ABSTRACT

Title of dissertation:	QUANTUM ALGORITHMS FOR DIFFERENTIAL EQUATIONS
	Aaron Jacob Ostrander, Doctor of Philosophy, 2019
Dissertation directed by:	Professor Andrew Childs University of Maryland Institute for Advanced Computer Studies

This thesis describes quantum algorithms for Hamiltonian simulation, ordinary differential equations (ODEs), and partial differential equations (PDEs).

Product formulas are used to simulate Hamiltonians which can be expressed as a sum of terms which can each be simulated individually. By simulating each of these terms in sequence, the net effect approximately simulates the total Hamiltonian. We find that the error of product formulas can be improved by randomizing over the order in which the Hamiltonian terms are simulated. We prove that this approach is asymptotically better than ordinary product formulas and present numerical comparisons for small numbers of qubits.

The ODE algorithm applies to the initial value problem for time-independent first order linear ODEs. We approximate the propagator of the ODE by a truncated Taylor series, and we encode the initial value problem in a large linear system. We solve this linear system with a quantum linear system algorithm (QLSA) whose output we perform a post-selective measurement on. The resulting state encodes the solution to the initial value problem. We prove that our algorithm is asymptotically optimal with respect to several system parameters.

The PDE algorithms apply the finite difference method (FDM) to Poisson's equation, the wave equation, and the Klein-Gordon equation. We use high order FDM approximations of the Laplacian operator to develop linear systems for Poisson's equation in cubic volumes under periodic, Neumann, and Dirichlet boundary conditions. Using QLSAs, we output states encoding solutions to Poisson's equation. We prove that our algorithm is exponentially faster with respect to the spatial dimension than analogous classical algorithms. We also consider how high order Laplacian approximations can be used for simulating the wave and Klein-Gordon equations. We consider under what conditions it suffices to use Hamiltonian simulation for time evolution, and we propose an algorithm for these cases that uses QLSAs for state preparation and post-processing.

QUANTUM ALGORITHMS FOR DIFFERENTIAL EQUATIONS

by

Aaron Jacob Ostrander

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2019

Dedication

If I have seen farther than others, it is because of my collaborators.

Acknowledgments

I owe a great deal to my advisor, Andrew Childs, who has given me an immense amount of intellectual freedom to work on a very large array of problems and collaborate with other scientists at QuICS and JQI. Throught his advising and his courses I have gained a broad view of theoretical quantum computing, the scope of which cannot be reflected in just this thesis. He has also consistently supported me throughout my time at QuICS even when life was exceedingly difficult.

About a month after I started at QuICS, I began working with Guoming Wang, who taught me a great deal about the basics of quantum algorithms and complexity theory. Over two years, he helped provide me with a strong foundation in the field.

I thank Chris Monroe for being my departmental advisor and giving me the opportunity to pull my head out of the theoretical clouds and see how a theorist can work in a lab.

I thank Dominic Berry, Stephen Jordan, and Carl Miller for the opportunity to work with them. I also thank Brad Lackey for many stimulating conversations over the years.

I thank the members of the committee, Alexey Gorshkov, Mohammad Hafezi, and Xiaodi Wu, for agreeing to read and critique my thesis. A writer needs an audience.

I thank Bill Fefferman, Shelby Kimmel, Cedric Lin, and Julien Ross, who were post-docs when I joined QuICS and who were very supportive of the new graduate students.

I thank my other collaborators, Pedro Costa, Amir Kalev, Yuan Su, Jin-Peng Liu, Caroline Figgatt, Kevin Landsman, and Norbert Linke.

I thank Linda Macri, Heather Blain Vorhies, and the many fellows of UMD's Graduate Writing Center for helping me grow as a writing consultant and writer. I thank Dave Buehrle for being a great boss during the time I worked as a teaching assistant.

Table of Contents

List of Figures

Chapter 1: Introduction

It is a truth universally acknowledged, that anything written about quantum simulation must begin with the quote

> "... nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical..." - Richard Feynman [42]

from the keynote where Feynman first proposed using quantum computers to simulate quantum mechanics. This thesis presents algorithms to simulate Hamiltonians and to simulate systems described by differential equations other than Schrödinger's equation. In particular we will consider algorihtms for linear ordinary differential equations (ODEs) and linear partial differential equations (PDEs). Most of these algorithms appear in previous publications, namely Refs. [16, 28, 33], and a forthcoming article on PDEs with Andrew Childs and Jinpeng Liu.

Hamiltonian simulation has received a great deal of attention, from Lloyd's original algorithm for local Hamiltonians [67] to the more recent quantum signal processing approach of Low and Chuang [68]. Hamiltonian simulation is an algorithmic primitive that is used for a variety of other algorithms, including quantum linear systems algorithms (QLSAs) which appear throughout this dissertation. We use QLSAs as primitives for the ODE and PDE algorithms; in fact the ODE algorithm amounts to solving a cleverly designed linear system and then post-selecting. QLSAs will also be used in the pre- and post-processing steps of certain cases of the wave equation algorithm. QLSAs provide a crucial perspective for studying finite difference operators in other partial differential equation algorithms.

1.1 Hamiltonian Simulation

We state the problem of Hamiltonian simulation as follows.

HAMILTONIAN SIMULATION (GENERAL STATEMENT): Given the ability to produce an initial state $|\psi_0\rangle$, a Hamiltonian H, an evolution time T, and an error parameter ε, output a state that is ε close to $\exp(-iHT)|\psi_0\rangle$.

There are several details left out of this statement: How is the initial state prepared? How do you compute the Hamiltonian? What norm is used to measure the distance between states? Initial states of interest must be efficiently preparable, and for our purposes it is sufficient to assume that we can implement a *state preparation oracle*, ie, an efficiently implementable unitary U_{prep} such that $U_{prep}|0\rangle^{\otimes n} = |\psi_0\rangle$. Note that the problem is how to implement a unitary, not a quantum channel, which requires the simulation of a Lindblad master equation.

There are many norms that can be used to measure the distance between states. Since pure states exist in a Hilbert space, it is natural to use the ℓ^2-norm. For mixed states, the distance between density matrices can be measured using fidelity or trace distance [74]. In Chapter 2 we will use the diamond norm, which is actually a distance measure for quantum channels which implies an error bound on the states output by the channel. This is because the algorithm we present outputs a distribution of states generated by an algorithm using classical randomness.

The first breakthrough algorithm in Hamiltonian simulation appears in Ref. [67], which proposes using product formulas [89] to simulate local Hamiltonians, ie, Hamiltonians which can be expressed as a sum of other Hamiltonians each of which acts on a limited number of qubits. Moving beyond the local model of Hamiltonians, Ref. [4] proposed the first algorithm for sparse Hamiltonians, ie, Hamiltonians with a constant number, d, of efficiently computable non-zero entries in each row/column.

The sparse model of Hamiltonians makes use of two *oracles* for computing the locations and values of non-zero entries [15]. Formally, the entry location oracle acts as $\mathcal{O}_{A1}|j,l\rangle = |j,\nu(j,l)\rangle$ where $\nu(j,l)$ is the column index of the lth non-zero entry in the jth row. The entry value oracle acts as $\mathcal{O}_{A2}|j,k,z\rangle = |j,k,z+A_{j,k}\rangle$. In general we might only refer to a single query oracle $\mathcal{O}_A \equiv |0\rangle\langle 0|\otimes\mathcal{O}_{A1} + |1\rangle\langle 1|\otimes\mathcal{O}_{A2}$ which can implement $\mathcal{O}_{A1}$ and $\mathcal{O}_{A2}$ with only a single control qubit. The number of times an algorithm uses $\mathcal{O}_{A1}$ and $\mathcal{O}_{A2}$ is the *query complexity* of the algorithm and constitutes an efficiency measure of the algorithm. The *time complexity* of an algorithm can be taken to be (without loss of generality) the total number of 1 and 2-qubit gates (eg H, T, and $CNOT$) used in the circuit implementing it.

Since the pioneering Hamiltonian simulation algorithms of Refs. [4, 67], several improvements have been made in the asymptotic query and time complexity of these algorithms, specifically how they scale with the evolution time, the sparsity, and the error. Ref. [24] achieved linear scaling in t but polynomial scaling in $1/\varepsilon$, which was improved on by Refs. [13–15] which achieved $O(\log(1/\varepsilon))$ scaling. Ref.[13] uses the linear combination of unitaries (LCU) technique of Ref. [62] to approximate the Taylor series of $\exp(iHt)$. Ref. [15] also uses a LCU where the unitaries are quantum walk operators. Ref. [12] was the first algorithm to achieve linear scaling in the sparsity, $O(d)$, and also considered the complexity of simulating non-sparse Hamiltonians and implementing general unitaries (possibly non-sparse) given oracle access to their entries.

Although the previously mentioned algorithms scale optimally with respect to either t or ε individually, they mutually scaled multiplicatively (eg $\tilde{O}(t\log(1/\varepsilon))$) when additive scaling ($\tilde{O}(t+\log(1\varepsilon))$) is not ruled out by a no-go theorem. This additive scaling was achieved in Ref. [9], which builds on the quantum walk algorithm of Ref. [15], and in Refs. [68, 69], which provide algorithms in the block encoding model (qubitization or quantum signal processing (QSP)). In the block encoding model, one considers implementing Hamiltonians, H, which are subblocks of a unitary operator, eg, $U = \begin{bmatrix} H & A \\ B & C \end{bmatrix}$ where A, B, and C are only restricted by the fact that U is unitary. This is modeled using an oracle $\mathcal{O}_G$, which acts as $\mathcal{O}_G|0\rangle^m = |G\rangle$, and a $(n+m)\times(n+m)$ unitary U. Then the $n \times n$ Hamiltonian which is simulated can be expressed as $\langle G|U|G\rangle$ where the inner product is understood to be an isometric projection of U.

While the asymptotic complexity of an algorithm is important for designing efficient algorithms, the exact number of gates needed to implement the algorithm is what matters in practice. Ref. [27] empirically estimates the efficiency of various Hamiltonian simulation algorithms when applied to n-qubit Heisenberg spin chains ($H = \sum_{i=1}^{n} X_i \otimes X_{i+1 \mod n} + Y_i \otimes Y_{i+1 \mod n} + Z_i \otimes Z_{i+1 \mod n} + r_i Z_i$ with r_i randomly sampled from $[-1,1]$) for n ranging from ~ 10 to 100's of qubits for a fixed error of 10^{-3}. For this benchmark problem, although the QSP and Taylor series approaches have better asymptotic scaling than product formula algorithms, product formulas are more efficient in terms of the number of qubits, T-gates, and $CNOT$-gates. This kind of comparison is closest to how we numerically compare deterministic and randomized product formulas in Chapter 2. For a Hamiltonian $H = \sum_j H_j$, we measure the efficiency by counting the total number of times a unitary of the form $\exp(iH_j t)$ is implemented.

1.2 QLSAs

We state the quantum linear system problem as follows.

QUANTUM LINEAR SYSTEM PROBLEM (QLSP) (GENERAL STATEMENT): Given an oracle to produce a state $|b\rangle$ which is proportional to a vector $\vec{b}$, oracle access to the matrix A, and an error parameter ε, output a state ε-close to $|A^{-1}\vec{b}\rangle$, the state proportional to $A^{-1}\vec{b}$.

More generally if A is singular, we can consider outputing a state proportional to $A^{+}\vec{b}$

where A^+ is the Moore-Penrose pseudoinverse of A. In the following quantum algorithms this applies if $\vec{b}$ does not have any support on the null space of A.

The existing QLSAs all use Hamiltonian simulation as a subroutine. If the matrix A is Hermitian, then A is the Hamiltonian which is simulated. Otherwise, we can transform the linear system

$$A\vec{x} = \vec{b} \mapsto \begin{bmatrix} 0 & A \\ A^\dagger & 0 \end{bmatrix} \begin{bmatrix} \vec{x} \\ \vec{0} \end{bmatrix} = \begin{bmatrix} \vec{0} \\ \vec{b} \end{bmatrix} \tag{1.1}$$

so that we can assume without loss of generality that A is Hermitian (with the only overhead being a single qubit in the formalization of the problem).

Ref. [52] proposed the first QLSA which is now commonly refered to as HHL. It works by preparing $|b\rangle$, applying phase estimation (PE) [60] to A, then performing a controlled rotation on an ancilla qubit (controlled by the eigenvalue estimated by PE), and then post-selecting on that qubit being either $|0\rangle$ or $|1\rangle$ (with only one of these corresponding to success).

Ref. [52] also proved that estimating observables of solutions of well-conditioned linear systems is BQP-complete. They construct a linear system whose solution is of the form $\sum_t \alpha_t |t\rangle|\psi_t\rangle$ where t indexes the gates of a circuit (possibly identity gates) and $|\psi_t\rangle$ is the state after the tth gate has been applied. By measuring the t register and post-selecting on a successful measurement, they produce the state output from a quantum circuit, which can then be measured. This implies BQP-hardness. The completeness follows from the facts that the linear system is efficienctly solvable and that the post-selective measurement has high success probability.

The algorithm proposed in Ref. [52] had query complexity $O(\kappa^2)$ where κ is the condition number of A. This was improved to $O(\kappa)$ scaling using variable time amplitude amplification in Ref. [5]. This linear scaling is optimal as was proven in Ref. [52], unless $\mathsf{BQP} = \mathsf{PSPACE}$, since a sublinear QLSA could simulate quantum mechanics in sublinear time, violating the no-fast-forwarding principle [11]. Using the BQP-complete linear system problem, they also argue that any quantum algorithm for *estimating an entry* of $|A^{-1}\vec{b}\rangle$ can at best have $\mathrm{poly}(1/\varepsilon)$ scaling unless $\mathsf{BQP} = \mathsf{PP}$. This argument however does not rule out $\mathrm{poly}(\log(1/\varepsilon))$ algorithms that only produce the state $|A^{-1}\vec{b}\rangle$. Developing algorithms with this scaling is beneficial since they can be used as subroutines in algorithms for estimating functions of $|A^{-1}\vec{b}\rangle$; this reduction in the complexity of subroutines helps reduce the complexity of the entire algorithm.

This $\mathrm{poly}(\log(1/\varepsilon))$ scaling for producing $|A^{-1}\vec{b}\rangle$ is achieved in Refs. [25, 86], which also propose $O(\kappa)$ algorithms. Both algorithms of Ref. [25] use the LCU technique [62]. One approximates A^{-1} as a linear combination of $\exp(iAt_j)$ terms for various evolution times t_j (essentially as a truncated Fourier series for $1/x$). The other algorithm uses a linear combination of quantum walk operators, which naturally allow for a Chebyshev polynomial approximation of A^{-1}. The algorithm of Ref. [86] is inspired by adiabatic computation and only requires a single ancilla qubit (in addition to any ancilla used for Hamiltonian simulation subroutines).

Some of the previously mentioned QLSAs can also be applied to non-sparse matrices. For example, the Fourier algorithm Ref. [25] only requires the ability to simulate the Hamiltonian A with high accuracy but with no restrictions on the Hamiltonian model, so a non-sparse Hamiltonian simulation technique could simply be used as a subroutine. Likewise, the Chebyshev approach only requires the ability to perform a certain quantum walk which is not required to have sparse structure.

Ref. [96] considers inverting non-sparse matrices assuming access to oracles that output states proportional to the rows of A and a state whose amplitudes are proportional to the ℓ^2 norms of the rows. Their algorithm scales linearly with the Frobenius norm $||A||_F = \sum_{i,j}|A_{i,j}|^2$, which in the worst case is $O(\sqrt{n})$ for an $n \times n$ matrix, so their algorithm in general only provides a quadratic speed-up. Ref. [96] comments on using the non-sparse

Hamiltonian simulation algorithm of Ref. [12] as a subroutine in the HHL algorithm.

There is a very important caveat to make about QLSAs that also applies to the algorithms presented in this thesis: These algorithms 'solve' linear systems and ODEs in the sense that they output a quantum state proportional to the solution; they do not output the value of every entry of the solution. These states do not encode information about the size of the solution, but in the case of HHL the ℓ^2-norm is proportional to the success probability of the post-selection step.

This means that if we want to extract any useful information about the solution, we must make measurements on the state. Because of the additional costs incurred by post-processing, it is important to specify what kind of output is needed for applications of algorithms.

One of the first papers to consider the end-to-end application of a QLSA was Ref. [31] which considered estimating the radar scattering cross section of an object, specifically by estimating the fraction of energy scattered in a direction input to the algorithm. Although they don't give a complete analysis of the complexity of their algorithm, they illustrate the importance of considering how to estimate a function of interest and not just output quantum states.

Because this paper did not fully analyze the complexity of their algorithm, it was not clear what kind of quantum speed-up could be hoped for. Ref. [73] addressed this by considering the complexity of outputting functionals of the solutions of PDEs up to error ε. They phrase their results in terms of this error since the error of the output is what matters in practice (as opposed to, eg, the number of mesh points in a lattice, which are indirectly related to the error). They argue that for PDEs with a fixed number of dimensions, at most a polynomical speed-up is achievable. They also present a no-go argument that forbids estimating an observable of a PDE solution with $\mathrm{poly}(\log(1/\varepsilon))$ copies of the solution state, since doing so could distinguish between states that are distance δ apart using only

$\mathsf{poly}(\log(1/\delta))$ copies. As we already mentioned, we can at best hope for $\mathsf{poly}(1/\varepsilon)$ scaling for the most general post-processing problem unless $\mathsf{BQP} = \mathsf{PP}$ [52].

Another linear system based algorithm with a specific output appears in Ref. [94], which developed an algorithm for estimating effective resistance, but the computational hardness of doing so remained an open question. Although well-conditioned matrix inversion is BQP-complete [52], estimating effective resistance specifically requires solving a linear system where the matrix is a graph Laplacian and the input vector only has support on 2 entries. Subsequently, Ref. [36] presented a probabilistic log-space algorithm for Laplacian linear systems that may be time efficient depending on parameters such as the error tolerance; complementing this, Refs. [37, 38] studied BPL-completeness of approximating Laplacian eigenvalues. Motivated by these results, Appendix A argues for the BPP-hardness of inverting exponentially large Laplacian linear systems and corresponds to estimating a multiterminal effective resistance.

Ref. [55] developed a recommendation system algorithm. It takes as input a customer and a preference matrix indicating which customers prefer which goods, and it outputs a good which the input customer is likely to prefer. This algorithms is based on the singular value decomposition of the preference matrix, which is assumed to be low rank. At the time of its proposal, this algorithm performed faster than existing classical algorithms. Drawing inspiration from this quantum algorithm, Ref. [90] presented a classical algorithm for the same problem assuming access to the same data structure. One the one hand this is a perfect demonstration of the importance classical lower bounds for the purpose of proving quantum speed-ups; on the other hand, it is a success story about quantum approaches motivating classical methods. Ref. [48] applied similar techniques for sampling from the inverse of a low rank matrix.

1.3 Outline

The previous section reviewing Hamiltonian simulation is by no means comprehensive (eg it doesn't mention algorithms for quantum chemistry or quantum field theories) but is intended to give an idea of the figures of merit relevant to assessing our algorithms (time and query complexity, elementary exponentials). The previous section on QLSAs gives an overview of their development over the last decade. See Ref. [35] for a pedagogical introduction. In each subsequent chapter we review the literature relevant to the class of DEs addressed in that chapter.

Chapter 2 presents results for simulating Hamiltonians using randomized Lie-Suzuki-Trotter product formulas based on Ref. [28]. Deterministic product formulas produce coherent errors. By randomizing over product formulas we generate a distribution of states averaged over certain coeherent errors, which reduces the total error of the output state.

Chapter 3 presents an algorithm for simulating (solving the initial value problem) linear time-independent ODEs of the form $\frac{d\vec{x}}{dt} = A\vec{x} + \vec{b}$ based on Ref. [16]. This involves constructing a large linear system whose solution encodes $\vec{x}(t)$ for several values of t. After producing this state, a post-selective measurement is used to produce a state proportional to $\vec{x}(t)$ at the final time. The time evolution operator $\exp(At)$ is approximated using a truncated Taylor series, which has the advantage that the accuracy can scale exponentially in the truncation order. Analyzing this algorithm requires bounding the approximation errors, condition number of the linear system, and success probability of the post-selective measurement.

Chapter 4 presents algorithms for linear PDEs which are a component of a paper in preparation with Andrew Childs and Jin-Peng Liu. We use the finite difference method (FDM) to approximate Poisson's equation for $u(\vec{x})$ at $\mathcal{X}$, a finite set of points in the domain. This implies a linear system which we solve with QLSAs to output a state whose amplitudes encode $u(\vec{x})$ at $\mathcal{X}$. Analyzing this algorithm requires bounding FDM errors and bounding the condition number of the linear system.

Chapter 5 presents algorithms for the wave equation and Klein-Gordon equation based on Ref. [33]. We again apply the FDM and show how to transform a second order PDE into a first order ODE, which can then be solved using existing algorithms. We pay special attention to the case where the time evolution is simply Hamiltonian simulation, and we also discuss the role of QLSAs in state preparation and post-processing.

Chapter 6 concludes by discussing recent algorithmic progress, nonlinear differential equations (NLDEs), and open problems. The Appendices include ancillary material on linear systems and NLDEs.

1.4 Additional Research

This thesis covers the research I've done on differential equation algorithms; however, in my time at QuICS I've worked on other research projects, most of which falls in the intersection of graph theory and quantum information.

Carl Miller, Amir Kalev, and I studied extensions of the magic square and magic pentagram games to more general two-party binary constraint games on graphs [6]. We proved that the magic square and magic pentagram games are unique in that they are the only games in this class which certify that Alice and Bob share 2 Bell pairs (for the magic square) or 3 Bell pairs (for the magic pentagram).

On the experimental side, I did control theory for Chris Monroe's 'Gates Lab' (primarily working with Caroline Figgatt, Kevin Landsman, and Norbert Linke) which uses a blade trap to form an ion crystal with qubits encoded in electron levels and gates realized via Raman beams and electron-phonon coupling. We realized the first instance of parallel entangling gates between two pairs of ions as described in Ref. [43].

Chapter 2: Hamiltonian Simulation by Randomization

We first briefly review product formulas. We then present the randomization lemma and illustrate how it can be used in a simple case. We prove technical lemmas pertaining to how randomizing symmetrically reduces approximation error, and then we prove our main result. We present numerics projecting how our algorithm compares to normal product formulas for Heisenberg spin chains with hundreds of qubits, the benchmark system used in Ref. [27].

We restate the theorems of Ref. [28] verbatim, but for some results we give an intuitive sketch in lieu of a proof and refer to Ref. [28] or other papers for the technical details.

2.1 Product Formulas

Lie-Suzuki-Trotter product formulas were first applied to quantum algorithms in Ref. [67] which proposed simulating

$$\exp(-itH) = \exp(-it\sum_{j=1}^{L} H_j) \tag{2.1}$$

where each H_j can be individually simulated, by applying the unitary $(\prod_{j=1}^{L} \exp(-itH_j/r))^r$ for a large integer r. This approximation becomes exact in the limit $r \to \infty$. This is a *first order* formula in the sense that it is accurate to first order in t, ie, the error is $O(t^2)$.

Refs. [87–89] provided a recursive method formula for product formulas of arbitrarily high accuracy. We denote the kth order accurate formula by S_k, and they are defined by

$$S_1(\lambda) := \prod_{j=1}^{L} \exp(\lambda H_j) \tag{2.2}$$

$$S_2(\lambda) := \prod_{j=1}^{L} \exp(\lambda H_j/2) \prod_{j=L}^{1} \exp(\lambda H_j) \tag{2.3}$$

$$S_{2k}(\lambda) := S_{2k-2}(p_k\lambda)^2 S_{2k-2}((1-4p_k)\lambda) S_{2k-2}(p_k\lambda)^2 \tag{2.4}$$

where $p_k := 1/(4-4^{1/(2k-1)})$. In general we assume that the target Hamiltonian is $H = \sum_{j=1}^{L} H_j$ and use λ in place of $-it$ since these formulas apply generally for $\lambda \in \mathbb{C}$. We define the target operator $V(\lambda) := \exp(\lambda H)$. As in the first order case, $V(\lambda)$ is well approximated by applying $S_{2k}(\lambda/r)^r$. The error from these formulas was bounded in Ref. [27], which also provides bounds that take advantage of when some of the H_j commute pairwise.

2.2 Randomized Algorithms

During the writing of Ref. [28], we became aware of Ref. [99] which presents a randomized Hamiltonian simulation algorithm which is equivalent to the algorithm we present here for randomizing over first order product formulas to generate a second order accurate channel. Ref. [76] also proposes a random product formula algorithm for Hamiltonian simulation. This algorithm is different from ours in that it evolves Hamiltonians in a fixed order for random amounts of time whereas our algorithm evolves Hamiltonians in a random order for fixed amounts of time.

Ref. [20] proposes a randomized algorithm for approximate unitary synthesis where, instead of implementing a unitary using a fixed sequence of 1- and 2-qubit gates, randomized sequences are applied. Ref. [20] uses the following lemma, which is crucial for proving our results.

Lemma 1 ([20, 53])**.** *Let V be a target unitary, with associated channel $\mathcal{V}(\rho) = V\rho V^\dagger$. Let $a, b > 0$ and $\{U_1, U_2, \ldots, U_n\}$ be a set of unitaries such that*

1. *for all $j \in \{1, \ldots, n\}$ we have $||U_j - V|| \leq a$;*

2. *there exist positive numbers $\{p_j\}$ such that $\sum_{j=1}^{n} p_j = 1$ and $||(\sum_j p_j U_j) - V|| \leq b$.*

It follows that $\mathcal{E} = \sum_j p_j \mathcal{U}_j$ satisfies

$$||\mathcal{E} - \mathcal{V}||_\diamondsuit \leq a^2 + 2b. \tag{2.5}$$

The feature that makes this Lemma useful is that the channel $\mathcal{E}$ has error linear in the error of the average unitary but quadratic in the error of each U_j. Thus, even if the individual U_j are only $O(\sqrt{\varepsilon})$ close to V, as long as their average $\sum_j p_j U_j$ is $O(\varepsilon)$ close to V, the channel will be $O(\varepsilon)$ close to V.

2.3 The First Order Case

To see why randomization is useful, we will consider the simple case of a Hamiltonian with two terms, $H = H_1 + H_2$. The first order (deterministic) product formula for simulating H is

$$S_1(\lambda) = \exp(\lambda H_1)\exp(\lambda H_2) \tag{2.6}$$

$$= 1 + \lambda(H_1 + H_2) + \lambda^2\left(\frac{H_1^2}{2} + \frac{H_2^2}{2} + H_1 H_2\right) + O(\lambda^3) \tag{2.7}$$

which has $O((\Lambda|\lambda|)^2)$ error where $\Lambda = \max\{\|H_1\|, \|H_2\|\}$. We define its reverse

$$S_1^{\text{rev}}(\lambda) = \exp(\lambda H_2)\exp(\lambda H_1). \tag{2.8}$$

which also has $O((\Lambda|\lambda|)^2)$ error, so the error term corresponding to a in Lemma 1 is $O((\Lambda|\lambda|)^2)$. If we randomly apply either $S_1(\lambda)$ or $S_1^{\text{rev}}(\lambda)$, then the error term corresponding to b in Lemma 1 is

$$\|V(\lambda) - \frac{1}{2}(S_1(\lambda) + S_1^{\text{rev}}(\lambda))\| = O((\Lambda|\lambda|)^3). \tag{2.9}$$

This suggests that the channel corresponding to randomly applying $S_1(\lambda)$ and $S_1^{\text{rev}}(\lambda)$ should have $O((\Lambda|\lambda|)^3)$ error in the diamond norm, improving on the $O((\Lambda|\lambda|)^2)$ of just applying $S_1(\lambda)$ or $S_1^{\text{rev}}(\lambda)$.

The approach above only uses a single bit of randomness, but more generally we can subdivide the evolution time t into r segments of time t/r, and for each of these segments we apply $S_1(-it/r)$ and $S_1^{\text{rev}}(-it/r)$ randomly, requiring r bits of randomness. The error of this approach for Hamiltonians with more terms is characterized by the following theorem.

Theorem 1 (Randomized first-order error bound). *Let $\{H_j\}_{j=1}^L$ be Hermitian matrices. Let*

$$V(-it) := \exp\left(-it \sum_{j=1}^{L} H_j\right) \tag{2.10}$$

be the evolution induced by the Hamiltonian $H = \sum_{j=1}^{L} H_j$ for time $t \in \mathbb{R}$. Define

$$S_1(\lambda) := \prod_{j=1}^{L} \exp(\lambda H_j) \quad \textit{and} \quad S_1^{\text{rev}}(\lambda) := \prod_{j=L}^{1} \exp(\lambda H_j). \tag{2.11}$$

Let $r \in \mathbb{N}$ be a positive integer and $\Lambda := \max||H_j||$. Then

$$||\mathcal{V}(-it) - \frac{1}{2^r}(\mathcal{S}_1(-it/r) + \mathcal{S}_1^{\text{rev}}(-it/r))^r||_\diamond \le \frac{(\Lambda|t|L)^4}{r^3} \exp\left(2\frac{\Lambda|t|L}{r}\right) + \frac{2(\Lambda|t|L)^3}{3r^2} \exp\left(\frac{\Lambda|t|L}{r}\right) \tag{2.12}$$

where, for $\lambda = -it$, we associate channels $\mathcal{V}(\lambda)$, $\mathcal{S}_1(\lambda)$, and $\mathcal{S}_1^{\text{rev}}(\lambda)$ with the unitaries $V(\lambda)$, $S_1(\lambda)$, and $S_1^{\text{rev}}(\lambda)$, respectively.

The proof of this makes use of Lemma F.2 from Ref. [27].

Lemma 2 ([27])**.** *For any $x \in \mathbb{C}$ and $\kappa \in \mathbb{N}$, we have*

$$\left| \sum_{s=\kappa}^{\infty} \frac{x^s}{s!} \right| \leq \frac{|x|^\kappa}{\kappa!} \exp(|x|). \tag{2.13}$$

This lemma is used to bound the total error in Taylor expansions such as those appearing in Eqns. 2.6, 2.8, and 2.9 for $\lambda = -it/r$. Next the subadditivity of the diamond norm is applied for r applications of $\mathcal{V}(-it/r)$.

2.4 Randomized Operators

In our algorithm, the quantum channel simply consists of applying product formula unitaries for $H = \sum_j H_j$ with the order of the H_j terms rearranged. Generically such an operator takes the form

$$\begin{aligned}
&\exp(q_1 \lambda H_{\pi_1(1)}) \exp(q_1 \lambda H_{\pi_1(2)}) \cdots \exp(q_1 \lambda H_{\pi_1(L)}) \\
&\exp(q_2 \lambda H_{\pi_2(1)}) \exp(q_2 \lambda H_{\pi_2(2)}) \cdots \exp(q_2 \lambda H_{\pi_2(L)}) \\
&\vdots \\
&\exp(q_\kappa \lambda H_{\pi_\kappa(1)}) \exp(q_\kappa \lambda H_{\pi_\kappa(2)}) \cdots \exp(q_\kappa \lambda H_{\pi_\kappa(L)})
\end{aligned} \tag{2.14}$$

where the q_j are real numbers, and the π_j are permutations in $\mathrm{Sym}(L)$, the symmetric group on L items, over which the channel randomly samples permutations. We state a technical lemma for the average of this kind of operator over $\mathrm{Sym}(L)$, namely the average

$$\begin{aligned}\frac{1}{L!}\sum_{\sigma\in\mathrm{Sym(L)}} &\exp(q_1\lambda H_{\sigma(\pi_1(1))})\exp(q_1\lambda H_{\sigma(\pi_1(2))})\cdots\exp(q_1\lambda H_{\sigma(\pi_1(L))})\\ &\exp(q_2\lambda H_{\sigma(\pi_2(1))})\exp(q_2\lambda H_{\sigma(\pi_2(2))})\cdots\exp(q_2\lambda H_{\sigma(\pi_2(L))})\\ &\vdots\\ &\exp(q_\kappa\lambda H_{\sigma(\pi_\kappa(1))})\exp(q_\kappa\lambda H_{\sigma(\pi_\kappa(2))})\cdots\exp(q_\kappa\lambda H_{\sigma(\pi_\kappa(L))}).\end{aligned} \tag{2.15}$$

The order λ^s term in this operator's Taylor expansion contains products of s H_j's. We define it as a sum of a degenerate term and a nondegenerate term. The part of the order λ^s term where these H_j are pairwise distinct is known as the *nondegenerate* term; the remaining part (where at least one H_j appears twice in each contribution) is the *degenerate* term.

Lemma 3 (Randomization lemma)**.** *Define an average evolution operator as in Eqn. 2.15 and let $s \le L$ be a positive integer. The sth-order nondegenerate term of this operator is*

$$\frac{[(q_1+\cdots+q_\kappa)\lambda]^s}{s!}\sum_{\substack{m_1,\ldots,m_s\\ \textit{pairwise different}}} H_{m_1}\cdots H_{m_s}. \tag{2.16}$$

We refer the reader to Ref. [28] for the a line-by-line proof which amounts to counting all nondegenerate contributions to the order λ^s terms in Eqn. 2.15 and simplifying using the symmetrization over permutations.

When we apply this to analyze our algorithm, the symmetrization over permutations guarantees that there is no nondegenerate error term of order L or less. For the purposes of our algorithm, this lemma will be instantiated for a $2k$th order product formula.

2.5 Error

The following lemma bounds the contribution of low order degenerate terms to the operator $V(\lambda)$. This will in turn be used to bound how much these terms can contribute to the error of approximating $V(\lambda)$ by Eqn. 2.15.

Lemma 4. *Let $\{H_j\}_{j=1}^{L}$ be Hermitian operators with $\Lambda := \max_j ||H_j||$; let $q_1, \ldots, q_\kappa \in \mathbb{R}$ with $\max_k |q_k| \leq 1$; and let $s \leq L$ be a positive integer. Then the norm of the sth-order degenerate term of the ideal evolution operator $V(\lambda)$ as in Eqn. 2.1 is at most*

$$\frac{(\Lambda|\lambda|)^s}{s!}[L^s - L(L-1)\cdots(L-s+1)] \tag{2.17}$$

and the norm of the sth-order degenerate term of the average evolution operator as in Eqn. 2.15 is at most

$$\frac{(\kappa\Lambda|\lambda|)^s}{s!}[L^s - L(L-1)\cdots(L-s+1)]. \tag{2.18}$$

The basic idea of the proof is that there are L^s possible products of H_j's that appear in the order λ^s term of $V(\lambda)$. Of these, $L!/(L-s)!$ are nondegenerate and so can be ignored since this is a lemma bounding degenerate terms. Counting only the degenerate terms and applying the triangle inequality implies the stated bounds. We use this lemma to prove the following.

Lemma 5. *Let $\{H_j\}_{j=1}^{L}$ be Hermitian operators; let $\lambda \in \mathbb{C}$ and $k, s \in \mathbb{N}$. Define the target evolution $V(\lambda)$ as in Eqn. 2.1, and define the permuted $(2k)$th-order formula $S_{2k}^{\sigma}(\lambda)$ as in Eqn. 2.23. Then the sth-order error of the approximation*

$$V(\lambda) - \frac{1}{L!}\sum_{\sigma \in \mathrm{Sym}(L)} S_{2k}^{\sigma}(\lambda) \tag{2.19}$$

is at most

$$\begin{cases} 0 & 0 \le s \le 2k, \\ \frac{(2\cdot 5^{k-1}\Lambda|\lambda|)^s}{(s-2)!} L^{s-1} & s > 2k, \end{cases} \tag{2.20}$$

where $\Lambda := \max||H_j||$.

Proof. It is sufficient to prove the stronger bound

$$\begin{cases} 0 & 0 \le s \le 2k, \\ 2\frac{(2\cdot 5^{k-1}\Lambda|\lambda|)^s}{s!}[L^s - L(L-1)\cdots(L-s+1)] & 2k < s \le L, \\ 2\frac{(2\cdot 5^{k-1}\Lambda|\lambda|)^s}{s!} L^s & s > L. \end{cases} \tag{2.21}$$

The case where $0 \le s \le 2k$ follows from the fact that the $2k$th order product formula is accurate at the sth order.

The case $2k < s \le L$ follows from applying Lemma 4 to bound the norms of the degenerate terms in the exact evolution operator and the average operator (with $\kappa = 2 \cdot 5^{k-1}$ accounting for the periods of time evolution in the product formula operators being averaged over).

The proof for $s > L$ is similar to the proofs of Propositions F.3 and F.4 of Ref. [27]. □

2.6 Main Result

We will make use of Lemma F.2 from Ref. [27].

Theorem 2 (Randomized higher-order error bound)**.** *Let* $\{H_j\}_{j=1}^L$ *be Hermitian matrices. Let*

$$V(-it) := \exp\left(-it \sum_{j=1}^{L} H_j\right) \tag{2.22}$$

be the evolution induced by the Hamiltonian $H = \sum_{j=1}^{L} H_j$ *for time* t*. For any permutation*

$\sigma \in \mathrm{Sym}(L)$, define the permuted $(2k)$th-order formula recursively by

$$\begin{aligned} S_2^\sigma(\lambda) &:= \prod_{j=1}^{L} \exp\left(\frac{\lambda}{2} H_{\sigma(j)}\right) \prod_{j=L}^{1} \exp\left(\frac{\lambda}{2} H_{\sigma(j)}\right) \\ S_{2k}^\sigma(\lambda) &:= [S_{2k-2}^\sigma(p_k\lambda)]^2 S_{2k-2}^\sigma((1-4p_k)\lambda)[S_{2k-2}^\sigma(p_k\lambda)]^2, \end{aligned} \tag{2.23}$$

with $p_k := 1/(4-4^{1/(2k-1)})$ for $k > 1$. Let $r \in \mathbb{N}$ and $\Lambda := \max\|H_j\|$. Then

$$\begin{aligned} &\left\| \mathcal{V}(-it) - \left(\frac{1}{L!}\sum_{\sigma\in\mathrm{Sym}(L)} \mathcal{S}_{2k}^\sigma(-it/r)\right)^r \right\|_\diamond \\ &\quad \leq 4\frac{(2\cdot 5^{k-1}\Lambda|t|L)^{4k+2}}{((2k+1)!)^2 r^{4k+1}} \exp\left(4\cdot 5^{k-1}\frac{\Lambda|t|L}{r}\right) + 2\frac{(2\cdot 5^{k-1}\Lambda|t|)^{2k+1}L^{2k}}{(2k-1)!\,r^{2k}} \exp\left(2\cdot 5^{k-1}\frac{\Lambda|t|L}{r}\right) \end{aligned} \tag{2.24}$$

where, for $\lambda = -it$, we associate quantum channels $\mathcal{V}(\lambda)$ and $\mathcal{S}_{2k}^\sigma(\lambda)$ with the unitaries $V(\lambda)$ and $S_{2k}^\sigma(\lambda)$, respectively.

Proof. We prove the finally inequality with $-it$ replaced by a general $\lambda \in \mathbb{C}$. It arises from Lemma 1, so we must bound the errors associated with the average unitary and each unitary that is randomly applied.

The $2k$th order product formula matches all terms up to $O(\lambda^{2k})$, and more particularly the sth order term of $V(\lambda) - S_{2k}^\sigma(\lambda)$ is

$$\begin{cases} 0 & 0 \leq s \leq 2k, \\ \frac{2(2\cdot 5^{k-1}\Lambda|\lambda|)^s}{s!} L^s & s > 2k \end{cases} \tag{2.25}$$

By Lemma F.2 of Ref. [27] (restated here as Lemma 2) and analysis similar to Proposition F.3 of Ref. [27], we bound the error in each unitary as

$$||V(\lambda) - S_{2k}^\sigma(\lambda)|| \leq 2\frac{(2\cdot 5^{k-1}\Lambda|\lambda|L)^{2k+1}}{(2k+1)!} \exp(2\cdot 5^{k-1}\Lambda|\lambda|L). \tag{2.26}$$

This implies the first term on the right hand side of the main inequality.

To bound the error of the average unitary we apply, note that Lemma 5 implies the sth-order term of $V(\lambda) - \frac{1}{L!}\sum_{\sigma\in\mathrm{Sym}(L)} S_{2k}^{\sigma}(\lambda)$ is bounded by

$$\begin{cases} 0 & 0 \le s \le 2k, \\ \frac{(2\cdot 5^{k-1}\Lambda|\lambda|)^s}{(s-2)!}L^{s-1} & s > 2k, \end{cases} \tag{2.27}$$

so applying Lemma F.2 of Ref. [27] again we have

$$\left\| V(\lambda) - \frac{1}{L!}\sum_{\sigma\in\mathrm{Sym}(L)} S_{2k}^{\sigma}(\lambda) \right\| \le \frac{(2\cdot 5^{k-1}\Lambda|\lambda|)^{2k+1}L^{2k}}{(2k-1)!}\exp(2\cdot 5^{k-1}\Lambda|\lambda|L). \tag{2.28}$$

The theorem then follows from applying Lemma 1, setting $\lambda = -it/r$, and using the diamond norm's subadditivity

$$\left\| \mathcal{V}(-it) - \left(\frac{1}{L!}\sum_{\sigma\in\mathrm{Sym}(L)} \mathcal{S}_{2k}^{\sigma}(-it/r)\right)^r \right\|_{\diamond} \le r\left\| \mathcal{V}(-it/r) - \frac{1}{L!}\sum_{\sigma\in\mathrm{Sym}(L)} \mathcal{S}_{2k}^{\sigma}(-it/r) \right\|_{\diamond}. \tag{2.29}$$

□

2.7 Comparison of Algorithms

We compare deterministic and random product formulas by counting the number of *elementary exponentials*, ie, unitaries of the form $\exp(iH_jt)$ for any $t\in\mathbb{R}$. This is a different measure of efficiency from Ref. [27] which compares the efficiency of product formulas, Taylor series algorithms, and quantum signal processing in terms of the total number of elementary gates.

We use the same benchmark system as Ref. [27], a 1-D periodic Heisenberg spin chain

$$H = \sum_{j=1}^{n} \sigma_j^x \sigma_{j+1}^x + \sigma_j^y \sigma_{j+1}^y + \sigma_j^z \sigma_{j+1}^z + h_j \sigma_j^z \tag{2.30}$$

where the qubit index is $\mod n$ and h_j is chosen uniformly at random from $[-1,1]$ for different problem instances.

For the purpose of decomposing the Hamiltonian into smaller Hamiltonians corresponding to the elementary exponentials $\exp(iH_jt)$, we assume we can efficiently implement $\sigma_j^x \sigma_{j+1}^x, \sigma_j^y \sigma_{j+1}^y, \sigma_j^z \sigma_{j+1}^z$, and σ_j^z for all j. We fix the evolution time to be $t = n$ and the simulation precision to be $\varepsilon = 10^{-3}$. With these parameters fixed the only remaining one is n, the number of qubits, so we can compare algorithms based on how the number of elementary exponentials scales with n.

We compare these algorithms using both analytically provable and numerical estimates of the required number of elementary exponentials.

The analytical bounds of Ref. [27] for the deterministic (labelled det) product formulas take advantage of the fact that some of the terms in the Hamiltonian commute. Those for the random (labelled rand) product formulas are based on our main result (ee Ref. [28] for more details on the calculation). The number of elementary exponentials needed by the 2kth order formulas are $r_{2k}^{\text{det/rand}}$ and scale with n as below.

$$r_1^{\text{det}} = O(n^3) \qquad r_4^{\text{det}} = O(n^{2.4}) \qquad r_6^{\text{det}} = O(n^{2.28}) \tag{2.31}$$

$$r_1^{\text{rand}} = O(n^3) \qquad r_4^{\text{rand}} = O(n^{2.25}) \qquad r_6^{\text{rand}} = O(n^{2.17}) \tag{2.32}$$

Similar to Ref. [27] we empirically estimate the efficiency by extrapolating from simulations for around 5 instances of Heisenberg spin chains (sampling h_j again for each instance) on 6-10 qubits. We assume that the number of elementary gates scales as a power-law of the number of qubits and find the relevant constant and exponent to fit the data for 6-10 qubits. We use this to project for up to 100 qubits.

Ref. [27] gives empirical bounds for deterministic (labelled demp) product formulas. We make use of Lemma 7 of Ref. [15] which introduces a factor 2 to convert the bounds of Ref. [27] to the diamond norm. The random (labelled remp) product formulas show a significant improvement for the first order case but only marginal improvement at higher orders. The numbers of elementary exponentials scale as below.

$$r_1^{\text{demp}} = 4143n^{2.066} \qquad r_4^{\text{demp}} = 5.821n^{1.471} \qquad r_6^{\text{demp}} = 2.719n^{1.160} \tag{2.33}$$

$$r_1^{\text{remp}} = 300.0n^{1.806} \qquad r_4^{\text{remp}} = 5.458n^{1.439} \qquad r_6^{\text{remp}} = 2.804n^{1.152} \tag{2.34}$$

We plot the data for the empirical estimates on 6-10 qubits and display the power law line of fit. We also plot a comparison of analytical and empirical bounds projected for 10-100 qubits. From this second plot, it is apparent that the 6th order formulas (both deterministic and random) are more efficient than the other orders. This also shows that there is practically no advantage to be had by using randomized product formulas.

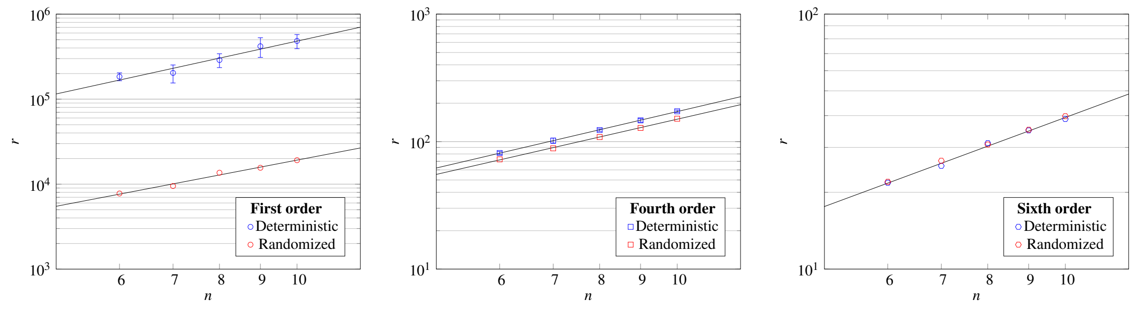

Figure 1: Numerical comparison and lines of fit for deterministic and random product formulas of the same order. Error bars (when included) account for error from sampling over a limited number of instances of Heisenberg spin models. Figure from Ref. [28] credit to Yuan Su.

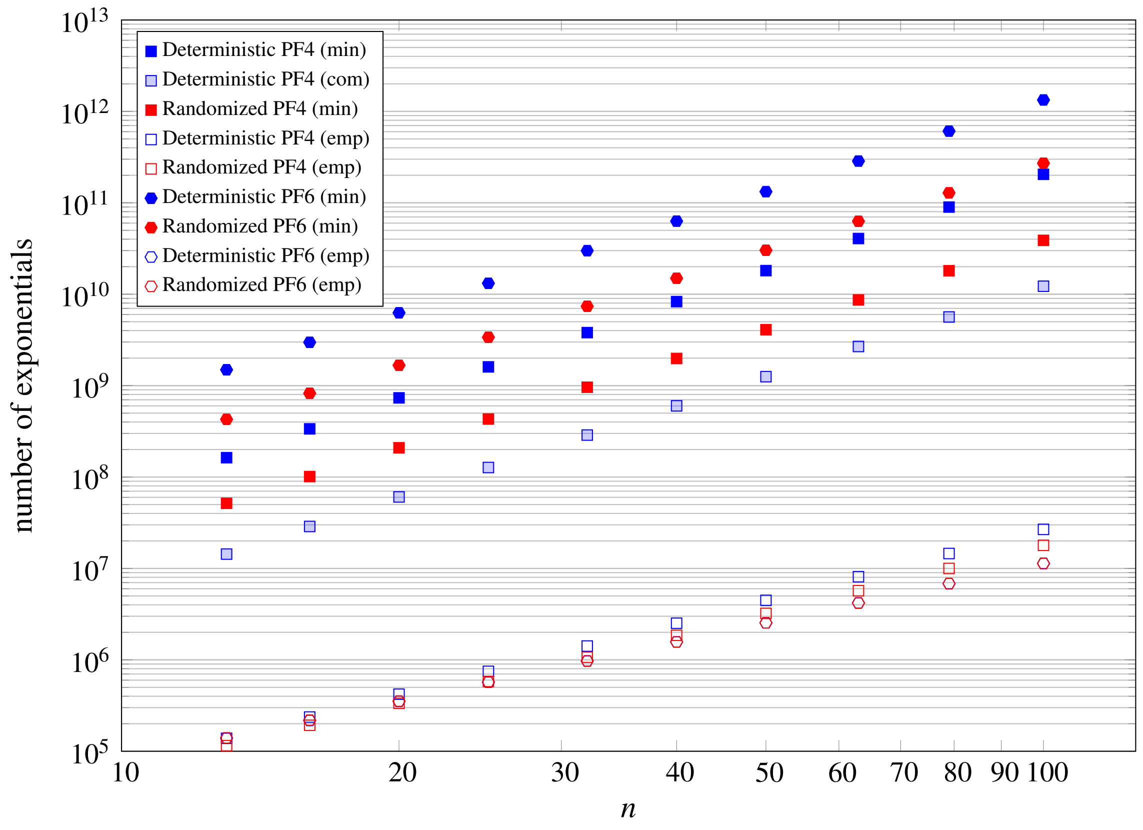

Figure 2: Elementary exponential counts for 4th/6th order deterministic/random product formulas. Solid markers indicate analytically proven bounds, and open markers indicate numerical estimates. The sixth order empirical data almost completely overlap. Figure from Ref. [28] credit to Yuan Su.

Chapter 3: Ordinary Differential Equations

The most general form of the time-independent Schrödinger's equation is

$$\frac{d}{dt}|\psi\rangle = -iH|\psi\rangle \tag{3.1}$$

where the only restriction on H is that it is Hermitian. This is a special case of the most general time-independent first-order linear ODE

$$\frac{d}{dt}\vec{x} = A\vec{x} + \vec{b} \tag{3.2}$$

where $\vec{b}$ can be any vector and A can be any invertible square matrix (this assumption implies the existence of a solution and is used in proving the efficiency of our algorithm). Given an initial condition, $\vec{x}(0)$, and an evolution time, t, the solution of this ODE is simply

$$\vec{x}(t) = \exp(At)\vec{x}(0) + (\exp(At) - I)A^{-1}\vec{b}. \tag{3.3}$$

When we defined the problem of Hamiltonian simulation, we were interested in outputting a quantum state that had undergone evolution, not outputting all the amplitudes of the quantum state. Likewise, QLSAs output quantum states proportional to solutions of linear systems. We define the problem of simulating an ODE in a similar manner.

QUANTUM ODE SIMULATION PROBLEM: Given a precision parameter ε, an evolution time T, and oracles, $\mathcal{O}_A, \mathcal{O}_b$, and $\mathcal{O}_x$, to compute the entries of A and to produce states proportional to $\vec{b}$ and $\vec{x}(0)$, produce a state ε-close to the state proportional to $\vec{x}(T)$.

The remainder of this chapter is based on Ref. [16]. First we briefly discuss the previous quantum algorithm for ODEs and how its assumptions compare to ours. We then outline the linear system which we solve with a QLSA and discuss the the post-selection of the state produced by the QLSA. Then we present a number of technical lemmas and theorems pertaining to the condition number of this linear system, the error due to the propagator approximation, the success probability associated with the post-selective measurement, and the complexity of the state preparation routine for the QLSA. We conclude with a statement of our main result.

Here we reproduce the main lemmas and theorems of Ref. [16] verbatim. Some of the proofs are rather technical, so in some cases we give an intuitive sketch of the proof and refer the reader to Ref. [16] for the complete analysis.

3.1 Previous Quantum Algorithms

The first quantum algorithm for solving linear ODEs was proposed by Berry in Ref. [10]. This algorithm uses linear multi-step methods (LMMs) (such as the Euler and Runge-Kutta methods) to approximate the time evolution. It encodes the multi-step method in a linear system $M\vec{y} = \vec{c}$ and uses a QLSA to produce a state proportional to $\vec{y}$. The linear system is constructed so that $\vec{y}$ encodes $\vec{x}(t)$ at several different values of $t \in [0, T]$ (what is known as a *history state*), so the QLSA is followed by a post-selection step which produces a state proportional to $\vec{x}(T)$. The algorithm of Ref. [10] has time and query complexities that are $O(\mathsf{poly}(1/\varepsilon))$.

This ODE algorithm was followed by improved algorithms for Hamiltonian simulation

[13, 14] which acheived $O(\text{poly}\log((1/\varepsilon)))$ scaling, naturally raising the question if the same scaling could be achieved for general ODEs. This question was answered in the positive by Ref. [16], which uses techniques similar in spirit to both the LMM algorithm for ODEs [10] and the truncated Taylor series algorithm for Hamiltonian simulation [13].

Similar to Ref. [10], we state our main result in terms of the sparse oracle model for A. However, both algorithm are amenable to non-sparse A provided that the Hamiltonian $\begin{bmatrix} 0 & A \\ A^\dagger & 0 \end{bmatrix}$ can be simulated using a non-sparse oracle model. Additionally, Ref. [10] assumed that $\vec{b}$ is sparse; however, we do not assume $\vec{b}$ is sparse and instead assume access to an oracle that prepares states proporitonal to $\vec{b}$. This is not a significant difference since the oracle for this paper could be realized with the sparse state preparation routine used in Ref. [10]. The algorithm of Ref. [10] would also work using an oracle to produce states proportional to $\vec{b}$, and stating that algorithm's complexity in terms of queries to $\mathcal{O}_b$ could only polynomially reduce the complexity's dependence on $\vec{b}$'s sparsity.

3.2 Linear Systems for ODEs

As in Ref. [10], we construct a large linear system whose solution is a history state; unlike Ref. [10], the time evolution under the propagator $\exp(At)$ is not realized with a LMM but instead by multiplying by a Taylor series approximation of $\exp(At)$ (as in Ref. [13]). The Taylor series approximation is better than the LMM approximation since the coefficients of the Taylor series of $\exp(z)$ decay as $\frac{1}{n!}$, so even a relatively low order truncation gives an accurate approximation. Specifically, we define the kth order truncated Taylor series of e^z as $T_k(z) := \sum_{j=0}^{k} \frac{z^j}{j!} \approx \exp(z)$. Approximating the propagator in this way also eliminates the need for additional hypotheses about A that are necessary for LMM algorithms to guarantee the stability of the particular linear multi-step formula being used. This Taylor series approximation is realized by inverting the following matrix.

Definition 3.1. *Let A be an $N \times N$ matrix, and let $m, k, p \in \mathbb{Z}^+$. Define*

$$
\begin{aligned}
C_{m,k,p}(A) := \sum_{j=0}^{d} |j\rangle\langle j| \otimes I - \sum_{i=0}^{m-1} \sum_{j=1}^{k} |i(k+1)+j\rangle\langle i(k+1)+j-1| \otimes A/j \\
- \sum_{i=0}^{m-1} \sum_{j=0}^{k} |(i+1)(k+1)\rangle\langle i(k+1)+j| \otimes I - \sum_{j=d-p+1}^{d} |j\rangle\langle j-1| \otimes I,
\end{aligned}
\tag{3.4}
$$

where $d := m(k+1)+p$, and I is the $N \times N$ identity matrix.

The full linear system we solve is

$$
C_{m,k,p}(Ah)|x\rangle = |0\rangle|x_{\text{in}}\rangle + h \sum_{i=0}^{m-1} |i(k+1)+1\rangle|b\rangle. \tag{3.5}
$$

For example, the linear system $C_{2,3,1}(Ah)|x\rangle = |0\rangle|x_{\text{in}}\rangle + h\sum_{i=0}^{1}|4i+1\rangle|b\rangle$ is

$$
\begin{bmatrix}
I &&&&&&&&& \\
-Ah & I &&&&&&&& \\
& -Ah/2 & I &&&&&&& \\
&& -Ah/3 & I &&&&&& \\
-I & -I & -I & -I & I &&&&& \\
&&&& -Ah & I &&&& \\
&&&&& -Ah/2 & I &&& \\
&&&&&& -Ah/3 & I && \\
&&&& -I & -I & -I & -I & I & \\
&&&&&&&& -I & I
\end{bmatrix}
|x\rangle =
\begin{bmatrix}
|x_{\text{in}}\rangle \\ h|b\rangle \\ 0 \\ 0 \\ 0 \\ h|b\rangle \\ 0 \\ 0 \\ 0 \\ 0
\end{bmatrix}
\tag{3.6}
$$

which has the solution

$$|x\rangle = \begin{bmatrix} |x_{\text{in}}\rangle \\ h|b\rangle + Ah|x_{\text{in}}\rangle \\ (Ah^2/2)|b\rangle + (A^2h^2/2)|x_{\text{in}}\rangle \\ (A^2h^3/3!)|b\rangle + (A^3h^3/3!)|x_{\text{in}}\rangle \\ A^{-1}(T_3(Ah)-I)|b\rangle + T_3(Ah)|x_{\text{in}}\rangle = |x_{1,0}\rangle \approx |x(h)\rangle \\ h|b\rangle + Ah|x_{1,0}\rangle \\ (Ah^2/2)|b\rangle + (A^2h^2/2)|x_{1,0}\rangle \\ (A^2h^3/3!)|b\rangle + (A^3h^3/3!)|x_{1,0}\rangle \\ A^{-1}(T_3(Ah)-I)|b\rangle + T_3(Ah)|x_{1,0}\rangle = |x_{2,0}\rangle \approx |x(2h)\rangle \\ |x_{2,0}\rangle \end{bmatrix}. \tag{3.7}$$

More generally, the solution to the linear system in Eqn. 3.5 is

$$|x\rangle = C_{m,k,p}(Ah)^{-1}\left[|0\rangle|x_{\text{in}}\rangle + h\sum_{i=0}^{m-1}|i(k+1)+1\rangle|b\rangle\right], \tag{3.8}$$

which can be written as

$$|x\rangle = \sum_{i=0}^{m-1}\sum_{j=0}^{k}|i(k+1)+j\rangle|x_{i,j}\rangle + \sum_{j=0}^{p}|m(k+1)+j\rangle|x_{m,j}\rangle \tag{3.9}$$

where the $|x_{i,j}\rangle$ are

$$|x_{0,0}\rangle = |x_{\text{in}}\rangle, \tag{3.10}$$

$$|x_{i,0}\rangle = \sum_{j=0}^{k}|x_{i-1,j}\rangle, \qquad 1 \le i \le m, \tag{3.11}$$

$$|x_{i,1}\rangle = Ah|x_{i,0}\rangle + h|b\rangle, \qquad 0 \le i < m, \tag{3.12}$$

$$|x_{i,j}\rangle = (Ah/j)|x_{i,j-1}\rangle, \qquad 0 \le i < m,\ 2 \le j \le k, \tag{3.13}$$

$$|x_{m,j}\rangle = |x_{m,j-1}\rangle, \qquad 1 \le j \le p. \tag{3.14}$$

From these equations, we obtain

$$
\begin{aligned}
|x_{0,0}\rangle &= |x_{\text{in}}\rangle, && && (3.15)\\
|x_{0,j}\rangle &= ((Ah)^j/j!)|x_{0,0}\rangle + ((Ah)^{j-1}/j!)h|b\rangle, && 1 \le j \le k, && (3.16)\\
|x_{1,0}\rangle &= T_k(Ah)|x_{0,0}\rangle + S_k(Ah)h|b\rangle && && \\
&\approx \exp Ah|x_{\text{in}}\rangle + (\exp Ah - I)A^{-1}|b\rangle, && && (3.17)\\
|x_{1,j}\rangle &= ((Ah)^j/j!)|x_{1,0}\rangle + ((Ah)^{j-1}/j!)h|b\rangle, && 1 \le j \le k, && (3.18)\\
|x_{2,0}\rangle &= T_k(Ah)|x_{1,0}\rangle + S_k(Ah)h|b\rangle && && \\
&\approx \exp 2Ah|x_{\text{in}}\rangle + (\exp 2Ah - I)A^{-1}|b\rangle, && && (3.19)\\
&\vdots && && \\
|x_{m-1,0}\rangle &= T_k(Ah)|x_{m-2,0}\rangle + S_k(Ah)h|b\rangle && && \\
&\approx \exp Ah(m-1)|x_{\text{in}}\rangle + (\exp Ah(m-1) - I)A^{-1}|b\rangle, && && (3.20)\\
|x_{m-1,j}\rangle &= ((Ah)^j/j!)|x_{m-1,0}\rangle + ((Ah)^{j-1}/j!)h|b\rangle, && 1 \le j \le k, && (3.21)\\
|x_{m,0}\rangle &= T_k(Ah)|x_{m-1,0}\rangle + S_k(Ah)h|b\rangle && && \\
&\approx \exp Ahm|x_{\text{in}}\rangle + (\exp Ahm - I)A^{-1}|b\rangle, && && (3.22)\\
|x_{m,j}\rangle &= |x_{m,0}\rangle && && \\
&\approx \exp Ahm|x_{\text{in}}\rangle + (\exp Ahm - I)A^{-1}|b\rangle, && 1 \le j \le p. && (3.23)
\end{aligned}
$$

As in Ref. [10], the solution of this linear system is a history state which encodes $\vec{x}(t)$ at several intermediate times. Only the $|x_{i,j}\rangle$ with $i = m, 1 \le j \le p$ correspond to approximations of $\vec{x}(T)/||\vec{x}(T)||$, so once we produce the state in Eqn. 3.9 we post-select on the first register being $m(k+1)$ or more. We use amplitude amplification to raise the success probability of this measurement to $\Theta(1)$.

3.3 Condition Number

In this section we present a number of lemmas used to bound the condition number of $C_{m,k,p}(A)$ under the assumptions A is diagonalizable and has eigenvalues with non-positive real part.

Lemma 6. *Let $\lambda \in \mathbb{C}$ such that $|\lambda| \leq 1$ and $\mathrm{Re}(\lambda) \leq 0$. Let $m, k, p \in \mathbb{Z}^+$ such that $k \geq 5$ and $(k+1)! \geq 2m$, and let $d = m(k+1)+p$. Then for any $n, l \in \{0, 1, \ldots, d\}$,*

$$\|C_{m,k,p}(\lambda)^{-1}|l\rangle\| \leq \sqrt{1.04 e I_0(2)(m+p)} \tag{3.24}$$

with $I_0(2) < 2.28$ a modified Bessel function of the first kind, and

$$|\langle n|C_{m,k,p}(\lambda)^{-1}|l\rangle| \leq \sqrt{1.04e}. \tag{3.25}$$

Proof. The proof of this lemma is rather extensive. For concision's sake, we give an intuitive explanation of the first inequality, which is the one we use in the next lemma. We refer the reader to Ref. [16] for the full proof.

In the same way Eqn. 3.9 can be constructed as the solution of Eqn. 3.5, we can construct the solution of $C_{m,k,p}(\lambda)|x\rangle = |l\rangle$.

When $l > m(k+1)$ the solution has its first $l-1$ amplitudes equal to 0 and the rest are all 1, and this solution has norm $\leq \sqrt{p}$.

When $0 \leq l \leq m(k+1)$ then the solution takes a form similar to (although not exactly) a history state such as Eqn. 3.9. This solution is close to the history state for the dynamical system obeying $x(t) = 0$ for $t \in [0, \lfloor l/(k+1) \rfloor h]$ and $\dfrac{dx}{dt} = \lambda x$ for later times with a $O(1)$ initial condition. Since λ has non-positive real part, the solution $x(t)$ won't grow in time, in which case the history state has at most m amplitudes encoding $x(t)$ at times between

$\lfloor l/(k+1) \rfloor h$ and T, the final time. Additionally there will be at most p amplitudes encoding copies of $x(T)$. This implies the $O(\sqrt{m+p})$ scaling in the first inequality. $\square$

Lemma 7. *Let $A = VDV^{-1}$ be a diagonalizable matrix, where $D = \operatorname{diag}(\lambda_0, \lambda_1, \dots, \lambda_{N-1})$ satisfies $|\lambda_i| \leq 1$ and $\operatorname{Re}(\lambda_i) \leq 0$ for $i \in \{0, 1, \dots, N-1\}$. Let $m, k, p \in \mathbb{Z}^+$ such that $k \geq 5$ and $(k+1)! \geq 2m$. Then*

$$||C_{m,k,p}(A)^{-1}|| \leq 3\kappa_V \sqrt{k}(m+p), \tag{3.26}$$

where $\kappa_V = \|V\| \cdot \|V^{-1}\|$ is the condition number of V.

Proof. Diagonalize $C_{m,k,p}(A)$ as $C_{m,k,p}(A) = \bar{V} C(D) \bar{V}^{-1}$ with $\bar{V} = \sum_{j=1}^{d} |j\rangle\langle j| \otimes V$ which has condition number $\kappa_{\bar{V}} = \kappa_V$, so

$$\begin{aligned} ||C_{m,k,p}(A)|| &\leq ||\bar{V}|| \cdot ||C(D)|| \cdot ||\bar{V}^{-1}|| &(3.27)\\ &\leq \kappa_V ||C(D)||. &(3.28) \end{aligned}$$

By definition $||C(D)^{-1}|| = \max_{|\psi\rangle} ||C(D)^{-1}|\psi\rangle|| / |||\psi\rangle||$ for $|\psi\rangle \in \mathbb{C}^{(d+1)N}$ which we can write as $|\psi\rangle = \sum_{l=0}^{d} |l\rangle |\psi_l\rangle = \sum_{l=0}^{d} \psi_{j,l} |l\rangle |j\rangle$. Then we have (by Cauchy-Schwarz and other simple inequalities)

$$\begin{aligned} ||C(D)^{-1}|\psi\rangle||^2 &= ||\sum_{l=0}^{d} C(D)^{-1} |l\rangle |\psi_l\rangle||^2 \\ &\leq (d+1) \sum_{l=0}^{d} ||C(D)^{-1} |l\rangle |\psi_l\rangle||^2. &(3.29) \end{aligned}$$

Now using the fact that D is diagonal we have

$$\begin{aligned}||C(D)^{-1}|l\rangle|\psi_l\rangle||^2 &= ||\sum_{j=0}^{N-1}\psi_{j,l}C(\lambda_j)^{-1}|l\rangle|j\rangle||^2\\ &= \sum_{j=0}^{N-1}|\psi_{j,l}|^2||C(\lambda_j)^{-1}|l\rangle||^2\\ &\leq 1.04eI_0(2)(m+p)\sum_{j=0}^{N-1}|\psi_{j,l}|^2\\ &= 1.04eI_0(2)(m+p)|||\psi_l\rangle||^2. \qquad (3.30)\end{aligned}$$

which implies

$$\begin{aligned}||C(D)^{-1}|\psi\rangle||^2 &\leq 1.04eI_0(2)(d+1)(m+p)\sum_{l=0}^{d}|||\psi_l\rangle||^2\\ &= 1.04eI_0(2)(m(k+1)+p+1)(m+p)|||\psi\rangle||^2\\ &\leq \frac{6}{5}\times 1.04eI_0(2)k(m+p)^2|||\psi\rangle||^2. \qquad (3.31)\end{aligned}$$

This suffices to prove the theorem. □

Lemma 8. *Let A be an $N\times N$ matrix such that $\|A\|\leq 1$. Let $m,k,p\in\mathbb{Z}^+$, and $k\geq 5$. Then*

$$\|C_{m,k,p}(A)\|\leq 2\sqrt{k}. \qquad (3.32)$$

Proof. The matrix $C_{m,k,p}(A)$ is the sum of the matrices

$$C_1 = \sum_{j=0}^{d} |j\rangle\langle j| \otimes I \tag{3.33}$$

$$C_2 = -\sum_{i=0}^{m-1}\sum_{j=0}^{k} |(i+1)(k+1)\rangle\langle i(k+1)+j| \otimes I \tag{3.34}$$

$$C_3 = -\sum_{i=0}^{m-1}\sum_{j=1}^{k} |i(k+1)+j\rangle\langle i(k+1)+j-1| \otimes A/j - \sum_{j=d-p+1}^{d} |j\rangle\langle j+1| \otimes I \tag{3.35}$$

with $d = m(k+1)+p$. Since C_1 is just the identity matrix, $||C_1|| = 1$. The matrix C_2 has a block structure with $k+1$ identity matrix blocks in select block rows, which implies $||C_2|| = \sqrt{k+1}$. The matrix C_3 contains the blocks on the first subdiagonal relevant to the Taylor series approximation $T_k(A)$ and relevant to how much of the history state encodes $\vec{x}(T)/||\vec{x}(T)||$. We have $||C_3|| = \max\{1, ||A||\} = 1$. Thus we have $||C_{m,k,p}(A)|| \leq 2 + \sqrt{k+1} \leq 2\sqrt{k}$. $\square$

Our bounds on $||C_{m,k,p}(A)||$ and $||C_{m,k,p}(A)^{-1}||$ imply the following condition number bound.

Theorem 3. *Let $A = VDV^{-1}$ be a diagonalizable matrix such that $\|A\| \leq 1$, $D = \mathrm{diag}(\lambda_0, \lambda_1, \ldots, \lambda_{N-1})$ and $\mathrm{Re}(\lambda_i) \leq 0$, for $i \in \{0, 1, \ldots, N-1\}$. Let $m, k, p \in \mathbb{Z}^+$ such that $k \geq 5$ and $(k+1)! \geq 2m$. Let $C := C_{m,k,p}(A)$, and let $\kappa_C = \|C\| \cdot \|C^{-1}\|$ be the condition number of C. Then*

$$\kappa_C \leq 6\kappa_V k(m+p), \tag{3.36}$$

where $\kappa_V = \|V\| \cdot \|V^{-1}\|$ is the condition number of V.

3.4 Approximation Error

Theorem 4. *Let $A = VDV^{-1}$ be a diagonalizable matrix, where $D = \mathrm{diag}(\lambda_0, \lambda_1, \ldots, \lambda_{N-1})$ satisfies $\mathrm{Re}(\lambda_i) \le 0$ for $i \in \{0, 1, \ldots, N-1\}$. Let $h \in \mathbb{R}^+$ such that $\|Ah\| \le 1$. Let $|x_{\mathrm{in}}\rangle, |b\rangle \in \mathbb{C}^N$, and let $|x(t)\rangle$ be defined by Eq. (3.3). Let $m, k, p \in \mathbb{Z}^+$ such that $k \ge 5$ and $(k+1)! \ge 2m$. Let $|x_{i,j}\rangle$ be defined by Eqs. (3.8) and (3.9). Then for any $j \in \{0, 1, \ldots, m\}$,*

$$\| |x(jh)\rangle - |x_{j,0}\rangle \| \le 2.8 \kappa_V j (\| |x_{\mathrm{in}}\rangle \| + mh \| |b\rangle \|)/(k+1)!, \tag{3.37}$$

where $\kappa_V = \|V\| \cdot \|V^{-1}\|$ is the condition number of V.

Proof. The solution of the differential equation $|x(jh)\rangle$ satisfies

$$|x((j+1)h)\rangle = \exp Ah |x(jh)\rangle + (\exp Ah - I)A^{-1}|b\rangle, \tag{3.38}$$

whereas $|x_{j,0}\rangle$ in Eqn. 3.9 satisfies

$$|x_{j+1,0}\rangle = T_k(Ah)|x_{j,0}\rangle + S_k(Ah)h|b\rangle \tag{3.39}$$

both with the initial condition $|x(0)\rangle = |x_{0,0}\rangle = |x_{\mathrm{in}}\rangle$. Since A is diagonal, we transform to its eigenbasis defining $|y(t)\rangle = V^{-1}|x(t)\rangle$, $|y_{i,j}\rangle = V^{-1}|x_{i,j}\rangle$, and $|c\rangle = V^{-1}|b\rangle$ which satisfy

$$|y((j+1)h)\rangle = \exp Dh |y(jh)\rangle + (\exp Dh - I)D^{-1}|c\rangle \tag{3.40}$$

$$|y_{j+1,0}\rangle = T_k(Dh)|y_{j,0}\rangle + S_k(Dh)h|c\rangle. \tag{3.41}$$

Since $||Dh||< 1$, technical lemmas in the appendix of Ref. [16] imply

$$||\exp Dh - T_k(Dh)|| \leq 1/(k+1)! \tag{3.42}$$

$$||S_k(Dh) - (\exp Dh - I)D^{-1}h^{-1}|| \leq 1/(k+1)!\,. \tag{3.43}$$

These are used to bound the error $\delta_j = |||y(jh)\rangle - |y_{j,0}\rangle||$ as

$$\delta_j \leq \frac{j}{(k+1)!} \max_{0\leq i\leq j} |||y_{i,0}\rangle|| + h|||c\rangle||. \tag{3.44}$$

So we must bound $|||y_{i,0}\rangle||$. For any $i \in \{0, 1, \ldots, m\}$ we have

$$|y_{i,0}\rangle = \langle i(k+1)|C_{m,k,p}(D)^{-1}|z\rangle, \tag{3.45}$$

where $\langle i(k+1)|$ acts as an isometry projecting out the first register, and

$$|z\rangle = |0\rangle|y_{\text{in}}\rangle + h\sum_{j=0}^{m-1} |j(k+1)+1\rangle|c\rangle. \tag{3.46}$$

By applying the triangle inequality and Lemma 6, we arrive at the bound

$$|||y_{i,0}\rangle|| \leq \sqrt{1.04e}(|||y_{\text{in}}\rangle|| + mh|||c\rangle||). \tag{3.47}$$

This implies a bound on δ_j. When that bound is taken into account along with the basis change from x to y coordinates (introducing a factor of κ_V), we arrive at the stated bound.

$\square$

3.5 Success Probability

Theorem 5. *Let $A = VDV^{-1}$ be a diagonalizable matrix, where $D = \mathrm{diag}(\lambda_0, \lambda_1, \ldots, \lambda_{N-1})$ satisfies $\mathrm{Re}(\lambda_i) \leq 0$ for $i \in \{0, 1, \ldots, N-1\}$. Let $h \in \mathbb{R}^+$ such that $\|Ah\| \leq 1$. Let $|x_{\mathrm{in}}\rangle, |b\rangle \in \mathbb{C}^N$, and let $|x(t)\rangle$ be defined by Eq. (3.3). Let $m, k, p \in \mathbb{Z}^+$ such that $(k+1)! \geq 70\kappa_V m(\||x_{\mathrm{in}}\rangle\| + mh\||b\rangle\|)/\||x(mh)\rangle\|$, where $\kappa_V = \|V\| \cdot \|V^{-1}\|$ is the condition number of V. Let $g = \max_{t\in[0,mh]} \||x(t)\rangle\|/\||x(mh)\rangle\|$. Let $|x\rangle$ be defined by Eq. (3.8) and let $|x_{i,j}\rangle$ be defined by Eq. (3.9). Then for any $j \in \{0, 1, \ldots, p\}$,*

$$\frac{\||x_{m,j}\rangle\|}{\||x\rangle\|} \geq \frac{1}{\sqrt{p + 77mg^2}}. \tag{3.48}$$

Proof. Note that all $|x_{m,j}\rangle$ are equal for $j \in \{0, 1 \ldots p\}$ so we can take $j = 0$. Define

$$|x_{\mathrm{good}}\rangle = \sum_{j=0}^{p} |m(k+1)+j\rangle |x_{m,j}\rangle = (\sum_{j=0}^{p} |m(k+1)+j\rangle)|x_{m,0}\rangle \tag{3.49}$$

$$|x_{\mathrm{bad}}\rangle = \sum_{i=0}^{m-1} \sum_{j=0}^{k} |i(k+1)+j\rangle |x_{i,j}\rangle. \tag{3.50}$$

So $|x\rangle = |x_{\mathrm{good}}\rangle + |x_{\mathrm{bad}}\rangle$ and

$$\begin{aligned} |||x\rangle||^2 &= |||x_{\mathrm{good}}\rangle||^2 + |||x_{\mathrm{bad}}\rangle||^2 \\ &= (p+1)|||x_{m,0}\rangle||^2 + |||x_{\mathrm{bad}}\rangle||^2. \end{aligned} \tag{3.51}$$

Next we lower bound $|||x_{m,0}\rangle||$ and upper bound $|||x_{\mathrm{bad}}\rangle||$. Let $q = |||x(mh)\rangle||$. Using

Thm. 4, the definition of g, and the choice of k, we find

$$|||x_{i,0}\rangle|| \leq 1.04gq, \qquad 0 \leq i \leq m-1 \tag{3.52}$$

$$0.96q \leq |||x_{m,0}\rangle|| \leq 1.04q. \tag{3.53}$$

Likewise, we can bound $|||x_{i,j}\rangle||$ as

$$|||x_{i,j}\rangle|| \leq \frac{2.08gq}{j!\,(3-e)}, \qquad 0 \leq i \leq m-1,\ 1 \leq j \leq k. \tag{3.54}$$

These bounds imply

$$|||x_{\text{bad}}\rangle||^2 \leq 70.9mg^2q^2 \tag{3.55}$$

so we have

$$\frac{|||x_{m,0}\rangle||^2}{|||x\rangle||^2} \geq \frac{(0.96q)^2}{p(0.96q)^2+70.9mg^2q^2} \geq \frac{1}{p+77mg^2}. \tag{3.56}$$

□

This implies that setting $p = m$ gives a $\Theta(1/g^2)$ success probability, which can be boosted to $\Theta(1)$ using amplitude amplification with $O(g)$ repetitions of of the QLSA.

3.6 State Preparation

Lemma 9. *Let $\mathcal{O}_x$ be a unitary that maps $|1\rangle|\phi\rangle$ to $|1\rangle|\phi\rangle$ for any $|\phi\rangle$ and maps $|0\rangle|0\rangle$ to $|0\rangle|\bar{x}_{\text{in}}\rangle$, where $\bar{x}_{\text{in}} = \vec{x}_{\text{in}}/||\vec{x}_{\text{in}}||$. Let $\mathcal{O}_b$ be a unitary that maps $|0\rangle|\phi\rangle$ to $|0\rangle|\phi\rangle$ for any $|\phi\rangle$ and maps $|1\rangle|0\rangle$ to $|1\rangle|\bar{b}\rangle$, where $\bar{b} = \vec{b}/||\vec{b}||$. Suppose we know $||\vec{x}_{\text{in}}||$ and $||\vec{b}||$. Then the state proportional to*

$$|0\rangle|x_{\text{in}}\rangle + h\sum_{i=0}^{m-1}|i(k+1)+1\rangle|b\rangle \tag{3.57}$$

can be produced with a constant number of calls to $\mathcal{O}_x$ and $\mathcal{O}_b$, and $\mathsf{poly}(\log(mk))$ *elementary gates.*

Proof. Start with $|0\rangle|0\rangle$ where the first register lies in $\mathbb{C}^d$. Apply the following unitary to the first register

$$\begin{aligned}U = &\left(\frac{||\vec{x}_{\text{in}}||}{\sqrt{||\vec{x}_{\text{in}}||^2+mh^2||\vec{b}||^2}}|0\rangle + \frac{\sqrt{m}h||\vec{b}||}{\sqrt{||\vec{x}_{\text{in}}||^2+mh^2||\vec{b}||^2}}|1\rangle\right)\langle 0| \\ &+ \left(\frac{||\vec{x}_{\text{in}}||}{\sqrt{||\vec{x}_{\text{in}}||^2+mh^2||\vec{b}||^2}}|1\rangle - \frac{\sqrt{m}h||\vec{b}||}{\sqrt{||\vec{x}_{\text{in}}||^2+mh^2||\vec{b}||^2}}|0\rangle\right)\langle 1| \\ &+ \sum_{j=2}^{d}|j\rangle\langle j|. \end{aligned} \tag{3.58}$$

Then apply the oracles $\mathcal{O}_x$ and $\mathcal{O}_b$ to produce the state

$$|\psi'\rangle = \frac{1}{\sqrt{||\vec{x}_{\text{in}}||^2+mh^2||\vec{b}||^2}}\left(|0\rangle|x_{\text{in}}\rangle + \sqrt{m}h|1\rangle|b\rangle\right). \tag{3.59}$$

Now apply a unitary mapping $|0\rangle$ to $|0\rangle$ and $|1\rangle$ to $\frac{1}{\sqrt{m}}\sum_{j=0}^{m-1}|j(k+1)+1\rangle$ on the first register to produce the state needed for the QLSA. This can be done with $\mathsf{poly}(\log(mk))$ elementary gates. □

3.7 Main Result

Theorem 6. *Suppose $A = VDV^{-1}$ is an $N \times N$ diagonalizable matrix, where $D = \operatorname{diag}(\lambda_0, \lambda_1, \ldots, \lambda_{N-1})$ satisfies* $\operatorname{Re}(\lambda_j) \le 0$ *for any $j \in \{0, 1, \ldots, N-1\}$. In addition, suppose A has at most s nonzero entries in any row and column, and we have an oracle $\mathcal{O}_A$ that computes these entries. Suppose $\vec{x}_{\text{in}}$ and $\vec{b}$ are N-dimensional vectors with known* *norms and that we have two controlled oracles, $\mathcal{O}_x$ and $\mathcal{O}_b$, that prepare the states proportional to $\vec{x}_{\text{in}}$ and $\vec{b}$, respectively. Let $\vec{x}$ evolve according to the differential equation*

$$\frac{d\vec{x}}{dt} = A\vec{x} + \vec{b} \tag{3.60}$$

with the initial condition $\vec{x}(0) = \vec{x}_{\text{in}}$. Let $T > 0$ and

$$g := \max_{t \in [0,T]} \|\vec{x}(t)\| / \|\vec{x}(T)\|. \tag{3.61}$$

Then there exists a quantum algorithm that produces a state ε-close to $\vec{x}(T)/\|\vec{x}(T)\|$ in ℓ^2 norm, succeeding with probability $\Omega(1)$, with a flag indicating success, using

$$O(\kappa_V sgT\|A\| \cdot \mathsf{poly}(\log(\kappa_V sg\beta T\|A\|/\varepsilon))) \tag{3.62}$$

queries to $\mathcal{O}_A$, $\mathcal{O}_x$, and $\mathcal{O}_b$, where $\kappa_V = \|V\| \cdot \|V^{-1}\|$ is the condition number of V and $\beta = (\||x_{\text{in}}\rangle\| + T\||b\rangle\|)/\||x(T)\rangle\|$. The gate complexity of this algorithm is larger than its query complexity by a factor of $\mathsf{poly}(\log(\kappa_V sg\beta T\|A\|N/\varepsilon))$.

Proof. Fix the parameters $h = T/\lceil T\|A\|\rceil$, $m = p = T/h = \lceil T\|A\|\rceil$, $\delta = \varepsilon/(25\sqrt{m}g)$, $\varepsilon \leq 1/2$, and $k = \left\lfloor \frac{2\log(\Omega)}{\log(\log(\Omega))} \right\rfloor$ where $\Omega = 70g\kappa_V m^{3/2}(\||x_{\text{in}}\rangle\| + T\||b\rangle\|)/(\varepsilon\||x(T)\rangle\|)$. Let $d = m(k+1) + p$

Consider applying the QLSA from Ref. [25] to solve Eqn. 3.5. If we measure the first register, we are successful and measure a state in $S = \{m(k+1) \dots m(k+1) + p\}$ with probability $\Omega(1/g^2)$, which we boost with amplitude amplification making $O(g)$ uses of the QLSA. Additionally, the post-selected state has error at most ε. The details of the proof of correctness can be found in Ref. [16].

The matrix $C_{m,k,p}(A)$ is $(d+1)N \times (d+1)N$ with $O(ks)$ sparsity and a condition number that is $O(\kappa_V km)$. Using the sparse QLSA of Ref. [25] with $O(g)$ steps of amplitude amplification as described gives the result stated in the theorem. $\square$

This expression for the complexity has some quantities that are unique to ODE algorithms and do not appear in the complexities of Hamiltonian simulation algorithms.

The linear scaling with g results from the fact that we use a history state and have a post-selective measurement. If the ODE is simply a case of Hamiltonian simulation then unitarity guarantees that $g = 1$. Under the assumption $\mathsf{NP} \not\subseteq \mathsf{BQP}$ we can rule out the possibility of algorithms running in time $\mathsf{poly}(\log(g))$ since otherwise we would be able to solve NP-hard problems by solving $\frac{d\vec{x}}{dt} = -C\vec{x}$ with $C_{i,j} = \delta_{i,j}(1-c(x_i))$ where x_i is the binary encoding of i and $c(x_i)$ outputs 1(0) if it satisfies (or does not) a $3-\mathsf{SAT}$ clause. An even stronger argument is that producing a state that has decayed to be exponentially small would allow us to post-select on exponentially small amplitudes, which would impliy $\mathsf{BQP} = \mathsf{PP}$ by Theorem 3.4 of Ref. [1].

Similar to g, the quantity β is related to the norms of the states as well as the inhomogeneity, but it is less concerning since the complexity scales as $\mathsf{poly}(\log(\beta))$.

The condition number κ_V appears since we assume that $A = VDV^{-1}$ is diagonalizable and then use this to bound the condition number of the $C_{m,k,p}(A)$. This factor is 1 for Hamiltonian simulation since Hamiltonians are diagonalized by unitaries which have condition numbers of 1.

This algorithm scales optimally with respect to $\|A\|T$ since Hamiltonian simulation is a special case, and the no-fast-forwarding theorem [11] prohibits sublinear scaling in $\|A\|T$.

This algorithm has a number of advantages over the algorithm of Ref. [10]. Most notably it scales as $\mathsf{poly}(\log(1/\varepsilon))$, but there are also polynomial improvements with respect to $s, \|A\|T$, and κ_V (Note: Some of the s dependence in Ref. [10] is due to the assumption that $\vec{b}$ is s sparse.). The term g does not appear in Ref. [10] because one of the hypotheses of the main result is that $g = O(1)$. The term β also does not appear; however, the complexity does scale polynomially with a similar term associated with the initial condition and inhomogeneity, whereas the algorithm presented here has $\mathsf{poly}(\log(\beta))$ scaling.

Another advantage is that the only hypothesis required for stability of the Taylor series method is that $\mathrm{Re}(\lambda_j) \leq 0$ for all eigenvalues λ_j of A, ie, the arguments of the λ_j must lie in $[\pi/2, 3\pi/2]$. In contrast, since Ref. [10] uses linear multistep methods, additional hypotheses are needed to guarantee stability of the method which further restrict the arguments of the λ_j to proper subintervals of $[\pi/2, 3\pi/2]$.

Chapter 4: FDM for Poisson's Equation

In the previous chapter we used QLSAs as subroutines for ODEs. This involved the construction of a linear system whose solution was a history state which we performed a post-selective measurement on. QLSAs are also a natural subroutine to consider for PDEs, since a vector can represent a solution to a PDE sampled at a set of points in the domain. This representation of the data lends itself most naturally to algorithms using the finite difference method (FDM). In this chapter we apply the FDM to solve Poisson's equation. The material in this chapter is originally written by A Ostrander; however, some of the material is the same verbatim as in 'High precision quantum algorithms for partial differential equations' (currently unpublished) by AM Childs, JP Liu, and A Ostrander.

4.1 Finite Difference Formulas

The FDM approximates the derivatives of a function f at a point $\vec{x}_i$ in terms of f's values on a finite set of points $\{\mathcal{X}\} = \{\vec{x}_j\}$ near $\vec{x}_i$. Generally there are no restrictions on where these points are relative to $\vec{x}_i$, but it is simplest to sample from uniformly spaced points. In the case we consider, this corresponds to discretizing $[-1,1]^d$ (or $[0,2\pi)^d$) to a d-dimensional rectangular (cyclic) lattice/mesh.

The spacing between points, which we denote as h, is one of the algorithmic parameters that can be adjusted depending on the error tolerance. The other relevant algorithmic parameter is the order of accuracy of the formula, which determines how the error term scales as a power of the lattice spacing. If a formula uses more sample points, it can typically have a higher order of accuracy than a formula using fewer sample points. One exception to this is that the trapezoid approximation of the first derivative is higher order than the forward and backward difference formulas, but all these formulas only use 2 sample points. However, given an order of accuracy, generally there is no unique finite difference formula for that order since a jth order formula and a kth order formula can be combined to give a $\min\{j,k\}$th order formula; however, the kth order formula with the smallest radius is unique since combining this formula with a higher order formula will only increase the radius.

Refs. [32, 72] consider generating coefficients starting from a Discrete Variable Respresentation (DVR) of Laplacian operators (the operators relevant to Poisson's equation). Ref. [32] uses a truncated Lagrange interpolation formula to derive coefficients. This is done by assuming interpolation data, eg, $\{(ih, f(ih)\}_{i=1}^{n}$ for n equidistant sample points. The function f can then be approximated by the polynomial

$$f(x) \approx \sum_{k=-N}^{N} f(kh) \prod_{l=-N, l\neq k}^{N} \left(\frac{x - lh}{kh - lh} \right). \tag{4.1}$$

This polynomial can be differentiated twice, and coefficients can be extracted from the resulting expression.

Coefficients can also be derived by writing the derivatives of f as linear combinations of f at different sample points with undertermined coefficients. By expanding these function values as Taylor series, equations restricting the coefficients can be derived. This approach leads to a Vandermonde system [63], which is the framework of several papers on finite difference formulas of arbitrarily high order. Ref. [57, 58] presents coefficients for up to 8th order finite difference formulas (some discovered numerically) and conjecture formulas for arbitrary orders. Ref. [59] rigorously proved the correctness of these formulas for first derivatives. Ref. [65] extended these results to formulas for non-uniform grids and more asymmetric stencils (instead of just forward, backward, and central difference stencils). These formulas have previously been used in quantum algorithms for estimating gradients Ref. [47] and for simulating quantum many body systems Ref. [61]. This algorithm for many body systems differs from most quantum chemistry algorithms, which work in the second quantized electron orbital basis instead of working in the position basis where it is natural to use the FDM.

4.2 Previous Quantum Algorithms

Within the quantum algorithms literature, linear PDEs have received a fair amount of attention. Although this chapter is about PDEs without time evolution, here we also review the literature for PDEs with time evolution, which is the focus of the next chapter. A large part of the literature on quantum simulation considers quantum systems described by PDEs: quantum chemistry and quantum field theories. Because the literature for these applications is so large, we do not review it here and focus on 'non-quantum' PDEs.

Ref. [31] proposes applying preconditioning matrices and HHL to solve finite element method (FEM) linear systems. The resource requirements of this algorithm are analyzed in Ref. [80]. Ref. [31] plausibly argues that sparse pre-conditioners can be implemented within the framework of QLSAs for sparse matrices (with some overhead for solving an optimization problem whose size scales with the sparsity). However, they do not analyze the complexity of their algorithm in terms of the error required to actually estimate a quantity of interest, eg, the error in a scattering cross section or a functional of the electric field generated by stationary charges. Ref. [73] accounts for the error in these tasks and analyzes what kind of quantum speed-ups can be achieved using the FEM and preconditioners. They argue that at most a polynomial improvement over classical algorithms can be expected.

Ref. [22] considers solving Poisson's equation over rectangular domains $\Omega = [-1,1]^d$ under Dirichlet boundary conditions $u(\partial\Omega) = 0$ using 2nd order accurate FDM Laplacians. The algorithm is essentially the same as HHL in that it diagonalizes the matrix and them performs controlled rotations on an ancillary qubit; although the authors give an explicit circuit for computing the inverse eigenvalues (whereas Ref. [52] merely assumed the ability to do this). Ref. [22] claims that their circuit uses $\mathsf{poly}(\log(1/\varepsilon))$ gates; this analysis ignores the fact that the success probability is $\mathsf{poly}(1/\varepsilon)$. A similar error is made in Ref. [71], which ignores the condition number and success probability.

Ref. [82] considers the problem of finding ascattering structures that would lead an initial set of EM fields to scatter into a final field configuration; however, they do not prove how efficient the algorithm is.

Refs. [44, 45] apply the reservoir method to the Dirac equation with a classical gauge field (as input) and to general hyperbolic differential equations. These papers state the speedup of the algorithm in terms of a quantity defined in Ref. [75] which is called the *quantum speedup*. Ref. [45] generally claims exponential speedups over classical algorithms; more specifically they achieve an exponential speedup with respect to the spatial dimension, but they only achieve a polynomial speedup with respect to the simulation error.

4.3 FDM Linear Systems

For a scalar field $u(\vec{x}) \in \mathbb{C}$, the canonical elliptic PDE is Poisson's equation,

$$\left(\sum_{j=1}^{d} \frac{\partial^2}{\partial x_j^2}\right) u(x) = f(\vec{x}), \tag{4.2}$$

which we consider solving on $[0,2\pi)^d$ with periodic boundary conditions. This also implies results for the domain $\Omega = [-1,1]^d$ under Dirichlet ($u(\partial\Omega) = 0$) and Neumann boundary conditions ($\hat{n}\cdot\nabla u(\partial\Omega)=0$, i.e., the normal derivative is 0, which for domain $[-1,1]^d$ is equivalent to $\frac{\partial u}{\partial x_j}|_{x_j=\pm 1}=0$ for $j\in[d]$).

Since Poisson's equation only involves second derivatives, the relevant 1D (central) finite difference formula is (taking $x_j = jh$ for a mesh with uniform spacing h)

$$u''(0)\approx\frac{1}{h^2}\sum_{j=-k}^{k} r_j u(jh) \tag{4.3}$$

where the coefficients are [61, 65]

$$r_j := \begin{cases} \frac{2(-1)^{j+1}(k!)^2}{j^2(k-j)!(k+j)!} & j\in[k] \\ -2\sum_{j=1}^{k} r_j & j=0 \\ r_{-j} & j\in -[k]. \end{cases} \tag{4.4}$$

We leave the dependence on the approximation order, k, implicit in this notation. The following lemma characterizes the error of this formula.

Lemma 10 ([61, Theorem 7])**.** *Let $k\geq 1$ and suppose $u(x)\in C^{2k+1}$ for $x\in\mathbb{R}$. Define the coefficients r_j as in Eq. 4.4. Then*

$$\frac{d^2u(x_i)}{dx^2}=\frac{1}{h^2}\sum_{j=-k}^{k} r_j u(x_i+jh)+O\Big(\Big|\frac{d^{2k+1}u}{dx^{2k+1}}\Big|\Big(\frac{eh}{2}\Big)^{2k-1}\Big) \tag{4.5}$$

where

$$\Big|\frac{d^{2k+1}u}{dx^{2k+1}}\Big| := \max_{y\in[x_i-kh,x_i+kh]}\Big|\frac{d^{2k+1}u}{dx^{2k+1}}(y)\Big|. \tag{4.6}$$

Since we assume periodic boundary conditions and apply the same FDM formula at each lattice site, the matrices we consider will be circulant. Define the $2n\times 2n$ matrix S to have entries $S_{i,j}=\delta_{i,j+1 \mod 2n}$. If we represent the solution $u(x)$ as a vector $\vec{u}=$

$\sum_{j=1}^{2n} u(\pi j/n)\vec{e}_j$, then we can write (approximately) Poisson's equation as

$$\frac{1}{h^2}L\vec{u} = \frac{1}{h^2}[r_0 I + \sum_{j=1}^{k} r_j(S^j + S^- j)]\vec{u} = \vec{f} \tag{4.7}$$

where $\vec{f} = \sum_{j=1}^{2n} f(\pi j/n)\vec{e}_j$. This is the linear system of interest for our algorithm. For classical algorithms we would be interested in outputting the entries of $\vec{u}$; in the quantum case we output the state proportional to $\vec{u}$.

4.4 Condition Number

The following lemma characterizes the condition number of a circulant Laplacian on $2n$ points.

Lemma 11. *The matrix $L = r_0 I + \sum_{j=1}^{k} r_j(S^j + S^- j)$ has condition number $\kappa(L) = \Theta(n^2)$ where $r_j = r_{-j} = \frac{2(-1)^{j+1}(k!)^2}{j^2(k-j)!(j+k)!}$ for $1 \leq j \leq k$, $r_0 = -2\sum_{j=1}^{k} r_j$, and $k = o(n^{2/3})$.*

Proof. We directly consider L's eigenvalues since it is symmetric so its singular values are the absolute values of its eigenvalues.

We first upper bound $|L|$ using Gershgorin's circle theorem [54] similar to Ref. [61]. Note that $|r_j| = \frac{2(k!)^2}{j^2(k-j)!(j+k)!} \leq 2/j^2$ since $\frac{(k!)^2}{(k-j)!(j+k)!} = \frac{k!/(k-j)!}{(k+j)!/k!} < 1$. The radius of the Gershgorin disc is

$$2\sum_{j=1}^{k}|r_j| \leq 2\sum_{j=1}^{k}\frac{2}{j^2} \leq \frac{2\pi^2}{3}. \tag{4.8}$$

The center of the disc is r_0, and

$$|r_0| \leq 2\sum_{j=1}^{k}|r_j| \leq \frac{2\pi^2}{3}, \tag{4.9}$$

so $|L| \leq \frac{4\pi^2}{3}$.

To lower bound $|L^{-1}|$ we lowerbound the (absolute value of the) smallest non-zero eigenvalue of L (since by construction the all-ones vector is a zero eigenvector).

Because L is a linear combination of the S which are circulant, L's eigenvalues are sums of roots of unity

$$\lambda_l = r_0 + \sum_{j=1}^{k} r_j(\omega^{lj} + \omega^{-lj}) \tag{4.10}$$

$$= r_0 + \sum_{j=1}^{k} 2r_j \cos(\frac{\pi l j}{n}) \tag{4.11}$$

$$= r_0 + \sum_{j=1}^{k} 2r_j \left(1 - \frac{\pi^2 l^2 j^2}{2n^2} + \frac{(\pi c_j)^4}{4!n^4} \cos(\frac{\pi c_j}{n})\right) \tag{4.12}$$

$$= -\frac{\pi^2 l^2}{n^2} \sum_{j=1}^{k} r_j j^2 + O\left(\frac{l^4 k^3}{n^4}\right) \tag{4.13}$$

where $\omega = \exp(\pi i/n)$ and the $c_j \in [0, lj]$ dependent terms arise from the Taylor remainder theorem. The last line follows from the fact that $|r_j| = O(1/j^2)$.

The sum we are concerned with is

$$-\sum_{j=1}^{k} r_j j^2 = \sum_{j=1}^{k} j^2 \frac{2(-1)^j (k!)^2}{j^2 (k+j)!\,(k-j)!} \tag{4.14}$$

$$= 2(k!)^2 \sum_{j=1}^{k} \frac{(-1)^j}{(k+j)!\,(k-j)!} \tag{4.15}$$

$$= 2(k!)^2 \frac{1}{(2k)!} \sum_{j=1}^{k} (-1)^j \frac{(2k)!}{(k+j)!\,(k-j)!} \tag{4.16}$$

$$= 2(k!)^2 \frac{1}{(2k)!} \sum_{j=1}^{k} (-1)^j \frac{(2k)!}{(k\pm j)!\,(2k-(k\pm j))!} \tag{4.17}$$

$$= (k!)^2 \frac{(-1)^k}{(2k)!} \left(\sum_{j=1}^{k} (-1)^{k+j} \binom{2k}{k+j} + \sum_{j=1}^{k} (-1)^{k-j} \binom{2k}{k-j} \right) \tag{4.18}$$

$$= (k!)^2 \frac{(-1)^k}{(2k)!} \sum_{j=0, j\neq k}^{2k} (-1)^j \binom{2k}{j} \tag{4.19}$$

$$= (k!)^2 \frac{(-1)^k}{(2k)!} ((1-1)^{2k} - (-1)^k \binom{2k}{k}) \tag{4.20}$$

$$= -1 \tag{4.21}$$

Then we have

$$\lambda_l = -\frac{\pi^2 l^2}{n^2} + O\left(\frac{l^4 k^3}{n^4}\right) \tag{4.22}$$

$$\lambda_1 = -\frac{\pi^2}{n^2} + O\left(\frac{k^3}{n^4}\right) \tag{4.23}$$

and the last term is vanishingly small since $k = o(n^{2/3})$ $\square$

The following lemma extends this to d dimensions,

Lemma 12. *Let $L = r_0 I + \sum_{j=1}^{k} r_j (S^j + S^- j)$ where $r_j = r_{-j} = \frac{2(-1)^{j+1}(j!)^2}{j^2(j-k)!(j+k)!}$ for $1 \le j \le k$, $r_0 = -2\sum_{j=1}^{k} r_j$, and $k = o(n^{2/3})$. The matrix $L' = L \otimes I^{\otimes d-1} + I \otimes L \otimes I^{\otimes d-2} + \cdots + I^{\otimes d-1} \otimes L$ has condition number $\kappa(L') = \Theta(d/n^2)$*

Proof. By the triangle inequality for spectral norms, $|L'| \leq d|L|$. Since L has zero sum rows by construction, the all-ones vector lies in its kernel, and thus the smallest non-zero eigenvalue of L is the same as that of L'. $\square$

4.5 Error Analysis

We introduce several states for the purpose of error analysis. Let $|u\rangle$ be the state that is proportional to $\vec{u} = \sum_{j=1}^{2n} u(\pi j/n)\vec{e}_j$ for the exact solution of the differential equation. Let $|\bar{u}\rangle$ be the state output by a QLSA that exactly solves the linear system. Let $|\tilde{u}\rangle$ be the state output by a QLSA with error. Then the total error of approximating $|u\rangle$ by $|\tilde{u}\rangle$ is bounded by

$$\begin{aligned} ||u\rangle - |\tilde{u}\rangle| &\leq ||u\rangle - |\bar{u}\rangle| + ||\bar{u}\rangle - |\tilde{u}\rangle| && (4.24) \\ &= \varepsilon_{FDM} + \varepsilon_{QLSA} && (4.25) \end{aligned}$$

and without loss of generality we can take ε_{FDM} and ε_{QLSA} to be of the same order of magnitude.

Lemma 13. *Let $u(\vec{x})$ be the exact solution of $(\sum_{i=1}^{d} \frac{d^2}{dx_i^2})u(\vec{x}) = f(\vec{x})$. Let $\vec{u} \in \mathbb{R}^{(2n)^d}$ encode the exact solution in the sense that $\vec{u} = \sum_{j \in \mathbb{Z}_{2n}^d} u(\pi j/n) \bigotimes_{i=1}^{d} e_{j_i}$. Let $\bar{u} \in \mathbb{R}^{(2n)^d}$ be the exact solution of the FDM linear system $\frac{1}{h^2}L'\bar{u} = \vec{f}$, where L' is as above a d-dimensional $(2k)$th order Laplacian where $k = o(n^{2/3})$ and $\vec{f} = \sum_{j=1}^{2n} f(\pi j/n)\vec{e}_j$. Then $||\vec{u} - \bar{u}|| \leq O(2^{d/2} n^{(d/2)-2k+1} |\frac{d^{2k+1}u}{dx^{2k+1}}| (e^2/4)^k)$.*

Proof. The remainder term of the central difference formula is $O(|\frac{d^{2k+1}u}{dx^{2k+1}}| h^{2k-1}(e/2)^{2k})$, so that

$$\frac{1}{h^2}L'\vec{u} = \vec{f} + O\Big(\Big|\frac{d^{2k+1}u}{dx^{2k+1}}\Big|(eh/2)^{2k-1}\Big)\vec{\varepsilon} \tag{4.26}$$

where $\vec{\varepsilon}$ is a $(2n)^d$ dimensional vector whose entries are $O(1)$. This implies

$$\frac{1}{h^2}L'(\vec{u}-\bar{u}) = O\Big(\Big|\frac{d^{2k+1}u}{dx^{2k+1}}\Big|(eh/2)^{2k-1}\Big)\vec{\varepsilon} \tag{4.27}$$

so that

$$||\vec{u}-\bar{u}|| = O\Big(\Big|\frac{d^{2k+1}u}{dx^{2k+1}}\Big|(eh/2)^{2k+1}\Big)||(L')^{-1}\vec{\varepsilon}|| \tag{4.28}$$

$$= O\Big((2n)^{d/2}\Big|\frac{d^{2k+1}u}{dx^{2k+1}}\Big|(eh/2)^{2k+1}/\lambda_1\Big). \tag{4.29}$$

since in the worst case $\vec{\varepsilon}$ is supported in the lowest lying eigenspace of L'. From the proof of Lemma 12 we have $\lambda_1 = \Theta(1/n^2)$, and since $h = \Theta(1/n)$, we have

$$||\vec{u}-\bar{u}|| = O\Big(2^{d/2}n^{(d/2)-2k+1}\Big|\frac{d^{2k+1}u}{dx^{2k+1}}\Big|(e/2)^{2k}\Big) \tag{4.30}$$

which proves the theorem. □

4.6 FDM Algorithms for Poisson's Equation

To apply QLSAs, we must consider the complexity of simulating Hamiltonians which correspond to Laplacian FDM operators. For periodic boundary conditions, the Laplacians are circulant, so they can be diagonalized by the quantum Fourier transform (QFT), ie, $L = FDF^\dagger$. In this case the simplest way to simulate $\exp(iLt)$ is to perform the inverse QFT, apply controlled phase rotations $\exp(iDt)$, and perform the QFT. Ref. [81] shows how to exactly implement arbitrary diagonal unitaries on m qubits using $O(2^m)$ gates. Since we consider Laplacians on n mesh sites, simulating $\exp(iLt)$ takes $O(n)$ gates with the dominant contribution coming from the phase rotations (which might be improved using Ref. [95] or Ref. [13]).

Theorem 7. *(Poisson equation with periodic boundary conditions) There exists a quantum algorithm that outputs a state ε-close to $|u\rangle$ that runs in time*

$$\tilde{O}(d^2 \log^{9/2+\gamma}(\sqrt{d}|\frac{d^{2k+1}u}{dx^{2k+1}}|/\varepsilon)\sqrt{\log(1/\varepsilon)}) \tag{4.31}$$

and makes

$$\tilde{O}(d \log^{3+\beta}(\sqrt{d}|\frac{d^{2k+1}u}{dx^{2k+1}}|/\varepsilon)\sqrt{\log(1/\varepsilon)}) \tag{4.32}$$

queries to the oracle for $\vec{f}$ for arbitrarily small $\gamma, \beta > 0$.

Proof. We use the Fourier series based QLSA from Ref. [25]. By Theorem 3 of that work, the QLSA makes $O(\kappa\sqrt{\log(\kappa/\varepsilon_{\text{QLSA}})})$ uses of a Hamiltonian simulation algorithm and uses of the oracle for the inhomogeneity. For Hamiltonian simulation we use d parallel QFTs and phase rotations as described in Ref. [81], for a total of $O(dn\kappa\sqrt{\log(\kappa/\varepsilon_{\text{QLSA}})})$ gates. The condition number for the d-dimensional Laplacian scales as $\kappa = O(dn^2)$.

We take ε_{FDM} and $\varepsilon_{\text{QLSA}}$ to be of the same order and just write ε. Then the QLSA has time complexity $O(n^3\sqrt{\log(n^2/\varepsilon)})$ and query complexity $O(n^2\log(n^2/\varepsilon))$. The parameters we can adjust for the algorithm are the number of sites on the lattice n and the order of the finite difference formula. To keep the error below the target error of ε we require

$$2^{d/2}n^{(d/2)-2k+1}\left|\frac{d^{2k+1}u}{dx^{2k+1}}\right|(e/2)^{2k} = O(\varepsilon), \tag{4.33}$$

which is equivalent to

$$(-d/2)+(-(d/2)+2k-1)\log(n)-2k\log(e/2) = \Omega\Big(\log\Big(\left|\frac{d^{2k+1}u}{dx^{2k+1}}\right|/\varepsilon\Big)\Big). \tag{4.34}$$

To satisfy this condition, we set $k = dn^b$ for some constant $b < 2/3$ (so that our condition number lemmas hold) and n sufficiently large, giving

$$n^b \log(n) = \Omega\Big(\frac{1}{d}\log\Big(\Big|\frac{d^{2k+1}u}{dx^{2k+1}}\Big|/\varepsilon\Big)\Big). \tag{4.35}$$

We set $n = \Theta(\log^{3/2+\delta}(|\frac{d^{2k+1}u}{dx^{2k+1}}|/\varepsilon))$ to satisfy this criterion for arbitrary (b-dependent) $\delta > 0$. This suffices to prove the theorem. $\square$

This can be compared to the cost of using the conjugate gradient method to solve the same linear system classically. The sparse conjugate gradient algorithm for an $N \times N$ matrix has time complexity $O(Ns\sqrt{\kappa}\log(1/\varepsilon))$. For arbitrary dimension $N = \Theta(n^d)$, we have $s = dk = d^2n^b$ and $\kappa = O(dn^2)$, so that the time complexity is
$O(d^{2.5}\log(1/\varepsilon)\log^{3+1.5d+1.5b+\gamma'}(\sqrt{d}|\frac{d^{2k+1}u}{dx^{2k+1}}|/\varepsilon))$ for arbitrary $\gamma' > 0$. Alternatively, d fast Fourier transforms could be used, although this will generally take
$\Omega(n^d) = \Omega(\log^{3d/2}(\sqrt{d}|\frac{d^{2k+1}u}{dx^{2k+1}}|/\varepsilon))$ time.

The sparse conjugate gradient algorithm has complexity $O(Ns\sqrt{\kappa}\log(1/\varepsilon))$ time complexity. For arbitrary dimension $N = \Theta(n^d)$, $s = dk = dn^b$, $\kappa = O(dn^2)$, so that the time complexity is $O(d^2n^{2+d+b}\log(1/\varepsilon)) = O(d^2\log(1/\varepsilon)\log^{3+1.5d+1.5b+\gamma'}(\sqrt{d}|\frac{d^{2k+1}u}{dx^{2k+1}}|/\varepsilon))$ for arbitray $\gamma' > 0$.

Alternatively, d fast Fourier transforms could be used, although this will generally take $\Omega(n^d) = \Omega(\log^{3d/2}(\sqrt{d}|\frac{d^{2k+1}u}{dx^{2k+1}}|/\varepsilon)$ time.

4.7 Boundary Conditions via the Method of Images

We can apply the method of images [78] to deal with the boundary conditions. In Ref. [83] the method of images is applied on graphs to analyze the spectrum of the graph Laplacian of a path graph. In the method of images, the domain $[-1,1]$ is extended to include all of $\mathbb{R}$, and the boundary conditions are related to symmetries of the solutions.

For two Dirichlet boundary conditions there are two symmetries: the solutions are anti-symmetric about -1, $f(-x-1) = -f(x-1)$, and anti-symmetric about 1, $f(1+x) = -f(1-x)$. Continuity and anti-symmetry about -1 and 1 immediately imply the $f(-1) = f(1) = 0$, and this more generally implies $f(x) = 0$ for all odd $x \in \mathbb{Z}$ and that $f(x+4) = f(x)$. For Neumann boundary conditions, the solutions are instead symmetric about -1 and 1, which implies $f(x+2) = f(x)$

We would like to combine the method of images with the FDM to arrive at finite difference formulas for this special case. In both cases, the method of images implies that the solutions are periodic, so without loss of generality we can consider a mesh on $[0, 2\pi)$ instead of a mesh on $\mathbb{R}$. We will find it useful to think of this mesh in terms of the cycle graph on $2n$ vertices, ie $(V,E) = (\mathbb{Z}_{2n}, \{(i,i+1) | i \in \mathbb{Z}_{2n}\})$, which means that the vectors encoding the solution $u(x)$ will lie in $\mathbb{R}^{2n}$. Let each vector $\vec{e}_j$ correspond to the vertex j. Then we divide $\mathbb{R}^{2n}$ into a symmetric and and anti-symmetric subspace, namely $\text{span}\{e_j + e_{2n+1-j}\}_{j=1}^{n}$ and $\text{span}\{e_j - e_{2n+1-j}\}_{j=1}^{n}$. Vectors lying in the symmetric subspace correspond to solutions that are symmetric about 0 and π, so they obey Neumann boundary conditions at 0 and π; similarly, vectors in the anti-symmetric space correspond to solutions obeying Dirichlet boundary conditions at 0 and π.

Restricting to a subspace of vectors reduces the size of the FDM vectors and matrices we consider, and the symmetry of that subspace indicates how to adjust the coefficients. If the FDM linear system is $L''\vec{u}'' = \vec{f}''$ then L'' has entries

$$L''_{i,j} = \begin{cases} r_{|i-j|} \pm r_{i+j-1} & i \leq k \\ r_{|i-j|} & k < i \leq n-k \\ r_{|i-j|} \pm r_{2n-i-j+1} & n-k \leq i \end{cases}$$

where $+(-)$ is chosen for Neumann (Dirichlet) boundary conditions and due to the truncation order k, $r_j = 0$ for any $j > k$.

For the purpose of solving the new linear systems using quantum algorithms, we still treat these cases as obeying periodic boundary conditions. We assume access to an oracle that produces states $|f''\rangle$ proportional to the inhomogeneity $f''(x)$. Then we apply the QLSA for periodic boundary conditions using $|f''\rangle|\pm\rangle$ to encode the inhomogeneity, which will output solutions of the form $|u''\rangle|\pm\rangle$. Here the ancilla is chosen to be $|+\rangle(|-\rangle)$ for Neumann (Dirichlet) boundary conditions.

The graph Laplacian (second order) for the path graph with Dirichlet boundary conditions has diagonal entries which are all equal to 2; however, using the above specification for the entries of L leads to the $(1,1)$ and (n,n) entries being 3 while the rest of the diagonal entries are 2.

To reproduce the graph Laplacian, we must consider an alternative subspace restriction used in Ref. [83]. In this case it is easiest to consider the mesh of a cycle graph on $2n+2$ vertices, where the vertices 0 and $n+1$ are selected as boundary points where the field takes the value 0. The relevant antisymmetric subspace is now $\operatorname{span}\{e_j - e_{2n+2-j}\}_{j=1}^{n}$ (which has no support on e_0 and e_{n+1}). If we again write the linear system as $L''\vec{u}'' = \vec{f}''$, then the Laplacian has entries

$$L''_{i,j} = \begin{cases} r_{|i-j|} - r_{i+j} & i \leq k \\ r_{|i-j|} & k < i \leq n-k \\ r_{|i-j|} - r_{2n-i-j+2} & n-k \leq i \end{cases} . \tag{4.36}$$

We again assume access to an oracle producing states proportional to $f''(x)$; however, we assume that this oracle operates in a Hilbert space with one additional dimension compared to the previous approaches, ie, previously we considered just implementing U whereas here we consider implementing $\begin{bmatrix} U & \vec{0} \\ \vec{0}^T & 1 \end{bmatrix}$. With this oracle we again prepare the state $|f''\rangle|-\rangle$ and solve Poisson's equation for perioidc boundary conditions to output a state $|u''\rangle|-\rangle$ (where $|u''\rangle$ lies in an $n+1$ dimensional Hilbert space but has no support on the $n+1$th basis state).

We in fact expect algorithms for Dirichlet boundary conditions to have lower (by a factor of d) time and query complexity compared to periodic and Neumann boundary conditions. This is because the $|-\rangle$ that appears in the inhomogeneous part of the linear system guarantees that $\vec{f}$ is orthogonal to the all-ones vector, which spans the kernel of L'. Within this anti-symmetric subspace, the lowest eigenvalues is $\Theta(d/n^2)$ instead of $\Theta(1/n^2)$, so the condition number is effectively reduced by a factor of d.

4.8 First Order PDEs

We conclude this chapter by considering what form linear systems might take for applying the FDM to first order PDEs. We use the backward difference approximation of the derivative $\frac{df}{dx}|_{x_i} \approx \frac{f(x_i)-f(x_i-h)}{h}$ because the form is simple and even a first order approximation gives some intuition for bounds on the condition number of higher order FDM matrices with similar structure.

Using the backward difference method with a uniform square $n \times n$ grid for the equation

$$(\frac{\partial}{\partial x} + \alpha \frac{\partial}{\partial x})u(x,y) = f(x,y) \tag{4.37}$$

$$u(x,0) = a(x), u(0,y) = b(y) \tag{4.38}$$

we have the following linear system

$$D\vec{u} = \frac{1}{h}\begin{bmatrix} 1+\alpha & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 \\ -1 & 1+\alpha & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & -1 & 1+\alpha & \dots & 0 & 0 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ -\alpha & 0 & 0 & \dots & 1+\alpha & 0 & 0 & \dots & 0 & 0 \\ 0 & -\alpha & 0 & \dots & -1 & 1+\alpha & 0 & \dots & 0 & 0 \\ 0 & 0 & -\alpha & \dots & 0 & -1 & 1+\alpha & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & -\alpha & 0 & 0 & \dots & 1+\alpha & 0 \\ 0 & 0 & 0 & \dots & 0 & -\alpha & 0 & \dots & -1 & 1+\alpha \end{bmatrix}\begin{bmatrix} u_{1,1} \\ u_{2,1} \\ u_{3,1} \\ \vdots \\ u_{1,2} \\ u_{2,2} \\ u_{3,2} \\ \vdots \\ u_{1,3} \\ u_{2,3} \end{bmatrix}$$

$$= \begin{bmatrix} f_{1,1} + \frac{\alpha a_1 + b_1}{h} \\ f_{2,1} + \frac{\alpha a_2}{h} \\ f_{3,1} + \frac{\alpha a_3}{h} \\ \vdots \\ f_{1,2} + \frac{b_2}{h} \\ f_{2,2} \\ f_{3,2} \\ \vdots \\ f_{1,3} + \frac{b_3}{h} \\ f_{2,3} \end{bmatrix} = \begin{bmatrix} g_{1,1} \\ g_{2,1} \\ g_{3,1} \\ \vdots \\ g_{1,2} \\ g_{2,2} \\ g_{3,2} \\ \vdots \\ g_{1,3} \\ g_{2,3} \end{bmatrix} \tag{4.39}$$

Since this matrix is lower triangular it can be explicitly inverted. Alternatively due to the square mesh and the backward difference formula, the value $u_{i,j}$ only depends on $f_{i,j}$ and two of $u_{i-1,j}$, $u_{i,j-1}$, $a_i = a(ih)$, $b_i = b(ih)$. Thus we can write a recurrence relation with two indices

$$u_{1,1} = \frac{hf_{1,1} + \alpha a_1 + b_1}{1+\alpha} = \frac{hg_{1,1}}{1+\alpha} \tag{4.40}$$

$$u_{i,1} = \frac{hf_{i,1} + \alpha a_i + u_{i-1,1}}{1+\alpha} = \frac{hg_{i,1}}{1+\alpha} + \frac{u_{i-1,1}}{1+\alpha} \qquad 2 \leq i \leq n \tag{4.41}$$

$$u_{1,i} = \frac{hf_{1,i} + b_i + \alpha u_{1,i-1}}{1+\alpha} = \frac{hg_{1,i}}{1+\alpha} + \frac{\alpha u_{1,i-1}}{1+\alpha} \qquad 2 \leq i \leq n \tag{4.42}$$

$$u_{i,j} = \frac{hf_{i,j} + \alpha u_{i,j-1} + u_{i-1,j}}{1+\alpha} = \frac{hg_{i,j}}{1+\alpha} + \frac{\alpha u_{i,j-1} + u_{i-1,j}}{1+\alpha} \qquad 2 \leq i,j \leq n \tag{4.43}$$

From this and the fact that $h = 1/n$ it can be proven that $||D|| = O(n)$ and $||D^{-1}|| = O(n)$. Generalizing to d dimensions, the differential equation becomes $(\sum_i \alpha_i \frac{\partial}{\partial x_i})u(\vec{x}) = f(\vec{x})$, from which we construct the linear system $D_d\vec{u} = \vec{g}$. The new linear system implies a recurrence relation with d indices which can be used to bound the condition number of the linear system.

Chapter 5: FDM for the Wave and Klein-Gordon Equations

In the previous chapters, we considered algorithms for linear ODEs, which can easily be used as primitive subroutines for solving PDEs with time evolution. We consider this problem for the wave equation and Klein-Gordon equation. We begin by discussing how to adapt first order ODE algorithms to PDEs that are second order in time, and this will help motivate our choice of equations. FDM Laplacian matrices will again play a crucial role in our algorithms. We discuss how to factor graph Laplacians and then consider how the choice of boundary conditions affects the Laplacian and factorization. We discuss higher order Laplacians and their Gram factorizations, and we present a physically motivated way to impose Dirichlet boundary conditions. We then discuss state preparation and post-processing routines and under what conditions they are used. We analyze the complexity of the different steps in the algorithm and finally compare it to alternative algorithms.

5.1 Second Order Equations with First Order Algorithms

For PDEs with time evolution, the existing Hamiltonian simulation or ODE algorithms [10, 16, 26] can be combined with the FDM to produce algorithms. Since the wave equation and Klein-Gordon equations are second order equations, we cannot directly apply Hamiltonian simulation or ODE algorithms but must first transform to a first order equation. In general, any dynamical system can be transformed to one that is first order in time by introducing auxiliary variables. For instance to reduce a nth order equation to a $n-1$th order equation, we transform $\frac{d^n u}{dt^n} = f(u)$ to the system $\frac{d^{n-1}u}{dt^{n-1}} = v$, $\frac{dv}{dt} = f(u)$ where v is the auxiliary variable. This transformation introduces additional overhead which shows up in our algorithms as additional state preparation or post-selection steps.

One familiar approach to simulating the Klein-Gordon equation is to simulate the Dirac equation and output just one of the spinor components since each component individually obeys the wave equation. This is appealing since simulating the Dirac equation just requires Hamiltonian simulation and not an ODE algorithm which has additional overhead costs. Instead of the Dirac equation, another equation which implies the Klein-Gordon equation could be simulated. This still requires introducing an auxiliary variable to reduce to a first order equation.

We now consider the most general form of the first order ODE and argue about how to simplify it with the goals of (1) simulating second order equations and (2) eliminating ancilla qubit overheads. For a solution $\phi(\vec{x})$, we define $\vec{\phi} = \sum_{x_j \in \mathcal{X}} \phi(x_j)\vec{e}_j$ where $\mathcal{X}$ is the domain of interest and $\vec{e}_j$ is jth basis vector (similar to the representation in the previous chapter). The auxiliary variable also has a corresponding vector, $\vec{\theta}$, which can in principle be of higher dimension than $\vec{\phi}$. We can in general write the FDM ODE as

$$\frac{d}{dt}\begin{bmatrix}\vec{\phi}\\ \vec{\theta}\end{bmatrix} = \begin{bmatrix}A & B\\ C & D\end{bmatrix}\begin{bmatrix}\vec{\phi}\\ \vec{\theta}\end{bmatrix} \tag{5.1}$$

where $\begin{bmatrix}\vec{\phi}\\ \vec{\theta}\end{bmatrix}$ denotes the vertical concatenation of $\vec{\phi}$ and $\vec{\theta}$. This implies the second order equation

$$\frac{d^2}{dt^2}\begin{bmatrix}\vec{\phi}\\ \vec{\theta}\end{bmatrix} = -\begin{bmatrix}A^2+BC & AB+BD\\ CA+DC & D^2+CB\end{bmatrix}\begin{bmatrix}\vec{\phi}\\ \vec{\theta}\end{bmatrix}. \tag{5.2}$$

We would like this to decouple into separate second order equations for $\vec{\phi}$ and $\vec{\theta}$, which implies $AB+BD = CA+DC = 0$. Two simple ways to satisfy these conditions are to set

$B = C = 0$ or $A = D = 0$. Note that these are not the only assignments; the Dirac equation has the different spinor components coupled non-trivially. Setting $B = C = 0$ nullifies the need to introduce auxiliary variables since $\vec{\phi}$ and $\vec{\theta}$ are then decoupled in the original ODE. Setting $A = D = 0$ we have that

$$\frac{d}{dt}\begin{bmatrix}\vec{\phi}\\ \vec{\theta}\end{bmatrix} = \begin{bmatrix}0 & B\\ C & 0\end{bmatrix}\begin{bmatrix}\vec{\phi}\\ \vec{\theta}\end{bmatrix} \tag{5.3}$$

and

$$\frac{d^2}{dt^2}\begin{bmatrix}\vec{\phi}\\ \vec{\theta}\end{bmatrix} = \begin{bmatrix}BC & 0\\ 0 & CB\end{bmatrix}\begin{bmatrix}\vec{\phi}\\ \vec{\theta}\end{bmatrix}. \tag{5.4}$$

The naive way to define the auxiliary vector $\vec{\theta}$ is to set $\vec{\theta} = \frac{d}{dt}\vec{\phi}$ in which case $B = I$, which implies that C is a FDM Laplacian matrix. This results in the ODE

$$\frac{d}{dt}\begin{bmatrix}\vec{\phi}\\ \vec{\theta}\end{bmatrix} = \begin{bmatrix}0 & I\\ \frac{-1}{h^2}L & 0\end{bmatrix}\begin{bmatrix}\vec{\phi}\\ \vec{\theta}\end{bmatrix} \tag{5.5}$$

where L is defined as in the previous chapter. This in fact simulates wave equation evolution for both $\vec{\phi}$ and $\vec{\theta}$. Since the matrix of this ODE is not anti-Hermitian, it is not amenable to Hamiltonian simulation and a general ODE algorithm must be used.

All existing ODE algorithms involve solving highly structured linear systems. The two we have discussed in this thesis [10, 16] produce a history state of the ODE evolution, which requires additional qubit overheads for the time register (and the Taylor series truncation order register for Ref. [16]). The algorithm of Ref. [16] depends on using an exponentially precise QLSA, and the LCU based QLSA [25], uses additional ancilla for the LCU; although, Ref. [86] subsequently presented a QLSA with constant ancilla overheads.

Because of these overheads, it would be more efficient if the ODE were in fact an instance of Schrödinger's equation. Assuming anti-Hermiticity (ie $C = -B^\dagger$) and $A = D = 0$

(which suffices for decoupling) in the original ODE, we have

$$\frac{d}{dt}\begin{bmatrix}\vec{\phi}\\ \vec{\theta}\end{bmatrix} = \begin{bmatrix}0 & B\\ -B^{\dagger} & 0\end{bmatrix}\begin{bmatrix}\vec{\phi}\\ \vec{\theta}\end{bmatrix} \tag{5.6}$$

$$\frac{d^2}{dt^2}\begin{bmatrix}\vec{\phi}\\ \vec{\theta}\end{bmatrix} = \begin{bmatrix}-BB^{\dagger}\vec{\phi}\\ -B^{\dagger}B\vec{\theta}\end{bmatrix} \tag{5.7}$$

so the matrix $-BB^{\dagger}$ must be a FDM Laplacian operator.

5.2 Factoring Graph Laplacians

The Gram factorization of graph Laplacians is well understood [30, 34, 49] . Given a graph $G = (V,E)$, we define the graph Laplacian as the $|V|\times|V|$ matrix $L(G) = D(G) - A(G)$ where $D(G)$ is the degree matrix with $D_{i,j} = \delta_{i,j}d_j$ where d_j is the number of edges incident with vertex j and $A(G)$ is the adjacency matrix with $A_{i,j} = 1$ iff $(i,j) \in E$ and $A_{i,j} = 0$ otherwise.

The Laplacian of G is Gram factored by signed incidence matrices, which are not unique to the graph. They are defined by first creating a directed graph $G' = (V,E')$ by assigning arbitrary orientations to the edges of G. The $|V|\times|E|$ signed incidence matrix W associated with this graph has entries

$$W_{i,j} = \begin{cases} 1 & \text{if } j \in E \text{ points away from } i \\ -1 & \text{if } j \in E \text{ points towards } i \\ 0 & \text{else} \end{cases} . \tag{5.8}$$

This also works for graphs with positively weighted edges and with loops (edges of the form (i,i)) where the entries of A are equal to the edge weights. In this case the incidence matrix has entries

$$W_{i,j} = \begin{cases} \sqrt{w_j} & \text{if } j \in E \text{ is not a loop and points away from } i \\ -\sqrt{w_j} & \text{if } j \in E \text{ is not a loop and points towards } i \\ \sqrt{w_j} & \text{if } j \in E \text{ is a loop and points towards } i \\ 0 & \text{else} \end{cases} \tag{5.9}$$

where w_j is the weight on edge j. Note that because we set $L = WW^\dagger$, L does not depend on the edge orientations.

5.3 Boundary Conditions and Mass Terms

When approximating the FDM Laplacian in 1D, the two graph classess of interest are the cycle graphs and path graphs. As in the last chapter, let S be the $n \times n$ circulant matrix with entries $S_{i,j} = \delta_{i,j+1 \mod n}$. Then the graph Laplacian of the cycle on n vertices, C_n, is $L(C_n) = 2I - S - S^\dagger$. So $\frac{-1}{h^2}L(C_n) = -BB^\dagger$ for $B = (1/h)(I - S)$ corresponds to the FDM Laplacian operator using the central difference formula $f''(x_i) \approx \frac{-2f(x_i)+f(x_i-h)+f(x_i+h)}{h^2}$ under periodic boundary conditions.

For Neumann boundary conditions, we use arguments similar to those presented in the previous chapter for the method of images, and we can write the Laplacian approximation for the unit interval as $\frac{-1}{h^2}L(P_n)$. We begin by considering a Laplacian for an infinite path graph whose vertices are indexed by $\mathbb{Z}$. Suppose we place a boundary at the 0 vertex and impose Neumann boundary conditions ($\frac{d\phi}{dx}|_{x_0}= 0$). We can realize this by fixing the field to be ϕ_0 on vertices indexed by $\mathbb{Z}^-$. Physically this corresponds to the case where ϕ is the electrical potential and the boundary is of a perfect conductor. This effectively changes how the Laplacian acts on $\vec{\phi}$ as

$$L\vec{\phi} = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} \phi_{-2} \\ \phi_{-1} \\ \phi_0 \\ \phi_1 \\ \phi_2 \end{bmatrix} \tag{5.10}$$

$$\mapsto L_{Neumann}\vec{\phi} = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} \phi_0 \\ \phi_0 \\ \phi_0 \\ \phi_1 \\ \phi_2 \end{bmatrix} \tag{5.11}$$

$$= \begin{bmatrix} 0 \\ 0 \\ \phi_0 - \phi_1 \\ 2\phi_1 - \phi_0 - \phi_2 \\ \cdots \end{bmatrix} \tag{5.12}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} \phi_0 \\ \phi_0 \\ \phi_0 \\ \phi_1 \\ \phi_2 \end{bmatrix}. \tag{5.13}$$

Thus we see that the FDM Neumann Laplacian for an interval $[a, a+nh]$ under Neumann boundary conditions is $\frac{-1}{h^2}L(P_{n+1})$.

To impose $\phi = 0$ Dirichlet boundary conditions, we set $\phi_i = 0$ for $i \in \mathbb{Z}^-$, which changes how the Laplacian acts as

$$L\vec{\phi} = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} \phi_{-2} \\ \phi_{-1} \\ \phi_0 \\ \phi_1 \\ \phi_2 \end{bmatrix} \tag{5.14}$$

$$\mapsto L_{Dirichlet}\vec{\phi} = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \phi_0 \\ \phi_1 \\ \phi_2 \end{bmatrix} \tag{5.15}$$

$$= \begin{bmatrix} 0 \\ -\phi_0 \\ 2\phi_0 - \phi_1 \\ 2\phi_1 - \phi_0 - \phi_2 \\ \cdots \end{bmatrix} \tag{5.16}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \phi_0 \\ \phi_1 \\ \phi_2 \end{bmatrix}. \tag{5.17}$$

We can ignore the row indexed by -1 since it corresponds to a point in the boundary. This Dirichlet Laplacian has the form $2I - A(P_n)$ and the FDM matrix is $\frac{-1}{h^2}(2I - A(P_n))$, which differs from the Neumann Laplacian in the $1,1$ and n,n entries.

The matrix $2I - A(P_n)$ is referred to as a *generalized Lalpacian* [49] which generally takes the form $L(G) + M$ where $L(G)$ is the Laplacian of a weighted graph and M is a diagonal matrix, which can be interpreted as a potential energy operator in the vertex basis. The weight term can be accounted for by adding a loop of weight $M_{i,i}$ at every vertex i. These are special cases of *stoquastic matrices* [8, 19] which have non-positive off-diagonal elements.

Here we have used graph theoretic arguments to impose Neumann and Dirichlet boundary conditions, and we have seen how they affect the entries of the Laplacian. Taking an alternative approach for the path graph, P_n, Ref. [83] presents arguments for imposing Neumann boundary conditions that are the same as the method of image arguments we presented in the previous chapter, and they wind up producing the ordinary graph Laplacian $D-A$. For Dirichlet boundary conditions, in the previous chapter we presented 2 method of images arguments which lead to Laplacians $2I-A$ (the same as what we just derived) and $4I-D-A$.

In these examples, the different choices of boundary conditions only changed the value of the $1,1$ and n,n terms. Changing diagonal terms is also how to account for the mass term in the FDM approximation of the Klein-Gordon equation. This is accomplished by adding a loop at each vertex with weight m^2h^2, which changes the graph Laplacian by adding to it the matrix m^2h^2I.

5.4 Higher Order Laplacians

In the previous chapter we used higher order Laplacians to solve Poisson's equation. Using higher order approximations has the advantage of improving accuracy, but it has the disadvantage that imposing boundary conditions and Gram factoring are not as simple as for graph Laplacians. We first discuss imposing boundary conditions.

For higher order Laplacians, there are a few ways to deal with boundary conditions. We previously applied the method of images to modify the coefficients of the FDM linear system. Here we consider physically motivated ways to impose the boundary conditions. Maxwell's equations in the Lorenz gauge and without sources imply that the electric and magnetic fields evolve according to the wave equation. Assuming that the domain of interest is surrounded by perfect conductors, the electric field goes to zero everywhere outside the domain. This also implies that the electrical potential is constant everywhere outside the domain. These cases suggest that we implement Neumann and Dirichlet boundary conditions as in the previous section. We consider this for the $k = 2$ central difference formula.

For the electrical potential we implement Neumann boundary conditions as in the previous section, so that we change the way the FDM Laplacian matrix, $\frac{-1}{h^2}L$, acts as below (excluding the prefactor of $-1/h^2$).

$$L\vec{\phi} = \begin{bmatrix} 5/2 & -4/3 & 1/12 & 0 & 0 \\ -4/3 & 5/2 & -4/3 & 1/12 & 0 \\ 1/12 & -4/3 & 5/2 & -4/3 & 1/12 \\ 0 & 1/12 & -4/3 & 5/2 & -4/3 \\ 0 & 0 & 1/12 & -4/3 & 5/2 \end{bmatrix} \begin{bmatrix} \phi_{-2} \\ \phi_{-1} \\ \phi_0 \\ \phi_1 \\ \phi_2 \end{bmatrix} \tag{5.18}$$

$$\mapsto \begin{bmatrix} 5/2 & -4/3 & 1/12 & 0 & 0 \\ -4/3 & 5/2 & -4/3 & 1/12 & 0 \\ 1/12 & -4/3 & 5/2 & -4/3 & 1/12 \\ 0 & 1/12 & -4/3 & 5/2 & -4/3 \\ 0 & 0 & 1/12 & -4/3 & 5/2 \end{bmatrix} \begin{bmatrix} \phi_0 \\ \phi_0 \\ \phi_0 \\ \phi_1 \\ \phi_2 \end{bmatrix} \tag{5.19}$$

$$= \begin{bmatrix} 0 \\ (5/2 - 4/3 - 4/3 + 1/12)\phi_0 + (1/12)\phi_1 \\ (5/2 - 4/3 + 1/12)\phi_0 - (4/3)\phi_1 + (1/12)\phi_2 \\ \cdots \end{bmatrix} \tag{5.20}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1/12 & 1/12 & 0 \\ 0 & 0 & 5/4 & -4/3 & 1/12 \\ 0 & 0 & -5/4 & 5/2 & -4/3 \\ 0 & 0 & 1/12 & -4/3 & 5/2 \end{bmatrix} \begin{bmatrix} \phi_0 \\ \phi_0 \\ \phi_0 \\ \phi_1 \\ \phi_2 \end{bmatrix} \tag{5.21}$$

Even ignoring the vertex indexed by -1, the relevant submatrix is not symmetric, so it can never by written as $BB^\dagger$. This case falls outside of the scope of algorithms which we consider, where the time evolution can be performed using Hamiltonian simulation.

For $\phi = 0$ Dirichlet boundary conditions we modify the Laplacian as below.

$$L\vec{\phi} = \begin{bmatrix} 5/2 & -4/3 & 1/12 & 0 & 0 \\ -4/3 & 5/2 & -4/3 & 1/12 & 0 \\ 1/12 & -4/3 & 5/2 & -4/3 & 1/12 \\ 0 & 1/12 & -4/3 & 5/2 & -4/3 \\ 0 & 0 & 1/12 & -4/3 & 5/2 \end{bmatrix} \begin{bmatrix} \phi_{-2} \\ \phi_{-1} \\ \phi_0 \\ \phi_1 \\ \phi_2 \end{bmatrix} \tag{5.22}$$

$$\mapsto \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5/2 & -4/3 & 1/12 \\ 0 & 0 & -4/3 & 5/2 & -4/3 \\ 0 & 0 & 1/12 & -4/3 & 5/2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \phi_0 \\ \phi_1 \\ \phi_2 \end{bmatrix} \tag{5.23}$$

The resulting Laplacians have the advantage that they are symmetric (since they are based on truncated central difference formulas) and positive-semidefinite, so they can be Gram factored, so Hamiltonian simulation can be used for their time evolution. In contrast, there are non-symmetric Laplacian approximations where the derivative of the field at x_i is approximated using values of the field at points only to the right of x_i.

Unlike graph Laplacians, the Gram factorization of these higher order Laplacians is not as straight forward and requires a classical computation prior to implementing our algorithm. The basic approach is to use the method of undetermined coefficients to set up a quadratic system of equations for the entries of the Gram matrices. This is most simply done when we assume the discretization mesh is a cycle so that L will be circulant and we can write it as a sum of powers of S. For example, for the $k = 2$ Laplacian, we have (again excluding the $-1/h^2$ prefactor)

$$L = ((5/2)I - (4/3)(S + S^\dagger) + (1/12)(S^2 + (S^\dagger)^2)). \tag{5.24}$$

Since this matrix is circulant, we also assume that its Gram matrix is circulant. We define the ansatz $B = cS - (c+b)I + bS^\dagger$ to find a system of equations

$$2(c^2 + b^2 + cb) = 5/2 \tag{5.25}$$

$$cb = 1/12 \tag{5.26}$$

$$(c+b)^2 = 4/3 \tag{5.27}$$

which can then be solved for the Gram matrix entries. Since B has 3 (more generally $k+1$) non-zero entries in each column, we interpret it as a hypergraph where each hyperedge is incident with $k+1$ vertices. Numerical values for the Laplacian coefficients and the entries of their Gram matrices up to $k = 5$ appear in Appendix B of Ref. [33].

This suffices for the Gram matrices of circulant Laplacians. To generate Gram matrices for Laplacians on a path graph mesh, P_n, under Dirichlet boundary conditions, we consider the following procedure.

1. Embed P_n in a cycle C_m for $m > n+k$.
2. Find the Gram matrix B' for the higher order Laplacian on C_m (as described above).
3. Remove from B' any rows for vertices not in P_n and any columns for hyperedges not incident with vertices in P_n.

What remains is the Gram matrix for the Laplacian under Dirichlet boundary conditions. Note that for general kth order Laplacians, this procedure specifies a weighted hypergraph on n vertices with hyperedges that are incident with $k+1$ or fewer vertices. In fact there will be 2 hyperedges incident with j vertices for $1 \leq j \leq k$, and each of these edges will be incident with one of the two end vertices of P_n.

5.5 Multiple Dimensions and Non-Convex Domains

As in the previous chapter, graphs sums [34] (eg, a $n \times n$ grid is the sum of two path graphs on n vertices) allow for simple extensions of 1D Laplacians to higher dimensional Laplacians when the domain is rectangular, ie, of the form $[-1,1]^d$. This is because the d-dimensional Laplacian operator is a sum of 1 dimensional Laplacians, $\nabla^2 = \sum_j \frac{\partial^2}{\partial x_j^2}$, so we can write the FDM matrix as in the statement of Lemma 12, $\frac{-1}{h^2}L' = \frac{-1}{h^2}(L \otimes I^{\otimes d-1} + I \otimes L \otimes I^{\otimes d-2} + \cdots + I^{\otimes d-1} \otimes L)$. This expression applies for periodic boundary conditions (when L is a Laplacian for a cycle graph) and for Dirichlet and Neumann boundary conditions (when L is one of the path graph Laplacians).

Our algorithm can be applied to convex domains other than $[-1,1]^d$ and to non-convex domains by considering appropriate meshes. This mesh can be constructed as follows.

1. Embed the domain in $[-1,1]^d$.
2. Construct the hypergraph for $L \otimes I^{\otimes d-1} + I \otimes L \otimes I^{\otimes d-2} + \cdots + I^{\otimes d-1} \otimes L$ with vertices arranged as in a cubic mesh and corresponding to points in $[-1,1]^d$.
3. Remove all vertices in the mesh whose corresponding points do not lie in the domain. Eliminate the incidence of these vertices with any hyperedge.
4. Remove any hyperedges not incident with one of the domain points.

What remains after this procedure is a weighted hypergraph, and this hypergraph structure provides the natural way to Gram factor the Laplacian matrix for the domain. As a test of this approach for non-convex domains, in Fig. 3 we present numerical simulations of how a wave propagates according to Eqn. 5.6 from its initial condition to 3 later points in time.

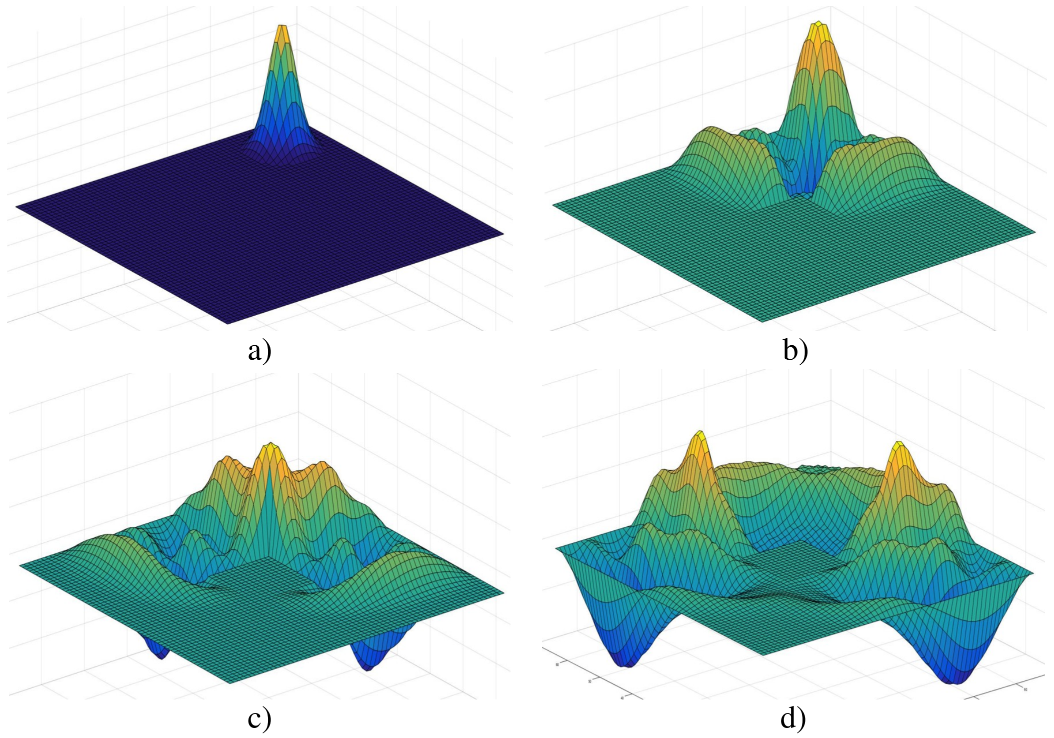
Figure 3: Wave propagation in non-convex domain. Subfigure a) shows the initial condition, and b), c), and d) show the field at later points in time. Figure from Ref. [33] credit to Pedro Costa.

5.6 Initial Conditions

We assume access to controlled oracles that produce states proportional to the initial conditions of the field, $\vec{\phi}(0), \frac{d\vec{\phi}}{dt}|_{t=0}$, and we assume knowledge of the ℓ^2 norms of these vectors. Knowing the norms we can synthesize the unitary

$$U = \frac{1}{\sqrt{||\vec{\phi}(0)||^2 + ||\frac{d\vec{\phi}}{dt}|_{t=0}||^2}} \begin{bmatrix} ||\vec{\phi}(0)|| & -||\frac{d\vec{\phi}}{dt}|_{t=0}|| \\ ||\frac{d\vec{\phi}}{dt}|_{t=0}|| & ||\vec{\phi}(0)|| \end{bmatrix}. \tag{5.28}$$

We solve the Schrödinger equation (instantiating B in Eqn. 5.6)

$$\frac{d}{dt}\begin{bmatrix} \vec{\phi} \\ \vec{\theta} \end{bmatrix} = \frac{-i}{h}\begin{bmatrix} 0 & W \\ W^\dagger & 0 \end{bmatrix}\begin{bmatrix} \vec{\phi} \\ \vec{\theta} \end{bmatrix} \tag{5.29}$$

where W is the incidence matrix or hypergraph incidence matrix for a Laplacian. This implies the following equation for the initial conditions

$$\begin{bmatrix} \frac{d}{dt}\vec{\phi}|_{t=0} \\ \frac{d}{dt}\vec{\theta}|_{t=0} \end{bmatrix} = \frac{-i}{h} \begin{bmatrix} W\vec{\theta}|_{t=0} \\ W^{\dagger}\vec{\phi}|_{t=0} \end{bmatrix} \tag{5.30}$$

so to initialize a state proportional to $\vec{\theta}|_{t=0}$ we must use a QLSA to invert W since we are only given access to an oracle for $\frac{d\vec{\phi}}{dt}|_{t=0}$. So the full state preparation routine is to apply the unitary in Eqn. 5.28 to an ancilla qubit, apply the controlled unitaries for the initial conditions, and then apply the pseudoinverse of B.

5.7 Post-Processing

The post-processing step depends on the state we want to output. Initial value problems are typically stated in terms of outputting the field, which in our case corresponds to outputting $|\vec{\phi}(T)\rangle$; however, due to the introduction of the auxiliary variable we can also output a state proportional to $|\frac{\vec{\phi}(T)}{dt}\rangle$.

If our goal is to output $|\vec{\phi}(T)\rangle$, then we measure the qubit to which we applied the unitary in Eqn. 5.28 during state preparation. This qubit indexes the parts of the state encoding $\vec{\phi}$ and $\vec{\theta}$, and measuring 0 indicates that the remaining state is proportional $\vec{\phi}(T)$.

In general, the success probability of this measurement will depend on the system itself, similar to how our ODE algorithm's success probability depends on the solution's final amplitude relative to its maximum amplitude during its evolution. For example, the field $\phi(x,t) = \sin(x)\cos(t)$ satisfies the wave equation, but when $t = \frac{\pi}{2} + n\pi$ for $n \in \mathbb{Z}$ the field is uniformly zero while its time derivative is non-zero. At these points in time, an accurate simulation of the field would have $\vec{\phi} = \vec{0}$, so the final measurement would almost always output 1, indicating failure.

If our goal is to output a state proportional to $\frac{d\vec{\phi}}{dt}$, then we must post-process the part of the state proportional to $\vec{\theta}$ and multiply by W. A simply way to perform this multiplication is to use a variation of the HHL QLSA as follows (using phase estimation and a controlled rotation):

$$|W^{-1}d\vec{\phi}/dt\rangle|0\rangle|0\rangle = \sum_j \alpha_j|\Lambda_j\rangle|0\rangle|0\rangle \tag{5.31}$$

$$\mapsto \sum_j \alpha_j|\Lambda_j\rangle|\tilde{\lambda}_j\rangle|0\rangle \tag{5.32}$$

$$\mapsto \sum_j \alpha_j|\Lambda_j\rangle|\tilde{\lambda}_j\rangle \left(\frac{\tilde{\lambda}_j}{C}|0\rangle + \frac{\sqrt{C^2-\tilde{\lambda}_j^2}}{C}|1\rangle\right). \tag{5.33}$$

Note that this mainly differs from the HHL procedure in that the eigenvalue now appears in the numerator of the amplitude, corresponding to multiplication instead of inversion. This rotation is followed by post-selecting on the ancilla qubit being $|0\rangle$, which in the worst case occurs with success probability $\min_j|\lambda_j|^2/||W||^2 = \kappa(W)^2$, which can be improved to linear in $\kappa(W)$ scaling as in Ref. [5].

5.8 Complexity Analysis

Instead of stating the complexity for arbitrary domains we consider the complexity of simulating the wave equation in a cubic region with side length l in d dimensions.

A $2k+1$ sparse Laplacian with $O(h^{2k})$ error in 1D has a $k+1$ sparse incidence matrix. Concatenating d such incidence matrices for a d-dimensional Laplacian leads to sparsity $s = d(k+1)$. Assuming the $2k+1$th derivative (with respect to any coordinate) of the solution is bounded by a constant, the error that accumulates after evolving for time T is $\varepsilon = O(Th^{2(s/d)-2})$, which implies the grid spacing scales as $h = \Theta((\varepsilon/T)^{d/2(s-d)})$.

The state preparation procedure has time complexity $\widetilde{O}(sd^{3/2}\ell(T/\varepsilon)^{d/2(s-d)})$, and the time evolution via Hamiltonian simulation has time complexity $\widetilde{O}(sdT(T/\varepsilon)^{d/2(s-d)})$. The query complexities are similarly bounded since we use $\widetilde{O}$ which suppresses $\mathsf{poly}(\log(\cdot))$ overhead factors.

The Hamiltonian simulation algorithm of [15] has gate complexity

$$O\left[\tau\left[n+\log^{5/2}(\tau/\varepsilon)\right]\frac{\log(\tau/\varepsilon)}{\log\log(\tau/\varepsilon)}\right], \tag{5.34}$$

where $\tau = s\|H\|_{\max}t$, $\|H\|_{\max}$ is the largest norm of a matrix element of H, s is the sparsity of H, and n is the number of qubits. For a first order Laplacian, the Hamiltonian we simulate for time evolution has parameters $s = 2d$ (since each vertex is incident with 2 edges), $\|H\|_{\max} = 1/h$, and $n = \log_2\left[(1+d)(l/h)^d\right]$, so the time evolution step has complexity

$$O\left[\frac{dt}{h}\left(\log\left((1+d)(l/h)^d\right)+\log^{5/2}\left(\frac{2dt}{h\varepsilon}\right)\right)\frac{\log\left(\frac{2dt}{h\varepsilon}\right)}{\log\log\left(\frac{2dt}{h\varepsilon}\right)}\right]=\widetilde{O}\left[\frac{td^2}{h}\right] \tag{5.35}$$

and the query complexity for the state preparation is the same up to $\mathsf{poly}(\log(\cdot))$ factors.

5.9 Comparison to ODE approaches

We already discussed the ancilla qubit overheads that a Hamiltonian simulation based algorithm avoids compared to an ODE algorithm for the equation

$$\frac{d}{dt}\begin{bmatrix}\vec{\phi}\\ \vec{\theta}\end{bmatrix}=\begin{bmatrix}0 & I\\ \frac{-1}{h^2}L & 0\end{bmatrix}\begin{bmatrix}\vec{\phi}\\ \vec{\theta}\end{bmatrix}. \tag{5.36}$$

This in fact implies that both $\vec{\phi}$ and $\vec{\theta}$ evolve according to FDM wave equation approximations. Instantiating this for $t = 0$ we find

$$\begin{bmatrix}\frac{d\vec{\phi}}{dt}|_{t=0}\\ \frac{d\vec{\theta}}{dt}|_{t=0}\end{bmatrix}=\begin{bmatrix}\vec{\theta}(0)\\ \frac{-1}{h^2}L\vec{\phi}(0)\end{bmatrix}. \tag{5.37}$$

To prepare the initial state for the ODE we must assume oracle access. We can assume access to oracles for $\vec{\phi}$ and $\frac{d\phi}{dt}$ without loss of generality, in which case the state preparation involves an application of the unitary in Eqn. 5.28 followed by the oracles for the initial conditions. This takes advantage of the asymmetry of the ODE, since assuming oracles for $\vec{\theta}$ and $\frac{d\theta}{dt}$ would require applying the pseudoinverse of L as part of the state preparation, which introduces additional overhead. So we see that using this ODE requires no additional state preparation or post-processing (such as applying QLSAs), but it incurs the overheads of using an ODE algorithm.

Chapter 6: Conclusion

We begin with a summary of the main chapters of this thesis. We follow that with a discussion of how the field has progressed following these results. We conclude by presenting several open problems.

6.1 Summary of Results

In Chapter 2 we described the Hamiltonian simulation algorithm of Ref. [28] which is based on randomly applying different product formulas. We proved technical lemmas characterizing degenerate and non-degenerate terms in the averages of product formulas, and we bounded error terms at different orders in the Taylor expansions of these average operators. This allowed us to characterize the total error for the quantum channel applying randomized product formulas, and we found asymptotically better scaling compared to deterministic product formulas. We also presented numerics estimating the number of elementary exponentials needed to simulate a Heisenberg spin chain using deterministic and randomized product formulas.

In Chapter 3 we presented the ODE algorithm of Ref. [16] which used a QLSA to generate a history state of the ODE solution which we then perform a post-selective measurement on. We proved several technical lemmas bounding the condition number of the linear system, the error due to the Taylor series approximaiton of $\exp(At)$, the success probability of the post-selective measurement, and the complexity of state preparation for the QLSA. We proved that our algorithm scales optimally with respect to the matrix norm, evolution time, and error.

In Chapter 4 we considered applying QLSAs to the Poisson equation. We constructed linear systems where the matrix was a FDM operator approximating the Laplacian and the solution had components equal to the PDE solution on uniformly spaced points in the domain. We bounded the condition number of this matrix and its approximation error. We used these bounds to prove the existence of a time and query efficient quantum algorithm whose complexity is $\mathrm{poly}(\log(1/\varepsilon))$ and exponentially faster with respect to the spatial dimension than comparable classical algorithms. We argued for how to use the method of images to extend our algorithm from periodic boundary conditions to Dirichlet and Neumann boundary conditions. We also illustrated how the FDM can be extended to more general first order BVPs.

In Chapter 5 we presented the wave and Klein-Gordon equation algorithms of Ref. [33] which also used FDM Laplacian approximations. We considered how to transform equations that are second order in time to equations that are first order in time. We found the wave equation and Klein-Gordon equation to be natural candidates, when restricting to Hamiltonian simulation with certain block structure that leads to decoupled second order equations. We saw how to impose Neumann and Dirichlet boundary conditions for first order Laplacians, and we considered Dirichlet boundary conditions for higher order Laplacians. We also considered how QLSAs are used in state preparation and post-processing, depending on oracle assumptions and the desired output.

6.2 Progress on Randomization for Hamiltonian Simulation

Following our work on randomization for Hamiltonian simulation, Campbell proposed an algorithm [21] where evolution under the terms of the Hamiltonian occurs with probability proportional to the strength of each term. Ref. [17] uses similar randomization techniques to simulate time-dependent Hamiltonians.

6.3 Progress on ODEs

Since publishing the ODE algorithm, Childs and Liu [26] have developed algorithms for time-dependent ODEs which also achieve $O(\mathsf{poly}(\log(1/\varepsilon)))$ query complexity. Their algorithm for the initial value problem also achieves optimal scaling with respect to the matrix norm. They also consider the problem of outputing a state at an intermediate time given the initial and final states.

6.4 Nonlinear Differential Equations

For general nonlinearities, the scope of possible quantum speed-ups is limited since there are strong no-go arguments for the simulation of generic NLDEs [3, 7, 29]. The intuition underlying these arguments is that NLDEs can have trajectories that diverge exponentially quickly (ie they can have positive Lyapunov exponents) and that this exponential divergence can be leveraged to speed-up computations and violate known lower bounds (such as for unstructured search or state discrimination).

Despite these limitations, several algorithms for NLDEs have been proposed. The first of these algorithms appeared in Ref. [64] that uses the Euler method to solve the initial value problem for NLDEs, but its resources scale exponentially in the system parameters (as required by the no-go arguments). Somma et. al. [97] study a quantum version of a classical Monte Carlo algorithm for fluid flow. This algorithm achieves a quadratic speed-up over the classical version since quantum computers can sample from Markov chains quadratically faster. Other heuristic algorithms for computational fluid dynamics have been proposed in Refs. [84, 85, 91]. A special linear case of the Vlasov equation (which is generally nonlinear) is treated in the algorithm of Ref. [39]. Ref. [70] uses variational algorithms to find the ground state of the nonlinear Schrödinger equation and uses the FDM; although efficient time evolution algorithms for the nonlinear Scrhödinger equation are forbidden by no-go arguments.

Since we can not hope to efficiently simulate arbitrary NLDEs, we can ask the question "What kinds of NLDEs can quantum computers simulate efficiently?" The no-go arguments rely on the existence of positive Lyapunov exponents, so they do not apply to certain dissipative systems (as an example) which might be efficiently simulable. This raises the question of what other lower bounds and no-go arguments can be developed for NLDEs without positive Lyapunov exponents.

This also raises the question of how imposing other kinds of structure on NLDEs might make them efficently simulable. Conservation laws are one source of additional structure; however, just having a conserved quantity such as energy is not sufficient since Ref. [77] shows that simulating point masses evolving classically under gravitational or Coulomb interactions is PSPACE-hard.

Another source of structure is to impose that the NLDE is a transformation of a linear DE. For example, the Korteweg-de Vries equation is equivalent to the heat equation under the Cole-Hopf transformation [40]. Another such equation is the Riccati equation. Radon's Lemma [2] characterizes the solutions of the Riccati equation in terms of the solutions of a linear ODE. In this case, existing algorithms for Hamiltonian simulation, ODEs, and linear systems can be pieced together to form quantum algorithms for the original nonlinear problem (albeit with caveats). We provide a sketch of this algorithm in Appendix B.

The algorithm for the Riccati equation is basically several standard quantum algorithms pieced together in a natural way. Another case where standard quantum algorithms can be applied is to bound the moments of NLDEs, which can be done using semidefinite programs (SDPs) as shown in Refs. [23, 41, 50]. Quantum algorithms for SDPs have been studied in Refs. [18, 56, 92, 93] and allow for polynomial speed-ups over classical algorithms. It is natural to ask if this speed-up still holds when bounding the moments of NLDEs (and under what conditions or oracle models).

6.5 Other Open Questions

After developing our Hamiltonian simulation algorithm, Ref. [66] described 'quantum interpolation', a method to implement certain unitaries which can be specialized to Hamiltonian simulation to generate new product formulas. This result can naturally be combined with the randomization tools we used for our Hamiltonian simulation algorithm. When initially developing our randomized algorithm, we found continuous families of 'randomized quantum interpolation product formula'. However, even characterizing these algorithms requires classical precomputation which seems to be prohibitively costly. Is there a way to get around this precomputation? One of the advantages of ordinary Suzuki-Trotter formulas is that they are defined recursively, so formulas of arbitrarily high accuracy can be developed. How can this be done for quantum interpolation?

Our ODE algorithm has multiplicative scaling with respect to t and ε (eg $\tilde{O}(t\log(1/\varepsilon))$) while the state of the art for Hamiltonian simulation is additive with respect to these terms (eg $\tilde{O}(t+\log(1\varepsilon))$ [9, 68, 69]). Can additive scaling be achieved for ODEs?

We only considered the FDM for uniformly spaced lattices; however, using a non-uniformly spaced mesh might be more natural in some applications (eg having a denser grid where the field is expected to change the most). It is natural to ask what kind of promises must be made about a class of DEs such that non-uniform mesh algorithms are faster within that class. The analysis of non-uniform mesh algorithms is complicated by the lack of symmetry in the formulas (symmetry which we made use of and in practice may be easier to implement). What techniques can be used to bound their complexities?

Our FDM algorithms assumed oracles for producing the states corresponding to inhomogeneities and initial conditions. For practical problems, what methods might be used to realize these oracles beyond the methods of Refs. [51, 79, 98]?

Our PDE algorithms provide exponential speed-ups with respect to the dimension, but for fixed dimension they may only give polynomial improvements with respect to the approximation error. What kind of quantum speed-ups can be achieved in fixed dimension? What classical lower bounds and quantum upper bounds can we prove for the complexity of these PDE problems?

In the algorithms for the wave equation and Klein-Gordon equation, we imposed $\phi = 0$ Dirichlet boundary conditions on high order Laplacians by truncating a central difference formula FDM matrix. We derived this by starting with arbitrarily large rectangular meshes encompassing the domain mesh and fixing the field to be 0 on mesh points lying outside the domain of the differential equation (motivated by the physical example of electric fields going to 0 inside conductors). An alternative approach would be to start with the nodes in the domain and only add a single layer of ghost nodes that lie on the boundary. Then the value of the field at the boundary points can be fixed, and the derivatives can be approximated near the boundary using formulas other than the central difference formulas. This would in fact allow for generic Dirichlet boundary conditions to be imposed. The downside to this approach is that the Laplacian operators are not symmetric, so instead of Hamiltonian simulation an ODE algorithm must be used including an inhomogeneous term for non-trivial boundary conditions.

In the wave equation algorithm, to use higher order Laplacians, we must find their Gram factorizations, which requires a classical pre-computation. What is the most efficient way to perform this? Are there simple analytical expressions for the entries of the Gram matrices? In Appendix B of Ref. [33], we discuss FDM Laplacians in 2+ dimensions that are not simply linear combinations of 1D Laplacians. How can these be efficiently Gram factored? How does their performance compare to the Laplacians we propose?

In Appendix A we sketch an argument for the BPP-hardness of inverting Laplacian matrices for general graphs (not just path graphs as considered for PDEs). Formalizing this remains to be done. Using the algorithm of Ref. [36], can this be made into a completeness result?

The algorithms we have presented solve problems where the output is a quantum state that is proportional to the solution of an ODE or a PDE. However, to learn anything about the solutions to the DEs we must post-process these states, eg, by estimating observables, which is a different computational task and will require additional overhead not considered here.

Appendix A: BPP-hardness of Laplacian Linear Systems

Following Refs. [36, 46] developing BPL algorithms for solving Laplacian linear systems, Ref. [37, 38], showed that approximating the second eigenvalue of graph Laplacians is BPL-hard. Here we sketch a heuristic argument that estimating $\vec{v}^T\vec{x}$ is BPP-hard where $\vec{x}$ is the solution of an exponentially large Laplacian linear system and $\vec{v}$ is a specially constructed vector.

This hardness reduction is inspired by the HHL construction that shows that inverting sparse, exponentially large matrices is BQP-complete. The HHL construction relies on a decomposition of the quantum circuit into a universal set of local unitaries, such as the Hadamard, phase, and controlled-not gates. For the BPP-hard construction we use the fact that the Toffoli (doubly controlled-not) gate is universal for classical deterministic computation. Since the Toffoli gate is deterministic but BPP is a language for random computation, we model the randomness by augmenting the computational bits with random coin bits; randomness is introduced into the computational bit space by controlling on the values of these coin bits.

Suppose we are given a classical circuit C operating on a space of computational bits and coin bits such that sampling over the space of coin bits approximates a BPP computation. Further suppose we have a decomposition of this circuit into a sequence of m Toffoli gates, ie, $C = \prod_{j=1}^{m} T_j$. Without loss of generality, we can assume that the circuit performs the computation, copies the output bit to an output register, and then uncomputes any 'garbage' in the computational bits. Then the output of the circuit is a set of bits all of which are set to 0 except for the output bit and the random coin bits.

As in HHL we first construct an asymmetric matrix.

$$M = I - \exp(-1/m)Q \tag{A.1}$$

$$Q \equiv \sum_{j=1}^{m} |j+1\rangle\langle j| \otimes T_j - \sum_{j=m+1}^{2m} |j+1\rangle\langle j| \otimes I - \sum_{j=2m+1}^{3m} |j+1 \mod 3m\rangle\langle j| \otimes T^{\dagger}_{3m+1-j} \tag{A.2}$$

The condition number of this matrix is $\Theta(m)$, ie, it scales with the number of Toffoli gates in the classical circuit.

To see how this matrix can be used to encode a computation, consider the linear system

$$M|x\rangle = |0\rangle_{clock}|0\rangle_{comp}|c\rangle_{coin} \tag{A.3}$$

where $|0\rangle_{clock}$ is in the $3m$-dimensional 'clock' Hilbert space, $|0\rangle_{comp}$ lies in the computational space, and $|c\rangle_{coin}$ lies in the coin space. The solution of this linear system is the history state

$$|x\rangle = \frac{1}{1-\exp(-3)} \sum_{j=1}^{3m} |j\rangle|\psi_j\rangle \tag{A.4}$$

where

$$|\psi_j\rangle = \begin{cases} |0\rangle|c\rangle & j=1 \\ \prod_{k=1}^{j-1} T_k|0\rangle|c\rangle & 2 \leq j \leq m \\ \prod_{k=1}^{m} T_k|0\rangle|c\rangle & m+1 \leq j \leq 2m \\ \prod_{k=1}^{3m+1-j} T_k|0\rangle|c\rangle & 2m+1 \leq j \leq 3m \end{cases} . \tag{A.5}$$

Note that when the clock register encodes a value between $m+1$ and $2m$, the rest of the state corresponds to the state after the computation where all the bits are guaranteed to be zero except for the output bit and the coin bits.

If the coin state $|c\rangle_{coin}$ corresponds to a distribution of coins, then the quantity $\langle y|x\rangle$ is proportional to the number of computational paths (in the distribution) outputing 0 minus the number of computational paths outputing 1, where

$$|y\rangle = \sum_{j=m+1}^{2m} |j\rangle|-\rangle_{output}|0\rangle_{comp\backslash output}|c\rangle_{coin}. \tag{A.6}$$

Thus if $|c\rangle_{coin}$ corresponds to a sufficiently large distribution of coins, then $\langle y|x\rangle$ is positive (or negative) if the output of the BPP computation is 0 (or 1) with high probability.

Now that we have argued that the solution of this linear system is BPP-hard, it remains to conver the linear system into a graph Laplacian linear system. We begin by symmetrizing the matrix

$$M^{\dagger}M = (1+\exp(-2/m))\mathbb{1} - \exp(-1/m)(Q+Q^{\dagger}) \tag{A.7}$$

where we have used the fact that Q is unitary. The linear system then reads $M^{\dagger}M|x\rangle = M^{\dagger}|0\rangle_{clock}|0\rangle_{comp}|c\rangle_{coin}$.

Since each row of Q has exactly one non-zero entry which is equal to 1, this matrix is symmetric diagonally dominant with negative entries off of the diagonal. Now the linear system can be converted to a Laplacian linear system using some standard tricks, namely we map the matrix and vectors according to

$$M^{\dagger}M \mapsto \exp(-1/m)(2I - Q + Q^{\dagger}) \otimes I_2 + (1+\exp(-2/m) - 2\exp(-1/m))I \otimes [(I_2 - X)/2] \tag{A.8}$$

$$\vec{v} \mapsto \vec{v} \otimes |-\rangle \tag{A.9}$$

where I_2 denotes the 2×2 identity matrix. Note that $[(I_2 - X)/2] = |-\rangle\langle -|$, so this new matrix acts on vectors of the form $\vec{v}\otimes|-\rangle$ equivalently to how $M^{\dagger}M$ acts on the vector $|v\rangle$. The matrix produced by this mapping corresponds to a weighted graph Laplacian.

Appendix B: Riccati Equation Algorithm

In this appendix we sketch a heuristic quantum algorithm for solving a special case of the Riccati equation and then comment on how this relates to the more general Riccati equation.

The problem we consider is:

QUANTUM RICCATI EQUATION SIMULATION: Let V, a $N \times N$ matrix, evolve according to the differential equation

$$\frac{dV}{dt} = I + iHV + V^2 \tag{B.1}$$

where H is a Hamiltonian which we can simulate. Let the initial condition $U(0)$ be a unitary. Given access to an oracle that implements a controlled applicaiton of $U(0)$, produce a quantum state proportional to one of the columns of $U(T)$.

B.1 Linearizing the Riccati Equation

Under the transformation $V = (\frac{dU}{dt})U^{-1}$, Eqn. B.1 is equivalent to the following ODE.

$$\frac{d}{dt}\begin{bmatrix} U \\ dU/dt \end{bmatrix} = \begin{bmatrix} 0 & I \\ -I & iH \end{bmatrix}\begin{bmatrix} U \\ dU/dt \end{bmatrix} \tag{B.2}$$

The square matrix on the RHS above is anti-Hermitian, so the columns of $\begin{bmatrix} U \\ dU/dt \end{bmatrix}$ obey the Schrödinger equation and we can use Hamiltonian simulation algorithms for the time evolution.

B.2 Qubitization and Matrix Inversion

One of the difficulties with this approach is that at the end of the linear evolution we have to transform back to the original V coordinates. This requires applying $U(T)^{-1}$; however, non-sparse models of Hamiltonian simulation (such as qubitization or block encoding) can be used to implement this transformation. Note that

$$X \otimes H = (1/\sqrt{2}) \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \\ 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \tag{B.3}$$

and define $\bar{W}(0)$ as

$$\bar{W}(0) = W_1 W_2 (X \otimes H \otimes I) W_1^\dagger W_2^\dagger \quad = \begin{bmatrix} 0 & 0 & U & U \\ 0 & 0 & (dU/dt) & -(dU/dt) \\ U^\dagger & (dU/dt)^\dagger & 0 & 0 \\ U^\dagger & -(dU/dt)^\dagger & 0 & 0 \end{bmatrix} \tag{B.4}$$

$$W_1 = |00\rangle\langle 00| \otimes U(0) + (I - |00\rangle\langle 00|) \otimes I \tag{B.5}$$

$$W_2 = |01\rangle\langle 01| \otimes (dU/dt)(0) + (I - |01\rangle\langle 01|) \otimes I \tag{B.6}$$

Since U and dU/dt are only given to us at $t = 0$ we need to evolve $\bar{W}$ forward in time. More specifically, we need to perform the inversion $U(T)^{-1}$. We can achieve this by evolving the relevant subblocks of this matrix according to Eqn. B.2.

$$\bar{W}(T) = \exp \begin{bmatrix} 0 & I & 0 & 0 \\ -I & iH & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & U & U \\ 0 & 0 & (dU/dt) & -(dU/dt) \\ U^\dagger & (dU/dt)^\dagger & 0 & 0 \\ U^\dagger & -(dU/dt)^\dagger & 0 & 0 \end{bmatrix} \exp \begin{bmatrix} 0 & -I & 0 & 0 \\ I & -iH & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{B.7}$$

$$= \begin{bmatrix} 0 & 0 & U(T) & Q(T) \\ 0 & 0 & (dU/dt)|_T & (dQ/dt)|_T \\ U(T)^\dagger & (dU/dt)^\dagger|_T & 0 & 0 \\ Q(T)^\dagger & (dQ/dt)^\dagger|_T & 0 & 0 \end{bmatrix} \tag{B.8}$$

where $Q(T)$ is the solution of the differential equation produced from the initial conditions $Q(0) = U(0)$ and $dQ/dt|_0 = -dU/dt|_0$. We can now apply block encoding to select the submatrix of $\bar{W}(T)$ that corresponds to the Hamiltonian which will be simulated when applying $U(T)^{-1}$.

$$[I \otimes \langle 0| \otimes I]\bar{W}(T)[I \otimes |0\rangle \otimes I] = \begin{bmatrix} 0 & U(T) \\ U(T)^\dagger & 0 \end{bmatrix} \tag{B.9}$$

This suffices to show that we can in principle simulate the Hamiltonian $\begin{bmatrix} 0 & U(T) \\ U(T)^\dagger & 0 \end{bmatrix}$.

B.3 The Full Algorithm

Suppose that we are asked to produce a state proportional to the jth column of $V(T) = (dU/dt)|_T U(T)^{-1}$.

1. Prepare the state $|j\rangle$.

2. Apply the inverse $U(T)^{-1}$ using a QLSA whose Hamiltonian simulation subroutine makes uses of the block encoding discussed above.

3. Multiply by $(dU/dt)|_T$ (eg by modifying HHL for multiplication instead of inver-

sion).

Alternatively, if we are asked to produce a state proportional to the jth row of $V(T)$ we can do the following.

1. Prepare the state $|01\rangle|j\rangle$.
2. Apply the unitary $\bar{W}(T)$.
3. Apply $(U(T)^{\dagger})^{-1}$ to the relevant subspace using a QLSA and the block encoding discussed above.
4. Measure the first two qubits, post-selecting on them being in the state $|10\rangle$.

These are merely heuristic outlines for the algorithms for simulating the Ricatti equation. A formal analysis of these algorithms will reveal that several algorithmic overheads exist, such as dependence on the condition numbers of $U(T)$ and $(dU/dt)_T$ and dependence on post-selection probabilities.

B.4 Radon's Lemma

The kind of linearization seen in the last section is summarized for more genera Riccati equations by Radon's Lemma. We restate this lemma here (Theorem 3.1.1 from Ref. [2]).

For the purposes of this lemma, we define the Riccati Differential Equation as

$$\frac{dW}{dt} = M_{2,1}(t) + M_{2,2}(t)W - WM_{1,1}(t) - WM_{1,2}(t)W \quad (RDE) \tag{B.10}$$

Lemma 14. *(Radon's Lemma) Let $M_{1,1} \in \mathbb{R}^{n\times n}$, $M_{1,2} \in \mathbb{R}^{n\times m}$, $M_{2,1} \in \mathbb{R}^{m\times n}$, $M_{2,2} \in \mathbb{R}^{m\times m}$, then the following holds:*

1. *Let $W(t) \in \mathbb{R}^{m\times n}$ be a solution of RDE in the interval $[t_0,t_f] \subset \mathbb{R}$. If $Q, Q(t) \in \mathbb{R}^{n\times n}$ is a solution of the IVP*

$$\frac{dQ}{dt} = (M_{1,1} + M_{1,2}W)Q, \;\; Q(t_0) = 1 \tag{B.11}$$

and $P(t) := W(t)Q(t)$, then $\begin{bmatrix} Q \\ P \end{bmatrix}$ is a solution of the associated linear system (of differential equations)

$$\frac{d}{dt}\begin{bmatrix} Q \\ P \end{bmatrix} = \begin{bmatrix} M_{1,1} & M_{1,2} \\ M_{2,1} & M_{2,2} \end{bmatrix}\begin{bmatrix} Q \\ P \end{bmatrix}. \qquad (L) \tag{B.12}$$

2. *If $\begin{bmatrix} Q \\ P \end{bmatrix}$ is a real solution of the system L such that $Q(t) \in \mathbb{R}^{n\times n}$ is regular for $t \in [t_0,t_f] \subset \mathbb{R}$, then*

$$W : [t_0,t_f] \to \mathbb{R}^{m,n}, \;\; t \mapsto P(t)Q^{-1}(t) = W(t) \tag{B.13}$$

is a real solution of RDE.

3. *In case of a complex parameter t and $M_{1,1} \in \mathbb{C}^{n\times n}$, $M_{1,2} \in \mathbb{C}^{n\times m}$, $M_{2,1} \in \mathbb{C}^{m\times n}$, $M_{2,2} \in \mathbb{C}^{m\times m}$, and also $W(t) \in \mathbb{C}^{m\times n}$, the assertions 1 and 2 remain valid if we replace therein the interval $[t_0,t_f]$ by an arbitrary domain $G \subset \mathbb{C}$ with $t_0 \in G$.*

Simulating the RDE requires modifying the algorithm that we sketched in the previous section. Linearizing the RDE produces an ODE which is not necessarily an instance of Schrödinger's equation, so a coherent ODE algorithm (such as that in Ref. [16]) must be used for the time evolution. If the algorithm in Ref. [16] is used, then during the matrix inversion step, the subblock selection must take into account the structure of the history state as well as the success/failure flag qubits used in the QLSA.

Bibliography

[1] Scott Aaronson. Quantum computing, postselection, and probabilistic polynomial-time. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 461(2063):3473–3482, 2005.

[2] Hisham Abou-Kandil, Gerhard Freiling, Vlad Ionescu, and Gerhard Jank. *Matrix Riccati Equations in Control and Systems Theory*. Birkhäuser, 2012.

[3] Daniel S Abrams and Seth Lloyd. Nonlinear Quantum Mechanics Implies Polynomial-Time Solution for *NP*-complete and *#P* problems. *Physical Review Letters*, 81(18):3992, 1998.

[4] Dorit Aharonov and Amnon Ta-Shma. Adiabatic quantum state generation and statistical zero knowledge. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 20–29. ACM, 2003.

[5] Andris Ambainis. Variable time amplitude amplification and a faster quantum algorithm for solving systems of linear equations. *arXiv preprint arXiv:1010.4458*, 2010.

[6] Alex Aleksandr Arkhipov. *Extending and Characterizing Quantum Magic Games*. PhD thesis, Massachusetts Institute of Technology, 2012.

[7] Ning Bao, Adam Bouland, and Stephen P Jordan. Grover Search and the No-Signaling Principle. *Physical Review Letters*, 117(12):120501, 2016.

[8] Michael Jarret Baume. *Spectral graph theory with applications to quantum adiabatic optimization*. PhD thesis, 2016.

[9] Dominic Berry and Leonardo Novo. Corrected quantum walk for optimal hamiltonian simulation. *Quantum Information & Computation*, 16(15-16):1295–1317, 2016.

[10] Dominic W Berry. High-order quantum algorithm for solving linear differential equations. *Journal of Physics A: Mathematical and Theoretical*, 47(10):105301, 2014.

[11] Dominic W Berry, Graeme Ahokas, Richard Cleve, and Barry C Sanders. Efficient Quantum Algorithms for Simulating Sparse Hamiltonians. *Communications in Mathematical Physics*, 270(2):359–371, 2007.

[12] Dominic W. Berry and Andrew M. Childs. Black-box Hamiltonian Simulation and Unitary Implementation. *Quantum Info. Comput.*, 12(1-2):29–62, January 2012.

[13] Dominic W Berry, Andrew M Childs, Richard Cleve, Robin Kothari, and Rolando D Somma. Simulating Hamiltonian Dynamics with a Truncated Taylor Series. *Physical Review Letters*, 114(9):090502, 2015.

[14] Dominic W Berry, Andrew M Childs, Richard Cleve, Robin Kothari, and Rolando D Somma. Exponential improvement in precision for simulating sparse hamiltonians. *Forum of Mathematics, Sigma*, 5:e8, 2017.

[15] Dominic W Berry, Andrew M Childs, and Robin Kothari. Hamiltonian Simulation with Nearly Optimal Dependence on all Parameters. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 792–809. IEEE, 2015.

[16] Dominic W Berry, Andrew M Childs, Aaron Ostrander, and Guoming Wang. Quantum Algorithm for Linear Differential Equations with Exponentially Improved Dependence on Precision. *Communications in Mathematical Physics*, 356(3):1057–1081, 2017.

[17] Dominic W Berry, Andrew M Childs, Yuan Su, Xin Wang, and Nathan Wiebe. Time-dependent Hamiltonian simulation with L^1-norm scaling. *arXiv preprint arXiv:1906.07115*, 2019.

[18] Fernando GSL Brandao, Amir Kalev, Tongyang Li, Cedric Yen-Yu Lin, Krysta M Svore, and Xiaodi Wu. Quantum SDP Solvers: Large Speed-ups, Optimality, and Applications to Quantum Learning. *arXiv preprint arXiv:1710.02581*, 2017.

[19] Sergey Bravyi, David P Divincenzo, Roberto Oliveira, and Barbara M Terhal. The Complexity of Stoquastic Local Hamiltonian Problems. *Quantum Information & Computation*, 8(5):361–385, 2008.

[20] Earl Campbell. Shorter gate sequences for quantum computing by mixing unitaries. *Physical Review A*, 95(4):042306, 2017.

[21] Earl Campbell. A random compiler for fast Hamiltonian simulation. *arXiv preprint arXiv:1811.08017*, 2018.

[22] Yudong Cao, Anargyros Papageorgiou, Iasonas Petras, Joseph Traub, and Sabre Kais. Quantum algorithm and circuit design solving the Poisson equation. *New Journal of Physics*, 15(1):013021, 2013.

[23] Sergei I Chernyshenko, P Goulart, D Huang, and Antonis Papachristodoulou. Polynomial sum of squares in fluid dynamics: a review with a look ahead. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 372(2020):20130350, 2014.

[24] Andrew M Childs. On the Relationship Between Continuous- and Discrete-Time Quantum Walk. *Communications in Mathematical Physics*, 294(2):581–603, 2010.

[25] Andrew M Childs, Robin Kothari, and Rolando D Somma. Quantum Algorithm for Systems of Linear Equations with Exponentially Improved Dependence on Precision. *SIAM Journal on Computing*, 46(6):1920–1950, 2017.

[26] Andrew M Childs and Jin-Peng Liu. Quantum spectral methods for differential equations. *arXiv preprint arXiv:1901.00961*, 2019.

[27] Andrew M Childs, Dmitri Maslov, Yunseong Nam, Neil J Ross, and Yuan Su. Toward the first quantum simulation with quantum speedup. *Proceedings of the National Academy of Sciences*, 115(38):9456–9461, 2018.

[28] Andrew M. Childs, Aaron Ostrander, and Yuan Su. Faster quantum simulation by randomization. *Quantum*, 3:182, September 2019.

[29] Andrew M Childs and Joshua Young. Optimal state discrimination and unstructured search in nonlinear quantum mechanics. *Physical Review A*, 93(2):022314, 2016.

[30] Fan RK Chung. *Spectral Graph Theory*. Number 92. American Mathematical Soc., 1997.

[31] B David Clader, Bryan C Jacobs, and Chad R Sprouse. Preconditioned Quantum Linear System Algorithm. *Physical Review Letters*, 110(25):250504, 2013.

[32] Daniel T Colbert and William H Miller. A novel discrete variable representation for quantum mechanical reactive scattering via the S-matrix Kohn method. *The Journal of chemical physics*, 96(3):1982–1991, 1992.

[33] Pedro CS Costa, Stephen Jordan, and Aaron Ostrander. Quantum algorithm for simulating the wave equation. *Physical Review A*, 99(1):012323, 2019.

[34] Dragoš Cvetković, Peter Rowlinson, and Slobodan Simić. An Introduction to the Theory of Graph Spectra. *Cambridge-New York*, 2010.

[35] Danial Dervovic, Mark Herbster, Peter Mountney, Simone Severini, Naïri Usher, and Leonard Wossnig. Quantum linear systems algorithms: a primer. *arXiv preprint arXiv:1802.08227*, 2018.

[36] Dean Doron, François Le Gall, and Amnon Ta-Shma. Probabilistic Logarithmic-Space Algorithms for Laplacian Solvers. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[37] Dean Doron, Amir Sarid, and Amnon Ta-Shma. On Approximating the Eigenvalues of Stochastic Matrices in Probabilistic Logspace. *computational complexity*, 26(2):393–420, 2017.

[38] Dean Doron and Amnon Ta-Shma. On the Problem of Approximating the Eigenvalues of Undirected Graphs in Probabilistic Logspace. In *International Colloquium on Automata, Languages, and Programming*, pages 419–431. Springer, 2015.

[39] Alexander Engel, Graeme Smith, and Scott E Parker. A Quantum Algorithm for the Vlasov Equation. *arXiv preprint arXiv:1907.09418*, 2019.

[40] Lawrence C. Evans. *Partial Differential Equations*. American Mathematical Society, Providence, R.I., 2010.

[41] Giovanni Fantuzzi, David Goluskin, Deqing Huang, and Sergei I Chernyshenko. Bounds for Deterministic and Stochastic Dynamical Systems using Sum-of-Squares Optimization. *SIAM Journal on Applied Dynamical Systems*, 15(4):1962–1988, 2016.

[42] Richard P Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6):467–488, 1982.

[43] Caroline Figgatt, Aaron Ostrander, Norbert M Linke, Kevin A Landsman, Daiwei Zhu, Dmitri Maslov, and Christopher Monroe. Parallel entangling operations on a universal ion-trap quantum computer. *Nature*, 572(7769):368–372, 2019.

[44] F Fillion-Gourdeau and Emmanuel Lorin. Simple digital quantum algorithm for symmetric first-order linear hyperbolic systems. *Numerical Algorithms*, pages 1–37, 2018.

[45] François Fillion-Gourdeau, Steve MacLean, and Raymond Laflamme. Algorithm for the solution of the Dirac equation on digital quantum computers. *Physical Review A*, 95(4):042343, 2017.

[46] François Le Gall. Solving Laplacian Systems in Logarithmic Space. *arXiv preprint arXiv:1608.01426*, 2016.

[47] András Gilyén, Srinivasan Arunachalam, and Nathan Wiebe. Optimizing quantum optimization algorithms via faster quantum gradient computation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1425–1444. Society for Industrial and Applied Mathematics, 2019.

[48] András Gilyén, Seth Lloyd, and Ewin Tang. Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension. *arXiv preprint arXiv:1811.04909*, 2018.

[49] Chris Godsil and Gordon F Royle. *Algebraic Graph Theory*, volume 207. Springer Science & Business Media, 2013.

[50] David Goluskin. Bounding Averages Rigorously Using Semidefinite Programming: Mean Moments of the Lorenz System. *Journal of NonLinear Science*, 28(2):621–651, 2018.

[51] Lov Grover and Terry Rudolph. Creating superpositions that correspond to efficiently integrable probability distributions. *arXiv preprint quant-ph/0208112*, 2002.

[52] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum Algorithm for Linear Systems of Equations. *Physical Review Letters*, 103(15):150502, 2009.

[53] Matthew B Hastings. Turning Gate Synthesis Errors into Incoherent Errors. *Quantum Information & Computation*, 17(5-6):488–494, 2017.

[54] Roger A Horn and Charles R Johnson. *Matrix Analysis*. Cambridge university press, 2012.

[55] Iordanis Kerenidis and Anupam Prakash. Quantum Recommendation Systems. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67, page 49. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017.

[56] Iordanis Kerenidis and Anupam Prakash. A Quantum Interior Point Method for LPs and SDPs. *arXiv preprint arXiv:1808.09266*, 2018.

[57] Ishtiaq Rasool Khan and Ryoji Ohba. Closed-form expressions for the finite difference approximations of first and higher derivatives based on Taylor series. *Journal of Computational and Applied Mathematics*, 107(2):179–193, 1999.

[58] Ishtiaq Rasool Khan and Ryoji Ohba. Taylor series based finite difference approximations of higher-degree derivatives. *Journal of Computational and Applied Mathematics*, 154(1):115–124, 2003.

[59] Ishtiaq Rasool Khan, Ryoji Ohba, and Noriyuki Hozumi. Mathematical proof of closed form expressions for finite difference approximations based on Taylor series. *Journal of Computational and Applied Mathematics*, 150(2):303–309, 2003.

[60] A Yu Kitaev. Quantum measurements and the Abelian Stabilizer Problem. *arXiv preprint quant-ph/9511026*, 1995.

[61] Ian D Kivlichan, Nathan Wiebe, Ryan Babbush, and Alán Aspuru-Guzik. Bounding the costs of quantum simulation of many-body physics in real space. *Journal of Physics A: Mathematical and Theoretical*, 50(30):305301, 2017.

[62] Robin Kothari. Efficient algorithms in quantum query complexity. 2014.

[63] Randall J LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*, volume 98. Siam, 2007.

[64] Sarah K Leyton and Tobias J Osborne. A quantum algorithm to solve nonlinear differential equations. *arXiv preprint arXiv:0812.4423*, 2008.

[65] Jianping Li. General explicit difference formulas for numerical differentiation. *Journal of Computational and Applied Mathematics*, 183(1):29–52, 2005.

[66] Yi-Xiang Liu, Jordan Hines, Ashok Ajoy, and Paola Cappellaro. Quantum interpolation for digital quantum simulation. *arXiv preprint arXiv:1903.01654*, 2019.

[67] Seth Lloyd. Universal Quantum Simulators. *Science*, pages 1073–1078, 1996.

[68] Guang Hao Low and Isaac L Chuang. Optimal Hamiltonian Simulation by Quantum Signal Processing. *Physical Review Letters*, 118(1):010501, 2017.

[69] Guang Hao Low and Isaac L Chuang. Hamiltonian Simulation by Qubitization. *Quantum*, 3:163, 2019.

[70] Michael Lubasch, Jaewoo Joo, Pierre Moinier, Martin Kiffner, and Dieter Jaksch. Variational Quantum Algorithms for Nonlinear Problems. *arXiv preprint arXiv:1907.09032*, 2019.

[71] Anuradha Mahasinghe. Vibration Analysis of Cyclic Symmetrical Systems by Quantum Algorithms. *Mathematical Problems in Engineering*, 2019, 2019.

[72] R Meyer. Trigonometric Interpolation Method for One-Dimensional Quantum-Mechanical Problems. *The Journal of Chemical Physics*, 52(4):2053–2059, 1970.

[73] Ashley Montanaro and Sam Pallister. Quantum algorithms and the finite element method. *Physical Review A*, 93(3):032324, 2016.

[74] Michael A Nielsen and Isaac Chuang. Quantum Computation and Quantum Information, 2002.

[75] Anargyros Papageorgiou and Joseph F Traub. Measures of quantum computing speedup. *Physical Review A*, 88(2):022316, 2013.

[76] David Poulin, Angie Qarry, Rolando Somma, and Frank Verstraete. Quantum Simulation of Time-Dependent Hamiltonians and the Convenient Illusion of Hilbert Space. *Physical Review Letters*, 106(17):170501, 2011.

[77] John H Reif and Stephen R Tate. The complexity of N-body simulation. In *International Colloquium on Automata, Languages, and Programming*, pages 162–176. Springer, Berlin, Heidelberg, 1993.

[78] Lorenzo Adlai Sadun. *Applied Linear Algebra: The Decoupling Principle*. American Mathematical Soc., 2007.

[79] Yuval R Sanders, Guang Hao Low, Artur Scherer, and Dominic W Berry. Black-Box Quantum State Preparation without Arithmetic. *Physical Review Letters*, 122(2):020502, 2019.

[80] Artur Scherer, Benoît Valiron, Siun-Chuon Mau, Scott Alexander, Eric Van den Berg, and Thomas E Chapuran. Concrete resource analysis of the quantum linear-system algorithm used to compute the electromagnetic scattering cross section of a 2D target. *Quantum Information Processing*, 16(3):60, 2017.

[81] Norbert Schuch and Jens Siewert. Programmable Networks for Quantum Algorithms. *Physical Review Letters*, 91(2):027902, 2003.

[82] Siddhartha Sinha and Peter Russer. Quantum computing algorithm for electromagnetic field simulation. *Quantum Information Processing*, 9(3):385–404, 2010.

[83] D. Spielman. Rings, Paths, and Cayley Graphs (Course Notes), 2014.

[84] René Steijl. Quantum Algorithms for Fluid Simulations. 2019.

[85] René Steijl and George N Barakos. Parallel evaluation of quantum algorithms for computational fluid dynamics. *Computers & Fluids*, 173:22–28, 2018.

[86] Yiğit Subaşı, Rolando D Somma, and Davide Orsucci. Quantum Algorithms for Systems of Linear Equations Inspired by Adiabatic Quantum Computing. *Physical Review Letters*, 122(6):060504, 2019.

[87] Masuo Suzuki. Generalized Trotter's formula and systematic approximants of exponential operators and inner derivations with applications to many-body problems. *Communications in Mathematical Physics*, 51(2):183–190, 1976.

[88] Masuo Suzuki. Fractal decomposition of exponential operators with applications to many-body theories and Monte Carlo simulations. *Physics Letters A*, 146(6):319–323, 1990.

[89] Masuo Suzuki. General theory of fractal path integrals with applications to many-body theories and statistical physics. *Journal of Mathematical Physics*, 32(2):400–407, 1991.

[90] Ewin Tang. A quantum-inspired classical algorithm for recommendation systems. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 217–228. ACM, 2019.

[91] Blaga Todorova and Rene Steijl. Quantum Algorithm for Collisionless Boltzmann Equation based on the Reservoir Technique. In *AIAA Scitech 2019 Forum*, page 1406, 2019.

[92] Joran van Apeldoorn and András Gilyén. Improvements in Quantum SDP-Solving with Applications. *arXiv preprint arXiv:1804.05058*, 2018.

[93] Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. Quantum SDP-Solvers: Better Upper and Lower Bounds. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 403–414. IEEE, 2017.

[94] Guoming Wang. Efficient quantum algorithms for analyzing large sparse electrical networks. *Quantum Information & Computation*, 17(11-12):987–1026, 2017.

[95] Jonathan Welch, Daniel Greenbaum, Sarah Mostame, and Alan Aspuru-Guzik. Efficient quantum circuits for diagonal unitaries without ancillas. *New Journal of Physics*, 16(3):033040, 2014.

[96] Leonard Wossnig, Zhikuan Zhao, and Anupam Prakash. Quantum Linear System Algorithm for Dense Matrices. *Physical Review Letters*, 120(5):050502, 2018.

[97] Guanglei Xu, Andrew J Daley, Peyman Givi, and Rolando D Somma. Turbulent Mixing Simulation via a Quantum Algorithm. *AIAA Journal*, pages 687–699, 2017.

[98] Christof Zalka. Efficient Simulation of Quantum Systems by Quantum Computers. *Fortschritte der Physik: Progress of Physics*, 46(6-8):877–879, 1998.

[99] Chi Zhang. Randomized Algorithms for Hamiltonian Simulation. In *Monte Carlo and Quasi-Monte Carlo Methods 2010*, pages 709–719. Springer, 2012.