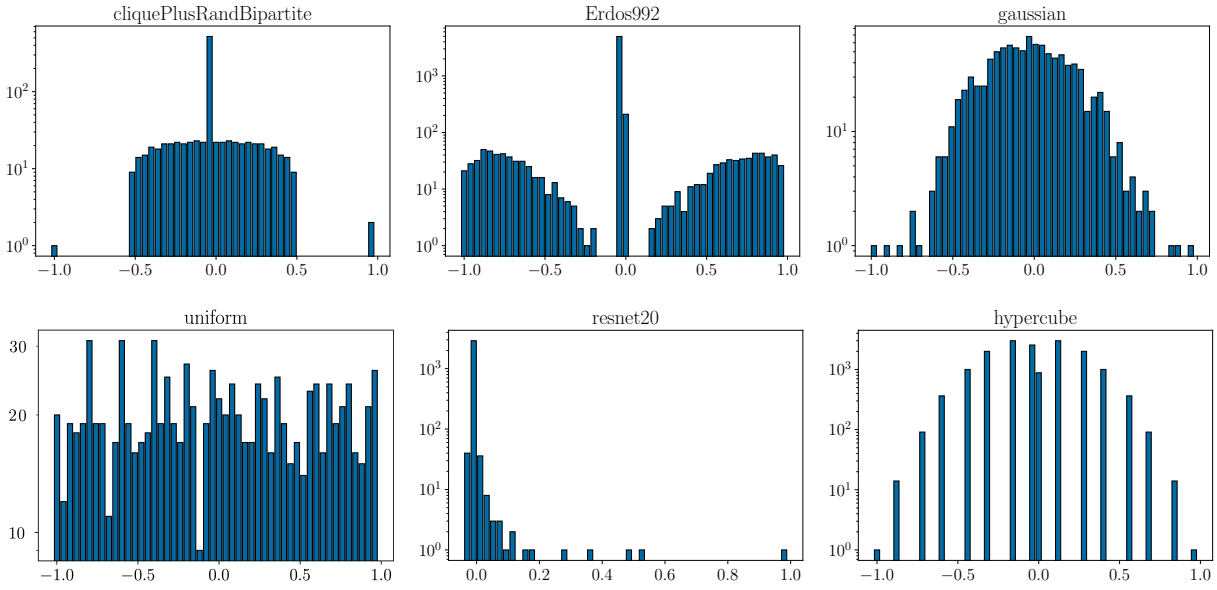

Figure 4.3: Histograms of the eigenvalues of `cliquePlusRandBipartite`, `Erdos992`, `gaussian`, `uniform`, `resnet20` and `hypercube` using 50 equally spaced buckets.

0 eigenvalue is largest, with eigenvalues closer to -1 and 1 having lower multiplicity.

- `Erdos992` is an undirected graph with 6100 vertices, containing 15030 edges from the sparse matrix suite of [45]. Its normalized adjacency matrix has ~ 5000 eigenvalues at 0, one at 1 and the rest evenly spread out between -0.5 and 0.5.

We consider three additional matrices to evaluate the performance of MM against KPM and SLQ when exact matrix-vector multiplies are used to estimate the Chebyshev moments:

- `gaussian` is a 1000×1000 matrix constructed by drawing $n = 1000$ Gaussian random variables $\lambda_1, \cdots, \lambda_n \sim \mathcal{N}(0, 1)$ and a random orthogonal matrix $U \in \mathbb{R}^{n \times n}$, and outputting $U \Lambda U^\top$ where Λ is a $n \times n$ diagonal matrix with entries $\frac{\lambda_1}{\max_i \lambda_i}, \ldots, \frac{\lambda_n}{\max_i \lambda_i}$.

- `uniform` is a 1000×1000 matrix constructed identically to `gaussian` except with $\lambda_1, \ldots, \lambda_n$ drawn independently and uniformly from the interval $[-1, 1]$.

- `resnet20` is a Hessian for the ResNet20 network [76] trained on the Cifar-10 dataset. The matrix is 3000×3000 and its eigenvalues have been normalized to lie between $[-1, 1]$ for our experiments.

For reference, the histogram of the eigenvalues for each matrix are shown in Figure 4.3 by breaking the range of the eigenvalues into 50 equally spaced intervals for each matrix.

In the first set of experiments, we compute the normalized Chebyshev moments $\tau_1, \ldots, \tau_N$ of each of the six aforementioned matrices using Hutchinson's moment estimator as in Algorithm 7, and, compute a spectral density estimate by passing these moments into Algorithm 6 for approximate Chebyshev moment matching method (MM)[10] and into Algorithm 11 for the Jackson damped kernel polynomial method (KPM). For KPM we compute the density with $N = 4, 6, 8, 10, \ldots, 52$ and for MM we compute it with $N = 4, 5, 6, 7, \ldots, 52$. We also compute the density estimate resulting from the stochastic Lanczos quadrature (SLQ) method of [31] with $N = 4, 5, 6, 7, \ldots, 52$ Lanczos iterations. We use $\ell = 5$ starting vectors (i.e., random vectors in Hutchinson's method, or random restarts of the SLQ method) for each method, except for the large `resnet20` and `hypercube` matrices, for which $\ell = 1$ random vector is used. Each experiment is repeated 10 times and the Wasserstein-error between the true density and the density estimate are shown in Figure 4.2. The results show that MM is more than 10x more accurate than KPM in almost all cases. The error of MM and SLQ are more comparable, except for `hypercube`, on which the errors are comparable for larger values of N. Both methods show an unusual convergence curve for this matrix, which we believe is related to the sparsify of its spectrum (a small number of distinct eigenvalues).

In our second set of experiments, we test the performance of our randomized sublinear time algorithm (Algorithm 9) for approximate matrix-vector multiplies with normalized

[10] We solve the optimization problem from Line 3 by formulating it as a linear program and using an off-the-shelf solver from `scipy`.

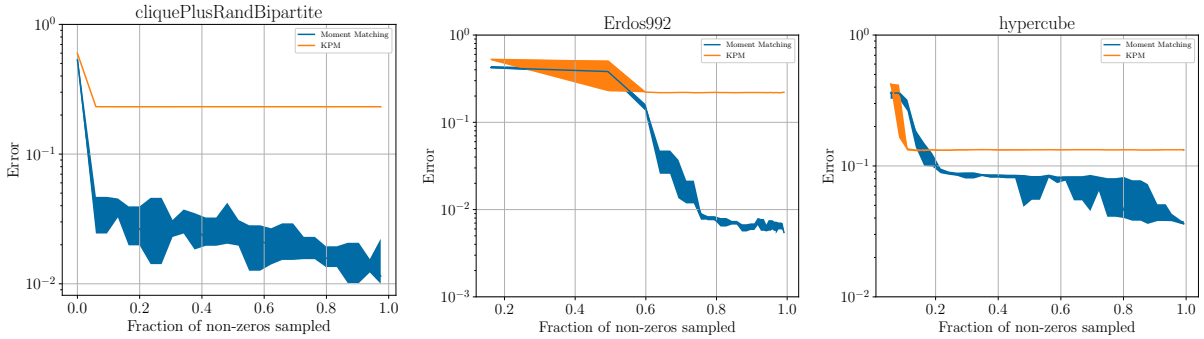

Figure 4.4: Wasserstein error of density estimate returned by MM and KPM on the `hypercube`, `cliquePlusRandBipartite` and `Erdos992` graphs using approximate matrix-vector multiplications (Algorithm 9) to estimate the Chebyshev moments. For both methods, $N = 32$ moments are computed using 5 random starting vectors for `cliquePlusRandBipartite` and `Erdos992` and 1 for `hypercube`. The x-axis corresponds to the average fraction of non-zeros sampled from the matrix and the y-axis is the Wasserstein error from the resulting density estimate. Each experiment is repeated for 10 trials: the solid line correponds to the median error of the 10 trials and the shaded region corresponds to the first and third quartiles.

graph adjacency matrices. This method is used to estimate Chebyshev moments in Algorithm 6 (MM) and in Algorithm 11 (KPM). We compute the normalized Chebyshev moments $\tau_1, \ldots, \tau_N$ for $N = 12$ using various values of the oversampling parameter t in the approximate matrix-vector multiplication method. We then compute, for each value of t, the average number of non-zero elements of A accessed by the method for each matrix-vector product, which reflects the runtime improvement over a full matrix-vector product. Figure 4.4 plots the Wasserstein error of the density estimate (y-axis) and the average fraction of non-zeros used in each matrix-vector multiplication (x-axis) to estimate the Chebyshev moments used by MM and KPM respectively.

The results show that the KPM method can achieve error nearly identical to that obtained when using exact matrix-vector multiplications, while only using a small fraction of non-zero entries for each approximate matrix-vector multiplication. Specifically, on the dense `cliquePlusRandBipartite` graph and even the relatively sparse `hypercube` graph,

KPM uses less than 15% of the non-zero entries on average to achieve nearly the same error as when using exact multiplies. On `cliquePlusRandBipartite`, the MM method achieves error close to that of the exact method while using $\sim 20\%$ of the non-zero entries on average. On the sparse `Erdos992` and `hypercube` graphs, the MM method requires $\sim 80\%$ of the non-zero entries on average to achieve error comparable to exact matrix-vector multiplications. However, it still obtains a good approximation (consistently better than the KPM method) when coarse matrix-vector multiplications are used (i.e., fewer non-zeros are sampled).

4.7 The Kernel Polynomial Method

In this section we show how to obtain a spectral density estimate based on a version of the kernel polynomial method that also approximates Chebyshev polynomial moments: $\operatorname{tr}(T_0(A)), \ldots, \operatorname{tr}(T_N(A))$. We again rely on Jackson's classic work on universal polynomial approximation bounds for Lipschitz functions: we take advantage of the fact that Jacksons construction of such polynomials is both linear and preserves positivity [83].

4.7.1 Idealized Kernel Polynomial Method

As an alternative to the moment matching method presented in Section 4.3, a natural approach to using computed Chebyshev moments is to construct a truncated Chebyshev series approximation to s (see Definition 4.2.1). To do so, note that the scaled moments $\frac{1}{n}\operatorname{tr}(\bar{T}_0(A)), \ldots, \frac{1}{n}\operatorname{tr}(\bar{T}_N(A))$ are exactly equal to the first N Chebyshev series coefficients of s/w, where $w(x) = \frac{1}{\sqrt{1-x^2}}$ is as defined in Section 4.2. Specifically, the eigenvalues of $\bar{T}_k(A)$ are equal to $\bar{T}_k(\lambda_1), \ldots, \bar{T}_k(\lambda_n)$, where $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of A. Since the trace of a diagonalizable matrix is the sum of its eigenvalues, we have $\frac{1}{n}\operatorname{tr}(\bar{T}_k(A)) = \frac{1}{n}\sum_{i=1}^{n}\bar{T}_k(\lambda_i) =$

$\langle s, \bar{T}_k \rangle = \langle s/w, w \cdot \bar{T}_k \rangle$.

After using the scaled Chebyshev moments to construct a truncated Chebyshev series for s/w, i.e. a degree N polynomial approximation, we can then multiply the final result by w to obtain an approximation to s. Unfortunately, there are two issues with this approach: 1) it is difficult to analyze the quality of the Chebyshev series approximation, since s is not a smooth function, and 2) this approximation will not in general be a non-negative function, which is a challenge because our goal is to find *probability density* that well approximates s.

A common approach for dealing with the second issue is to instead use a *damped Chebyshev expansion* [157], where the Chebyshev coefficients are slightly reweighted to ensure that the resulting polynomial is always non-negative. Such non-negativity preserving damping schemes follow from the connection between Chebyshev and Fourier series: we refer the reader to Appendix B.1 for details. In short, by the convolution theorem, Fourier series truncation corresponds to convolution with a function whose Fourier support is bounded. If this function is also non-negative, convolution preserves non-negativity of the function being approximated, leading to truncated series that is guaranteed to be positive. One such damping schemes was introduced in classic work of Jackson [83]. For any positive integer z, let $N = 4z$. Then, for $k = 0, \ldots, N$, define the coefficient

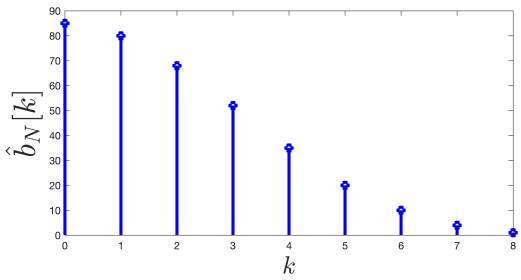

Figure 4.5: Jackson coefficients for $N = 8$.

$$\hat{b}_N[k] = \sum_{j=-\frac{N}{2}-1}^{\frac{N}{2}+1-k} \left(\frac{N}{2} + 1 - |j|\right) \cdot \left(\frac{N}{2} + 1 - |j+k|\right). \tag{4.12}$$

While (4.12) may look opaque, $\hat{b}_N[0], \ldots, \hat{b}_N[N]$ are actually equal to the result of a simple discrete convolution operation. Let $g \in \mathcal{F}(\mathbb{Z}, \mathbb{R})$ have $g[j] = 1$ for $j = -z, \ldots, z$, and $g[j] = 0$

otherwise. Then let $\hat{b}_N = (g * g) * (g * g)$ and $\hat{b}_N[0], \ldots, \hat{b}_N[N]$ be the values corresponding to non-negative indices.[11] See Fig. 4.5 for an illustration of these coefficients. They are all positive and $\hat{b}_N[0] > \hat{b}_N[1] > \ldots > \hat{b}_N[N]$. Jackson suggests approximating a function using the following truncation based on these coefficients:

Definition 4.7.1 (Jackson damped Chebyshev series)**.** Let $f \in \mathcal{F}([-1, 1], \mathbb{R})$ have Chebyshev series $\sum_{k=0}^{\infty} \langle f, w \cdot \bar{T}_k \rangle \cdot \bar{T}_k$. The Jackson approximation to f is a degree N polynomial $\bar{f}_N$ obtained via the following truncation with modified coefficients:

$$\bar{f}_N(x) \coloneqq \sum_{k=0}^{N} \frac{\hat{b}_N[k]}{\hat{b}_N[0]} \langle f, w \cdot \bar{T}_k \rangle \bar{T}_k(x). \tag{4.13}$$

Note that $\hat{b}_N[0]/\hat{b}_N[0] = 1$, and all other terms are strictly less than one. It is not hard to show this damped series preserves positivity. We prove the following fact as Lemma B.1.7 in the appendix:

Fact 4.7.2. *If $f : [-1, 1] \to \mathbb{R}^{\geq 0}$ is a non-negative function, then the polynomial $\bar{f}_N(x)$ defined in (4.13) is non-negative for all $x \in [-1, 1]$.*

Beyond preserving non-negativity, as claimed in Fact 4.3.2, the Jackson damped Chebyshev approximation is more well-known for the fact that it provably provides a good *uniform* polynomial approximation to any Lipschitz function. For completeness, we give a proof of this fact as Theorem B.1.6 in the appendix. With Facts 4.7.2 and 4.3.2 in place, we are ready to introduced the basic kernel polynomial method for approximating the spectral density s as Algorithm 10. This algorithm is identical to the "Jackson Kernel" KPM from [157]. Recall that, for now, we assume we have access to *exact* Chebyshev moment of the spectral

[11] This formulation allows the coefficients to be easily computed in most high-level programming languages. E.g., in `MATLAB` we can compute `g = ones(2*z+1,1); c = conv(conv(g,g),conv(g,g)); b = c(N+1:2*N+1);`.

density s for our matrix A. In Section 4.7.2 we prove that Algorithm 10 is robust to using approximate moments.

Algorithm 10 Idealized Jackson Damped Kernel Polynomial Method

Input: Symmetric $A \in \mathbb{R}^{n \times n}$ with spectral density $s : [-1, 1] \to \mathbb{R}^{\geq 0}$, degree $N \in 4\mathbb{N}^+$.
Output: Density function $q : [-1, 1] \to \mathbb{R}^{\geq 0}$.

1: For $k = 0, \ldots, N$ compute $\tau_k = \frac{1}{n} \operatorname{tr}(\bar{T}_k(A)) = \langle s, \bar{T}_k \rangle$.
2: For $k = 0, \ldots, N$ compute $\hat{b}_N[k]$ as is (4.12).
3: Return $q = w \cdot \sum_{k=0}^{N} \frac{\hat{b}_N[k]}{\hat{b}_N[0]} \cdot \tau_k \cdot \bar{T}_k$.

Lemma 4.7.3. *If $N \geq \frac{18}{\epsilon}$, then the function $q : [-1, 1] \to \mathbb{R}^{\geq 0}$ returned by Algorithm 10 is a probability density and satisfies:*

$$W_1(s, q) \leq \epsilon.$$

Proof. We first prove that q is a probability density. To see that it is positive, note that $h = \sum_{k=0}^{N} \frac{\hat{b}_N[k]}{\hat{b}_N[0]} \cdot \tau_k \cdot \bar{T}_k$ is a Jackson approximation to the positive function s/w, so is must be non-negative by Fact 4.7.2. Since w is also non-negative, we conclude that $q = w \cdot h$ is as well. Then we consider q's integral. We need to show that $\int_{-1}^{1} q(x)dx = 1 = \int_{-1}^{1} s(x)dx$. Since $\bar{T}_0$ is a scaling of the constant function, it suffices to show that $\langle \bar{T}_0, q \rangle = \langle \bar{T}_0, s \rangle$. We have:

$$\langle \bar{T}_0, q \rangle = \tau_0 \cdot \langle \bar{T}_0, w \cdot \bar{T}_0 \rangle = \langle \bar{T}_0, s \rangle \cdot 1.$$

The first step follows directly from the orthogonality of the Chebyshev polynomials under the weight function w, which implies that $\langle \bar{T}_0, w \cdot \bar{T}_k \rangle = 0$ for all $k > 0$. We also use that $\langle \bar{T}_0, w \cdot \bar{T}_0 \rangle = 1$.

Next, we prove the approximation guarantee. Referring to the formulation of Wasserstein-1 distance from equation (4.3), we have that $W_1(s, q) = \sup \langle s - q, f \rangle$ where f is a 1-Lipschitz function. So, we want to show that any 1-Lipschitz f has small inner product with the dif-

ference between s and its degree-N Jackson approximation, q. To do so, we show that this inner product is actually *exactly equal to* the inner product between s and a degree-N Jackson approximation to f. Since f is 1-Lipschitz, this approximation is guaranteed to be have small error. This key equivalency follows because, like a standard Chebyshev series approximation, the Jackson approximation can be viewed as the output of a symmetric linear operator applied to s.

Formally, we introduce notation for several linear operators needed to analyze (4.13). Let $\bar{\mathcal{T}} : \mathcal{F}([-1,1],\mathbb{R}) \to \mathcal{F}(\mathbb{N},\mathbb{R})$ be the operator mapping a function $f \in \mathcal{F}([-1,1],\mathbb{R})$ to its inner-product with the normalized Chebyshev polynomials. Define the transpose operator $\bar{\mathcal{T}}^* : \mathcal{F}(\mathbb{N},\mathbb{R}) \to \mathcal{F}([-1,1],\mathbb{R})$ to satisfy $\langle \bar{\mathcal{T}} f, g\rangle = \langle f, \bar{\mathcal{T}}^* g\rangle$ for any $g \in \mathcal{F}(\mathbb{N},\mathbb{R})$. Concretely, for $i \in \mathbb{N}$ and $x \in [-1,1]$,

$$[\bar{\mathcal{T}} f][i] \coloneqq \int_{-1}^{1} \bar{T}_i(x) f(x) dx \qquad \text{and} \qquad [\bar{\mathcal{T}}^* g](x) \coloneqq \sum_{i=0}^{\infty} \bar{T}_i(x) g[i]. \tag{4.14}$$

We also define operators $\mathcal{W} : \mathcal{F}([-1,1],\mathbb{R}) \to \mathcal{F}([-1,1],\mathbb{R})$ and $\mathcal{I} : \mathcal{F}(\mathbb{N},\mathbb{R}) \to \mathcal{F}(\mathbb{N},\mathbb{R})$ as follows:

$$[\mathcal{W} f](x) \coloneqq w(x) f(x) = \frac{1}{\sqrt{1-x^2}} f(x) \qquad \text{and} \qquad [\mathcal{I} g][i] \coloneqq g[i].$$

Note that $\mathcal{I}$ is an identity operator. For any $N \in 4\mathbb{N}$, we define $\mathcal{B}_N : \mathcal{F}(\mathbb{N},\mathbb{R}) \to \mathcal{F}(\mathbb{N},\mathbb{R})$ as:

$$[\mathcal{B}_N g](i) \coloneqq \begin{cases} \frac{\hat{\beta}_N[i]}{\hat{\beta}_N[0]} g(i) & \text{for } 0 \le i \le N \\ 0 & i > N. \end{cases}$$

The operators $\mathcal{W}$, $\mathcal{I}$, and $\mathcal{B}_N$ are all commutative with respect to the inner-products in their respective spaces. Specifically, for $f_1, f_2 \in \mathcal{F}([-1,1],\mathbb{R})$ and $g_1, g_2 \in \mathcal{F}(\mathbb{N},\mathbb{R})$, $\langle f_1, \mathcal{W} f_2\rangle =$

$\langle \mathcal{W} f_1, f_2 \rangle$, $\langle g_1, \mathcal{I} g_2 \rangle = \langle \mathcal{I} g_1, g_2 \rangle$, and $\langle g_1, \mathcal{B}_N g_2 \rangle = \langle \mathcal{B}_N g_1, g_2 \rangle$. Also note that by orthogonality of the Chebyshev polynomials under w, $\bar{\mathcal{T}}^* \bar{\mathcal{T}} \mathcal{W}$ is the identity operator on $\mathcal{F}([-1,1], \mathbb{R})$ and so is $\mathcal{W} \bar{\mathcal{T}}^* \bar{\mathcal{T}}$.

With these operators defined, the remainder of the proof is short. We have via direct calculation:

$$\begin{aligned}\langle f, s - q \rangle &= \langle f, s - \mathcal{W} \bar{\mathcal{T}}^* \mathcal{B}_N \bar{\mathcal{T}} s \rangle \\ &= \langle f, \mathcal{W} \bar{\mathcal{T}}^* (\mathcal{I} - \mathcal{B}_N) \bar{\mathcal{T}} s \rangle = \langle \bar{\mathcal{T}}^* (\mathcal{I} - \mathcal{B}_N) \bar{\mathcal{T}} \mathcal{W} f, s \rangle = \langle f - \bar{\mathcal{T}}^* \mathcal{B}_N \bar{\mathcal{T}} \mathcal{W} f, s \rangle.\end{aligned}$$

Note that $\bar{\mathcal{T}}^* \mathcal{B}_N \bar{\mathcal{T}} \mathcal{W} f$ is exactly the degree-N Jackson approximation to f. So by Fact **??**, if f is a 1-Lipschitz function, $\|f - \bar{\mathcal{T}}^* \mathcal{B}_N \bar{\mathcal{T}} \mathcal{W} f\|_\infty \leq 18/N$. Since s is a non-negative function that integrates to 1, it follows that $\langle f, s - q \rangle = \langle f - \bar{\mathcal{T}}^* \mathcal{B}_N \bar{\mathcal{T}} \mathcal{W} f, s \rangle \leq 18/N$. Since $W(s,q) = \sup_{\text{1-Lipschitz } f} \langle f, s - q \rangle$, we conclude that $W(s,q) \leq \epsilon$ as long as as long as $N \geq 18/\epsilon$. □

Remark. Given access to the Chebyshev polynomial moments, $\mathrm{tr}(\bar{T}_0(A)), \ldots, \mathrm{tr}(\bar{T}_N(A))$, Algorithm 10 can be implemented in $O(1/\epsilon)$ additional time. The function it returns is an $O(1/\epsilon)$ degree polynomial times the closed form function w. The polynomial can be represented as a sum of Chebyshev polynomials, or converted to standard monomial form in $O(1/\epsilon^2)$ time. The function is easily plotted or integrated over a range – see discussion around Fact 4.8.2 for more details.

4.7.2 Full Kernel Polynomial Method

Since it is not possible to efficiently compute the exact Chebyshev polynomial moments, we need to show that the kernel polynomial method can work with approximations to these

moments, computed e.g. using a stochastic trace estimator as described in Section 4.4. Here, we first prove a general result on the accuracy of approximation needed to ensure we obtain a good spectral density estimation. Specifically, we analyze the following "robust" version of Algorithm 10.

Algorithm 11 Jackson Damped Kernel Polynomial Method

Input: Symmetric $A \in \mathbb{R}^{n\times n}$ with spectral density $s : [-1,1] \to \mathbb{R}^{\geq 0}$, degree parameter $N \in 4\mathbb{N}^+$, algorithm $\mathcal{M}(A)$ that computes moment approximations $\tilde{\tau}_1, \ldots, \tilde{\tau}_N$ with the guarantee that $|\tilde{\tau}_k - \frac{1}{n}\operatorname{tr}(\bar{T}_k(A))| \leq 1/N^2$ for all k.

Output: Density function $q : [-1,1] \to \mathbb{R}^{\geq 0}$.

1: For $k = 1, \ldots, N$ use $\mathcal{M}$ to compute $\tilde{\tau}_1, \ldots, \tilde{\tau}_N$ as above. Set $\tilde{\tau}_0 = 1/\sqrt{\pi}$.

2: For $k = 0, \ldots, N$ compute $\hat{b}_N[k]$ as is (4.12).

3: Compute polynomial $\tilde{s}_N = w \cdot \sum_{k=0}^{N} \frac{\hat{b}_N[k]}{\hat{b}_N[0]} \cdot \tilde{\tau}_k \cdot \bar{T}_k$.

4: Return the probability density $q = \left(\tilde{s}_N + \frac{w\sqrt{2}}{N\sqrt{\pi}}\right) / \left(1 + \frac{\sqrt{2\pi}}{N}\right)$.

The final transformation of $\tilde{s}_N$ in Line 4 of Algorithm 11 ensures that we return a proper density, since error incurred by approximating $\frac{1}{n}\operatorname{tr}(\bar{T}_k(A)) = \langle s, \bar{T}_k\rangle$ could leave the function with negative values. So, we shift by a small positive function, and rescale to maintain unit integral. Our main result on the error of Algorithm 11, which parallels Lemma 4.7.3 for Algorithm 10, is as follows:

Lemma 4.7.4. *If $N \geq \frac{18}{\epsilon}$, then the function $q : [-1,1] \to \mathbb{R}^{\geq 0}$ returned by Algorithm 11 is a probability density and satisfies:*

$$W_1(s, q) \leq 2\epsilon.$$

Proof. We first prove that q is a probability distribution. Let s_N denote the ideal distribution returned by Algorithm 10 if exact Chebyshev moments were used. I.e.,

$$s_N = w \cdot \sum_{k=0}^{N} \frac{\hat{b}_N[k]}{\hat{b}_N[0]} \cdot \tau_k \cdot \bar{T}_k$$

where $\tau_k = \frac{1}{n}\operatorname{tr}(\bar{T}_k(A)) = \langle s, \bar{T}_k\rangle$. Note that for any density s, $\tau_0 = \langle s, \bar{T}_0\rangle = 1/\sqrt{\pi}$. Let

$\Delta_k = \tilde{\tau}_k - \tau_k$. We have $\tilde{s}_N(x) = s_N + \sum_{k=1}^N \Delta_k \frac{\hat{b}_N[k]}{\hat{b}_N[0]} w(x)\bar{T}_k(x)$. Define functions $\eta = s_N/w$ and $\tilde{\eta} = \tilde{s}/w$. It follows that for any $x \in [-1, 1]$,

$$|\tilde{\eta}(x) - \eta(x)| = |\sum_{k=1}^N \frac{\hat{b}_N[k]}{\hat{b}_N[0]} \Delta_k \bar{T}_k(x)| \leq \frac{\sqrt{2}}{N\sqrt{\pi}}. \tag{4.15}$$

The last inequality uses that $0 \leq \hat{b}_N[k]/\hat{b}_N[0] \leq 1$ and for $x \in [-1, 1]$, $\bar{T}_k(x) \leq \sqrt{2/\pi}$ for $k \geq 1$. Since η is a non-negative function, from (4.15) we can conclude that the function $\tilde{\eta} + \frac{\sqrt{2}}{N\sqrt{\pi}}$ is non-negitive, and thus $w \cdot (\tilde{\eta} + \frac{\sqrt{2}}{N\sqrt{\pi}}) = \tilde{s}_N + w\frac{\sqrt{2}}{N\sqrt{\pi}}$ is also non-negative. The density of this function is $\int_{-1}^1 \tilde{s}_N(x)dx + \frac{\sqrt{2}}{N\sqrt{\pi}} \int_{-1}^1 w(x)dx = 1 + \frac{\sqrt{2\pi}}{N}$, so dividing by $1 + \frac{\sqrt{2\pi}}{N}$ gives a probability density.

Next we prove the approximation guarantee. By Lemma 4.7.3 we know that $W_1(s, s_N) \leq \epsilon$, so if we can show that $W_1(s_N, q) \leq \epsilon$, then by triangle inequality we will have shown that $W_1(s, q) \leq W_1(s, s_N) + W_1(s_N, q) \leq 2\epsilon$.

To bound $W_1(s_N, q)$, we need to show that $\langle f, s_N - q\rangle \leq \epsilon$ for any 1-Lipschitz function $f \in \mathcal{F}([-1, 1], \mathbb{R})$. Without loss of generality, we can assume that $\int_{-1}^1 f(x)dx = 0$, as the 1-Lipschitz function $f' = f - \int_{-1}^1 f(x)dx$ satisfies $\langle f, s_N - q\rangle = \langle f', s_N - q\rangle$ (since s_N and q are both probability densities). If $\int_{-1}^1 f(x)dx = 0$, $f(x)$ must be zero for some $x \in [-1, 1]$, and since it is also 1-Lipschitz we can in turn bound $\|f\|_\infty \leq 1$.[12] We can then bound the inner product:

$$\langle f, s_N - q\rangle \leq \|f(\bar{s}_N - q)\|_1 \leq \|f\|_\infty \|\bar{s}_N(x) - q(x)\|_1 \leq \left\| w \cdot \left(\eta - \frac{\tilde{\eta} + \frac{\sqrt{2}}{N\sqrt{\pi}}}{1 + \frac{\sqrt{2\pi}}{N}} \right) \right\|_1$$

$$\leq \underbrace{\left\| w \cdot \left(\eta - \tilde{\eta} - \frac{\sqrt{2}}{N\sqrt{\pi}} \right) \right\|_1}_{z_1} + \underbrace{\left\| \frac{\sqrt{2\pi}}{N} \cdot w \cdot \left(\tilde{\eta} + \frac{\sqrt{2}}{N\sqrt{\pi}} \right) \right\|_1}_{z_2}$$

[12] Let z maximize $f(x)$. Since f is 1-Lipschitz we have $f(z) \leq |x - z| - f(x)$ for all x. Integrating both sides from -1 to 1, we have $2f(z) \leq (z^2 + 1) - 0 \leq 2$. So, $f(z) \leq 1$.

The last inequality uses the fact that $1 - \frac{1}{1+\gamma} \leq \gamma$ for $0 \leq \gamma \leq 1$, which we apply with $\gamma = \frac{\sqrt{2\pi}}{N}$. Using the fact that $\|w\|_1 = \int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} dx = \pi$ and the bound on $\|\eta - \tilde{\eta}\|_\infty$ from (4.15), we have

$$z_1 \leq \|w\|_1 \cdot \left\| \eta - \tilde{\eta} - \frac{\sqrt{2}}{N\sqrt{\pi}} \right\|_\infty \leq \frac{2\pi\sqrt{2}}{N\sqrt{\pi}}.$$

Examining z_2, recall that we showed earlier that $w(\tilde{\eta} + \frac{\sqrt{2}}{N\sqrt{\pi}}) = \tilde{s}_N + \frac{\sqrt{2}}{N\sqrt{\pi}} w$ has ℓ_1 norm $1 + \frac{\sqrt{2\pi}}{N}$. So we have $z_2 \leq \frac{\sqrt{2\pi}}{N}(1 + \frac{\sqrt{2}}{N\sqrt{\pi}}) \leq \frac{2\sqrt{2\pi}}{N}$ for all $N \geq 1$.

Compiling the bounds on z_1 and z_2, we have that for all 1-Lipschitz f, $\langle f, s_N - q \rangle \leq \frac{4\sqrt{2\pi}}{N} \leq \frac{11}{N}$, and thus $W_1(\bar{s}_N, q) \leq \frac{11}{N}$. For $N \geq \frac{18}{\epsilon}$ we conclude that $W_1(\bar{s}_N, q) \leq \epsilon$. Applying triangle quality as discussed above completes the proof. □

Lemma 4.7.4 is exactly analogous to Lemma 4.3.4. We can take advantage of the result by using the Hutchinson's based method from Section 4.4 or the sublinear time method from Section 4.5 to obtain the approximations for the Chebyshev moments required by Algorithm 11. The end result is that we can obtain the same bounds as Theorem 4.1.4 and Theorem 4.1.3 with $\ell = \max(1, \frac{C'}{n}\epsilon^{-4}\log^2(\frac{1}{\epsilon\delta}))$ and $\epsilon_{\mathsf{MV}} = C''\epsilon^{-4}$, respectively. The slightly worse ϵ dependence follows from the fact that Algorithm 11 has a more stringent requirement on the approximate Chebyshev moments used than Algorithm 6.

4.8 Approximate Eigenvalues from Spectral Density Estimate

Algorithm 10 and Algorithm 11 in the previous sections output a closed form representation of a distribution q which is close in Wasserstein-1 distance to s. In particular, the distribution

output is continuous. Alternatively, we describe a simple greedy algorithm (Algorithm 12) that recovers a list of n eigenvalues $\tilde{\Lambda} = [\tilde{\lambda}_1, \ldots, \tilde{\lambda}_n]$ such that $\|\Lambda - \tilde{\Lambda}\|_1 \leq 3n\epsilon$, which implies that the discrete distribution associated with $\tilde{\Lambda}$ is 3ϵ close to s in Wassersetin-1 distance. Formally:

Theorem 4.8.1. *Let s be a spectral density and let q be a density on $[-1, 1]$ such $W_1(s, q) \leq \epsilon$ for $\epsilon \in (0, 1)$. As long as q can be integrated over any subinterval of $[-1, 1]$ (e.g., has a closed form antiderivative), there is an algorithm (Algorithm 12) that computes $1/\epsilon$ such integrals and in $O\left(n + 1/\epsilon\right)$ additional time outputs a list of n values $\tilde{\Lambda} = [\tilde{\lambda}_1, \ldots, \tilde{\lambda}_n]$ such that $\|\Lambda - \tilde{\Lambda}\|_1 \leq 3n\epsilon$.*

At a high-level, Algorithm 12 computes a grid with spacing ϵ for the interval $[-1, 1]$, "snaps" the mass of the continuous density onto the nearest point in the grid, and then readjusts the resulting point masses to a distribution where every point mass is divisible by $1/n$ (and can therefore be represented by a certain number of eigenvalues). It does so by iteratively shifting fractional masses to the next point in the grid so that the mass at the current point is divisible by $1/n$.

The method requires computing the mass $\int_a^b q(x)dx$ where $-1 \leq a < b \leq 1$. For Algorithms 10 and 11, q is written as $q = w \cdot p$ where p is a degree N polynomial written as a sum of the first $N + 1$ Chebyshev polynomials. So to compute the integral $\int_a^b q(x)dx$, we just need to compute the integral $\int_a^b T_k(x)w(x)dx$ for any $k \in 0, \ldots, N$. We can do so using the following closed form expression (see Appendix B.2 for a short derivation):

Fact 4.8.2. *For $k \in \mathbb{N}^{>0}$ and $-1 \leq a < b \leq 1$ we have that*

$$\int_a^b \frac{T_k(x)}{\sqrt{1-x^2}}dx = \frac{-\cos(k \sin^{-1} b)}{k} - \frac{-\cos(k \sin^{-1} a)}{k}$$

For $k = 0$, $T_k(x) = 1$ for all x and we have that $\int_a^b T_k(x)w(x)dx = \sin^{-1}(b) - \sin^{-1}(a)$.

Using the above fact, when $q = w \cdot p$ for a degree N polynomial p, we can compute $\int_a^b q(x)dx$ in $O(N)$ time. In our main results $N = O(1/\epsilon)$, so this cost is small.

Algorithm 12 Approximate Eigenvalues from Spectral Density

Input: Spectral density $q : [-1, 1] \to \mathbb{R}^+$, integer n.
Output: Vector $\tilde{\Lambda} = [\tilde{\lambda}_1, \ldots, \tilde{\lambda}_n]$.

1: compute $\vec{v} = (v_{-1+\epsilon}, v_{-1+2\epsilon} \ldots, v_0, v_\epsilon, \ldots, v_1)$ such that $v_t = \int_{t-\epsilon}^{t} q(x)dx$
2: **for** t in $(-1+\epsilon, -1+2\epsilon \ldots, 0, \epsilon, \ldots, 1)$ **do**
3: $\quad r \leftarrow v_t - \lfloor v_t \rfloor_{1/n}$ ▷ $\lfloor v_t \rfloor_{1/n}$ is the largest value $\leq v_t$ that is divisible by $\frac{1}{n}$
4: $\quad v_{t+\epsilon} \leftarrow r + v_{t+\epsilon}$
5: $\quad$ Set $n \cdot \lfloor v_t \rfloor_{1/n}$ values in $\hat{\Lambda}$ to be t
6: return $\hat{\Lambda}$

Proof of Theorem 4.8.1. Consider the output $\tilde{\Lambda}$ of Algorithm 12 with input q and n. Notice that $W_1(v, q) \leq \epsilon$ by the definition of v and the earthmover's definition of the Wasserstein distance. Hence, by triangle inequality, we have that $W_1(v, s) \leq 2\epsilon$. Let $\tilde{v}$ be the vector of masses after the shifting procedure (Line 4) in the for-loop of the algorithm. Notice that $\tilde{v}$ is the distribution corresponding to having n equally weighted point-masses on the points in $\tilde{\Lambda}$. Since the procedure in Line 4 moves at most $1/n$ mass at most ϵ distance in each iteration, we have $W_1(v, \tilde{v}) \leq \epsilon$ by the earthmover's distance definition of the Wasserstein-1 distance. It follows then that $W_1(\tilde{v}, s) \leq 3\epsilon$.

□

We note that there are other options beyond Algorithm 12 for discretizing a continuous density return by the Jackson damped kernel polynomial method – i) the optimal discretization of a continuous density on the interval $[-1, 1]$ into n equally-weighted point-masses, and ii) an algorithm by [41] that can be seen as a combination of Algorithm 12 and the optimal method.

Optimal Discretization. Given the continuous density q, consider the discrete density that results from the following procedure:

1. Initialize $t = -1$, then repeat the following steps until $t = 1$.
2. Let $t' \geq t$ be the smallest value such that $\int_t^{t'} q(x)dx = \frac{1}{n}$.
3. Place a point-mass at $\mathbf{E}_{x\sim q}\left[x \mid x \in [t, t']\right]$. I.e. a point-mass is placed in the interval $[t, t']$ at the point given by the conditional distribution of q on the interval.
4. Update $t \leftarrow t'$.

The values $\tilde{\Lambda} = \tilde{\lambda}_1, \ldots, \tilde{\lambda}_n$ given by the point-masses computed by the aforementioned procedure is a optimal discretization of q into n equally-weighted point-masses on $[-1, 1]$ in terms of Wasserstein-1 distance. To see why this is the case, consider the first $1/n$ fraction of the mass of the density q, i.e. the smallest $t > -1$ such that $\int_{-1}^{t} q(x)dx = 1/n$. The policy minimizing the earthmover's distance to any n equally-weighted point-wise masses must "move" the mass of q on the interval $[-1, t]$ to the point-mass closest to -1. Hence, it is sufficient to restrict our attention to the interval $[-1, t]$ when computing the smallest point-mass, i.e. the mass closest to -1. Now that we are constrained to looking at the interval $[-1, t]$ one can check that the point-mass minimizing the earthmover's distance to q, restricted to $[-1, t]$, is the point-mass at $\mathbf{E}_{x\sim q}\left[x \mid x \in [-1, t]\right]$. The optimality of the procedure follows from making this argument inductively for all n point-masses.

We note that all steps of the procedure takes roughly $O(n)$ time, although a numerical integration technique or binary search would need to be used to find each t' to sufficiently high accuracy.

A result combining the greedy discretization in Algorithm 12 and the optimal discretization is given in [41]. They compute a fractional discretization on an ϵ-spaced grid of $[-1, 1]$,

as in Algorithm 12, but then compute the eigenvalues using the conditional expectation of every $1/n$ fraction of mass based on the discrete density on the grid.

Appendix A

Additional Proofs for Chapter 2

A.1 Proof of Fact 2.2.1

Let $M = AA^\top$ be a PSD matrix, with eigenvalues $\lambda_1 \geq \ldots \geq \lambda_n \geq 0$. Let $\vec{m}, \vec{\lambda} \in \mathbb{R}^n$ be the vectors corresponding to the diagonal entries of M and the eigenvalues of M respectively, both in non-increasing order. Then, by Schur-Horn theorem (Theorem 4.3.26 in [77]), $\vec{\lambda}$ weakly majorizes $\vec{m}$, i.e. $\sum_{i=1}^r \lambda_i \geq \sum_{i=1}^r m_i$ for all $r \in [n]$.

Since $f(y) = \sum_{i=1}^n y_i^t$ is a Schur-convex function for $y \in \mathbb{R}^n$ and $t \geq 1$, we have that $\sum_{i=1}^n \lambda_i^t \geq \sum_{i=1}^n m_i^t$. The statement follows from the fact that $\sum_{i=1}^n \lambda_i^t = \|AA^\top\|_{S_t}^t = \|A\|_{S_{2t}}^{2t}$ and $m_i = \|a_i\|_2^2$ for all $i \in [n]$.

A.2 Proof of Theorem 2.2.2

It is easy to see that the estimator is unbiased; $\mathbf{E}[\hat{Z}] = \sum_{i\in[n]} \frac{z_i}{\tau_i} \cdot \tau_i = z$. Bounding the variance can be done as follows,

$$Var(\hat{Z}) \leq \mathbf{E}[(\hat{Z})^2] = \sum_{i\in[n]} \left(\frac{z_i}{\tau_i}\right)^2 \tau_i = \sum_{i\in[n]} \left(\frac{|z_i|}{\tau_i}\right)^2 \tau_i.$$

Since for each $i \in [n]$ we have $\tau_i \geq \frac{|z_i|}{\lambda z}$, we can bound $Var(\hat{Z}) \leq \sum_{i\in[n]} (\lambda z)^2 \tau_i = (\lambda z)^2$

A.3 Proof of Lemma 2.5.2

The expectation is straight forward. First assume $r = 1$:

$$\mathbf{E}[Y] = \mathbf{E}[\frac{z_{i,j}}{\tau_i \tau_j}] = \sum_{l\in[n],m\in[n]} \frac{z_{l,m}}{\tau_l \tau_m} \tau_l \tau_m = z$$

and then using the linearity of expectation,

$$\mathbf{E}[Y] = \frac{1}{r^2} \sum_{u\in[r],v\in[r]} \mathbf{E}[\frac{z_{i_u,j_v}}{\tau_{i_u}\tau_{j_v}}] = \frac{1}{r^2} \sum_{u\in[r],v\in[r]} z = z$$

For the variance,

$$\mathbb{E}Y^2 = \frac{1}{r^4} \sum_{u,v} \mathbf{E}[\sum_{\substack{u'\neq u \\ v'\neq v}} \frac{z_{i_u,j_v}}{\tau_{i_u}\tau_{j_v}} \cdot \frac{z_{i_{u'},j_{v'}}}{\tau_{i_{u'}}\tau_{j_{v'}}}] + \mathbf{E}[\left(\frac{z_{i_u,j_v}}{\tau_{i_u}\tau_{j_v}}\right)^2]$$
$$+ \mathbf{E}[\sum_{u\neq u'} \frac{z_{i_u,j_v}}{\tau_{i_u}\tau_{j_v}} \cdot \frac{z_{i_{u'},j_v}}{\tau_{i_{u'}}\tau_{j_v}}] + \mathbf{E}[\sum_{v\neq v'} \frac{z_{i_u,j_v}}{\tau_{i_u}\tau_{j_v}} \cdot \frac{z_{i_u,j_{v'}}}{\tau_{i_u}\tau_{j_{v'}}}]$$

As i_u and $i_{u'}$ are independent for $u \neq u'$, and similarly for j_v and $j_{v'}$ for $v \neq v'$, we get

$$
\begin{aligned}
&= \frac{1}{r^4}\Bigg(r^2(r-1)^2 z^2 + r^2 \sum_{l,m} \frac{z_{l,m}}{\tau_l} \cdot \frac{z_{l,m}}{\tau_m} + r^2(r-1) \sum_{l,m,m'} \frac{z_{l,m'}}{\tau_l} \cdot z_{l,m} \\
&+ r^2(r-1) \sum_{l,l',m} \frac{z_{l',m}}{\tau_m} \cdot z_{l,m} \Bigg) \\
&\leq z^2 + \frac{1}{r^2} \sum_{l,m} \frac{|z_{l,m}|}{\tau_l \tau_m} \cdot |z_{l,m}| + \frac{1}{r} \sum_{l,m,m'} \frac{|z_{l,m'}|}{\tau_l} \cdot |z_{l,m}| + \frac{1}{r} \sum_{l,l',m} \frac{|z_{l',m}|}{\tau_m} \cdot |z_{l,m}|
\end{aligned}
$$

As the first term is just $(\mathbf{E}[Y])^2$, it holds that

$$
\mathrm{Var}(Y) \leq \frac{1}{r^2} \sum_{l,m \in N(l)} \frac{|z_{l,m}|}{\tau_l \tau_m} \cdot |z_{l,m}| + \frac{1}{r} \sum_{l,m,m' \in N(l)} \frac{|z_{l,m'}|}{\tau_l} \cdot |z_{l,m}| + \frac{1}{r} \sum_{\substack{m,l \in N(m) \\ l' \in N(m)}} \frac{|z_{l',m}|}{\tau_m} \cdot |z_{l,m}|
$$

Recalling that $z_{l,m} = 0$ for all $(l,m) \notin E$, we can rewrite the above as

$$
= \frac{1}{r^2} \sum_{l,m \in N(l)} \frac{|z_{l,m}|}{\tau_l \tau_m} \cdot |z_{l,m}| + \frac{1}{r} \sum_{\substack{m,l \in N(l) \\ l' \in N(l)}} \frac{|z_{l,m'}|}{\tau_l} \cdot |z_{l,m}| + \frac{1}{r} \sum_{\substack{m,l \in N(m) \\ l' \in N(m)}} \frac{|z_{l',m}|}{\tau_m} \cdot |z_{l,m}|
$$

and using the bound on the probability,

$$
\leq \frac{\lambda^2 z}{r^2} \sum_{l,m \in N(l)} |z_{l,m}| + \frac{\lambda z}{r} \sum_{l,m \in N(l), m' \in N(l)} |z_{l,m}| + \frac{\lambda z}{r} \sum_{m,l \in N(m), l' \in N(m)} |z_{l,m}|
$$

Finally, using the bounds on maximum degrees, we get

$$
\leq \left(\frac{\lambda^2}{r^2} + \frac{2\lambda\Delta}{r} \right) z \sum_{i,j \in [n]} |z_{i,j}|
$$

A.4 Bounding the tail of the Estrada index Taylor expansion (Theorem 2.8.2)

We bound the tail of the Estrada index Taylor expansion (2.14), i.e. $\left|\sum_{p=m+1}^{\infty} \frac{\operatorname{tr}(A^p)}{p!}\right| \leq \varepsilon \left|\operatorname{tr}(\exp(A))\right|$ for $m = \lceil (e\theta+1)\log(1/\varepsilon) - 1 \rceil$.

$$\left|\sum_{p=m+1}^{\infty} \frac{\operatorname{tr}(A^p)}{p!}\right| \leq \left|\sum_{p=m+1}^{\infty} \frac{\operatorname{tr}(A^{m+1}A^{p-(m+1)})}{(m+1)!(p-(m+1))!}\right|.$$

Using $\operatorname{tr}(AB) \leq \|A\|_{S_\infty} \cdot \operatorname{tr}(B)$ which follows from Von Neuman's trace inequality (see [17]),

$$\leq \frac{\|A^{m+1}\|_{S_\infty}}{(m+1)!}\left|\sum_{p=m+1}^{\infty} \frac{\operatorname{tr}(A^{p-(m+1)})}{(p-(m+1))!}\right|,$$

and by the bound on the largest eigenvalue and Stirling's formula,

$$\begin{aligned}
&\leq \frac{(e\theta)^{m+1}}{(m+1)^{m+3/2}\sqrt{2\pi}}\left|\sum_{p=0}^{\infty} \frac{\operatorname{tr}(A^p)}{p!}\right| \\
&\leq \left(\frac{e\theta}{m+1}\right)^{m+1}\left|\operatorname{tr}(\exp(A))\right|
\end{aligned}$$

Setting $m = \lceil (e\theta+1)\log(1/\varepsilon) - 1 \rceil$ and using $(1-x^{-1})^x \leq e^{-1}$ (for $x > 0$) guarantees that

$$\left(\frac{e\theta}{m+1}\right)^{m+1} \leq \left(\frac{e\theta}{(e\theta+1)\log(1/\varepsilon)}\right)^{m+1} \leq \left(1 - \frac{e\theta}{e\theta+1}\right)^{(e\theta+1)\log(1/\varepsilon)} = \varepsilon.$$

Appendix B

Additional Proofs for Chapter 4

B.1 Positive Polynomial Approximation

In this section, we introduce Jackson's powerful result from 1912 on the uniform approximation of Lipschitz continuous periodic functions by low-degree trigonometric polynomials [83, 84]. This result will directly translate to the result for algebraic polynomials needed to analyze the kernel polynomial method. We start with basic definitions and preliminaries below.

B.1.1 Fourier Series Preliminaries

Definition B.1.1 (Fourier Series)**.** A function f with period 2π that is integrable on the length of that period can be written via the Fourier series:

$$f(x) = \frac{\alpha_0}{2} + \sum_{k=1}^{\infty} \alpha_k \cos(kx) + \beta_k \sin(kx)$$

where

$$\alpha_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(kx)dx \qquad\qquad \beta_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin(kx)dx.$$

Equivalently we can write f in *exponential form* as:

$$f(x) = \sum_{k=-\infty}^{\infty} \hat{f}_k e^{ikx}$$

where $i = \sqrt{-1}$, $\hat{f}_0 = \alpha_0/2$, $\hat{f}_k = \hat{f}^*_{|k|}$ for $k < 0$, and for $k > 0$,

$$\hat{f}_k = \frac{1}{2}(\alpha_k - i\beta_k).$$

If the Fourier series of a periodic function f has $\hat{f}_k = 0$ for $k > N$ (equivalently, $\alpha_k = \beta_k = 0$ for $k > N$), we say that f is a degree N trigonometric polynomial.

In working with Fourier series, we require the two standard convolution theorems:

Claim B.1.2 (First Convolution Theorem)**.** *Let f, g be integrable 2π-periodic functions with exponential form Fourier series coefficients $[\hat{f}_k]_{k=-\infty}^{\infty}$ and $[\hat{g}_k]_{k=-\infty}^{\infty}$, respectively. Let h be their convolution:*

$$h(x) = [f * g](x) = \int_{-\pi}^{\pi} f(u)g(x-u)du.$$

The exponential form Fourier series coefficients of h, $[\hat{h}_k]_{k=-\infty}^{\infty}$, satisfy:

$$\hat{h}_k = 2\pi \cdot \hat{f}_k \hat{g}_k$$

Claim B.1.3 (Second Convolution Theorem)**.** *Let f, g be integrable 2π-periodic functions*

with exponential form Fourier series coefficients $[\hat{f}_k]_{k=-\infty}^{\infty}$ *and* $[\hat{g}_k]_{k=-\infty}^{\infty}$*, respectively. Let* h *be their product:*

$$h(x) = f(x) \cdot g(x).$$

The exponential form Fourier series coefficients of h*,* $[\hat{h}_k]_{k=-\infty}^{\infty}$*, satisfy:*

$$\hat{h}_k = \sum_{j=-\infty}^{\infty} \hat{f}_j \cdot \hat{g}_{k-j}$$

In other words, the Fourier coefficients of h *are the discrete convolution of those of* f *and* g*.*

B.1.2 Jackson's Theorem for Trigonometric Polynomials

We seek a low-degree trigonometric polynomial $\tilde{f}$ that is a good *uniform* approximation to any sufficiently smooth periodic function f. I.e., we want $\|f - \tilde{f}\|_\infty < \epsilon$ where $\|z\|_\infty$ denotes $\|z\|_\infty = \max_x z(x)$. A natural choice for $\tilde{f}$ is the truncated Fourier series $\sum_{k=-N}^{N} c_k e^{ikx}$, but this does not lead to good uniform approximation in general. Instead, Jackson showed that better accuracy can be obtained with a Fourier series with *damped coefficients*, which is equivalent to the *convolution* of f with an appropriately chosen "bump" function (aka kernel), defined below:

Definition B.1.4 (Jackson Kernel)**.** For any positive integer m, let b be the $2m-2$ degree trigonometric polynomial:

$$b = \left(\frac{\sin(mx/2)}{\sin(x/2)}\right)^4 = \sum_{k=-2m+2}^{2m-2} \hat{b}_k e^{ikx},$$

which has exponential form coefficients $\hat{b}_{-2m+2}, \ldots, \hat{b}_0, \ldots, \hat{b}_{2m-2}$ equal to:

$$\hat{b}_{-k} = \hat{b}_k = \sum_{j=-m}^{m-k} (m - |j|) \cdot (m - |j+k|) \qquad \text{for } k = 0, \ldots, 2m-2. \tag{B.1}$$

When m is odd it is easy to see that b is a degree $2m-2$ trigonometric polynomial. Specifically, for odd m we have the well known Fourier series of the periodic sinc function $s(x) = \frac{\sin(mx/2)}{\sin(x/2)} = \sum_{k=-(m-1)/2}^{(m-1)/2} e^{ikx}$. We then apply the convolution theorem (Claim B.1.3) to $s(x) \cdot s(x)$. to see that $s^2(x) = \left(\frac{\sin(mx/2)}{\sin(x/2)}\right)^2$ is an $m-1$ degree trigonometric polynomial with coefficients $c_{-k} = c_k = m - k$. Applying it again to $s^2(x) \cdot s^2(x)$ yields (B.1). For a derivation of (B.1) when m is even, we refer the reader to [84] or [107].

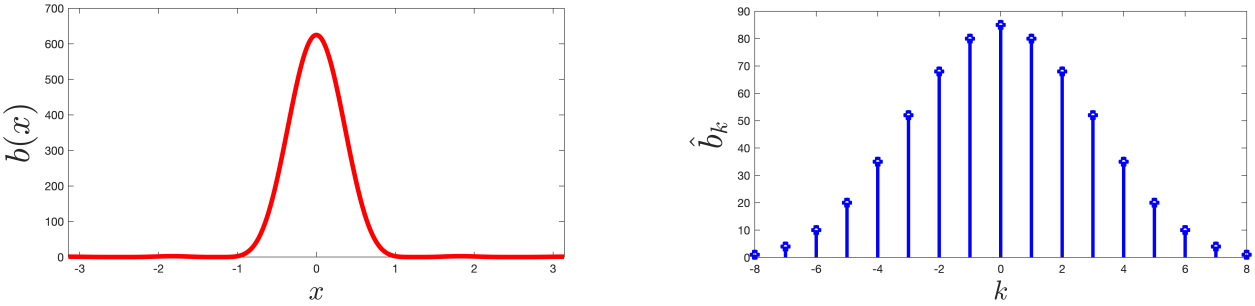

Figure B.1: Jackson's bump function $b(x)$ for $m = 5$, alongside its Fourier series coefficients.

Jackson's main result is as follows. We include a short proof for completeness.

Theorem B.1.5 (Jackson [83], see also [84]). *Let f be a 2π-periodic, Lipschitz continuous function with Lipschitz constant λ. I.e., $|f(x) - f(y)| \leq \lambda|x - y|$ for all x, y. For integer m, let b be the bump function from Definition B.1, with k^{th} Fourier ceofficients $\hat{b}_k$. The function $\tilde{f}(x) = \frac{1}{2\pi\hat{b}_0} \int_{-\pi}^{\pi} b(u) f(x-u) du$ satisfies:*

$$\|\tilde{f} - f\|_\infty \leq 9\frac{\lambda}{m}.$$

$\tilde{f}$ is a $2m-2$ degree trigonometric polynomial, and by the convolution theorem, its exponential

form Fourier series coefficients are given by $\hat{\tilde{f}}_k = \frac{\hat{b}_k}{\hat{b}_0} \cdot \hat{f}_k$ *for* $k = -2m+2, \ldots, 2m-2$.

Remark. The function $\tilde{f}$ takes the form of a *damped* truncation of f's Fourier series: $\frac{\hat{b}_0}{\hat{b}_0} = 1$ and $\frac{\hat{b}_k}{\hat{b}_0}$ falls off towards zero as $k \to 2m-2$. After $2m-2$, the Fourier series coefficients from f are fully truncated to 0.

Proof. Recalling that $\hat{b}_0 = \frac{1}{2\pi}\int_{-\pi}^{\pi} b(x)dx$, we have that $\int_{-\pi}^{\pi} \frac{1}{2\pi\hat{b}_0} b(u)du = 1$, and thus

$$|\tilde{f}(x) - f(x)| \leq \int_{-\pi}^{\pi} \frac{1}{2\pi\hat{b}_0} b(u) \cdot |f(x) - f(x-u)|du.$$

By our Lipschitz assumption of f, we can bound $|f(x) - f(x-u)| \leq \lambda|u|$ and thus have:

$$\max_x |\tilde{f}(x) - f(x)| = \|\tilde{f} - f\|_\infty \leq \lambda \cdot \frac{\int_{-\pi}^{\pi} |u| b(u)du}{2\pi\hat{b}_0} = \lambda \cdot \frac{\int_0^{\pi} ub(u)du}{\int_0^{\pi} b(u)du}. \tag{B.2}$$

In the last equality, we use that b is symmetric about zero. We have that $2 \cdot \sin\left(\frac{u}{2}\right) \leq u \leq \pi \cdot \sin\left(\frac{u}{2}\right)$ for $x \in [0, \pi]$ and thus:

$$\int_0^{\pi} ub(u)du \leq \pi^4 \int_0^{\pi} u\frac{\sin(mu/2)^4}{u^4}du = \pi^4 m^2 \int_0^{\pi m} \frac{\sin(v/2)^4}{v^3}dv \leq \pi^4 m^2 \int_0^{\infty} \frac{\sin(v/2)^4}{v^3}dv.$$

The last integral evaluates of $\frac{\ln 2}{4}$, so overall we have $\int_0^{\pi} b(u) \cdot udu \leq \frac{\pi^4 \ln 2}{4} \cdot m^2$. Moreover we can check that:

$$\int_0^{\pi} b(u)du = \pi \cdot (\frac{2}{3}m^3 + \frac{1}{3}m) \geq \frac{2\pi}{3}m^3.$$

Plugging into (B.2) we have that:

$$\|\tilde{f} - f\|_\infty \leq 8.06\frac{\lambda}{m}.$$

The result follows. We note that the constant above is loose: numerical results suggest the bound can be improved to $\leq \frac{\pi}{2}\frac{\lambda}{m}$. □

Theorem B.1.5 translates to a result for *algebraic polynomials* via a standard transformation between Fourier series and Chebyshev series, which we detail below.

B.1.3 Jackson's Theorem for Algebraic Polynomials

Theorem B.1.6. *Let $f \in \mathcal{F}([-1,1],\mathbb{R})$ be a Lipschitz continuous function on $[-1,1]$ with Lipschitz constant λ. I.e., $|f(x)-f(y)| \leq \lambda|x-y|$ for all x, y. For integer m, let $\hat{b}_0, \ldots, \hat{b}_{2m-2}$ be the coefficients from (B.1). Let $c_k = \langle f, w \cdot \bar{T}_k \rangle$ be the k^{th} coefficient in f's Chebyshev polynomial expansion, where w and $\bar{T}_k$ are as defined in Section 4.2. The degree $(2m-2)$ algebraic polynomial*

$$\tilde{f}(x) = \sum_{n=0}^{2m-2} \frac{\hat{b}_k}{\hat{b}_0} c_k \cdot \bar{T}_k(x)$$

satisfies $\|\tilde{f} - f\|_\infty \leq 9\frac{\lambda}{m}$.

Proof. To translate from the trigonometric case to the algebraic setting, we will use the identity that for all k,

$$T_k(\cos\theta) = \cos(k\theta). \tag{B.3}$$

Consider any function $r \in \mathcal{F}([-1,1],\mathbb{R})$ with Chebyshev expansion coefficients $c_0, c_1, \ldots$, where $c_k = \langle r, w \cdot \bar{T}_k \rangle$. Transform r into a periodic function as follows: let $g(\theta) = r(\cos\theta)$ for $\theta \in [-\pi, 0]$ and let $h(\theta) = g(-|\theta|)$ for $\theta \in [-\pi, \pi]$. The function $h(\theta)$ is periodic, and also

even, so its Fourier series has all coefficients $\beta_1, \beta_2, \ldots$ equal to 0. We thus have that

$$h(\theta) = \sum_{n=0}^{\infty} \alpha_k \cos(n\theta),$$

where

$$\alpha_0 = \frac{1}{2\pi} \int_{-\pi}^{\pi} h(\theta) \cos(k\theta) d\theta = \frac{1}{\pi} \int_{-\pi}^{0} g(\theta) \cos(k\theta) d\theta$$

and, for $n > 0$,

$$\alpha_k = \frac{1}{\pi} \int_{-\pi}^{\pi} h(\theta) \cos(k\theta) d\theta = \frac{2}{\pi} \int_{-\pi}^{0} g(\theta) \cos(k\theta) d\theta.$$

Using (B.3) and the fact that $\frac{d}{dx} \cos^{-1}(x) = \frac{1}{\sqrt{1-x^2}}$, we have:

$$\int_{-\pi}^{0} g(\theta) \cos(k\theta) d\theta = \int_{-1}^{1} r(x) T_k(x) \frac{1}{\sqrt{1-x^2}} dx.$$

We conclude that the Chebyshev coefficients of r are precisely a scaling of the Fourier coefficients of h. Specifically, since $\bar{T}_0 = \sqrt{\frac{2}{\pi}} T_0$ and $\bar{T}_k = \sqrt{\frac{1}{\pi}} T_k$, we have:

$$\sqrt{\frac{2}{\pi}} c_0 = \alpha_0, \qquad \sqrt{\frac{1}{\pi}} c_k = \alpha_k \text{ for } k > 0. \tag{B.4}$$

With this fact in hand, Theorem B.1.6 follows almost immediately from Theorem B.1.5. Specifically, given $f \in \mathcal{F}([-1,1], \mathbb{R})$ with Chebyshev series coefficients $c_0, c_1, \ldots$, we let $g(\theta) = f(\cos\theta)$ and $h(\theta) = g(-|\theta|)$. Let $\alpha_0, \alpha_1, \ldots$ denote h's non-zero Fourier coefficients. Then, let $\tilde{h}$ be the approximation to h given by Theorem B.1.5. $\tilde{h}$ is a $2m-2$ degree trigonometric polynomial and is even since h is even and the bump function b is symmetric. Denote $\tilde{h}$'s non-zero Fourier coefficients by $\tilde{\alpha}_0, \ldots, \tilde{\alpha}_{2m-2}$. We have that $\tilde{\alpha}_k = \frac{\hat{b}_k}{\hat{b}_0} \alpha_k$ for $0 \leq k \leq 2m-2$.

Finally, let $\tilde{f} \in \mathcal{F}([-1,1],\mathbb{R})$ be defined by $\tilde{f}(\cos(\theta)) = h(-\theta)$. By (B.4), we have that $\tilde{f}$ is a degree $2m-2$ polynomial and its Chebyshev series coefficients $\tilde{c}_0, \ldots, \tilde{c}_{2m-2}$ are exactly equal to $\frac{\hat{b}_k}{\hat{b}_0} c_k$.

Moreover, we have $\|f - \tilde{f}\|_\infty = \max_{x\in[-1,1]} |f(x) - \tilde{f}(x)| = \max_x |h(x) - \tilde{h}(x)|$. By Theorem B.1.5 we have $\max_x |h(x) - \tilde{h}(x)| < 9\frac{\lambda}{m}$, so we conclude that $\|f - \tilde{f}\|_\infty < 9\frac{\lambda}{m}$. □

In addition to the main result of Theorem B.1.6, our SDE algorithm alsos require an additional property of the damped Chebyshev approximation $\tilde{f}$:

Lemma B.1.7. *For any non-negative function $f \in \mathcal{F}([-1,1],\mathbb{R})$ (not necessarily Lipschitz), let $\tilde{f}$ be as in Theorem B.1.6. We have that $\tilde{f}$ is also non-negative on $[-1,1]$.*

Proof. Let $h(\theta)$ and $\tilde{h}(\theta)$ be the 2π perioduc functions as in the proof of Theorem B.1.6. I.e., $h(\theta) = g(-|\theta|)$ where $g(\theta) = f(\cos\theta)$ and $\tilde{h}$ is the truncated, Jackson-damped approximation to h from Theorem B.1.5. If f is non-negative, then so is h, and since $\tilde{h}$ is the convolution of h with a non-negative function, it is non-negative as well. Finally, since $\tilde{f}(\cos(\theta)) = h(-\theta)$, we conclude that $\tilde{f}(x) \geq 0$ for $x \in [-1,1]$. □

B.2 Derivation of Fact 4.8.2

Let $x = \sin(u)$ then we have that $dx = \cos(u)du$. Substituting the change of variable in the integral and noting the fact that $T_k(\cos\theta) = \cos(k\theta)$ for $\theta \in [-\pi,\pi]$ gives us that

$$\int_a^b \frac{T_k(x)}{\sqrt{1-x^2}}dx = \int_{\sin^{-1}a}^{\sin^{-1}b} \frac{\cos(k\cos^{-1}\sin(u))}{\sqrt{1-\sin^2(u)}}\cos(u)du = \int_{\sin^{-1}a}^{\sin^{-1}b} \cos(k(\pi/2-u))du$$
$$= \frac{-\sin(k(\pi/2-u))}{k}\Big|_{\sin^{-1}a}^{\sin^{-1}b} = \frac{-\cos(ku)}{k}\Big|_{\sin^{-1}a}^{\sin^{-1}b}$$

where we used the fact that $\cos^2(u) + \sin^2(u) = 1$ and $\int \cos(u)du = \sin(u) + c$.

B.3 Proof of Fact 4.3.3

Proof. We start by doing a change of variables; set $x = \cos\theta$ and note that $dx = -\sin\theta d\theta$. Substituting this into the expression for $\langle f, w \cdot \bar{T}_k \rangle$ and noting that $T_k(\cos\theta) = \cos k\theta$ gives us that

$$\sqrt{\frac{2}{\pi}} \int_{-1}^{1} f(x) \frac{T_k(x)}{\sqrt{1-x^2}} dx = \sqrt{\frac{2}{\pi}} \int_{-\pi}^{0} -f(\cos\theta)(\cos k\theta) d\theta$$

since $\sqrt{1-\cos^2\theta} = \sin\theta$ and $dx = -\sin\theta d\theta$. Integrating by parts and noting that

$$(f(\cos\theta) \int -\cos k\theta d\theta)|_{-\pi}^{0} = -f(\cos\theta)\frac{\sin k\theta}{k}|_{-\pi}^{0} = 0$$

gives us that

$$\langle f, w \cdot \bar{T}_k \rangle = \sqrt{\frac{2}{\pi}} \int_{-\pi}^{0} \frac{\sin k\theta}{k} \, df(\cos\theta).$$

We use the definition of the Riemann-Stieltjes integral and let $M \in \mathbb{N}^+$ be a parameter and $\mathcal{P}_M = \{-\pi = x_0 \leq \cdots \leq x_M = 0\}$ be the set of all M intervals partitioning the interval $[-\pi, 0]$. Then for a partition $P \in \mathcal{P}_M$ we denote $\text{norm}(P)$ to be the length of its longest sub-interval. The Riemann-Stieltjes integral $\int_{-\pi}^{0} \sin(k\theta) \, df(\cos\theta)$ can be written as

$$\int_{-\pi}^{0} \sin k\theta \, df(\cos\theta) = \lim_{\epsilon \to 0} \sup_{\substack{M, \ P \in \mathcal{P}_M \\ \text{s.t.norm}(P) \leq \epsilon}} \sum_{i=0}^{m-1} (f(\cos x_{i+1}) - f(\cos x_i)) \sin k x_i.$$

Since $f(x) \in \text{lip}_1$ and $|\sin k\theta| \leq 1$ we can bound the magnitude of the above summation as

$$\left| \sum_{i=0}^{m-1} (f(\cos x_{i+1}) - f(\cos x_i)) \sin k x_i \right| \leq \sum_{i=0}^{m-1} \lambda |\cos x_{i+1} - \cos x_i| \leq 2.$$

The last inequality follows from the fact that $\cos(\theta)$ is 1-Lipschitz. Putting these bounds together gives us that $|\langle f, w \cdot \bar{T}_k \rangle| \leq 2\lambda/k$. $\square$

Bibliography

[1] D. Achlioptas and F. McSherry. Fast computation of low-rank matrix approximations. *Journal of the ACM (JACM)*, 54(2):9–es, 2007.

[2] D. Achlioptas, Z. S. Karnin, and E. Liberty. Near-optimal entrywise sampling for data matrices. In *Advances in Neural Information Processing Systems*, pages 1565–1573, 2013.

[3] Z. Allen-Zhu and Y. Li. Lazysvd: Even faster svd decomposition yet without agonizing pain. *Advances in neural information processing systems*, 29, 2016.

[4] Z. Allen-Zhu and Y. Li. Faster principal component regression and stable matrix chebyshev approximation. In *International Conference on Machine Learning*, pages 107–115. PMLR, 2017.

[5] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999. doi: 10.1006/jcss.1997.1545.

[6] A. Andoni and H. L. Nguyen. Eigenvalues of a matrix in the streaming model. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013*, pages 1729–1737, 2013. doi: 10.1137/1.9781611973105.124.

[7] A. Andoni, R. Krauthgamer, and K. Onak. Streaming algorithms via precision sampling. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 363–372. IEEE, 2011.

[8] S. Arora, E. Hazan, and S. Kale. Fast algorithms for approximate semidefinite programming using the multiplicative weights update method. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 339–348. IEEE, 2005.

[9] S. Arora, E. Hazan, and S. Kale. A fast random sampling algorithm for sparsifying matrices. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 272–279. Springer, 2006.

[10] J. L. Aurentz, V. Kalantzis, and Y. Saad. Cucheb: A GPU implementation of the filtered Lanczos procedure. *Computer Physics Communications*, 220:332 – 340, 2017.

[11] H. Avron and S. Toledo. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *J. ACM*, 58(2), 2011.

[12] H. Avron, M. Kapralov, C. Musco, C. Musco, A. Velingker, and A. Zandieh. Random fourier features for kernel ridge regression: Approximation bounds and statistical guarantees. In *International conference on machine learning*, pages 253–262. PMLR, 2017.

[13] A. Banerjee. *The spectrum of the graph Laplacian as a tool for analyzing structure and evolution of networks*. PhD thesis, Verlag nicht ermittelbar, 2008.

[14] J. Banks, J. Vargas, A. Kulkarni, and N. Srivastava. Pseudospectral shattering, the sign function, and diagonalization in nearly matrix multiplication time. In *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2019.

[15] C. Bekas, A. Curioni, and I. Fedulova. Low cost high performance uncertainty quantification. In *Proceedings of the 2nd Workshop on High Performance Computational Finance*, pages 1–8, 2009.

[16] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

[17] C. Boutsidis, P. Drineas, P. Kambadur, E.-M. Kontopoulou, and A. Zouzias. A randomized algorithm for approximating the log determinant of a symmetric positive definite matrix. *Linear Algebra and its Applications*, 533:95–117, 2017.

[18] M. Braverman, E. Hazan, M. Simchowitz, and B. Woodworth. The gradient complexity of linear regression. In *Proceedings of the 33rd Annual Conference on Computational Learning Theory (COLT)*, volume 125, pages 627–647, 2020.

[19] V. Braverman. Approximations of schatten norms via taylor expansions. In *International Computer Science Symposium in Russia*, pages 70–79. Springer, 2019.

[20] V. Braverman and R. Ostrovsky. Measuring independence of datasets. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 271–280, 2010.

[21] V. Braverman, R. Ostrovsky, and G. Vorsanger. Weighted sampling without replacement from data streams. *Information Processing Letters*, 115(12):923 – 926, 2015. ISSN 0020-0190. doi: 10.1016/j.ipl.2015.07.007.

[22] V. Braverman, A. Roytman, and G. Vorsanger. Approximating subadditive hadamard functions on implicit matrices. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, 2016.

[23] V. Braverman, S. R. Chestnut, R. Krauthgamer, Y. Li, D. P. Woodruff, and L. Yang. Matrix norms in data streams: Faster, multi-pass and row-order. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 648–657, 2018. URL http://proceedings.mlr.press/v80/braverman18a.html.

[24] S. Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[25] M. Bury and C. Schwiegelshohn. Sublinear estimation of weighted matchings in dynamic data streams. In *23rd Annual European Symposium*, ESA'15, pages 263–274, 2015. doi: 10.1007/978-3-662-48350-3_23.

[26] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009.

[27] Y. Carmon, Y. Jin, A. Sidford, and K. Tian. Coordinate methods for matrix games. In *61st Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 283–293. IEEE, 2020.

[28] M. Charikar, K. C. Chen, and M. Farach-Colton. Finding frequent items in data streams. *Theor. Comput. Sci.*, 312(1):3–15, 2004. doi: 10.1016/S0304-3975(03)00400-6.

[29] I. Chavel. *Eigenvalues in Riemannian geometry*. Academic press, 1984.

[30] J. Chen. How accurately should I compute implicit matrix-vector products when applying the Hutchinson trace estimator? *SIAM J. Scientific Computing*, 38(6), 2016. doi: 10.1137/15M1051506.

[31] T. Chen, T. Trogdon, and S. Ubaru. Analysis of stochastic lanczos quadrature for spectrum approximation. In *Proceedings of the International Congress of Mathematicians 2021 (ICM)*, 2021.

[32] K. L. Clarkson and D. P. Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 205–214, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-506-2. doi: 10.1145/1536414.1536445.

[33] K. L. Clarkson and D. P. Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the 41st Annual ACM SIGACT Symposium on Theory of Computing*, pages 205–214, 2009.

[34] K. L. Clarkson and D. P. Woodruff. Low-rank approximation and regression in input sparsity time. *Journal of the ACM (JACM)*, 63(6):1–45, 2017.

[35] C. W. Clenshaw. A note on the summation of chebyshev series. *Mathematics of Computation*, 9(51):118, 1955.

[36] M. B. Cohen. Nearly tight oblivious subspace embeddings by trace inequalities. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 278–287. SIAM, 2016.

[37] M. B. Cohen, R. Kyng, G. L. Miller, J. W. Pachocki, R. Peng, A. B. Rao, and S. C. Xu. Solving sdd linear systems in nearly m log1/2 n time. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 343–352, 2014.

[38] M. B. Cohen, S. Elder, C. Musco, C. Musco, and M. Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 163–172, 2015.

[39] M. B. Cohen, J. Nelson, and D. P. Woodruff. Optimal approximate matrix product in terms of stable rank. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[40] M. B. Cohen, C. Musco, and C. Musco. Input sparsity time low-rank approximation via ridge leverage score sampling. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1758–1777. SIAM, 2017.

[41] D. Cohen-Steiner, W. Kong, C. Sohler, and G. Valiant. Approximating the spectrum of a graph. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1263–1271, 2018.

[42] D. Cohen-Steiner, W. Kong, C. Sohler, and G. Valiant. Approximating the spectrum of a graph. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pages 1263–1271. ACM, 2018. ISBN 978-1-4503-5552-0. doi: 10.1145/3219819.3220119.

[43] G. Cormode and H. Jowhari. Lp samplers and their applications: A survey. *ACM Comput. Surv.*, 52(1):16:1–16:31, 2019. ISSN 0360-0300. doi: 10.1145/3297715.

[44] A. d'Aspremont. Subsampling algorithms for semidefinite programming. *Stochastic Systems*, 1(2):274–305, 2011.

[45] T. A. Davis and Y. Hu. The university of florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1–25, 2011.

[46] J. Demmel, I. Dumitriu, and O. Holtz. Fast linear algebra is stable. *Numerische Mathematik*, 108(1):59–91, 2007.

[47] P. Dharangutte and C. Musco. Dynamic trace estimation. *Preprint*, 2021.

[48] E. Di Napoli, E. Polizzi, and Y. Saad. Efficient estimation of eigenvalue counts in an interval. *Numerical Linear Algebra with Applications*, 2016.

[49] K. Dong, A. R. Benson, and D. Bindel. Network density of states. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1152–1161, 2019.

[50] P. Drineas and M. W. Mahoney. Randnla: randomized numerical linear algebra. *Communications of the ACM*, 59(6):80–90, 2016.

[51] P. Drineas and A. Zouzias. A note on element-wise matrix sparsification via a matrix-valued Bernstein inequality. *Information Processing Letters*, 111(8):385–389, 2011.

[52] P. Drineas, M. W. Mahoney, and N. Cristianini. On the nyström method for approximating a gram matrix for improved kernel-based learning. *journal of machine learning research*, 6(12), 2005.

[53] P. Drineas, R. Kannan, and M. W. Mahoney. Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication. *SIAM Journal on Computing*, 36(1):132–157, 2006.

[54] P. Drineas, M. W. Mahoney, and S. Muthukrishnan. Sampling algorithms for l 2 regression and applications. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1127–1136, 2006.

[55] Z. Du and Z. Liu. On the Estrada and Laplacian Estrada indices of graphs. *Linear Algebra and its Applications*, 435(8):2065–2076, 2011.

[56] D. Feldman, M. Schmidt, and C. Sohler. Turning big data into tiny data: Constant-size coresets for k-means, pca, and projective clustering. *SIAM Journal on Computing*, 49(3):601–657, 2020.

[57] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[58] A. Frieze, R. Kannan, and S. Vempala. Fast Monte-Carlo algorithms for finding low-rank approximations. *Journal of the ACM (JACM)*, 51(6):1025–1041, 2004.

[59] R. Frostig, R. Ge, S. Kakade, and A. Sidford. Un-regularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization. In *International Conference on Machine Learning*, pages 2540–2548, 2015.

[60] R. Frostig, C. Musco, C. Musco, and A. Sidford. Principal component projection without principal component analysis. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 2291–2299, 2016.

[61] J. Ganitkevitch, B. van Durme, and C. Callison-Burch. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764. Association for Computational Linguistics, 2013. URL https://www.aclweb.org/anthology/N13-1092.

[62] M. Ghashami, E. Liberty, and J. M. Phillips. Efficient frequent directions algorithm for sparse matrices. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 845–854, 2016.

[63] B. Ghorbani, S. Krishnan, and Y. Xiao. An investigation into neural net optimization via hessian eigenvalue density. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97, pages 2232–2241, 2019.

[64] P. B. Gibbons and Y. Matias. Synopsis data structures for massive data sets. In *External Memory Algorithms*, volume 50 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1999. ISBN 9780821870938. URL http://theory.stanford.edu/~matias/papers/synopsis.pdf.

[65] A. Gittens and J. A. Tropp. Error bounds for random matrix approximation schemes. *arXiv preprint arXiv:0911.4108*, 2009.

[66] G. H. Golub and C. F. Van Loan. *Matrix computations*. JHU press, 2013.

[67] N. Goyette, P. Jodoin, F. Porikli, J. Konrad, and P. Ishwar. Changedetection.net: A new change detection benchmark dataset. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8, June 2012. doi: 10.1109/CVPRW.2012.6238919.

[68] A. Gronemeier. Asymptotically optimal lower bounds on the NIH-multi-party information complexity of the AND-function and Disjointness. In *26th International Symposium on Theoretical Aspects of Computer Science STACS 2009*, volume 3, pages 505–516. IBFI Schloss Dagstuhl, 2009. doi: 10.4230/lipiCS.stacs.2009.1846.

[69] G. Guennebaud, B. Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[70] N. Gupta and A. Sidford. Exploiting numerical sparsity for efficient learning: faster eigenvector computation and regression. In *Advances in Neural Information Processing Systems*, pages 5269–5278, 2018.

[71] I. Gutman. The energy of a graph: old and new results. In *Algebraic combinatorics and applications*, pages 196–211. Springer, 2001.

[72] I. Gutman, H. Deng, and S. Radenkovic. The Estrada index: An updated survey. *Selected Topics on Applications of Graph Spectra, Math. Inst., Beograd*, pages 155–174, 2011.

[73] N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[74] I. Han, D. Malioutov, and J. Shin. Large-scale log-determinant computation through stochastic Chebyshev expansions. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 908–917, 2015.

[75] I. Han, D. Malioutov, H. Avron, and J. Shin. Approximating spectral sums of large-scale matrices using stochastic Chebyshev approximations. *SIAM J. Scientific Computing*, 39(4), 2017. doi: 10.1137/16M1078148.

[76] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[77] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge university press Cambridge, Cambridge, 1985.

[78] P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5(9), 2004.

[79] Z. Huang. Near optimal frequent directions for sketching dense and sparse matrices. *Journal of Machine Learning Research*, 20(56):1–23, 2019.

[80] N. Hurley and S. Rickard. Comparing measures of sparsity. *IEEE Transactions on Information Theory*, 55(10):4723–4741, 2009.

[81] M. F. Hutchinson. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 19(2):433–450, 1990.

[82] P. Indyk and A. McGregor. Declaring independence via the sketching of sketches. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 737–745. SIAM, 2008. URL http://dl.acm.org/citation.cfm?id=1347082.1347163.

[83] D. Jackson. On approximation by trigonometric sums and polynomials. *Transactions of the American Mathematical society*, 13(4):491–515, 1912.

[84] D. Jackson. *The Theory of Approximation*, volume 11 of *Colloquium Publications*. American Mathematical Society, 1930.

[85] T. S. Jayram. Hellinger strikes back: A note on the multi-party information complexity of AND. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 12th International Workshop, APPROX 2009, and 13th International Workshop, RANDOM 2009, Berkeley, CA, USA, August 21-23, 2009. Proceedings*, pages 562–573, 2009. doi: 10.1007/978-3-642-03685-9_42.

[86] T. S. Jayram and D. P. Woodruff. The data stream space complexity of cascaded norms. In *50th Annual IEEE Symposium on Foundations of Computer Science*, pages 765–774, 2009. doi: 10.1109/FOCS.2009.82.

[87] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013.

[88] L. V. Kantorovich and G. S. Rubinshtein. On a functional space and certain extremum problems. In *Doklady Akademii Nauk*, volume 115, pages 1058–1061. Russian Academy of Sciences, 1957.

[89] L. Katzir, E. Liberty, and O. Somekh. Estimating sizes of social networks via biased sampling. In *Proceedings of the 20th International World Wide Web Conference (WWW)*, pages 597–606, 2011.

[90] A. Khetan and S. Oh. Spectrum estimation from a few entries. *Journal of Machine Learning Research*, 20(21):1–55, 2019. URL http://jmlr.org/papers/v20/18-027.html.

[91] W. Kong and G. Valiant. Spectrum estimation from samples. *The Annals of Statistics*, 45, 01 2016. doi: 10.1214/16-AOS1525.

[92] W. Kong and G. Valiant. Spectrum estimation from samples. *Ann. Statist.*, 45(5): 2218–2247, 10 2017.

[93] I. Koutis, A. Levin, and R. Peng. Faster spectral sparsification and numerical algorithms for sdd matrices. *ACM Transactions on Algorithms (TALG)*, 12(2):1–16, 2015.

[94] A. Kundu and P. Drineas. A note on randomized element-wise matrix sparsification. *arXiv preprint arXiv:1404.0320*, 2014.

[95] A. Kundu, P. Drineas, and M. Magdon-Ismail. Recovering PCA and sparse PCA via hybrid-(l1, l2) sparse sampling of data elements. *The Journal of Machine Learning Research*, 18(1):2558–2591, 2017.

[96] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[97] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data*, 1(1):2?es, Mar. 2007. ISSN 1556-4681. doi: 10.1145/1217299.1217301. URL https://doi.org/10.1145/1217299.1217301.

[98] R. Li, Y. Xi, L. Erlandson, and Y. Saad. The eigenvalues slicing library (EVSL): Algorithms, implementation, and software. *SIAM Journal on Scientific Computing*, 41(4):C393–C415, 2019.

[99] Y. Li and D. P. Woodruff. On approximating functions of the singular values in a stream. In *Proceedings of the 48th annual ACM symposium on Theory of Computing (STOC)*, pages 726–739. ACM, 2016. doi: 10.1145/2184319.2184343.

[100] Y. Li and D. P. Woodruff. Tight bounds for sketching the operator norm, Schatten norms, and subspace embeddings. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016*, pages 39:1–39:11, 2016. doi: 10.4230/LIPIcs.APPROX-RANDOM.2016.39.

[101] Y. Li and D. P. Woodruff. Embeddings of Schatten norms with applications to data streams. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017*, pages 60:1–60:14, 2017. doi: 10.4230/LIPIcs.ICALP.2017.60.

[102] Y. Li, H. L. Nguyen, and D. P. Woodruff. On sketching matrix norms and the top singular vector. In *Proceedings of the 25th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1562–1581. SIAM, 2014. doi: 10.1137/1.9781611973402.114.

[103] E. Liberty. Simple and deterministic matrix sketching. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '13, pages 581–588. ACM, 2013. ISBN 978-1-4503-2174-7. doi: 10.1145/2487575.2487623.

[104] L. Lin, Y. Saad, and C. Yang. Approximating spectral densities of large matrices. *SIAM Review*, 58(1):34–65, 2016.

[105] X. Liu, G. Zhao, J. Yao, and C. Qi. Background subtraction based on low-rank and structured sparse decomposition. *IEEE Transactions on Image Processing*, 24(8): 2502–2514, Aug 2015. ISSN 1057-7149. doi: 10.1109/TIP.2015.2419084.

[106] M. Lopes. Estimating unknown sparsity in compressed sensing. In *International Conference on Machine Learning*, pages 217–225, 2013.

[107] G. G. Lorentz. *Approximation of Functions*. American Mathematical Society, second edition, 1966.

[108] A. Magen and A. Zouzias. Low rank matrix-valued Chernoff bounds and approximate matrix multiplication. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1422–1436. SIAM, 2011.

[109] M. Mahoney and C. Martin. Traditional and heavy tailed self regularization in neural network models. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97, pages 4284–4293, 2019.

[110] X. Meng and M. W. Mahoney. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 91–100, 2013.

[111] R. A. Meyer, C. Musco, C. Musco, and D. Woodruff. Hutch++: Optimal stochastic trace estimation. *Proceedings of the 4th Symposium on Simplicity in Algorithms (SOSA)*, 2020.

[112] M. Monemizadeh and D. P. Woodruff. 1 pass relative-error Lp-sampling with applications. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 1143–1160, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics. ISBN 978-0-898716-98-6. URL http://dl.acm.org/citation.cfm?id=1873601.1873693.

[113] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995. doi: 10.1017/CBO9780511814075.

[114] Y. Mroueh, E. Marcheret, and V. Goel. Co-Occurring Directions Sketching for Approximate Matrix Multiply. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54, pages 567–575. PMLR, 2017.

[115] C. Musco and C. Musco. Randomized block Krylov methods for stronger and faster approximate singular value decomposition. In *Advances in Neural Information Processing Systems 28 (NeurIPS)*, pages 1396–1404, 2015.

[116] C. Musco and C. Musco. Recursive sampling for the nystrom method. *Advances in neural information processing systems*, 30, 2017.

[117] C. Musco and C. Musco. Projection-cost-preserving sketches: Proof strategies and constructions. *arXiv preprint arXiv:2004.08434*, 2020.

[118] C. Musco and D. P. Woodruff. Sublinear time low-rank approximation of positive semidefinite matrices. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 672–683. IEEE, 2017.

[119] C. Musco, C. Musco, and A. Sidford. Stability of the Lanczos method for matrix function approximation. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1605–1624, 2018.

[120] C. Musco, P. Netrapalli, A. Sidford, S. Ubaru, and D. P. Woodruff. Spectrum approximation beyond fast matrix multiplication: Algorithms and hardness. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, volume 94 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 8:1–8:21, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-060-6. doi: 10.4230/LIPIcs.ITCS.2018.8.

[121] C. Musco, P. Netrapalli, A. Sidford, S. Ubaru, and D. P. Woodruff. Spectrum approximation beyond fast matrix multiplication: Algorithms and hardness. *Proceedings of the 9th Conference on Innovations in Theoretical Computer Science (ITCS)*, 2018.

[122] C. Musco, P. Netrapalli, A. Sidford, S. Ubaru, and D. P. Woodruff. Spectrum approximation beyond fast matrix multiplication: Algorithms and hardness. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, volume 94 of *Leibniz International Proceedings in Informatics (LIPICS)*, pages 8:1–8:21. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018.

[123] E. D. Napoli, E. Polizzi, and Y. Saad. Efficient estimation of eigenvalue counts in an interval. *Numerical Lin. Alg. with Applications.*, 23(4):674–692, 2016. doi: 10.1002/nla.2048.

[124] N. H. Nguyen, P. Drineas, and T. D. Tran. Tensor sparsification via a bound on the spectral norm of random tensors. *Information and Inference: A Journal of the IMA*, 4(3):195–229, 2015.

[125] R. O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, New York, NY, USA, 2014. ISBN 1107038324, 9781107038325.

[126] B. N. Parlett. *The symmetric eigenvalue problem*. SIAM, 1998.

[127] B. A. Pearlmutter. Fast exact multiplication by the hessian. *Neural computation*, 6 (1):147–160, 1994.

[128] R. Peng. Approximate undirected maximum flows in o (m polylog (n)) time. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 1862–1867. SIAM, 2016.

[129] R. Peng and S. 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.

[130] J. Pennington, S. Schoenholz, and S. Ganguli. The emergence of spectral universality in deep networks. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1924–1932, 2018.

[131] F. Roosta-Khorasani and U. M. Ascher. Improved bounds on sample size for implicit matrix trace estimators. *Foundations of Computational Mathematics*, 15(5):1187–1212, 2015.

[132] M. Rudelson and R. Vershynin. Hanson-Wright inequality and sub-Gaussian concentration. *Electronic Communications in Probability*, 18, 2013.

[133] S. M. Ruiz. An algebraic identity leading to Wilsons theorem. *The Mathematical Gazette*, 80(489):579–582, 1996.

[134] Y. Saad and M. H. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

[135] W. Schudy and M. Sviridenko. Concentration and moment inequalities for polynomials of independent random variables. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 437–446. SIAM, 2012. URL http://dl.acm.org/citation.cfm?id=2095116.2095153.

[136] O. Shamir. Fast stochastic algorithms for svd and pca: Convergence properties and convexity. In *International Conference on Machine Learning*, pages 248–256. PMLR, 2016.

[137] R. Silver and H. R oder. Densities of states of mega-dimensional hamiltonian matrices. *International Journal of Modern Physics C*, 5(4):735–753, 1994.

[138] M. Simchowitz, A. El Alaoui, and B. Recht. Tight query complexity lower bounds for pca via finite sample deformed wigner law. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1249–1259, 2018.

[139] J. Skilling. *The Eigenvalues of Mega-dimensional Matrices*, pages 455–466. Springer Netherlands, 1989.

[140] D. A. Spielman and S.-H. Teng. Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *SIAM Journal on Matrix Analysis and Applications*, 35(3):835–885, 2014.

[141] X. Sun, D. P. Woodruff, G. Yang, and J. Zhang. Querying a matrix through matrix-vector products. In *Proceedings of the 46th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 132, pages 94:1–94:16, 2019.

[142] L. N. Trefethen. Is gauss quadrature better than clenshaw–curtis? *SIAM review*, 50 (1):67–87, 2008.

[143] J. A. Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*, 12(4):389–434, 2012.

[144] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991.

[145] S. Ubaru and Y. Saad. Fast methods for estimating the numerical rank of large matrices. In *International Conference on Machine Learning*, pages 468–477. PMLR, 2016.

[146] S. Ubaru and Y. Saad. Applications of trace estimation techniques. In *High Performance Computing in Science and Engineering - Third International Conference, HPCSE 2017, Karolinka, Czech Republic, May 22-25, 2017, Revised Selected Papers*, pages 19–33, 2017. doi: 10.1007/978-3-319-97136-0_2.

[147] S. Ubaru, J. Chen, and Y. Saad. Fast estimation of $tr(f(a))$ via stochastic Lanczos quadrature. *SIAM J. Matrix Analysis Applications*, 38(4):1075–1099, 2017. doi: 10.1137/16M1104974.

[148] S. Ubaru, J. Chen, and Y. Saad. Fast estimation of $tr(f(a))$ via stochastic lanczos quadrature. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1075–1099, 2017.

[149] S. Vassilvitskii and D. Arthur. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, 2006.

[150] E. Verbin and W. Yu. The streaming complexity of cycle counting, sorting by reversals, and other problems. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 11–25, 2011. URL http://dl.acm.org/citation.cfm?id=2133036.2133038.

[151] J. S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1): 37–57, 1985. doi: 10.1145/3147.3165.

[152] J. S. Vitter. External memory algorithms and data structures: Dealing with massive data. *ACM Computing Survey.*, 33(2):209–271, 2001. ISSN 0360-0300. doi: 10.1145/384192.384193.

[153] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4): 395–416, 2007.

[154] L.-W. Wang. Calculating the density of states and optical-absorption spectra of large quantum systems by the plane-wave moments method. *Phys. Rev. B*, 49:10154–10158, 1994.

[155] W. Wang and M. A. Carreira-Perpinán. Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. *arXiv preprint arXiv:1309.1541*, 2013.

[156] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. *Advances in neural information processing systems*, 21, 2008.

[157] A. Weiße, G. Wellein, A. Alvermann, and H. Fehske. The kernel polynomial method. *Reviews of modern physics*, 78(1):275, 2006.

[158] D. P. Woodruff. Sketching as a tool for numerical linear algebra. *Found. Trends Theor. Comput. Sci.*, 10:1–157, Oct. 2014. ISSN 1551-305X. doi: 10.1561/0400000060. URL http://dx.doi.org/10.1561/0400000060.

[159] D. P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1–2):1–157, 2014.

[160] J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. *Advances in neural information processing systems*, 22, 2009.

[161] S. J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

[162] L. Wu, J. Laeuchli, V. Kalantzis, A. Stathopoulos, and E. Gallopoulos. Estimating the trace of the matrix inverse by interpolating from the diagonal of an approximate inverse. *J. Comput. Physics*, 326:828–844, 2016. doi: 10.1016/j.jcp.2016.09.001.

[163] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[164] Y. Xie, S. Gu, Y. Liu, W. Zuo, W. Zhang, and L. Zhang. Weighted Schatten p-norm minimization for image denoising and background subtraction. *IEEE Transactions on Image Processing*, 25:4842–4857, 2016.

[165] Z. Yao, A. Gholami, K. Keutzer, and M. W. Mahoney. Pyhessian: Neural networks through the lens of the hessian. In *IEEE BigData*, 2020.

[166] Q. Ye, L. Luo, and Z. Zhang. Frequent direction algorithms for approximate matrix multiplication with applications in CCA. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI*, pages 2301–2307. IJCAI/AAAI Press, 2016.

[167] Y. Zhang, M. J. Wainwright, and M. I. Jordan. Distributed estimation of generalized matrix rank: Efficient algorithms and lower bounds. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 457–465. JMLR.org, 2015. URL http://dl.acm.org/citation.cfm?id=3045118.3045168.