Quantum Algorithms for Scientific Computing and Approximate Optimization

Stuart Andrew Hadfield

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2018

ABSTRACT

Quantum Algorithms for Scientific Computing and Approximate Optimization

Stuart Andrew Hadfield

Quantum computation appears to offer significant advantages over classical computation and this has generated a tremendous interest in the field. In this thesis we study the application of quantum computers to computational problems in science and engineering, and to combinatorial optimization problems. We outline the results below.

Algorithms for scientific computing require modules, i.e., building blocks, implementing elementary numerical functions that have well-controlled numerical error, are uniformly scalable and reversible, and that can be implemented efficiently. We derive quantum algorithms and circuits for computing square roots, logarithms, and arbitrary fractional powers, and derive worst-case error and cost bounds. We describe a modular approach to quantum algorithm design as a first step towards numerical standards and mathematical libraries for quantum scientific computing.

A fundamental but computationally hard problem in physics is to solve the time-independent Schrödinger equation. This is accomplished by computing the eigenvalues of the corresponding Hamiltonian operator. The eigenvalues describe the different energy levels of a system. The cost of classical deterministic algorithms computing these eigenvalues grows exponentially with the number of system degrees of freedom. The number of degrees of freedom is typically proportional to the number of particles in a physical system. We show an efficient quantum algorithm for approximating a constant number of low-order eigenvalues of a Hamiltonian using a perturbation approach. We apply this algorithm to a special case of the Schrödinger equation and show that our algorithm succeeds with high probability, and has cost that scales polynomially with the number of degrees of freedom and the reciprocal of the desired accuracy. This improves and extends earlier results on quantum algorithms for estimating the ground state energy.

We consider the simulation of quantum mechanical systems on a quantum computer. We show a novel divide and conquer approach for Hamiltonian simulation. Using the Hamiltonian structure, we can obtain faster simulation algorithms. Considering a sum of Hamiltonians we split them into groups, simulate each group separately, and combine the partial results. Simulation is customized to take advantage of the properties of each group, and hence yield refined bounds to the overall simulation cost. We illustrate our results using the electronic structure problem of quantum chemistry, where we obtain significantly improved cost estimates under mild assumptions.

We turn to combinatorial optimization problems. An important open question is whether quantum computers provide advantages for the approximation of classically hard combinatorial problems. A promising recently proposed approach of Farhi et al. is the Quantum Approximate Optimization Algorithm (QAOA). We study the application of QAOA to the Maximum Cut problem, and derive analytic performance bounds for the lowest circuit-depth realization, for both general and special classes of graphs. Along the way, we develop a general procedure for analyzing the performance of QAOA for other problems, and show an example demonstrating the difficulty of obtaining similar results for greater depth.

We show a generalization of QAOA and its application to wider classes of combinatorial optimization problems, in particular, problems with feasibility constraints. We introduce the Quantum Alternating Operator Ansatz, which utilizes more general unitary operators than the original QAOA proposal. Our framework facilitates low-resource implementations for many applications which may be particularly suitable for early quantum computers. We specify design criteria, and develop a set of results and tools for mapping diverse problems to explicit quantum circuits. We derive constructions for several important prototypical problems including Maximum Independent Set, Graph Coloring, and the Traveling Salesman problem, and show appealing resource cost estimates for their implementations.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I am immensely thankful to my wonderful academic advisors Professor Alfred V. Aho, Professor Joseph F. Traub, and Dr. Anargyros Papageorgiou for their support, mentorship, and patience. Together, they taught me the importance of seeking impactful research problems, and perhaps more importantly, perseverance. It has been a privilege and a pleasure to work with each of them.

I am very grateful to Dr. Eleanor G. Rieffel and the other members of the NASA Quantum Artificial Intelligence Laboratory, where I spent a stimulating summer in 2016, for countless insightful interactions. Two chapters of this thesis resulted from research collaborations which began there.

I would like to especially thank my dissertation committee members Prof. Aho, Dr. Papageorgiou, Dr. Rieffel, Professor Mihalis Yannakakis, and Professor Rocco Servedio for their service and valuable comments on this thesis.

I am thankful to my many colleagues from the computer science department, and from the wider Columbia community, for countless stimulating discussions and interactions, both professionally and socially. Additionally, I wish to generally thank my collaborators and the many members of the broader quantum computing community who have positively affected my growth as a researcher along the way.

Chapter 2 of this thesis is a joint work with with Mihir Bhaskar, Anargyros Papageorgiou, and Iasonas Petras [35]. The results of Chapters 3 and 4 are collaborations with Anargyros Papageorgiou [112, 113]. Chapters 5 and 6 are based on joint works with Eleanor Rieffel, Zhihui Wang, Zhang Jiang, Bryan O'Gorman, Davide Venturelli, and Rupak Biswas [114, 115, 238].

Finally, most importantly, I thank my family and friends, especially Bahareh, for their unending patience and understanding during the often arduous PhD process. Without your constant love, support, and encouragement, I would not be where I am today.

This thesis is dedicated to my family, I couldn’t have done it without you.

In memory of J. F. Traub.

Epigraph

The importance of accelerating approximating and computing mathematics by factors like 10,000 or more, lies not only in that one might thereby do in 10,000 times less time problems which one is now doing...but rather in that one will be able to handle problems which are considered completely unassailable at present.

The projected device, or rather the species of which it is to be the first representative, is so radically new that many of its uses will become clear only after it has been put into operation, and after we have adjusted our mathematical habits and ways of thinking to its existence and possibilities. Furthermore, these uses which are not, or not easily, predictable now, are likely to be the most important ones...because they are farthest removed from what is now feasible.

John von Neumann to Lewis L. Strauss, 1945 [235]

The 'paradox' is only a conflict between reality and your feeling of what reality 'ought to be.'

Richard Feynman, *The Feynman Lectures on Physics*, 1965 [96]

Chapter 1

Introduction

The potential advantages of quantum over classical computers for solving hard problems has generated tremendous interest in quantum computation. Efficient quantum algorithms have been derived not only for discrete problems [172], such as, famously, integer factorization [209], but also for important computational problems in science and engineering, such as quantum simulation, eigenvalue estimation, integration, partial differential equations, and numerical linear algebra problems [102, 173, 185]. Thus quantum computers have immense potential impact on fields ranging from quantum chemistry to computer security to machine learning [36, 52, 154].

Indeed, as prototype quantum computers begin to emerge over the next few years [43, 135, 169, 207], quantum computing is finally posed to transition from a theoretical model to impactful practical computing devices. It is important to both characterize the power of such devices, and to find new and improved algorithms particularly applicable to both small- and medium-scale quantum hardware to take advantage of the emerging technologies. In this thesis, we take a number of steps toward these goals. We study five problems. The first three deal with quantum algorithms for computational problems in science and engineering. The remaining two deal with quantum algorithms for approximate optimization. In particular, we study quantum algorithms and circuits for scientific computing, the approximation of ground and excited state energies of the Schödinger equation, algorithms for Hamiltonian simulation, performance analysis of the quantum approximate optimization algorithm (QAOA), and a generalization of QAOA particularly suitable for constrained optimization problems and low-resource implementations. We briefly describe each of these problems below, providing motivation and summarizing the respective results. For the interest of the reader, a brief overview of quantum computation is included as Appendix A.

1.1 Quantum Algorithms and Circuits for Scientific Computing

The need for quantum algorithms for scientific computing, to be used as modules in other quantum algorithms, is apparent. For example, a recent paper deals with the solution of linear systems on a quantum computer [121]. The authors present an algorithm that requires the (approximate) calculation of the reciprocal of a number followed by the calculation of trigonometric functions needed in a controlled rotation on the way to the final result. However, the paper does not give any details about how these operations are to be implemented. From a complexity theory point of view this may not be a complication, but certainly there is a lot of work that is left to be done before one is able to implement the linear systems algorithm and eventually use it on a quantum computer.

In solving scientific and engineering problems, classical algorithms typically use floating point arithmetic and numerical libraries of special functions. The IEEE Standard for Floating Point Arithmetic (IEEE 754-2008) [136] ensures that such calculations are performed with well-defined precision. A similar standard is needed for quantum computing. Many quantum algorithms use the quantum circuit model of computation, typically employing a fixed-precision representation of numbers. Yet there is no standard specifying how arithmetic operations between numbers (of possibly disproportionate magnitudes) held in registers of finite length are to be performed, and how to deal with error. In designing algorithms for scientific computing the most challenging task is to control the error propagation. Since registers have finite length it is not reasonable to expect to calculate exactly and propagate all the results of intermediate calculations throughout all the stages of an algorithm. Intermediate approximations have to be made. Most importantly, there are no existing libraries of quantum circuits with performance guarantees, implementing functions such as the square root of a number, the logarithm, or other similar elementary functions. Such quantum circuits should be uniformly scalable, and at the same time make efficient use of quantum resources to meet physical constraints of potential quantum computing devices of the foreseeable future.

It is worthwhile remarking on the direct applicability of classical algorithms to quantum computation. It is known that classical computation is subsumed by quantum computation, i.e., that for any classical algorithm, there exists a quantum algorithm which performs the same computation [173].

This follows from the fact that any classical algorithm (or circuit) can be implemented reversibly, in principle, but with the additional overhead of a possibly large number of *ancilla* qubits that must be carried forward throughout the computation. For simple circuits consisting of the composition of basic logical operations, this overhead grows with the number of gates. On the other hand, scientific computing algorithms are quite different. They typically involve a large number of floating point arithmetic operations computed approximately according to rules that take into account the relative magnitudes of the operands requiring mantissa shifting, normalization and rounding. Thus they are quite expensive to implement reversibly because this would require many registers of large size to store all the intermediate results. Moreover, a mechanism is necessary for dealing with roundoff error and overflow which are not reversible operations. Hence, the direct simulation on a quantum computer of classical algorithms for scientific computing that have been implemented in floating point arithmetic quickly becomes quite complicated and prohibitively expensive.

With this motivation, in Chapter 2 we show a modular approach forming a basis towards a standard for quantum scientific computing. We give efficient quantum algorithms and circuits for computing square roots, logarithms, and arbitrary fractional powers. We derive worst-case error and cost bounds, providing performance guarantees with respect to the desired accuracy. We further illustrate the performance of our algorithms with tests comparing them to the respective floating point implementations found in widely used numerical software. Our results are important first steps towards mathematical libraries and numerical standards for quantum computing.

1.2 Approximating Ground and Excited State Energies on a Quantum Computer

Quantum mechanical systems are governed by the Schrödinger equation, where the evolution in time of a quantum system is determined by the *Hamiltonian* operator. Hamiltonian eigenvalues describe the system energy levels. For example, computing the energy levels is one of the most important tasks in chemistry because they are required for predicting reaction rates and electronic structure; both of which, in particular, depend principally on the low-order energy levels. Computing the energy levels is a very hard problem in general. The best classical algorithms known have costs that grow exponentially in the number of system degrees of freedom [154]. Therefore, efficient quantum algorithms would be an extremely powerful tool for new science and technology, having tremendous potential impact on the design of new medicines and advanced materials, and improving the efficiency of important chemical processes such as nitrogen fixation [198].

On the other hand, there are a number of recent results in discrete complexity theory suggesting that many eigenvalue problems are very hard even for quantum computers because they are QMA-complete [61, 148, 205, 243]. (Roughly speaking, QMA is the quantum analog of NP, i.e., the class of decision problems that can be efficiently *verified* on a quantum computer.) However, discrete complexity theory deals with the worst case over large classes of Hamiltonians. It does not provide methods or necessary conditions determining when an eigenvalue problem is hard. In fact, there is a dichotomy between theory and practice. As stated in [164], "complexity theoretic proofs of the advantage of many widely used classical algorithms are few and far between." Therefore, it is important to develop new quantum algorithms and to use them for solving eigenvalue problems for which quantum computers can be shown to have a significant advantage over classical computers.

In [183] the authors developed an algorithm and proved a strong exponential quantum speedup for approximating the ground state energy (i.e., the smallest eigenvaue) of the time-independent Schrödinger equation under certain assumptions. In [184] it is explained why this problem is different from the QMA-complete problems of discrete complexity theory. In [182] an important assumption of [183] was relaxed and the results extended to the ground state energy approximation for the time-independent Schrödinger equation with a convex potential.

An important advance would be to obtain analogous results for approximating excited state energies under weakened assumptions. The techniques used previously for the ground state energy do not extend to excited state energies. Similarly, in computational chemistry, for instance, Hohenberg-Kohn density functional theory (DFT) is strictly limited to ground states [100, 132]. There are other flavors of DFT that may provide approximations of excited state energies. However, in general, approximate methods in computational chemistry often succeed in predicting chemical properties yet their level of accuracy varies with the nature of the species and may fail in important instances; see [15, 154] and the references therein. Obtaining conditions allowing one to approximate excited state energies with a guaranteed accuracy and a reasonable cost would provide a valuable insight into the complexity of these problems.

To this end, in Chapter 3, under general conditions, and using a perturbation approach, we provide a quantum algorithm that produces estimates of a constant number of different low-order eigenvalues. The algorithm relies on a set of trial eigenvectors, whose construction depends on the particular Hamiltonian properties. We illustrate our results by considering a special case of the time-independent Schrödinger equation with d degrees of freedom. Our algorithm computes estimates of a constant number of different low-order energy levels with error ε and success probability at least $3/4$, with cost polynomial[1] in ε^{-1} and d. This extends earlier results on algorithms for estimating the ground state energy. The technique we present is sufficiently general to apply to problems beyond the application studied in Chapter 3.

1.3 Divide and Conquer Hamiltonian Simulation

Simulating quantum mechanical systems using classical computers appears to be a very hard problem. The description of general quantum states grows exponentially with the system size and so does the computational cost of the best classical algorithms known for simulation. This difficulty led Feynman to propose simulation on a quantum computer, i.e., using one quantum system to simulate another. He conjectured that quantum computers might be able to carry out the simulation more efficiently than classical algorithms. Subsequently, a long line of research has shown efficient quantum algorithms for simulation for a variety of important applications. For an overview of quantum simulation see, e.g., [54,95]. Early simulation results can be found in [5,162,257,258]. More recent developments can be found in [29–31,33,186]. Related applications to physics and chemistry can be found in [2,55,140,141,146,147,177,233,240,242,246].

In the Hamiltonian simulation problem one is given a Hamiltonian H acting on q qubits, a time $t \in \mathbb{R}$, and an accuracy demand ε, and the goal is to derive an algorithm that constructs an operator $\widetilde{U}$ that approximates the unitary operator e^{-iHt} with error $\|\widetilde{U} - e^{-iHt}\| \leq \varepsilon$ measured in the spectral norm. The operator e^{-iHt} corresponds to quantum evolution under the Hamiltonian H for time t. When the Hamiltonian is given explicitly, the size of the quantum circuit realizing the algorithm is its cost. In particular, the cost depends on the complexity parameters q, t and ε^{-1}. On the other hand, when the Hamiltonian is given by an oracle, the number of queries (oracle calls)

[1]We say a quantity $c(n)$ is *polynomial* in n if as n becomes large it grows at most as $c(n) = O(n^k)$ for some $k \in \mathbb{N}$. We will often write $c = \mathrm{poly}(n)$ to denote such polynomial scaling.

used by the algorithm plays a major role in its cost, in addition to the number of qubits and the other necessary quantum operations. Different types of queries have been considered in the literature.

Of particular interest are Hamiltonians H that can be expressed as a sum

$$H = \sum_{j=1}^{m} H_j,$$

where the H_j are Hamiltonians which each can be simulated efficiently, i.e., we have an explicit quantum circuit for implementing each exponential (query) e^{-iH_jt}, $j = 1, \ldots, m$. The Born-Oppenheimer electronic Hamiltonian in the second-quantized form, which describes molecular systems, enjoys this property, see, e.g., [246]. Suzuki-Trotter formulas [214, 215] are typically used in the simulation of these Hamiltonians. Using them Berry et al. [29] showed a quantum algorithm for Hamiltonian simulation with cost (number of exponentials e^{-iH_jt}) polynomial in m, t, $\|H\|$ and in ε^{-1}. These results were subsequently improved in [186] where the authors observed that if we order the Hamiltonians so that $\|H_1\| \geq \|H_2\| \geq \cdots \|H_m\|$ and if $\|H_2\| \to 0$ then a single exponential would suffice for the simulation of H, and hence showed improved cost bounds polynomial in m, t, $\|H_1\|$, $\|H_2\|$ and in ε^{-1}. The dependence of the cost on the norms of the individual Hamiltonians composing H is very important since it can reduce the simulation cost significantly.

For applications such as simulating the second-quantized electronic Hamiltonian where m is typically very large, the simulation cost can be prohibitive [240]. Moreover, if many Hamiltonians have small, or even negligible, norms compared to the largest, it may be possible to take advantage of this disparity to derive faster algorithms. Indeed, this situation is common in chemistry applications involving the second-quantized electronic Hamiltonian, and heuristics have been proposed to take advantage of this and reduce the simulation cost; see, e.g., [18, 193, 240].

The goal, then, is to construct algorithms that improve known simulation cost estimates, particularly when m is huge while relatively few Hamiltonians have large norms and many have small norms. In Chapter 5, we accomplish this goal applying a divide and conquer approach. We partition the Hamiltonians into groups, simulate the sum of the Hamiltonians in each group separately, and then combine the partial results. Simulation is customized to take advantage of the properties of each group, and hence yields refined bounds to the overall simulation cost that reflect these properties.

We illustrate our results using the electronic structure problem of quantum chemistry. For the second-quantized electronic Hamiltonian describing a molecular system, the number of Hamiltonians in the sum is $m = \Theta(\mathcal{N}^4)$, where $\mathcal{N}$ is a parameter proportional to the number of particles. For many important problems in chemistry, simulating this Hamiltonian is well beyond the reach of the best classical algorithms. Standard quantum algorithms for simulation have polynomial cost that, roughly speaking, grows as $\mathcal{N}^8$ or $\mathcal{N}^9$. For moderately sized problems of interest where, say, $\mathcal{N} = 100$, this cost dependence is already prohibitive and hence simulation is considered to be a cost bottleneck [240]. Using our divide and conquer approach, we show under mild assumptions that the cost estimates for our algorithms scale with $\mathcal{N}$ in the range $\mathcal{N}^5$ to $\mathcal{N}^7$. Hence, our approach may reduce the simulation cost by several orders of magnitude, allowing the simulation of much larger chemical systems, especially on early quantum computers.

1.4 Quantum Approximate Optimization

Combinatorial optimization problems are ubiquitous in science, engineering and operations research. Many important problems are not only NP-hard to solve, but are NP-hard even to approximate better than some factor. Two well-known examples are Maximum Satisfiability, where given a Boolean formula in conjunctive normal form we seek an assignment of the variables satisfying as many clauses as possible, and the Traveling Salesman problem, where given a list of cities and the distances between them we seek a tour visiting all cities with total length as small as possible. Indeed, for many important applications we must settle for algorithms or heuristics producing approximate solutions, and this has led to rich theories of approximation algorithms, approximation complexity, and the hardness of approximation [14, 16, 226, 229, 251].

The difficulty in solving optimization problems has generated much excitement about the possibility of using quantum devices to approximately solve them. A particularly prominent metaheuristic is quantum annealing (or, more specifically, adiabatic quantum optimization) [90, 142], which can be implemented on quantum devices that are much simpler than universal quantum computers and hence easier in principle to design and engineer. Currently, the commercially available family of D-WAVE machines provide quantum annealing with up to 2000 qubits, but with restricted connectivity and other limitations [42]. Google and the IARPA Quantum Enhanced Optimization program, for example, are working towards building the next generation of quantum annealing devices. However, these quantum annealers are heuristic solvers, since generally we do not have performance guarantees, and these machines must be characterized empirically; see, e.g., [168]. These devices remain at an early stage, with hardware constraints severely limiting the classes of and sizes of problem to which they are applicable. It remains an important open problem whether or not such devices, or even future improved versions, can provide advantages for real-world optimization problems [42, 78]. Since the D-WAVE and other proposed quantum annealers are not believed to be universal for quantum computation,[2] the pertinent question is: do quantum computers based on the quantum circuit model of computation, which are universal and potentially more powerful than existing quantum annealers, offer significant advantages for approximate optimization?

More concretely, we pose two general motivating questions.

- Can quantum computers find good approximate solutions *faster* than classical computers?
- Do there exist important problems that can be approximated *better* on a quantum computer than by any classical algorithm?

For some problems, we have tight classical results where the best algorithm known achieves the best possible approximation ratio[3] (under a standard assumption from computational complexity theory such as P$\neq$NP). For such cases, we do not believe that quantum computers could achieve a much better approximation ratio, since this would imply that quantum computers could efficiently solve NP-hard problems; see e.g. [16, 226]. On the other hand, for some problems there exist substantial gaps between the ratio achieved by the best classical algorithms known and the sharpest hardness of approximation result; problems in this category are a promising class where quantum computers may offer an advantage. Indeed, an illuminating example is the problem Max-E3Lin2 where we seek to maximally satisfy a set of three-variable linear equations over $\mathbb{Z}_2$. Remarkably, an efficient quantum algorithm producing a better approximation ratio than the best classical algorithm known was found [87], only to subsequently inspire an even better (by a logarithmic factor) classical algorithm [22]. An important research direction, generally, is to find hard problems where quantum algorithms may provide advantages for approximation.

[2]A closely related computational model, adiabatic quantum computation [6], is universal.

[3]An algorithm achieves an R-approximation if it always produces a solution within a multiplicative factor of R or better of the optimal solution, in time polynomial in the problem size; see e.g. [16].

Recently, Farhi et al. [86] proposed a new class of quantum algorithms, the Quantum Approximate Optimization Algorithm (QAOA), to tackle challenging approximate optimization problems on gate model quantum computers. A handful of recent papers suggest QAOA circuits are powerful for computation [87, 91, 138, 254]. In QAOA, the algorithms require the determination of certain parameters, and their success relies on one being able to find a good set of such parameters. QAOA algorithms are characterized by their depth p, which we write QAOA$_p$. In particular, the algorithm consists of applications of a phase operator and a mixing operator, applied in alternation p times each. Generally, the performance of the algorithm improves with higher p. The lowest depth version QAOA$_1$ has provable performance guarantees for certain problems [86,87,238]. Characterizing the performance of QAOA$_p$ circuits for $p > 1$, which is the regime where we expect to see the most advantage for applications, remains the most important open problem towards understanding whether these circuits can outperform classical algorithms.

In Chapter 5, we study the application of QAOA to the Maximum Cut problem, the original application considered in [86]. We derive novel analytic results towards characterizing the algorithm's performance and finding the optimal choice of parameters, for the $p = 1$ case. In particular, we show bounds for the expected approximation ratio on both general and restricted classes of graphs. Our results extend earlier numerical results for special cases [86]. We apply our technique to a particular case of Maximum Cut for $p = 2$, obtaining a complicated expression which demonstrates the difficulty of obtaining similar results for $p > 1$. Along the way, we provide a procedure which can be used to derive similar results for other problems of interest. Our results significantly expand the known performance bounds for QAOA, and are important steps towards developing techniques to characterize its power in more general applications.

In Chapter 6, we show a generalization of QAOA, the *Quantum Alternating Operator Ansatz*, enabling the exploration of QAOA approaches to a much broader selection of problems. In particular, we consider *constrained* optimization problems, where we are additionally given feasibility constraints and we seek the best solution within the feasible subset. An example application to such a problem was considered in [86, Sec. VII], but without giving the details. We carefully specify design criteria and requirements for general problems. Specifically, we allow for much more general mixing operators than those considered in [86]. Importantly, we show constructions that restrict the evolution of the QAOA algorithm to the subspace of states corresponding to feasible solutions, which avoids altogether the difficulty and cost of dealing with infeasible states directly. (Without this property, many algorithm measurement outcomes could yield invalid solutions, which would have to be carefully accounted for in analyzing the success probability of the algorithm, or dealt with by some other means, often with considerable additional cost.) Our constructions facilitate low-resource implementations, which is particularly advantageous for early quantum computers. Moreover, we provide a toolkit of results for mapping Boolean and real functions to Hamiltonians, which may be useful for other quantum algorithms for approximation such as quantum annealing. We then realize our approach by deriving explicit constructions for a sequence of important prototypical problems, including Maximum Independent Set, several optimization problems related to Graph Coloring, and the Traveling Salesman problem. In each case we show that the numbers of qubits and basic quantum gates required are relatively low, e.g., scaling linearly or quadratically with the problem parameters, and hence these resource counts enable us to identify problems especially suitable for implementation on near-term quantum computing hardware.

Chapter 2

Quantum Algorithms and Circuits for Scientific Computing

2.1 Introduction

Recent results [56,121,218] suggest that quantum computers may have a substantial advantage over classical computers for solving numerical linear algebra problems and, more generally, problems of computational science and engineering [185]. Scientific computing applications require the evaluation of elementary functions like those found in mathematics libraries of programming languages, where the calculations are performed using floating point arithmetic. Although in principle quantum computers can always directly simulate any classical algorithm, generally there is no guarantee that such simulations remain practical. Hence, it is important to develop efficient quantum algorithms and circuits implementing such functions, towards the goal of establishing a standard for numerical computation on quantum computers.

In designing algorithms for scientific computing the most challenging task is to control the error propagation and to do so efficiently. In this sense, it is important to derive reusable modules with well-controlled error bounds. In this chapter we continue this line of work of [56] by deriving quantum circuits which, given a number w (represented as a fixed-precision binary number), compute the functions $w^{1/2^i}$ for $i = 1, \dots, k$, $\ln(w)$ (and thereby the logarithm in different bases), and w^f with $f \in [0, 1)$. Our design is modular, combining a number of elementary quantum circuits to implement the functions, and for each circuit we provide cost and worst-case error estimates. We also illustrate the accuracy of our algorithms through examples comparing their error with that of widely used numerical software such as Matlab. In summary, our tests show that using a moderate amount of resources, our algorithms compute the values of the functions matching the corresponding values obtained using scientific computing software (using floating point arithmetic) with 12 to 16 decimal digits of accuracy. Our circuits complement those given in [56] for computing the reciprocal and basic trigonometric functions, and together are important first steps towards establishing libraries of quantum circuits for mathematical functions.

We consider the quantum circuit model of computation where arithmetic operations are performed with fixed precision. We use a small number of elementary modules, or building blocks, to implement fundamental numerical functions. Within each module the calculations are performed exactly. The results are logically truncated by selecting a desirable number of significant bits which are passed on as inputs to the next stage, which is also implemented using an elementary module. We repeat this procedure until we obtain the final result. This way, it suffices to implement quantum mechanically a relatively small number of elementary modules, which can be done once, and then to combine them as necessary to obtain the quantum circuits implementing the different functions. The elementary modules carry out certain basic tasks such as shifting the bits of a number held in a quantum register, or counting bits, or computing expressions involving addition and/or multiplication of the inputs. The benefit of using only addition and multiplication is that in fixed-precision arithmetic the format of the input specifies exactly the format of the output, i.e., the location of the decimal point in the result. There exist numerous quantum circuits in the literature for addition and multiplication; see e.g. [13, 25, 68, 73, 83, 84, 199, 219, 221].

There are three advantages to this approach. The first is that one can derive error estimates by treating the elementary modules as black boxes and considering only the truncation error in the output of each. The second is that it is easy to obtain total resource estimates by adding the resources used by the individual modules. The third advantage is that the modular design allows one to modify or improve the implementation of the individual elementary modules in a transparent way. Such an approach was used in [56] that deals with a quantum algorithm and circuit design for solving the Poisson equation. The results of this chapter can also be found in [35].

2.2 Algorithms

We derive quantum algorithms and circuits computing approximately $w^{1/2^i}$, $i = 1, \ldots, k$, $\ln(w)$ and w^f, $f \in [0, 1)$, for a given input w. We provide pseudocode[1] and show how the algorithms are obtained by combining elementary quantum circuit modules. We provide error and cost estimates.

The input of the algorithms is a fixed-precision binary number. It is held in an n qubit quantum register whose state is denoted $|w\rangle$ as shown in Figure 2.1. The m left most qubits are used to represent the integer part of the number and the remaining $n-m$ qubits represent its fractional part.

$$|w\rangle = \underbrace{\left|w^{(m-1)}\right\rangle \otimes \left|w^{(m-2)}\right\rangle \otimes \cdots \otimes \left|w^{(0)}\right\rangle}_{\text{integer part}} \otimes \underbrace{\left|w^{(-1)}\right\rangle \otimes \cdots \otimes \left|w^{(m-n)}\right\rangle}_{\text{fractional part}},$$

Fig. 2.1: An n-qubit fixed-precision representation of a number $w \geq 0$ on a quantum register.

Thus $|w\rangle = \left|w^{(m-1)}w^{(m-2)} \cdots w^{(0)}w^{(-1)} \cdots w^{(m-n)}\right\rangle$, where $w^{(j)} \in \{0, 1\}$, $j = m-n, m-n+1, \ldots, 0, \ldots, m-1$ and $w = \sum_{j=m-n}^{m-1} w^{(j)} 2^j$. Since fewer than n bits may suffice for the representation of the input, a number of leftmost qubits in the register may be set to $|0\rangle$. In general, a leading qubit may hold the sign of w, but since for the functions in this chapter w is a nonnegative number we have omitted the sign qubit for simplicity.

Our algorithms use elementary modules that perform certain basic calculations. Some are used to shift the contents of registers, others are used as counters determining the position of the most significant bit of a number. An important elementary module computes expressions of the form $xy + z$ exactly in fixed-precision arithmetic.

Following our convention concerning the fixed precision representation of numbers as we introduced it in Fig. 2.1, let x, y, and z be represented using n_1-bits, of which m_1 bits are used to represent the integer part. (It is not necessary to use the same number of bits to represent all three numbers and this might be useful in cases where we know that their magnitudes are significantly different). The expression $xy+z$ can be computed exactly as long as we allocate $2n_1+1$ bits to hold the result. In this case, the rightmost $2(n_1 - m_1)$ bits hold the fractional part of the result. Such compu-

[1]We present our algorithms using standard high-level mathematical and programming language expressions, traditionally known as *Pidgin ALGOL* [7].

tations can be implemented reversibly. There are numerous quantum circuit designs in the literature implementing addition and multiplication [13, 25, 73, 83, 84, 149, 192, 199, 202, 219–221, 228, 230]. Therefore, we can use them to design a quantum circuit implementing $xy + z$. In fact, we can design a quantum circuit template for implementing such expressions and use it to derive the actual quantum circuit for any n_1, m_1 and values of the x, y, z represented with fixed precision. We use Figure 2.2 below to generically represent such a quantum circuit.

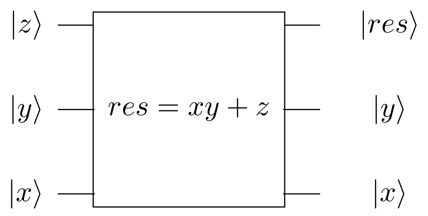


Fig. 2.2: Elementary module using fixed-precision arithmetic to implement exactly $res \leftarrow xy + z$ for x, y, and z. Note that register sizes, ancilla registers, and their values are not indicated.

Note that Fig. 2.2 is an abstraction of an elementary module computing $res \leftarrow xy + z$. It is not meant to reveal or imply any of the implementation decisions including ancilla registers, saved values, and other details used for addition and multiplication. Any desired number b of significant digits after the decimal point in the result $|res\rangle$ can be selected and passed on to the next stage of the computation. This corresponds to a truncation of the result to the desired accuracy.

We emphasize that although there are numerous ways of implementing basic arithmetic operations, there are trade-offs for the resulting quantum circuits in terms of the types of quantum gates used, the number of ancilla qubits required, and the resulting circuit size and depth. For example, Table 1 in [221] summarizes some of the trade-offs for quantum circuits for addition. There we see that the circuit from [83] does not require Toffoli gates or ancilla qubits, but has size $O(n^2)$, while the circuit in [73] uses one ancilla qubit, $O(n)$ Toffoli gates, and has size $O(n)$. It is reasonable to believe that low-level implementation details will have to be decided taking into account the target architecture because the unit cost of the different resources is unlikely to be equal. Our algorithms, in addition to the applications of the elementary module of Fig. 2.2, use a constant number of extra arithmetic operations. Thus the cost of the algorithms can be expressed in a succinct and fairly accurate way by counting the number of arithmetic operations required, which is the approach taken in classical algorithms for scientific computing. The number of arithmetic operations leads to precise resource estimates once particular choices for the quantum circuits implementing these operations have been made. Hence, for the remainder of this chapter we will not be concerned with low-level implementation details or resource optimizations.

We remark that particularly for the algorithms in this chapter the integer parts of the inputs and outputs of the instances of the quantum circuit of Fig. 2.2 can be represented exactly using an equal number of qubits to that used for $|w\rangle$, i.e., m qubits; we deal with the fractional parts separately. In [56], such elementary modules were cascaded to derive the quantum algorithm, INV, computing the reciprocal function. The algorithms of this chapter compute approximations of functions which depend, to a certain extent, on INV, so we review its details in the next subsection.

Without loss of generality and for brevity we only deal with the case $w > 1$ in the functions computing the roots and the logarithm. Indeed, if $0 < w < 1$ one can suitably shift it to the left ν times to become $2^\nu w > 1$. After obtaining the roots or the logarithm of the shifted number $2^\nu w$ the final result for w can be obtained in a straightforward way, either by shifting it to the right in the case of the roots, or by subtracting $\nu \ln(2)$ in the case of the logarithm.

In the following subsections we discuss each of our algorithms providing its details along with pseudocode and quantum circuits. We give theorems establishing the performance of each algorithm in terms of their accuracy in relation to the number of required qubits. To avoid distracting technical details, the proofs are deferred to Appendix C. Our results are summarized in Table 2.1.

2.2.1 Reciprocal

An algorithm computing the reciprocal of a number is shown in [56]. The algorithm is based on Newton iteration. Below we provide a slight modification of that algorithm.

Recall that w is represented with n bits of which the first m correspond to its integer part. Algorithm 0 INV below approximates the reciprocal of a number $w \geq 1$, applying Newton iteration to the function $f(x) = \frac{1}{w} - x$. This yields a sequence of numbers x_i, $i = 1, 2, \dots, s$, according to the iteration

$$x_i = g_1(x_{i-1}) := -w\hat{x}_{i-1}^2 + 2\hat{x}_{i-1}. \tag{2.1}$$

Observe that the expression above can be computed using two applications of a quantum circuit of the type shown in Fig. 2.2.

Function	Requirements	Algorithm and Parameters	Idea	Error
$1/w$	$w \geq 1$	$\mathrm{INV}(w, n, m, b)$ $b \geq m$ $s = \lceil \log_2 b \rceil$	1. Newton iteration. Calculate $x_i = -w\hat{x}_{i-1}^2 + 2\hat{x}_{i-1}, i = 1, \ldots, s$ 2. Return $\hat{x}_s$	$\leq \left(\frac{1}{2}\right)^b (2 + \log_2 b)$
$\sqrt{w}$	$w \geq 1$	$\mathrm{SQRT}(w, n, m, b)$ $b \geq \max\{2m, 4\}$ $s = \lceil \log_2 b \rceil$	1. Call INV(w, n, m, b) 2. Newton iteration. Calculate $y_j = \frac{1}{2}(3\hat{y}_{j-1} - \hat{x}_s\hat{y}_{j-1}^3), j = 1, \ldots, s$ 3. Return $\hat{y}_s$	$\leq \left(\frac{3}{4}\right)^{b-2m} (2 + b + \log_2 b)$
$w^{1/2^i}$ $i = 1, \ldots, k$	$w \geq 1$	Powerof2Roots(w, k, n, m, b) $b \geq \max\{2m, 4\}$ $s = \lceil \log_2 b \rceil$ for each call of SQRT()	1. $z_1 =$ SQRT(w, n, m, b) 2. Call SQRT() repeatedly, i.e., $z_i =$ SQRT$(z_{i-1}, m + b, m, b), i = 1, \ldots, k$ 3. Return $\{z_i\}$	$\leq \left(\frac{3}{4}\right)^{b-2m} 2(2 + b + \log_2 b)$
$\ln w$	$w \geq 1$	$\mathrm{LN}(w, n, m, \ell)$ $b = \max\{5\ell, 25\}$ $\ell \geq \lceil \log_2 8n \rceil$ $r \approx \ln 2$, with b bits accuracy $p = \lceil \log_2 w \rceil$	1. $w_p = w \cdot 2^{1-p}$ 2. Call PowerOf2Roots$(w_p, \ell, n, 1, b)$ and let $\hat{t}_p$ be the $\frac{1}{2^\ell}$th root of w_p 3. Approx. $\ln \hat{t}_p$ with $\hat{y}_p$, the first two terms of its power series expansion. Return $z_p = 2^\ell \hat{y}_p + (p-1)r$	$\leq \left(\frac{3}{4}\right)^{5\ell/2} \left(m + \frac{32}{9} + 2\left(\frac{32}{9} + \frac{n}{\ln 2}\right)^3\right)$
w^f	$w \geq 1$ $f \in [0, 1]$ f is n_f bits	FractionalPower(w, f, n, m, n_f, ℓ) $b = \max\{n, n_f, \lceil 5(\ell, 2m, \ln n_f) \rceil\}$ $\ell \in \mathbb{N}$ determines the error	1. For $i = 1, \ldots, n_f$ calculate $\hat{w}_i =$ PowerOf2Roots(w, n_f, n, m, b) 2. Return $\Pi_{i \in \mathcal{P}} \hat{w}_i$ where $\mathcal{P} = \{1 \leq i \leq n_f : f_i = 1\}$	$\leq \left(\frac{1}{2}\right)^{\ell-1}$
w^f	$0 \leq w < 1$ $f \in [0, 1]$ f is n_f bits	FractionalPower2(w, f, n, m, n_f, ℓ) $b = \max\{n, n_f, \lceil 2\ell + 6m + 2 \ln n_f \rceil, 40\}$ $\ell \in \mathbb{N}$ determines the error	1. Compute $w' \geq 1$ by left shifting w 2. Call FractionalPower(w', f, n, m, n_f, ℓ) 3. *Undo* the initial shift of w using right shifts, FractionalPower, and INV, and return	$\leq \left(\frac{1}{2}\right)^{\ell-3}$

Table 2.1: Summary of algorithms. All parameters are polynomial in n and b and so is the cost of all algorithms.

Algorithm 0 INV(w, n, m, b)

```
Require: $w \geq 1$, held in an $n$ qubit register with $m$ qubits for its integer part.
Require: $b \in \mathbb{N}$, $b \geq m$. We perform fixed-precision arithmetic and results are truncated to $b$ bits
    of accuracy after the decimal point.
 1: if $w = 1$ then
 2:     return 1
 3: end if
 4: $\hat{x}_0 \leftarrow 2^{-p}$, where $p \in \mathbb{N}$ such that $2^p > w \geq 2^{p-1}$
 5: $s \leftarrow \lceil \log_2 b \rceil$
 6: for $i = 1$ to $s$ do
 7:     $x_i \leftarrow -w\hat{x}_{i-1}^2 + 2\hat{x}_{i-1}$
 8:     $\hat{x}_i \leftarrow x_i$ truncated to $b$ bits after the decimal point
 9: end for
10: return $\hat{x}_s$
```

The initial approximation used is $\hat{x}_0 = 2^{-p}$, with $2^p > w \geq 2^{p-1}$. The number of iterations s is specified in Algorithm 0 INV. Note that $x_0 < 1/w$ and the iteration converges to $1/w$ from below, i.e., $\hat{x}_i \leq 1/w$. Within each iterative step the arithmetic operations are performed in fixed precision and x_i is computed exactly. We truncate x_i to $b \geq n$ bits after the decimal point to obtain $\hat{x}_i$ and pass it on as input to the next iterative step. Each iterative step is implemented using an elementary module of the form given in Fig. 2.2 that requires only addition and multiplication. The final approximation error is

$$|\hat{x}_s - \frac{1}{w}| \leq \frac{2 + \log_2 b}{2^b}.$$

For the derivation of this error bound see Corollary 8 in Appendix C. We remark that although the iteration function (2.1) is well known in the literature [225, Ex. 5-1], an important property of Algorithm 0 INV is the fixed-precision implementation of Newton iteration for a specific initial approximation and a prescribed number of steps, so that the error bound of Corollary 8 is satisfied.

Turning to the cost, we iterate $O(\log_2 b)$ times and as we mentioned each x_i is computed exactly. Therefore, each iterative step requires $O(n + b)$ qubits and a number of quantum operations for implementing addition and multiplication that is a low-degree polynomial in $n + b$.[2] The cost to obtain the initial approximation is relatively minor when compared to the overall cost of the

[2]For example, addition and multiplication can be performed with $O((n + b)^2)$ basic quantum gates using the grade-school algorithm.

multiplications and additions used in the algorithm.

2.2.2 Square Root

Computing approximately the square root $\sqrt{w}$, $w \geq 1$, can also be approached as a zero finding problem and one can apply Newton iteration to it. However, the selection of the function whose zero is $\sqrt{w}$ has to be done carefully so that the resulting iterative steps are easy to implement and analyze in terms of error and cost. Not all choices are equally good. For example, $f(x) = x^2 - w$, although well known in the literature [225, Ex. 5-1], is not a particularly good choice. The resulting iteration is $x_{i+1} = x_i - (x_i^2 - w)/(2x_i)$, $i = 0, 1, \ldots$, which requires a division using an algorithm such as Algorithm 0 INV at each iterative step. The division also requires circuits keeping track of the position of the decimal point in its result, because its location is not fixed but depends on the values w and x_i. Since the result of a division may not be represented exactly using an a priori chosen fixed number of bits, approximations are needed within each iterative step. This introduces error and overly complicates the analysis of the overall algorithm approximating $\sqrt{w}$ compared to an algorithm requiring only multiplication and addition in each iterative step. All these complications are avoided in our algorithm.

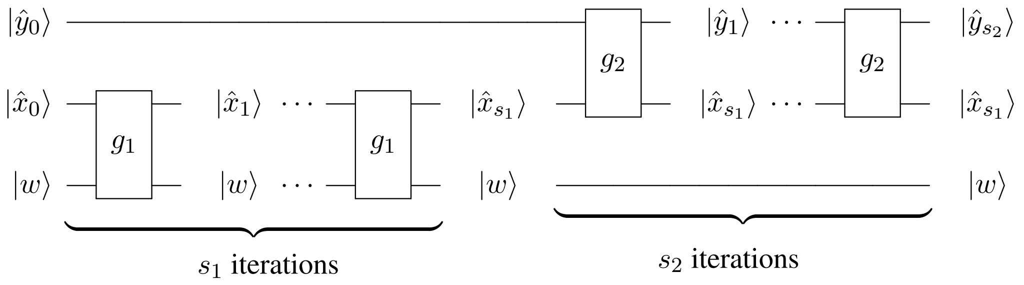


Fig. 2.3: Block diagram of the overall circuit computing $\sqrt{w}$. Two stages of Newton's iteration using the functions g_1 and g_2 are applied s_1 and s_2 times respectively. The first stage outputs $\hat{x}_{s_1} \approx \frac{1}{w}$, which is then used by the second stage to compute $\hat{y}_{s_2} \approx \frac{1}{\sqrt{\hat{x}_{s_1}}} \approx \sqrt{w}$.

Each of the steps of the algorithm we present below can be implemented using only multiplication and addition in fixed-precision arithmetic as in Fig. 2.2. This is accomplished by first approximating $1/w$ (applying iteration g_1 of equation (2.1)) using steps essentially identical to those in Algorithm 0 INV. Then we apply Newton iteration again to a suitably chosen function to approximate its zero and this yields the approximation of $\sqrt{w}$. In particular, in Algorithm 0 INV we set $s = s_1 = s_2$ and first we approximate $1/w$ by $\hat{x}_s$ which has a fixed precision representation with b bits after the decimal point (steps 4–10 of Algorithm 0 INV). Then applying the Newton method to $f(y) = \frac{1}{y^2} - \frac{1}{w}$ we obtain the iteration

$$y_j = g_2(y_{j-1}) := \frac{1}{2}(3\hat{y}_{j-1} - \hat{x}_s\hat{y}_{j-1}^3), \tag{2.2}$$

where $j = 1, 2, \dots, s$. Each y_i is computed exactly and then truncated to b bits after the decimal point to obtain $\hat{y}_i$, which is passed on as input to the next iterative step. See Algorithm 1 SQRT for the values of the parameters and other details. Steps 4 and 10 of the algorithm compute initial approximations for the two Newton iterations, the first computing the reciprocal and the second shown in (2.2). They are implemented using the quantum circuits of Figures 2.4 and 2.5, respectively. A block diagram of the overall circuit of the algorithm computing $\sqrt{w}$ is shown in Figure 2.3.

Algorithm 1 SQRT(w, n, m, b)

```
Require: $w \geq 1$, held in an $n$ qubit register with $m$ qubits for its integer part.
Require: $b \geq \max\{2m, 4\}$. Results are truncated to $b$ bits of accuracy after the decimal point.
 1: if $w = 1$ then
 2:     return 1
 3: end if
 4: $\hat{x}_0 \leftarrow 2^{-p}$, where $p \in \mathbb{N}$ such that $2^p > w \geq 2^{p-1}$
 5: $s \leftarrow \lceil \log_2 b \rceil$
 6: for $i = 1$ to $s$ do
 7:     $x_i \leftarrow -w\hat{x}_{i-1}^2 + 2\hat{x}_{i-1}$
 8:     $\hat{x}_i \leftarrow x_i$ truncated to $b$ bits after the decimal point
 9: end for
10: $\hat{y}_0 \leftarrow 2^{\lfloor (q-1)/2 \rfloor}$, where $q \in \mathbb{N}$ such that $2^{1-q} > \hat{x}_s \geq 2^{-q}$
11: for $j = 1$ to $s$ do
12:     $y_j \leftarrow \frac{1}{2}(3\hat{y}_{j-1} - \hat{x}_s\hat{y}_{j-1}^3)$
13:     $\hat{y}_j \leftarrow y_j$ truncated to $b$ bits after the decimal point
14: end for
15: return $\hat{y}_s$
```

For $w \geq 1$ represented with n bits of which the first m give its integer part, Algorithm 1 SQRT computes $\sqrt{w}$ by $\hat{y}_s$ in fixed precision with $b \geq \max\{2m, 4\}$ bits after its decimal point and we have

$$|\hat{y}_s - \sqrt{w}| \le \left(\frac{3}{4}\right)^{b-2m} (2 + b + \log_2 b)\,.$$

The proof can be found in Theorem 12 in Appendix C.

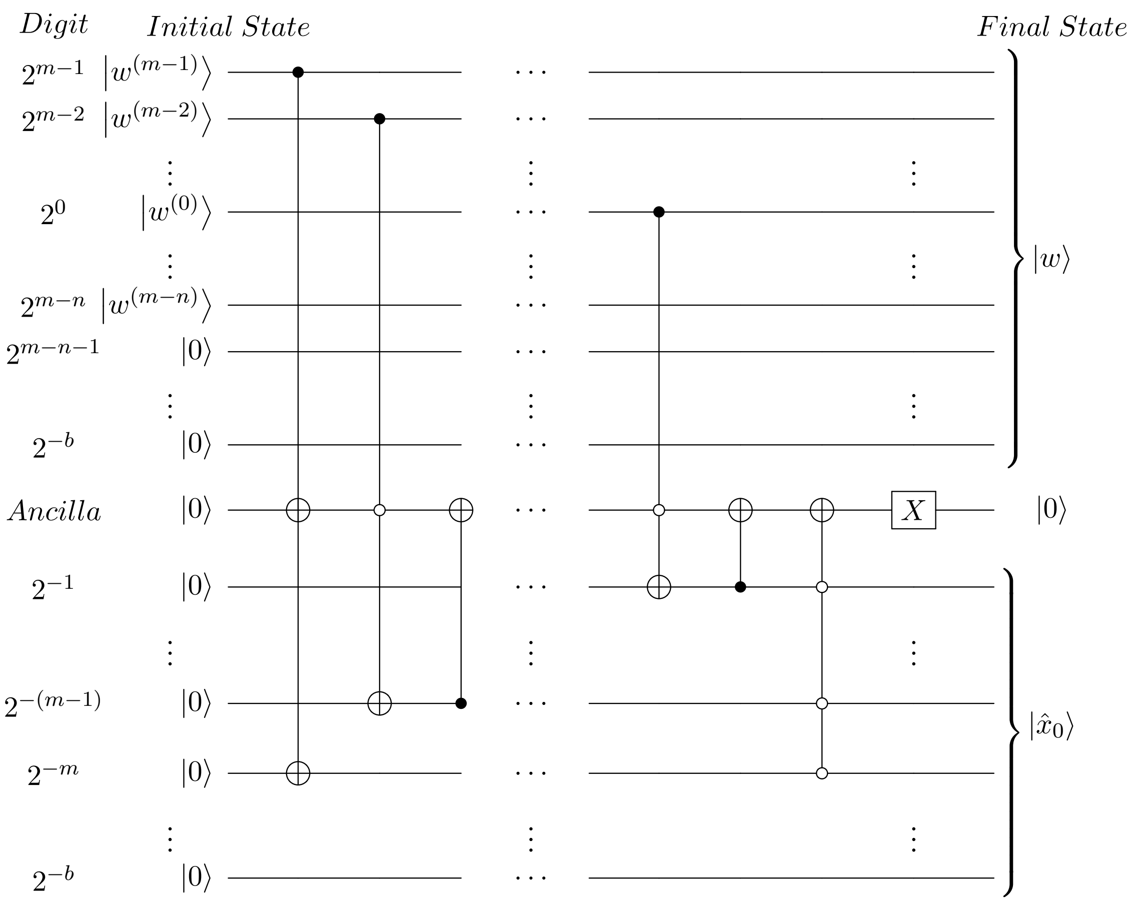
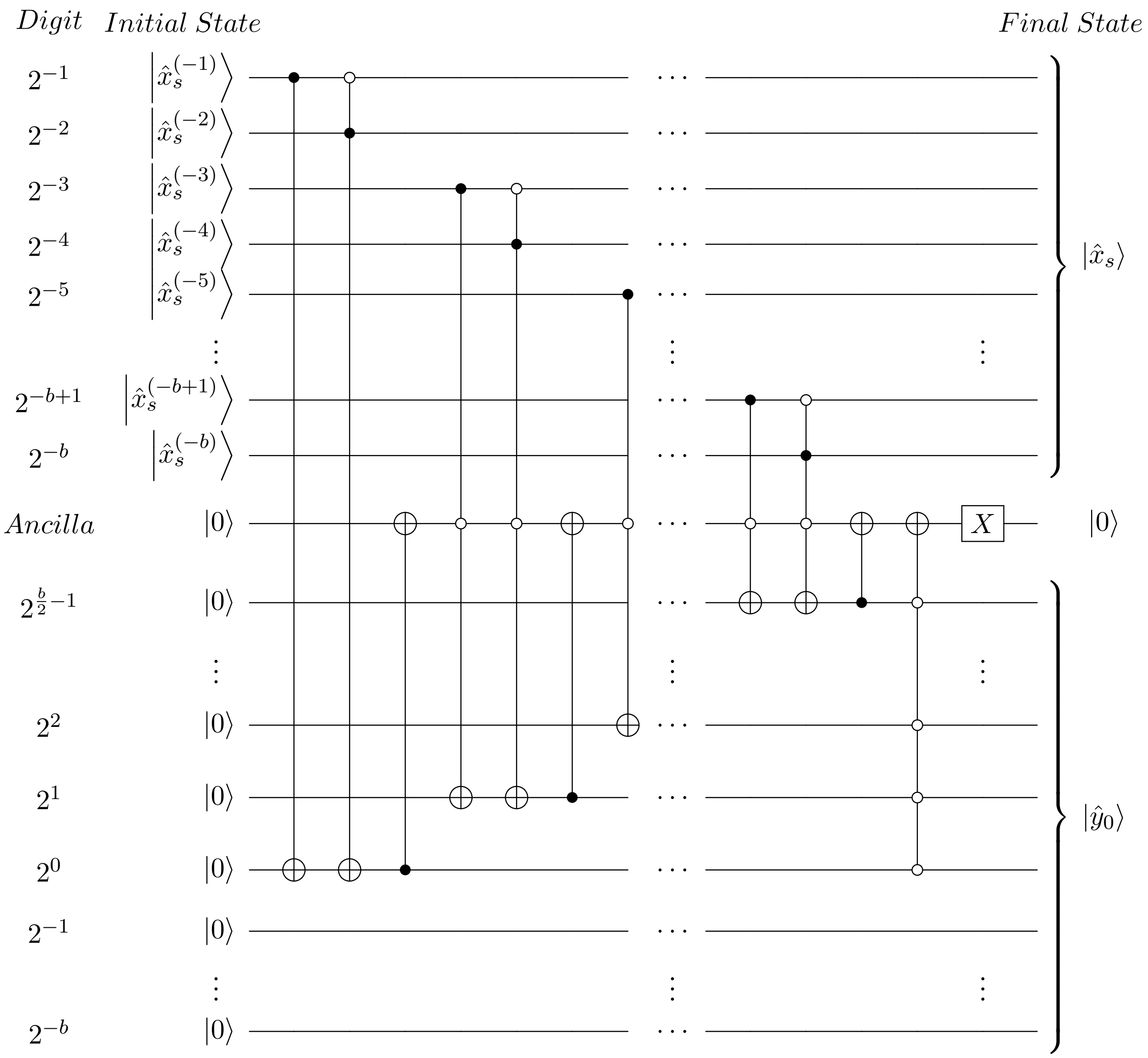


Fig. 2.4: A quantum circuit computing the initial state $|\hat{x}_0\rangle = |2^{-p}\rangle$, for $|w\rangle$ given by n bits of which m are for its integer part, where $p \in \mathbb{N}$ and $2^p > w \ge 2^{p-1}$. Here we have taken $b \ge n - m$. This circuit is used in step 4 of Algorithm 1 SQRT. Each horizontal line ("wire") represents a qubit. This circuit consists of controlled-not (CNOT) and controlled-controlled-not (Toffoli) gates. Each crossed circle indicates a target qubit, controlled by the qubit(s) indicated by black dots. White dots indicate inverted control. See Appendix A for a review of some important quantum gates.

Turning to the cost of the quantum circuit in Fig. 2.3 implementing Algorithm 1 SQRT, we set the number of iterative steps of each of the two iterations to be $s = s_1 = s_2 = O(\log_2 b)$. Observe that each of the iterations g_1 and g_2 can be computed using at most three applications of a quantum circuit of the type shown in Fig. 2.2. Therefore, each iterative step requires $O(n + b)$ qubits and a number of quantum operations for implementing addition and multiplication that is a low-degree polynomial in $n + b$. Observe that the cost to obtain the initial approximations is again relatively minor compared to the overall cost of the additions and multiplications in the algorithm.



Digit
Initial State
Final State
2^{-1}
$\left|\hat{x}_s^{(-1)}\right\rangle$
2^{-2}
$\left|\hat{x}_s^{(-2)}\right\rangle$
2^{-3}
$\left|\hat{x}_s^{(-3)}\right\rangle$
2^{-4}
$\left|\hat{x}_s^{(-4)}\right\rangle$
2^{-5}
$\left|\hat{x}_s^{(-5)}\right\rangle$
2^{-b+1}
$\left|\hat{x}_s^{(-b+1)}\right\rangle$
2^{-b}
$\left|\hat{x}_s^{(-b)}\right\rangle$
Ancilla
$|0\rangle$
$2^{\frac{b}{2}-1}$
2^2
2^1
2^0
2^{-1}
2^{-b}
X
$|\hat{x}_s\rangle$
$|0\rangle$
$|\hat{y}_0\rangle$

Fig. 2.5: A quantum circuit computing the state $|\hat{y}_0\rangle = \left|2^{\lfloor \frac{q-1}{2} \rfloor}\right\rangle$, for $0 < \hat{x}_s < 1$ given by b bits, where $q \in \mathbb{N}$ and $2^{1-q} > \hat{x}_s \geq 2^{-q}$. This circuit is used in step 10 of Algorithm 1 SQRT. It is for the case of even b; a similar circuit follows for odd b.

2.2.3 2^k-Root

Obtaining the roots, $w^{1/2^i}$, $i = 1, \dots, k$, $k \in \mathbb{N}$, is straightforward. It is accomplished by calling Algorithm 1 SQRT iteratively k times, since $w^{1/2^i} = \sqrt{w^{1/2^{i-1}}}$, $i = 1, \dots, k$. In particular, Algorithm 2 PowerOf2Roots calculates approximations of $w^{1/2^i}$, for $i = 1, 2, \dots, k$. The circuit implementing this algorithm consists of k repetitions of the circuit in Fig. 2.3 approximating the square root. The results are truncated to $b \geq \max\{2m, 4\}$ bits after the decimal point before passed on to the next stage. The algorithm produces k numbers $\hat{z}_i$, $i = 1, \dots, k$. We have

$$|\hat{z}_i - w^{1/2^i}| \leq 2 \left(\frac{3}{4}\right)^{b-2m} (2 + b + \log_2 b),$$

$i = 1, \dots, k$. The proof can be found in Theorem 14 in Appendix C.

```
Algorithm 2 PowerOf2Roots($w$, $k$, $n$, $m$, $b$)
Require: $w \geq 1$, held in an $n$ qubit register with $m$ qubits for its integer part.
Require: $k \geq 1$ an integer. The algorithm returns approximations of $w^{\frac{1}{2^i}}$, $i = 1, \dots, k$.
Require: $b \geq \max\{2m, 4\}$. Results are truncated to $b$ bits of accuracy after the decimal point.
1: $\hat{z}_1 \leftarrow$ SQRT($w$, $n$, $m$, $b$). Recall that SQRT returns a number with a fractional part $b$ bits long.
   The integer part of of $\hat{z}_1$ is represented by $m$ bits.
2: for $i = 2$ to $k$ do
3:     $\hat{z}_i \leftarrow$ SQRT($\hat{z}_{i-1}$, $m + b$, $m$, $b$). Note that $\hat{z}_1$ and the $\hat{z}_i$ are held in registers of size $m + b$
       bits of which the $b$ bits are for the fractional part.
4: end for
5: return $\hat{z}_1$,$\hat{z}_2$,...,$\hat{z}_k$
```

Algorithm 2 PowerOf2Roots uses k calls to Algorithm 1 SQRT. Hence it requires $k \log b \cdot O(n + b)$ qubits and $k \log b \cdot p(n + b)$ quantum operations, where p is a low-degree polynomial depending on the specific implementation of the circuit of Fig. 2.2.

2.2.4 Logarithm

To the best of our knowledge, the method presented in this section is entirely new. Let us first introduce the idea leading to the algorithm approximating $\ln(w)$, $w > 1$; the case $w = 1$ is trivial. First we shift w to the left, if necessary, to obtain the number $2^{-\nu} w \in [1, 2)$. It suffices to approximate $\ln(2^{-\nu} w)$ since $\ln(w) = \ln(2^{-\nu} w) + \nu \ln 2$ and we can precompute $\ln 2$ up to any desirable number of bits.

We use the following observation. When one takes the 2^ℓ-root of a number that belongs to the interval $(1, 2)$ the fractional part δ of the result is roughly speaking proportional to $2^{-\ell}$, i.e, it is quite small for relatively large ℓ. Therefore, for $1+\delta = [2^{-\nu}w]^{1/2^\ell}$ we use the power series expansion for the logarithm to approximate $\ln(1+\delta)$ by $\delta - \delta^2/2$, with any desired accuracy since δ can be made arbitrarily small by appropriately selecting ℓ. Then the approximation of the logarithm follows from

$$\ln(w) \approx \nu \ln 2 + 2^\ell(\delta - \frac{\delta^2}{2}). \tag{2.3}$$

In particular, Algorithm 3 LN approximates $\ln(w)$ for $w \geq 1$ represented by n bits of which the first m are used for its integer part. (The trivial case $w = 1$ is dealt with first.) In step 7, $p-1$ is the value of ν, i.e., $\nu = p-1$, where $2^p > w \geq 2^{p-1}$, $p \in \mathbb{N}$. We compute $w_p = 2^{1-p}w \in [1, 2)$ using a right shift of w. We explain how to implement the shift operation below. Then, $\hat{t}_p$, an approximation of $w_p^{1/(2\ell)}$, is calculated using Algorithm 2 PowerOf2Roots, where ℓ is a parameter that determines the error. Note that $\hat{t}_p$ can be written as $1 + \delta$, for $0 < \delta < 1$. We provide precise bounds for δ in Theorem 14 in Appendix C. Next, an approximation $\hat{y}_p$ of $\ln \hat{t}_p = \ln(1 + \delta)$ is calculated, using the first two terms of the power series for the logarithm as we explained above. The whole procedure yields an estimate of $(\ln w_p)/2^\ell$. Finally the algorithm in steps 16 – 20 uses equation (2.3), with $\nu = p - 1$, to derive $z_p + (p-1)r$ as an approximation to $\ln w$. All intermediate calculation results are truncated to b bits after the decimal point. The value of b is determined by the value of ℓ in step 1 of the algorithm. Note that the precision of the algorithm grows with ℓ, which is a user selected parameter that must satisfy $\ell \geq \lceil \log_2 8n \rceil$.

The block diagram of the overall circuit implementing Algorithm 3 LN can be found in Fig. 2.6. The first module is a right shift operation that calculates w_p. There are a number of ways to implement it. One is to implement the shift by a multiplication of w by a suitable negative power of two. This is convenient since we have elementary quantum circuits for multiplication. In Fig. 2.7 we show a circuit computing the necessary power of two, which we denote by x in that figure. In particular, for $w \geq 2$ we set $x = 2^{1-p}$, where $p - 1 = \lfloor \log_2 w \rfloor \geq 1$. For $1 \leq w < 2$ we set $x = 1$. Thus m bits are needed for the representation of x, with the first bit $x^{(0)}$ denoting its integer part and all the remaining bits $x^{(-1)}, \ldots, x^{-(m-1)}$ denoting its fractional part. We implement the shift of w in terms of multiplication between w and x, i.e., $w_p = w\,x$. Since w and x are held in registers of known size and we use fixed precision representation we know a priori the position of the decimal point in their product w_p. Moreover, since $w_p \in [1, 2 - 2^{1-n}]$ (w is an n bit number) we have that n bits (qubits), of which 1 is used for its integer part, suffice to hold w_p exactly.

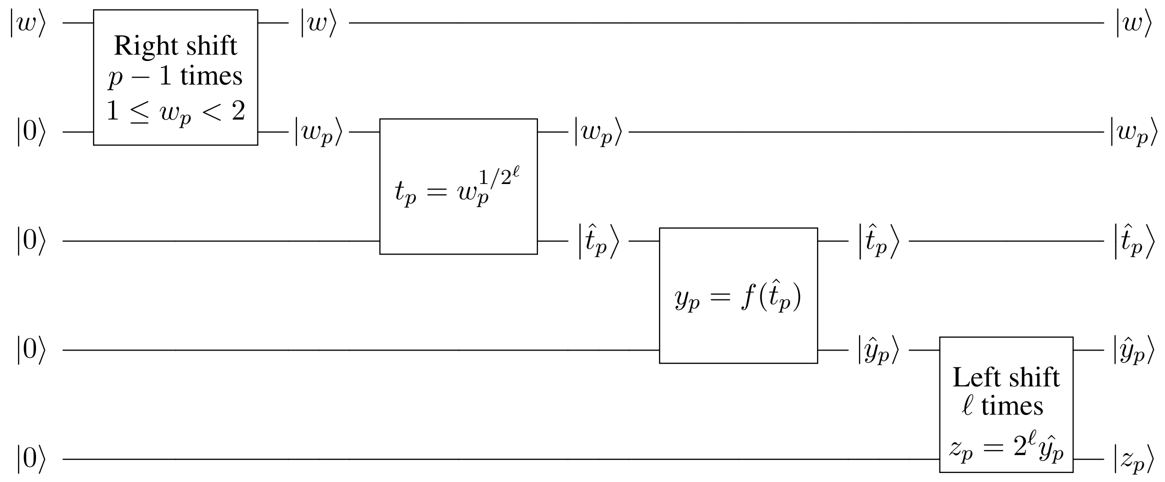


Fig. 2.6: Overall circuit schematic for approximating $\ln w$. The gate $f(\hat{t}_p)$ outputs $\hat{y}_p = (\hat{t}_p - 1) - \frac{1}{2}(\hat{t}_p - 1)^2$. Once z_p is obtained the approximation of $\ln w$ is computed by the expression $z_p + (p-1)r$, where r approximates $\ln 2$ with high accuracy and $p - 1$ is obtained from a quantum circuit as the one shown in Fig. 2.8.

The next module is the circuit for the PowerOf2Roots algorithm and calculates $\hat{t}_p$. The third module calculates $\hat{y}_p$ and is comprised of modules that perform subtraction and multiplication in fixed precision. The fourth module of the circuit performs a series of left shift operations. Observe that ℓ is an accuracy parameter whose value is set once at the very beginning and does not change during the execution of the algorithm. Thus the left shift ℓ times is, in general, much easier to implement than the right shift of w at the beginning of the algorithm and we omit the details.

In its last step Algorithm 3 LN computes the expression $\hat{z} := z_p + (p-1)r$ which is the estimate of $\ln w$. The value of $p - 1$ in this expression is obtained using the quantum circuit of Fig. 2.8. The error of the algorithm satisfies

$$|\hat{z} - \ln w| \le \left(\frac{3}{4}\right)^{5\ell/2} \left(m + \frac{32}{9} + 2\left(\frac{32}{9} + \frac{n}{\ln 2}\right)^3\right).$$

For the proof details see Theorem 14 in Appendix C.

Algorithm 3 LN(w, n, m, ℓ)

```
Require: $w \geq 1$, held in an $n$ qubit register with $m$ qubits for its integer part.
Require: $\ell \geq \lceil \log_2 8n \rceil$ is a parameter upon which the error will depend and which we use to
    determine the number of bits $b$ after the decimal points in which arithmetic will be performed.
 1: $b \leftarrow \max\{5\ell, 25\}$. Results are truncated to $b$ bits of accuracy after the decimal point in the
    intermediate calculations.
 2: $r \leftarrow \ln 2$ with $b$ bits of accuracy, i.e., $|r - \ln 2| \leq 2^{-b}$. An approximation of $\ln 2$ can be
    precomputed with sufficiently many bits of accuracy and stored in a register, from which we
    take the first $b$ bits.
 3: if $w = 1$ then
 4:     return $0$
 5: end if
 6: Let $p \in \mathbb{N}$ be such that $2^p > w \geq 2^{p-1}$
 7: if $p - 1 = 0$ then
 8:     $w_p \leftarrow w$. In this case $w = w_p \in [1, 2 - 2^{1-n}]$.
 9: else
10:     $w_p \leftarrow w 2^{1-p}$. Note that $w_p \in [1, 2 - 2^{1-n}]$ for $w \geq 2$. The number of bits (qubits) used for
        $w_p$ is $n$ of which 1 is for its integer part as explained in the text and the caption of Fig. 2.7.
11:     $x_p \leftarrow w_p - 1$. This is the fractional part of $w_p$.
12: end if
13: if $x_p = 0$ then
14:     $z_p \leftarrow 0$
15: else
16:     $\hat{t}_p \leftarrow$PowerOf2Roots($w_p$, $\ell$, $n$, $1$, $b$)[$\ell$]. The function $PowerOf2Roots$ returns a list of
        numbers and we take the last element, the $1/2^\ell$th root. Note that in this particular case
        $1 \leq \hat{t}_p < 2$.
17:     $\hat{y}_p \leftarrow (\hat{t}_p - 1) - \frac{1}{2}(\hat{t}_p - 1)^2$, computed to $b$ bits of accuracy after the decimal point. Note that
        $\hat{t}_p = 1 + \delta$ with $\delta \in (0, 1)$, and we approximate $\ln(1 + \delta)$ by $\delta - \frac{1}{2}\delta^2$, the first two terms of
        its power series expansion.
18:     $z_p \leftarrow 2^\ell \hat{y}_p$. This corresponds to a logical right shift of the decimal point.
19: end if
20: return $z_p + (p - 1)r$
```

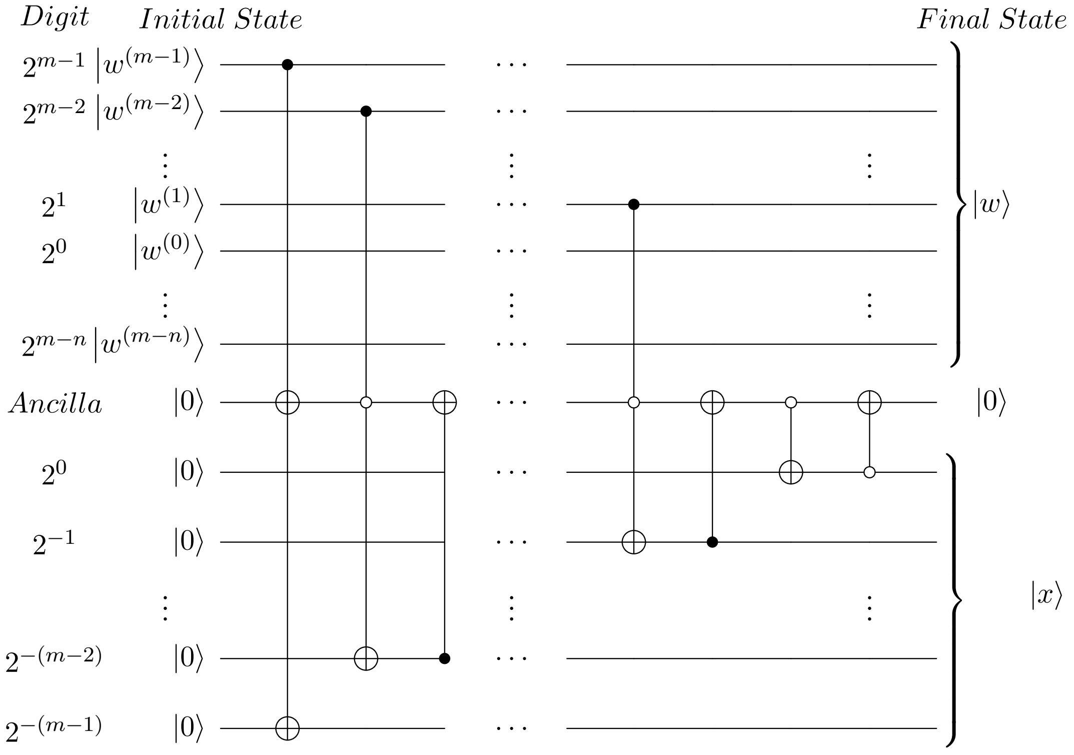
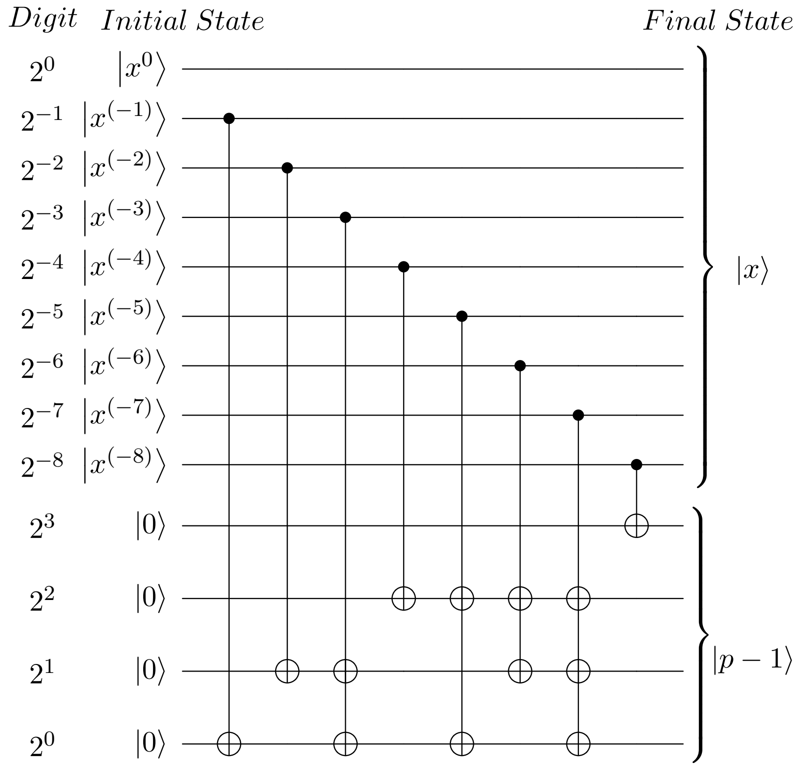


Fig. 2.7: For $w \geq 1$ this quantum circuit computes $|x\rangle$ where x is an m bit number $x \in [2^{1-m}, 1]$. For $w \geq 2$ we set $x = 2^{1-p}$, where $p - 1 = \lfloor \log_2 w \rfloor \geq 1$. For $1 \leq w < 2$ we set $x = 1$. Thus m bits are needed for the representation of x, with the first bit $x^{(0)}$ denoting its integer part and all the remaining bits $x^{(-1)}, \ldots, x^{-(m-1)}$ denoting its fractional part. This circuit is used in steps 6 – 10 of Algorithm 3 LN to derive $x = 2^{1-p}$ so one can implement the shift of w in terms of multiplication between w and x, i.e., $w_p = w\,x$.

Considering the cost of the quantum circuit in Fig. 2.6 implementing Algorithm 3 LN, we have that the cost of computing the initial and the last shifts (first and fourth modules in Fig. 2.6), as well as the cost of the arithmetic expression in the third module in Fig. 2.6, the cost of computing $p - 1$ (see the circuits in Fig. 2.7 and Fig. 2.8) and the cost of the expression $z_p + (p-1)r$, are each relatively minor compared to the other costs of the algorithm. The algorithm calls Algorithm 2 PowerOf2Roots with input parameter $k := \ell$, which requires $O(\ell(n+b)\log b)$ qubits and $\ell \log b \cdot p(n+b)$ quantum operations, as explained in the previous subsection. Observe that the expression in Step 17 of Algorithm 3 LN can be computed using a constant number of applications of a quantum circuit of the type shown in Fig. 2.2. Hence, the overall cost of Algorithm 3 LN is $O(\ell(n+b)\log b)$

qubits and requires a number of quantum operations proportional to $\ell \log b \cdot p(n+b)$.



$Digit$
$Initial\ State$
$Final\ State$
2^0 $|x^0\rangle$
2^{-1} $|x^{(-1)}\rangle$
2^{-2} $|x^{(-2)}\rangle$
2^{-3} $|x^{(-3)}\rangle$
2^{-4} $|x^{(-4)}\rangle$
2^{-5} $|x^{(-5)}\rangle$
2^{-6} $|x^{(-6)}\rangle$
2^{-7} $|x^{(-7)}\rangle$
2^{-8} $|x^{(-8)}\rangle$
2^3 $|0\rangle$
2^2 $|0\rangle$
2^1 $|0\rangle$
2^0 $|0\rangle$
$|x\rangle$
$|p-1\rangle$

Fig. 2.8: Example of a quantum circuit computing $p-1 \geq 0$ required in the last step of Algorithm 3 LN. The input to this circuit is the state $|x\rangle$ computed in Fig. 2.7 where $x = 2^{-(p-1)}$. Recall that m bits are used to store x, and clearly $\lceil \log_2 m \rceil$ bits suffice to store $p-1$ exactly. In this example, $m = 9$. It is straightforward to generalize this circuit to an arbitrary number m.

2.2.5 Fractional Power

Another application of interest is the approximation of fractional powers of the form w^f, where $w \geq 1$ and $f \in [0,1]$ a number whose binary form is n_f bits long. The main idea is to calculate appropriate powers of the form $w^{1/2^i}$ according to the value of the bits of f and multiply the results. Hence initially, the algorithm PowersOf2Roots is used to derive a list of approximations $\hat{w}_i$ to the powers $w_i = w^{1/2^i}$, for $i = 1, 2, \dots, n_f$. The final result that approximates w^f is $\Pi_{i \in \mathcal{P}} \hat{w}_i$, where $\mathcal{P} = \{1 \leq i \leq n_f : f_i = 1\}$ and f_i denotes the ith bit of the number f. The process is described in detail in Algorithm 4 Fractional Power.

Algorithm 4 FractionalPower(w, f, n, m, n_f, ℓ)

```
Require: $w \geq 1$, held in an $n$ qubit register with $m$ qubits for its integer part.
Require: $\ell \in \mathbb{N}$ is a parameter upon which the error will depend. We use it to determine the number
    of bits $b$ after the decimal points in which arithmetic will be performed.
Require: $1 \geq f \geq 0$ is a binary string corresponding to a fractional number given with $n_f$ bits of
    accuracy after the decimal point. The algorithm returns an approximation of $w^f$.
 1: $b \leftarrow \max\{n, n_f, \lceil 5(\ell + 2m + \ln n_f) \rceil, 40\}$. Results are truncated to $b$ bits of accuracy after the
    decimal point.
 2: if $f = 1$ then
 3:     return w
 4: end if
 5: if $f = 0$ then
 6:     return 1
 7: end if
 8: $\{\hat{w}_i\}$ ←PowerOf2Roots($w$, $n_f$, $n$, $m$, $b$). The function returns a list of numbers $\hat{w}_i$ approxi-
    mating $w_i = w^{\frac{1}{2^i}}$, $i = 1, \dots, n_f$.
 9: $\hat{z} \leftarrow 1$
10: for $i = 1$ to $n_f$ do
11:     if the $i$th bit of $f$ is 1 then
12:         $z \leftarrow \hat{z}\hat{w}_i$
13:         $\hat{z} \leftarrow z$ truncated to $b$ bits after the decimal point
14:     end if
15: end for
16: return $\hat{z}$
```

For $w > 1$ the value $\hat{z}$ returned by the algorithm satisfies

$$|\hat{z} - w^f| \leq \left(\frac{1}{2}\right)^{\ell-1},$$

where w is represented by n bits of which m are used for its integer part, n_f is number of bits in the representation of the exponent f, $\ell \in \mathbb{N}$ is a user selected parameter determining the accuracy of the result. The results of all intermediate steps are truncated to $b \geq \max\{n, n_f, [5(\ell+2m+\ln n_f)], 40\}$ bits before passing them on to the next step. The proof can be found in Theorem 15 in Appendix C.

The algorithm can be extended to calculate $w^{p/q}$, $w > 1$, where the exponent is a rational number $p/q \in [0, 1]$. First f, an n_f bit number, is calculated such that it approximates p/q within n_f bits of accuracy, namely $|f - \frac{p}{q}| \leq 2^{-n_f}$. Then f is used as the exponent in the parameters of Algorithm 4 FractionalPower to get an approximation of w^f, which in turn is an approximation of $w^{p/q}$. The value $\hat{z}$ returned by the algorithm satisfies

$$|\hat{z} - w^{p/q}| \leq \left(\frac{1}{2}\right)^{\ell-1} + \frac{w \ln w}{2^{n_f}}.$$

The proof can be found in Corollary 9 in Appendix C.

Remark 1. *For example, for $p/q = 1/3$, one can use Algorithm 0 INV to produce an approximation f of $1/3$ and pass that to the algorithm. In such a case, the approximation error $|w^{p/q} - w^f| \leq 2^{-n_f} w \ln w$, obtained using the mean value theorem for the function w^x, appears as the last term of the equation above. For the details see Corollaries 9 and 10 in Appendix C.*

When $w \in (0, 1)$ we can shift it appropriately to obtain a number greater than one to which we can apply Algorithm 4 FractionalPower. However, when approximating the fractional power, *undoing* the initial shift to obtain an estimate of w^f is a bit more involved than it is for the previously considered functions. For this reason we provide the details in Algorithm 5 FractionalPower2. The algorithm first computes k such that $2^k w \geq 1 > 2^{k-1} w$; see Fig. 2.9. Using k the algorithm shifts w to the left to obtain $x := 2^k w$. The next step is to approximate x^f. Observe that $x^f = 2^{kf} w^f$. Therefore to undo the initial shift we have to divide by $2^{kf} = 2^c 2^{\{c\}}$, where c denotes the integer part of kf and $\{c\}$ denotes its fractional part. Dividing by 2^c is straightforward and is accomplished using shifts. Dividing by $2^{\{c\}}$ is accomplished by first approximating $2^{\{c\}}$ and then multiplying by its approximate reciprocal. See Algorithm 5 FractionalPower2 for the details.

The value $\hat{t}$ the algorithm returns as an approximation of w^f, $w \in (0, 1)$, satisfies

$$|\hat{t} - w^f| \leq \frac{1}{2^{\ell-3}},$$

where ℓ is a user-defined parameter. The proof can be found in Theorem 16 in Appendix C.

Just like before we can approximate w^p/q, $w \in (0, 1)$, and a rational exponent $p/q \in [0, 1]$, by first approximating p/q and then calling Algorithm 5 FractionalPower2. The value $\hat{t}$ the algorithm returns satisfies

$$|\hat{t} - w^{p/q}| \leq \left(\frac{1}{2}\right)^{\ell-2} + \frac{w \ln w}{2^{n_f}}.$$

See Corollary 10 in Appendix C.

We now address the cost of our Algorithms computing w^f. First consider Algorithm 4 FractionalPower, which calls Algorithm 2 PowerOf2Roots with input parameters $k := n_f$ and $b :=$

$\max\{n, n_f, \lceil 5(\ell + 2m + \ln n_f)\rceil, 40\}$, which requires $O(n_f(n + b)\log b)$ qubits and of order $n_f \log b \cdot p(n + b)$ quantum operations, as explained in the previous subsections. At most n_f multiplications are then required, using a quantum circuit of the type shown in Fig. 2.2. Therefore, Algorithm 4 FractionalPower requires a number of qubits and a number of quantum operations that is a low-degree polynomial in n, n_f, and ℓ, respectively.

Algorithm 5 FractionalPower2(w, f, n, m, n_f, ℓ)

```
Require: $0 \le w < 1$, held in an $n$ qubit register with $m$ qubits for its integer part. (For $w \ge 1$, use
    Algorithm 4.)
Require: $\ell \in \mathbb{N}$ is a parameter upon which the error will depend. We use it to determine the number
    of bits $b$ after the decimal points in which arithmetic will be performed.
Require: $f \in [0, 1]$ specified to $n_f$ bits. The algorithm returns an approximation of $w^f$.
 1: $b \leftarrow \max\{n, n_f, \lceil 2\ell + 6m + 2\ln n_f\rceil, 40\}$. Results are truncated to $b$ bits of accuracy after the
    decimal point.
 2: if $w = 0$ then
 3:     return 0
 4: end if
 5: if $f = 1$ then
 6:     return w
 7: end if
 8: if $f = 0$ then
 9:     return 1
10: end if
11: $x \leftarrow 2^k w$, where $k$ is a positive integer such that $2^k w \ge 1 > 2^{k-1} w$. This corresponds to a
    logical right shift of the decimal point. An example of the quantum circuit computing $k$ is given
    in Fig. 2.9.
12: $c \leftarrow kf$
13: $\hat{z} \leftarrow$FractionalPower($x$, $f$, $n$, $m$, $n_f$, $\ell$). This computes an approximation of $z = x^f$.
14: $\hat{y} \leftarrow$FractionalPower(2, $\{c\}$, $n$, $m$, $n_f$, $\ell$). This computes an approximation of $y = 2^{\{c\}}$,
    where $\{c\} = c - \lfloor c \rfloor$ (for $c \ge 0$) denotes the fractional part of $c$. Since we use fixed-precision
    arithmetic, the integer and fractional parts of numbers are readily available.
15: $\hat{s} \leftarrow$ INV($\hat{y}, n, 1, 2\ell$). This computes an approximation of $s = \hat{y}^{-1}$.
16: $v \leftarrow 2^{-\lfloor c \rfloor}\hat{z}$. This corresponds to a logical left shift of the decimal point.
17: $t \leftarrow v\hat{s}$
18: $\hat{t} \leftarrow t$, truncated to $b$ bits after the decimal point.
19: return $\hat{t}$
```

Now consider Algorithm 5 FractionalPower2, which requires two calls to Algorithm 4 FractionalPower, one call to Algorithm 0 INV, and a constant number of calls to a quantum circuit of the type shown in Fig. 2.2. Hence, using the previously derived bounds for the costs of each of these modules, the cost of Algorithm 5 FractionalPower2 in terms of both the number of quantum operations and the number of qubits is a low-degree polynomial in n, n_f, and ℓ, respectively.

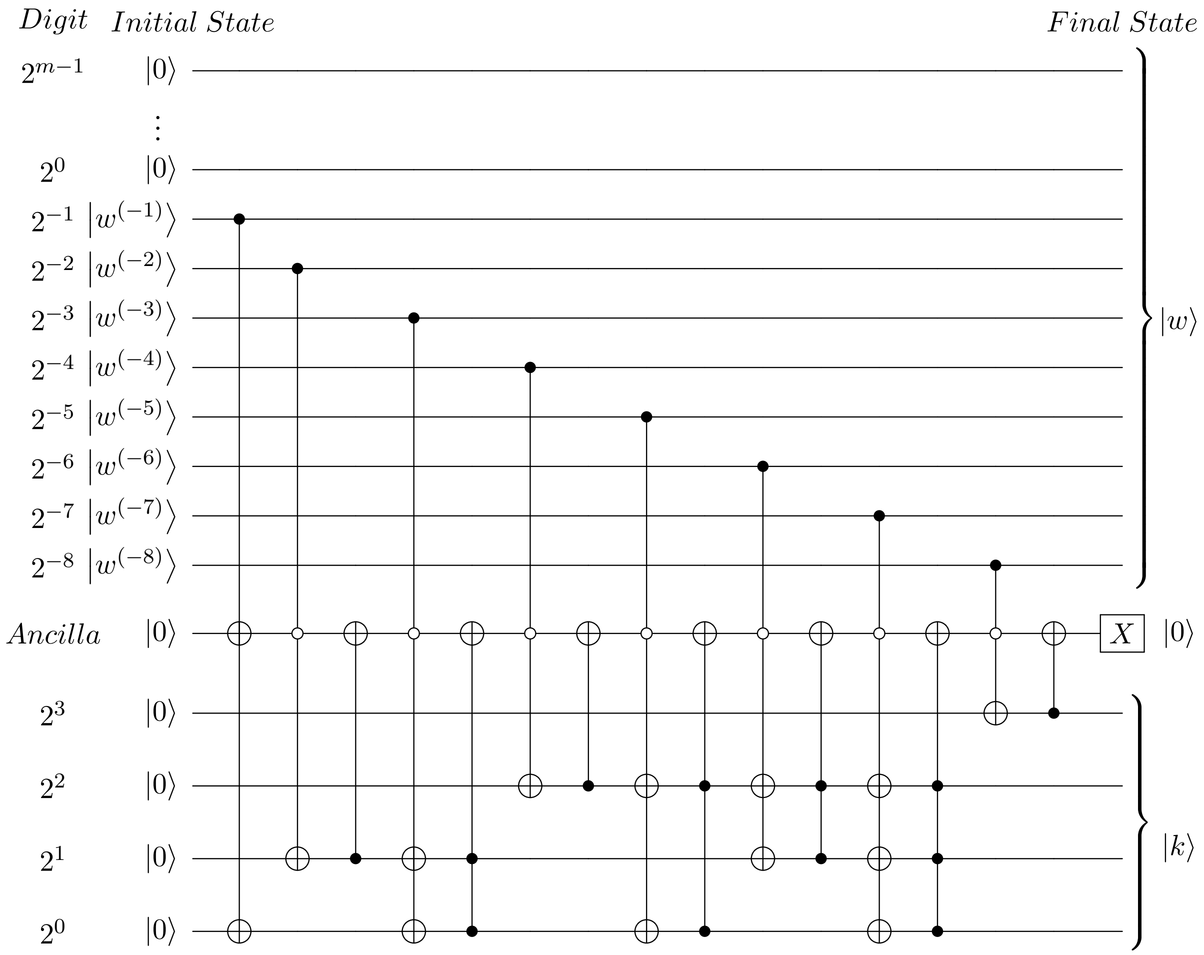

$Digit$
$Initial\ State$
$Final\ State$
2^{m-1}
$|0\rangle$
2^0
$|0\rangle$
2^{-1} $|w^{(-1)}\rangle$
2^{-2} $|w^{(-2)}\rangle$
2^{-3} $|w^{(-3)}\rangle$
2^{-4} $|w^{(-4)}\rangle$
2^{-5} $|w^{(-5)}\rangle$
2^{-6} $|w^{(-6)}\rangle$
2^{-7} $|w^{(-7)}\rangle$
2^{-8} $|w^{(-8)}\rangle$
$|w\rangle$
$Ancilla$ $|0\rangle$
X
$|0\rangle$
2^3 $|0\rangle$
2^2 $|0\rangle$
2^1 $|0\rangle$
2^0 $|0\rangle$
$|k\rangle$

Fig. 2.9: Example of a quantum circuit computing the positive integer k such that $2^k w \geq 1 > 2^{k-1} w$, for $0 < w < 1$ with $n - m$ bits after the decimal point. In this example, $n - m = 8$. It is straightforward to generalize this circuit to arbitrary numbers of bits. This circuit is used in Algorithm 5 FractionalPower2.

2.3 Numerical Results

We present a number of tests comparing our algorithms to the respective ones implemented using floating point arithmetic in Matlab. To ensure that we compare against highly accurate values in Matlab we have used variable precision arithmetic (vpa) with 100 significant digits. In particular, we simulate the execution of each of our algorithms and obtain the result. We obtain the corresponding result using a built-in function in Matlab. We compare the number of equal significant digits in the two results. Our tests show that using a moderate amount of resources, our algorithms compute values matching the corresponding ones obtained by commercial, widely used scientific computing software with 12 to 16 decimal digits. In our tests, the input w was chosen randomly.

Algorithm: SQRT

In Table 2.2, all calculations for SQRT are performed with $b = 64$ bits (satisfying $b > 2m$ for all inputs). The first column in the table gives the different values of w in our tests, the second and third columns give the computed value using Matlab and our algorithm respectively, and the last column gives the number of identical significant digits between the output of our algorithm and Matlab.

w	Matlab: $w^{1/2}$	Our Algorithm: $w^{1/2}$	# of Identical Digits
0.0198	0.140712472794703	0.140712472794703	16
48	6.928203230275509	6.928203230275507	15
91338	302.2217728754829	302.2217728754835	14
171234050	13085.64289593752	13085.64289596872	12

Table 2.2: Numerical results for SQRT.

In Fig. 2.10 we give simulation results showing the error of our algorithm for different values of the parameter b. In particular, we show how the worst-case error of the algorithm depends on b, and then, using the results of Matlab as the correct values of $\sqrt{w}$, we show the actual error of our algorithm in relation to b in a number of test cases.

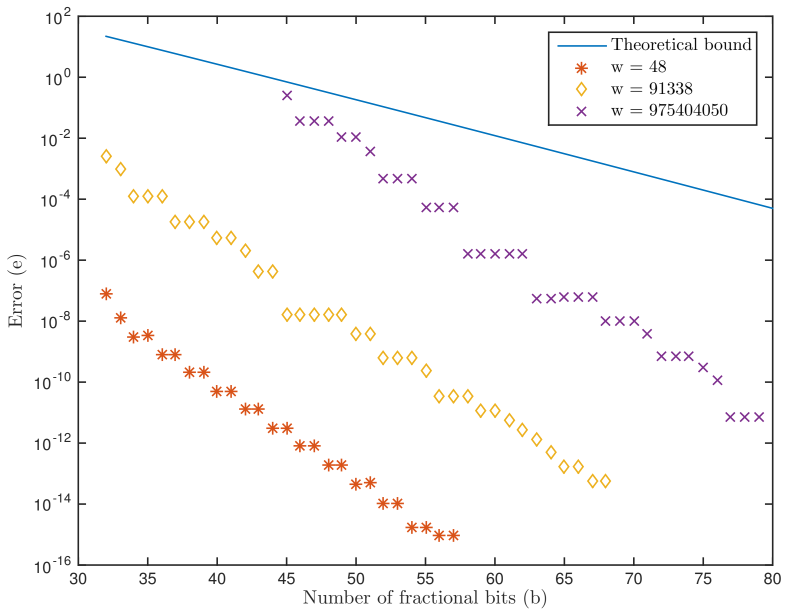


Fig. 2.10: Semilog plot showing the error of Algorithm 1 SQRT versus the number of precision bits b. The solid blue line is a plot of the worst-case error of Theorem 1, for $n = 2m = 64$. The three data sets represent the absolute value of the difference between Matlab's floating point calculation of $\sqrt{w}$ and our algorithm's result for three different values of w.

Algorithm: PowerOf2Roots

In Table 2.3, we show only the result for the 2^kth root (the highest root), where k was chosen randomly such that $5 \leq k \leq 10$. Again, all our calculations are performed with $b = 64$ bits.

w	k	Matlab: $w^{1/2^k}$	Our Algorithm: $w^{1/2^k}$	# of Identical Digits
0.3175	6	0.982233508377946	0.982233508377946	16
28	10	1.003259406317532	1.003259406317532	16
15762	5	1.352618595919273	1.352618595919196	13
800280469	8	1.083373703681284	1.083373703681403	13

Table 2.3: Numerical results for PowerOf2Roots.

Algorithm: LN

In Fig. 2.11 we give simulation results showing the error of our algorithm for different values of the parameter ℓ that controls the desired accuracy. In particular, we show how the worst-case error of the algorithm depends on ℓ, and then, using the results of Matlab as the correct values of $\ln(w)$, we show the actual error of our algorithm in relation to ℓ in a number of test cases.

For the tests shown in Table 2.4 below we have used $\ell = 50$.

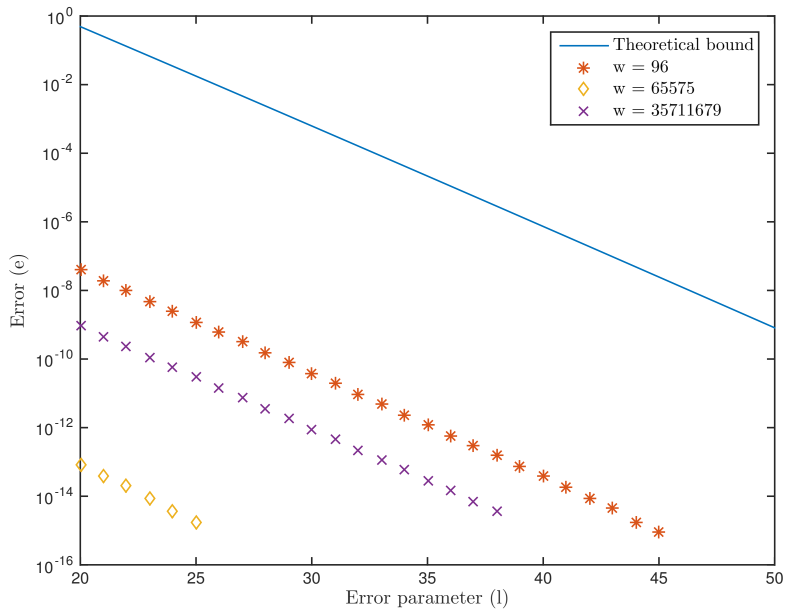


Fig. 2.11: Semilog plot showing the error of Algorithm 3 LN versus the parameter ℓ which controls the accuracy of the result. The solid blue line shows the worst-case error of Theorem 3, for $n = 2m = 64$, $b = \max\{5\ell, 25\}$. The three data sets represent the absolute value of the difference between Matlab's floating point calculation of $\ln(w)$ and our algorithm's result for three different values of w.

Algorithm: FractionalPower

The table below shows tests for the approximation of w^f for randomly generated $f \in (0, 1)$. This result of this algorithm also depends on a user-determined parameter ℓ controlling the accuracy. We have used $\ell = 50$ in the tests shown in Table 2.5.

w	Matlab: $\ln(w)$	Our Algorithm: $\ln(w)$	# of Identical Digits
96	4.564348191467836	4.564348191467836	16
65575	11.090949804735075	11.090949804735075	17
35711679	17.390988336107455	17.390988336107455	17

Table 2.4: Numerical results for LN.

w	f	Matlab: w^f	Our Algorithm: w^f	# of Identical Digits
0.7706	0.1839	0.953208384891996	0.953208384891998	15
76	0.7431	24.982309269657478	24.982309269657364	14
1826	0.1091	2.268975123215851	2.268975123215838	14
631182688	0.5136	33094.79142555447	33094.79142555433	14

Table 2.5: Numerical results for FractionalPower.

2.4 Discussion

In designing algorithms for scientific computing the most challenging task is to control the error propagation and to do so efficiently. In this sense, it is important to derive reusable modules with well-controlled error bounds. We have given efficient, reversible, and scalable quantum algorithms for computing a variety of basic numerical functions and derived worst-case error bounds. The designs are modular, combining a number of elementary quantum circuits to implement the functions, and the resulting algorithms have cost that is polynomial in the problem parameters.

It is worthwhile to comment on the implementation of our algorithms. Since we are interested in quantum algorithms, the algorithms must be reversible. To the extent that the modules are based on classical computations, their reversibility is not a major issue since [27, 158, 192] show how to simulate them reversibly. There are time/space trade-offs in an implementation that are of theoretical and practical importance. These considerations, as well as other constraints such as (fault-tolerant) gate sets or locality restrictions [221], should be optimized by the compiler, but we are not concerned with them here. For a discussion of the trade-offs, see, for example, [187] and the references therein.

We are at an early stage in the development of quantum computation, quantum programming languages and compilers. It is anticipated that implementation decisions will be made taking into account technological limitations concerning the different quantum computer architectures, but also it is expected that things will change with time and some of the present constraints may no longer be as important. Nevertheless, simple but concrete implementation of the algorithms in this chapter are outlined in [35], though there are alternatives that one may opt for in practice.

Furthermore, although optimization of the individual quantum circuits realizing the algorithms for scientific computing described in this chapter is desirable, to a certain degree, it is not a panacea. Quantum algorithms may use one or more of our modules in different ways while performing their tasks. Therefore, global optimization of the resulting quantum circuit and resource management is much more important and will have to be performed by the compiler and optimizer. Quantum programming languages and compilers is an active area of research with many open questions.

In summary, our results are important first steps towards the goal of ultimately developing a comprehensive library of quantum circuits for scientific computing. The immediate next step is to further extend our results to a wider class of numerical functions. An important open question is whether or not, and for which numerical functions, quantum computers can exploit quantum mechanical effects to compute such functions in a novel way that is more efficient than classical algorithms. For instance, using the Fourier transform; see, e.g., [250].

Finally, we remark that our algorithms have applications beyond quantum computation. For instance, they can be used in signal processing and easily realized on an FPGA (Field Programmable Gate Array), providing superior performance with low cost. Moreover, there is resurgent interest in fixed-precision algorithms for low power/price applications such as mobile or embedded systems, where hardware support for floating-point operations is often lacking [39].

Chapter 3

Approximating Ground and Excited State Energies on a Quantum Computer

3.1 Introduction

Computing eigenvalues of Hamiltonians with a large number of degrees of freedom is a very challenging problem in computational science and engineering. Hamiltonian eigenvalues give the system energy levels, corresponding to the *ground* and *excited* states of the system. For example, one of the most important tasks in chemistry is to calculate the energy levels of molecules, where the number of degrees of freedom is proportional to the number of particles, which are required for computing reaction rates and electronic structure properties that, in particular, depend principally on the low-order energy levels. The best classical algorithms known for such problems have cost that grows exponentially in the number of degrees of freedom [154]. Therefore, efficient quantum algorithms for computing Hamiltonian eigenvalues would be an extremely powerful tool for new science and technology.

On the other hand, there are a number of recent results in discrete complexity theory suggesting that many eigenvalue problems are very hard even for quantum computers because they are QMA-complete [44,61,148,205,243]. However, discrete complexity theory deals with the worst case over large classes of Hamiltonians. It does not provide methods or necessary conditions determining when an eigenvalue problem is hard. In fact, there is a dichotomy between theory and practice. As stated in [164], "complexity theoretic proofs of the advantage of many widely used classical algorithms are few and far between." Therefore, it is important to develop new quantum algorithms and to use them for solving eigenvalue problems for which quantum computing can be shown to have a significant advantage over classical computing.

Several recent works have made progress in this direction. In [183], the authors developed an algorithm and proved strong exponential quantum speedup for approximating the ground state energy (i.e., the smallest eigenvaue) of the time-independent Schrödinger equation under certain assumptions. Why this problem is different from the QMA-complete problems of discrete complexity theory was made clear in [184]. The authors of [182] relaxed the assumptions of [183] to give a ground state energy approximation algorithm for the time-independent Schrödinger equation with a convex potential.

An important advance would be to obtain analogous results for approximating excited state energies, under weakened assumptions. The techniques of [182–184] for approximating the ground state energy do not extend to excited state energies. In this chapter, we present an entirely new approach for approximating a constant number of low-order eigenvalues. Our approach applies to a general class of eigenvalue problems [112]. We illustrate our results by considering the time-independent Schrödinger equation with a number of degrees of freedom d, under weaker assumptions than those of [182, 183]. More precisely, we consider the eigenvalue problem

$$\begin{aligned} \left(-\frac{1}{2}\Delta + V\right)\Psi(x) &= E\,\Psi(x) \quad x \in I_d = (0,1)^d, && (3.1) \\ \Psi(x) &= 0 \qquad\quad\;\; x \in \partial I_d, && (3.2) \end{aligned}$$

where Δ denotes the Laplacian and Ψ is a normalized eigenfunction. (Here all masses and the normalized Planck constant are set to one, d is proportional to the number of particles, and we assume the potential V is smooth and uniformly bounded as we will explain later.) In general, this problem may have many *degenerate* energy levels (eigenvalues with multiplicity greater than one).

We give a quantum algorithm, and derive cost and success probability bounds, for approximating a constant number of low-order eigenvalues, ignoring eigenvalue multiplicities. For accuracy $O(\varepsilon)$ and success probability at least $3/4$, the cost and the number of qubits of our algorithm are each polynomial in d and ε^{-1}, and hence our algorithm is efficient. As the best classical algorithms known for this problem have costs that grow exponentially in d, our quantum algorithm gives an exponential speedup. The results of this chapter can also be found in [112].

3.2 Problem Definition

We consider an eigenvalue problem for a self-adjoint operator L with a discrete spectrum, which generalizes the problem of computing the ground state energy (lowest eigenvalue). Let

$$E_{(0)} < E_{(1)} < ... < E_{(i)} < ... \tag{3.3}$$

be the eigenvalues of L ignoring multiplicities, which we call the *energy levels* of L. We refer to energy levels E satisfying $E \leq E_{(j)}$, $j = O(1)$, as *low order*. Suppose we want to estimate the lower part of the spectrum with accuracy $O(\varepsilon)$. Since any two distinct eigenvalues of L can be arbitrarily close to each other, any algorithm that approximates the lower part of the spectrum with accuracy ε cannot be expected to distinguish between all eigenvalues $E_{(k)} \neq E_{(l)}$ with $|E_{(k)} - E_{(l)}| = O(\varepsilon)$. We call such eigenvalues *ε-degenerate*. So, the problem is to obtain an algorithm whose output will be j numbers

$$\tilde{E}_0 < \tilde{E}_1 < ... < \tilde{E}_{j-1} \tag{3.4}$$

satisfying with high probability the following conditions:

C1 For every $i \neq k \in \{0, \dots, j-1\}$, there exist $E_{(s_i)} \neq E_{(s_k)}$ such that $|E_{(s_i)} - \tilde{E}_i| = O(\varepsilon)$ and $|E_{(s_k)} - \tilde{E}_k| = O(\varepsilon)$, i.e., different outputs are approximations of different eigenvalues with error $O(\varepsilon)$, respectively.

C2 If $|\tilde{E}_{i+1} - \tilde{E}_i| = \omega(\varepsilon)$, there is no eigenvalue E of L satisfying $\tilde{E}_i < E < \tilde{E}_{i+1}$ and $\min(|\tilde{E}_{i+1} - E|, |\tilde{E}_i - E|) = \omega(\varepsilon)$.[1] Thus the algorithm doesn't miss (or skip) any eigenvalues in the lower part of the spectrum unless they are $O(\varepsilon)$ apart, i.e., ε-degenerate.

Clearly, if ε is sufficiently small such that the eigenvalues of L are well-separated, then the algorithm produces approximations with error $O(\varepsilon)$ of the j smallest distinct eigenvalues.

We will employ a perturbation-based approach. Assume that L^0 is another self-adjoint operator such that $L = L^0 + V$, where $V = L - L^0$, with each operator acting on the same domain. We also assume that L^0 and L have discrete spectra and that the eigenspaces associated with each eigenvalue are finite dimensional; see e.g. [110, 129, 223]. In the general case, selecting the partition of L to L^0

[1]For functions $f, g \geq 0$ defined on $\mathbb{R}_+$, the notation $f(\varepsilon) = \omega(g(\varepsilon))$ means that for any $M > 0$, arbitrarily large, we have $f(\varepsilon) \geq Mg(\varepsilon)$ for sufficiently small ε.

and V is not trivial and may significantly affect the problem complexity; we do not deal with this problem here. Our discussion in this section applies equally well to Hermitian matrices.

Let

$$\sigma \leq E_0 \leq E_1 \leq ... \leq E_i \leq ... \tag{3.5}$$

be the eigenvalues of L indexed in nondecreasing order, where σ is a given lower bound. Ignoring possible eigenvalue multiplicities we have the strictly increasing subsequence of eigenvalues (3.3). Similarly we denote by

$$E_0^0 \leq E_1^0 \leq ... \leq E_i^0 \leq ... \tag{3.6}$$

the eigenvalues of L^0 indexed in nondecreasing order, and by

$$E_{(0)}^0 < E_{(1)}^0 < ... < E_{(i)}^0 < ..., \tag{3.7}$$

the eigenvalues of L^0 ignoring multiplicities. Assume we know the eigenvalues and eigenvectors of L^0. Often this is a reasonable assumption. For example, this is true for the eigenvalues and eigenvectors of the Laplacian $L^0 = -\Delta$ defined on the d-dimensional unit cube with Dirichlet or Neumann boundary conditions.

3.3 Background

We briefly review algorithms for eigenvalue problems. Algorithms approximating eigenvalues use a discretization of L to obtain a matrix eigenvalue problem. For example, when L is a differential operator, one can use a finite difference discretization [157], or a finite element discretization [19, 211]. In particular, for the time-independent Schrödinger equation specified by equations (3.1) and (3.2), a finite difference discretization has been used in [182, 183]. Since L is self-adjoint, the resulting matrix is symmetric. Eigenvalue problems involving symmetric matrices are conceptually easy and methods such as the bisection method can be used to solve them with cost proportional to the matrix size, modulo polylog factors [77]. The difficulty is that the discretization leads to a matrix of size that is exponential in d. Hence, the cost for approximating the matrix eigenvalue is prohibitive when d is large. In fact, a stronger result is known, namely the cost of any deterministic classical algorithm approximating the ground state energy must be at least exponential in the number of degrees of freedom d, i.e., the problem suffers from the curse of dimensionality [181].

More precisely, the discretization must be sufficiently fine so that the eigenvalues of interest are approximated by eigenvalues of the resulting matrix within the specified accuracy ε. This increases the matrix size, which directly impacts the cost. Indeed, general matrix eigenvalue problems have been extensively studied in numerical linear algebra, and there are classical algorithms for approximating one, or some, or even all of the eigenvalues and/or the corresponding eigenvectors of a matrix [74, 77, 106, 188]. Examples of such algorithms include the power method, inverse iteration, the QR algorithm, and the bisection method for symmetric matrices. In particular, the bisection method for symmetric matrices can compute the eigenvalues that lie within a given range. However, the cost of each algorithm scales at least linearly in the size of the matrix.

It is important to point out that different approaches may lead to different matrix eigenvalue problems that have varying degrees of difficulty. For instance, in quantum chemistry, the first and second quantization approaches for computing energies of the electronic Hamiltonian, as described in [147], lead to completely different matrices with different notions of degrees of freedom. Moreover, discretizations of certain problems in physics may lead to eigenvalue problems for stoquastic matrices, that some believe are computationally easier to solve [49].

For example, in the estimation of the ground state energy (smallest eigenvalue) of the time-independent Schrödinger equation (3.1), (3.2) with V uniformly bounded by 1, the finite difference discretization on a grid will yield a matrix of size $m^d \times m^d$, $m = 2^{\lceil -\log_2 \varepsilon \rceil} - 1$ [183]. This means that the cost of the matrix eigenvalue algorithms mentioned above is bounded from below by a quantity proportional to ε^{-d}, i.e., the cost grows exponentially in d.

To approximate the ground state energy of the problem specified by equations (3.1) and (3.2) in the worst case with (relative) error ε, assuming V and its first-order partial derivatives are uniformly bounded by 1, and the function evaluations of V are supplied by an oracle, a much stronger result holds. The complexity (i.e., the minimum cost of any classical deterministic algorithm, and not just the eigenvalue algorithms mentioned above) is bounded from below by a quantity proportional to ε^{-d} as $d\varepsilon \to 0$ [181, 183]. So unless d is moderate, the problem is very hard and suffers from the curse of dimensionality. The same complexity lower bound applies to the approximation of low-order eigenvalues under the same, or more general, conditions on V. Finally, we point out that the complexity of this problem in the classical randomized case is an open question.

In certain cases quantum algorithms may break the curse of dimensionality by computing ε-

accurate eigenvalue estimates with cost polynomial in ε^{-1} and d. This was shown in [182, 183] where we saw that for smooth nonnegative potentials that are uniformly bounded by a relatively small constant, or are convex, there exists a quantum algorithm approximating the ground state energy with relative error $O(\varepsilon)$ and cost polynomial in d and ε^{-1}. In [183], the Laplacian was discretized using a $2d+1$ stencil on a grid, and V was discretized by evaluating it at the grid points. Our results of this chapter continue this line of research, yielding stronger results under weaker conditions than those of [182, 183].

Quantum Phase Estimation

There is a well-studied quantum algorithm, quantum phase estimation (QPE) [3, 15, 173], which can be used to approximate eigenvalues of a Hamiltonian H. More precisely, the algorithm approximates the phase corresponding to an eigenvalue of a unitary matrix, which in our case is e^{-iH}. QPE is efficient if two conditions are met. The first condition is that simulating a system evolving with Hamiltonian H can be done efficiently, i.e., we can approximate e^{-iHt}, $t \in \mathbb{R}$, accurately with low cost. The second condition requires that we are given a relatively good approximation of an eigenvector corresponding to the eigenvalue of interest. In addition, one should be able to implement this approximation as a quantum state efficiently. The approximate eigenvector is used to form the initial state of QPE. We also remark that QPE uses the (quantum) Fourier transform as a module. The Fourier transform can be implemented efficiently on a quantum computer [173].

We discuss the two required conditions for QPE further. Simulating the evolution of a system under a Hamiltonian H appears to be a difficult problem for classical computers when the size of H is large. As proposed by Feynman [95], quantum computers are able to carry out such simulation more efficiently in certain cases. For example, Lloyd [162] showed that local Hamiltonians can be simulated efficiently on a quantum computer. About the same time, Zalka [257, 258] showed that many-particle systems can be also be simulated efficiently on a quantum computer. Later, Aharonov and Ta-Shma [5] generalized Lloyd's results to sparse Hamiltonians. Berry et al. [29] extended the cost estimates of [5]. The results of [29] were in turn improved by Papageorgiou and Zhang in [186]. Although there has been more work on quantum Hamiltonian simulation since then, the approach of [29, 186] suffices for our discussion. These papers assume that H is given by a black-box (or oracle), and that H can be decomposed efficiently by a quantum algorithm, using oracle calls, into a finite sum of Hamiltonians that individually can be simulated efficiently. In this chapter, where $L = L^0 + V$, we assume that the Hamiltonians resulting from the discretizations of L^0 and V can be simulated efficiently on a quantum computer. Their sum, i.e., the Hamiltonian obtained from the discretization of L, can be simulated efficiently using splitting formulas such as the Trotter formula, the Strang splitting formula, or Suzuki's high-order formulas; see Appendix D.1 for a review. (We study Hamiltonian simulation in detail in Chapter 4.)

We now consider the second requirement of QPE, namely the availability of a good approximate eigenvector. QPE will produce an estimate of the eigenvalue λ (or more precisely, an estimate of the phase $\phi \in [0, 1)$ corresponding to λ through $\lambda = e^{2\pi i\phi}$) with success probability proportional to the quality of the approximate eigenvector [3, 173]. If the eigenvector providing the initial state of QPE is known exactly, the parameters of QPE can be set so its success probability is arbitrarily close to 1 [173]. If, on the other hand, we use an approximate eigenvector, the success probability is reduced proportionally to the square of the magnitude of the projection of the approximate eigenvector onto the actual eigenvector (i.e., the square of the *overlap* between the two vectors) [3]. As long as this overlap is not exponentially small, QPE is efficient.

We remark that obtaining a good approximate eigenvector required for QPE is a particularly difficult task, in general, when the matrix size is huge. Things are complicated further if one needs a number of different approximate eigenvectors, in order to use QPE to approximate the $j > 1$ lowest-order eigenvalues. We overcome this difficulty for $L = L^0 + V$ using the known eigenvalues and eigenvectors of L^0, and properties of V, as we discuss below.

3.4 Algorithm

We give our algorithm for eigenvalue estimation of a general self-adjoint operator $L = L^0 + V$ under the assumptions of Section 3.2. Recall that L^0 and L are self-adjoint operators on a Hilbert Space $\mathcal{H}$ with discrete spectra; e.g., see [110]. The eigenvalue problem for the time-independent Schrödinger equation (3.1) is a special case we consider in detail in Section 3.5.3.

Our goal is to use QPE to estimate j low-order energy levels of L. For this, we need relatively good approximations of the corresponding eigenvectors. Since $L = L^0 + V$ (i.e., L and L^0 differ by the perturbation V), and since we know the eigenvalues and the eigenvectors of L^0, we can use them to obtain the necessary approximate eigenvectors. We indicate how this can be done. For simplicity and notational convenience, we do not distinguish between operators and their matrix discretizations in the rest of this section, since it is not important for the moment. Let E denote one of the low-order eigenvalues of L (see equation (3.3)) that we wish to estimate. Intuitively, we expect a "small" and suitably well-behaved perturbation to have a proportionately "small" effect on the eigenvectors and eigenvalues of L^0. Let u be an arbitrary unit vector belonging to the eigenspace associated with E. Then there exists an eigenvector u_k^0 of L^0, similarly corresponding to a low-order eigenvalue, that has an *overlap* (magnitude of projection) with u that is nontrivial, i.e., $|\left\langle u_k^0|u\right\rangle|$ is not extremely small, as we will see later.

One of the keys to our approach is to form a collection $\mathcal{S}$ of the eigenvectors of L^0 that correspond to eigenvalues of L^0 that satisfy a certain property, which we specify in the next subsection. The goal is to have at least one element in $\mathcal{S}$ that has a reasonable overlap with a vector in the eigenspace corresponding to $E_{(i)}$, for each $i = 0, 1, ..., j-1$. We call $\mathcal{S}$ the *set of trial eigenvectors*. We will use each one of the elements of $\mathcal{S}$ repetitively as initial state in QPE, running QPE multiple times, to obtain a sequence of approximations that will lead us to estimates of each $E_{(i)}$.

Let us briefly discuss the idea for constructing $\mathcal{S}$. At one extreme, one could take $\mathcal{S}$ to be all of the eigenvectors of L^0, because not all of them have a negligible overlap with the eigenvectors of L corresponding to the eigenvalues of interest. However, then the size of $\mathcal{S}$ can be huge. To limit $|\mathcal{S}|$, we select eigenvectors of L^0 that correspond to eigenvalues that do not exceed a certain bound. Roughly speaking, we will be excluding eigenvalues of L^0 that correspond to energies grossly exceeding the energies of L that we wish to estimate. This idea is made precise in equation (3.11) in next section.

The cardinality of $\mathcal{S}$ depends on the eigenvalue distribution of L^0. If the cardinality of $\mathcal{S}$ is not prohibitively large, and if we can discretize its elements and efficiently prepare the corresponding quantum states, then we can run QPE repeatedly for the all elements of $\mathcal{S}$ to produce an estimate of $E_{(i)}$ among its different outputs with a sufficiently high probability, $i = 0, 1, ..., j-1$. This probability can be boosted to become arbitrarily close to 1 using further repetitions of the procedure. We remark that the cardinality of $\mathcal{S}$ depends on the distribution of eigenvalues of L^0 and the properties of V. Observe that detecting the desired estimates $\tilde{E}_0, ..., \tilde{E}_{j-1}$ from the outcomes obtained from the different runs of QPE is not a trivial task, and we will show how this is accomplished.

3.4.1 Preliminary Analysis

Suppose we wish to compute a specific eigenvalue E of L. Let u be a unit vector in the (possibly degenerate) subspace associated with E, i.e., satisfying $Lu = Eu$. Then we have

$$\|Lu - L^0 u\|^2 = \|(L^0 + V)u - L^0 u\|^2 = \|Vu\|^2 \leq \|V\|^2.$$

Expanding in the basis of unperturbed eigenvectors, we have $|u\rangle = \sum_i \beta_i |u_i^0\rangle$ where $\beta_i = \langle u_i^0 | u\rangle$. Then

$$\|Lu - L^0 u\|^2 = \|E \sum_i \beta_i |u_i^0\rangle - \sum_i \beta_i E_i^0 |u_i^0\rangle\|^2 = \|\sum_i \beta_i (E - E_i^0)|u_i^0\rangle\|^2$$

Combining these expressions and using the eigenvector orthonormality gives

$$\|V\|^2 \geq \|Lu - L^0 u\|^2 = \sum_i |\beta_i|^2 (E_i^0 - E)^2 \tag{3.8}$$

Assume we know an upper bound $B \geq E$ and a constant $c > 1$ such that their exist eigenvalues of L^0 larger than $c\|V\| + B$, and define the set $\mathcal{I} := \{i : E_i^0 > c\|V\| + B\}$. Observe that this is true for instances of the time-independent Schrödinger equation [223]. From (3.8), we obtain

$$\|V\|^2 \geq \sum_{i \in \mathcal{I}} |\beta_i|^2 (E_i^0 - E)^2 \geq \sum_{i \in \mathcal{I}} |\beta_i|^2 (c\|V\|)^2,$$

which we rearrange to give

$$\sum_{i \in \mathcal{I}} |\beta_i|^2 \leq \frac{1}{c^2}, \tag{3.9}$$

or, equivalently,

$$\sum_{i \notin \mathcal{I}} |\beta_i|^2 \geq 1 - \frac{1}{c^2} =: q. \tag{3.10}$$

Thus there must exist an index $k \notin \mathcal{I}$ such that $|\beta_k|^2 \geq \frac{q}{|S|}$. If $|S|$ is not extremely large, then one of the first $|S|$ eigenvectors of L^0 must have a reasonable overlap with u. (Here and elsewhere, by reasonable overlap we mean that the magnitude of the projection is not exponentially small in d.)

3.4.2 Algorithm Description

Let V be such that $\|V\| := \sup\{\|Vu\| : \|u\| = 1\} < \infty$ uniformly in d. Assume we are given (or we have derived) $c > 1$ and B a sufficiently large upper bound on the lower part of the spectrum of L which is of interest.[2] Consider the set of indices

$$\mathcal{I} = \{i : E_i^0 - B > c\|V\|\} \neq \emptyset. \tag{3.11}$$

We define $\mathcal{S}$ to be the set of eigenvectors of L^0 that correspond to eigenvalues E_i^0 with $i \notin \mathcal{I}$; in the case of degeneracy, it suffices to select any basis of the degenerate subspace. By constructing $\mathcal{S}$ in this way, we are guaranteed that at least one of its elements will overlap sufficiently with an element of the degenerate eigenspace corresponding to each $E_{(i)}$, for $i = 0, 1, ..., j-1$. We will show that the magnitude of this overlap is bounded from below by a positive constant.

We now give our quantum algorithms for approximating $j = O(1)$ low-order eigenvalues of the operator $L = L^0 + V$. Algorithm 1 deals with the special case of approximating the ground state energy $E_{(0)}$. This algorithm illustrates our idea of using a set of trial eigenvectors to approximate an eigenvalue of L. Algorithm 2 computes the sequence of approximations $\tilde{E}_1, \tilde{E}_2, ..., \tilde{E}_{j-1}$, where each $\tilde{E}_i$ is computed using the values $\tilde{E}_0$ through $\tilde{E}_{i-1}$. Thus the overall procedure consists of iterating Algorithm 2 until we obtain the j desired estimates of equation (3.4).

Let us pretend for the moment that L^0 and L are $N \times N$ matrices. In the next section we will show how to discretize them and obtain symmetric matrices such that each of the low-order eigenvalues of these matrices approximates the corresponding eigenvalue of the respective operator with error proportional to ε.

Our algorithms are based on QPE and require two quantum registers. The first (*top*) register contains sufficiently many qubits t to guarantee the required accuracy $O(\varepsilon)$ in the results with a reasonable success probability for QPE. The second (*bottom*) register contains the necessary number of qubits to hold an approximate eigenstate.

Algorithm 1. Ground State Energy:

1. Define $\mathcal{S}$, the *set of trial eigenvectors*, to be all eigenvectors of L^0 that correspond to eigenvalues E_i^0 $i \notin \mathcal{I}$ as defined in equation (3.11). We denote these eigenvectors by u_k^0 for $k = 0, 1, .., |\mathcal{S}| - 1$.

2. Set k = 0.

[2]We give an explicit construction for B in equation (3.16).

3. Prepare the initial quantum state $|0\rangle^{\otimes t}|u_k^0\rangle$. The value of t is chosen so that QPE, with relatively high probability, produces outcomes leading to energy estimates with error $O(\varepsilon)$.

4. Perform QPE with initial state $|0\rangle^{\otimes t}|u_k^0\rangle$ using the unitary matrix $U = e^{iA/R}$. $A = L$ if L is nonnegative definite, and otherwise $A = L - \sigma I$, where σ is a lower bound to the minimum eigenvalue of L as we have assumed in the previous section. The parameter σ is assumed to be known; see equation (3.5). The parameter R is an upper bound to the spectral norm of A, which can be obtained using the eigenvalues of L^0 and the properties of V. The purpose of R is to ensure that the resulting phases will lie in the interval $[0, 1)$.

5. Measure the first t qubits, which give the result of QPE, and store the resulting value classically. We assume that the measurement outcomes are truncated to b bits and we obtain nonnegative integers in the range $\{0, ..., 2^b - 1\}$, where $b < t$. The role of the extra qubits $t_0 = b - t$ is to increase the success probability of QPE.

6. $k \leftarrow k + 1$.

7. Repeat steps 3-6 while $k < |\mathcal{S}|$.

8. Repeat steps 2-7 r many times, where r is a number precomputed to ensure with high probability that the stored results after r runs contain an estimate of E_0.

9. Take the minimum value of the stored measurement outcomes, mark it as selected, and convert it to an eigenvalue estimate $\tilde{E}_0$ of E_0 using the values of σ and R in the definition of A.

10. Output $\tilde{E}_0$.

Note, the purpose of step 7 is to run QPE $|\mathcal{S}|$ many times, once with each $\left|u_k^0\right\rangle \in \mathcal{S}$ as input, because we do not know which of the elements of $\mathcal{S}$ has the largest overlap with the unknown ground state eigenvector, and the success probability of each run depends on this overlap. Since the largest overlap between the elements of $\mathcal{S}$ and the unknown eigenvector may not be sufficiently large so that the resulting success probability of the algorithm is bounded from below be a constant, say $\frac{3}{4}$, the purpose of step 8 is to repeat the entire procedure r many times to boost the success probability of computing $\tilde{E}_0$ correctly.

The following iterative algorithm extends Algorithm 1 to compute the sequence of approximations $\tilde{E}_1, .., \tilde{E}_{j-1}$ satisfying the conditions of equation (3.4), respectively. Every term of the computed sequence depends on all of the previously computed terms.

Algorithm 2. Excited State Energies:

1. Consider A as defined in Algorithm 1. Run Algorithm 1 and let $\tilde{E}_0$ be its output.

2. Set $i = 1$ and prepare to compute an estimate of $\tilde{E}_1$.

3. Repeat steps 1-8 of Algorithm 1, storing the outcome of every measurement. We assume that the measurement outcomes are truncated to b bits and we obtain nonnegative integers in the range $\{0, ..., 2^b - 1\}$, where $b < t$. The role of the extra qubits $t_0 = b - t$ is to increase the success probability of QPE.

4. Take the minimum of the measurement outcomes that exceeds by 2 the last selected outcome and mark it selected. This way, with high probability, for each eigenvalue the error will be $O(\varepsilon)$, and the algorithm will not produce two different estimates for the same eigenvalue. Note that by taking the minimum outcome relative to the previously selected outcome implies that the algorithm does not fail to produce estimates for consecutive eigenvalues, unless the eigenvalues differ by $O(\varepsilon)$. See Figure 3.1.

5. Use the values of σ and R in the definition of A to rescale and shift the newly selected outcome to obtain the estimate $\tilde{E}_i$.

6. Set $i \leftarrow i + 1$ and prepare to compute the estimate $\tilde{E}_i$.

7. Repeat steps 3 through 6 if $i < j$.

8. Output $\tilde{E}_1, ..., \tilde{E}_{j-1}$.

It is clear that our procedure as outlined by Algorithms 1 and 2 will produce the j desired estimates (3.4) of equation (3.3). However, its cost varies depending on V and the distribution of eigenvalues of L^0, which as we already mentioned determine the cardinality of $\mathcal{S}$. In the next sections we derive tight estimates for the cost and the success probability of our algorithm for particular choices of L^0 and L.

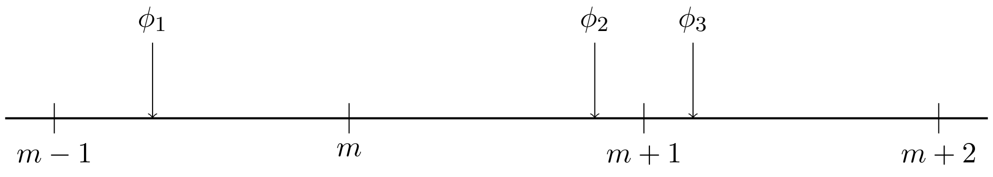


Fig. 3.1: Example of the selection of the measurement outcomes. Consider three phases ϕ_1, ϕ_2 and ϕ_3 as shown. Assume that the distance between possible outcomes corresponds to error $O(\varepsilon)$. If $m-1$ is selected to estimate ϕ_1, the next possible outcome the algorithm selects is $m+1$, which provides an estimate for both ϕ_2 and ϕ_3 in this example. Alternatively, if m is selected to provide an estimate of ϕ_1, the next possible outcome is $m+2$, which provides an estimate of ϕ_3, and the algorithm does not care to produce a separate estimate for ϕ_2. Note that ϕ_2 and ϕ_3 are ε-degenerate and either can be ignored.

We remark that in cases where L and L^0 are given explicitly, using their properties one may be able to obtain a set of trial eigenvectors $\mathcal{S}$ with significantly smaller cardinality, substantially improving the cost of the algorithm. For example, knowledge of the symmetry groups of L, L^0, and V could be used to immediately rule out candidate eigenvectors. It is important to observe that different partitionings of the Hamiltonian into L^0 and $L-L^0$ may lead to very different sets of trial eigenvectors. Given a Hamiltonian L, an important task is to select L^0 that will result in a relatively small set of trial eigenvectors which can be computed efficiently.

3.5 Application: Time-Independent Schrödinger Equation

Consider the time-independent Schrödinger equation on the d-dimensional unit cube with Dirichlet boundary conditions,

$$\begin{aligned} Lu(x) := (-\tfrac{1}{2}\Delta + V)u(x) &= Eu(x) \quad \text{for all } x \in I_d := (0,1)^d, \qquad (3.12) \\ u(x) &= 0 \quad \text{for all } x \in \partial I_d, \end{aligned}$$

where V is uniformly bounded by a constant M and has continuous first-order partial derivatives in each direction uniformly bounded by a constant C, i.e., $\|\frac{\partial}{\partial x_i}V\| \le C$. Thus, without loss of generality we assume that $V \geq 0$. We set

$$L^0 = -\frac{1}{2}\Delta = -\frac{1}{2}\sum_{i=1}^{d} \frac{\partial^2}{\partial x_i^2}.$$

Assume that the eigenvalues of L and L^0 are indexed in nondecreasing order. We want to approximate the first j excited state energies, $E_{(0)}, \ldots, E_{(j-1)}$, (i.e., the j smallest eigenvalues ignoring multiplicities) with error proportional to ε, modulo ε-degenerate eigenvalues as explained previously. Thus we are interested in low-order excited state energies because we have assumed that j is a constant.

3.5.1 Set of Trial Eigenvectors

As we already indicated, the cost of Algorithms 1 and 2 depends on the cardinality of a set of trial eigenvectors $\mathcal{S}$. We will now show that $|\mathcal{S}|$ is bounded by a polynomial in d in the case of the Schrödinger equation we are considering here.

The eigenvalues and eigenvectors of $L^0 = -\frac{1}{2}\Delta$ are known to be

$$E^0_{\vec{k}} = \frac{1}{2}(k_1^2 + k_2^2 + \ldots + k_d^2)\pi^2 \qquad \vec{k} = (k_1, ..., k_d) \in \mathbb{N}^d \tag{3.13}$$

$$u^0_{\vec{k}}(\vec{x}) = 2^{d/2}\prod_{i=1}^{d} \sin(k_i \pi x_i) \qquad \vec{x} = (x_1, ..., x_d) \in [0,1]^d \qquad \vec{k} = (k_1, ..., k_d) \in \mathbb{N}^d.$$

We may reindex them by considering the eigenvalues in nondecreasing order to obtain $E^0_0 \leq E^0_1 \leq ... \leq E^0_i \leq ...$ as in (3.7). Thus $E^0_0 = \frac{1}{2}d\pi^2 < \frac{1}{2}(d+3)\pi^2 = E^0_1$ and E^0_1 is a degenerate eigenvalue with dimension of its associated degenerate subspace equal to d. Similar considerations apply to the rest of the eigenvalues. We remark that the distribution of the eigenvalues of L^0 is known [223].

We will use (3.11) with $c = 2$ to derive a set of trial eigenvectors and bound its cardinality. In fact, we derive a set of trial eigenvectors that is slightly larger than the set obtained by strictly considering the indices in the complement of $\mathcal{I}$ in (3.11). Yet its size is polynomial in d as we will see, and for the sake of brevity, we also denote this set by $\mathcal{S}$. In particular, we construct the quantity B of equation (3.11) and show a $K = K(j, V)$ such that for $k \geq K$ we have $E^0_k > 2M + B$. So we obtain an upper bound for the jth largest eigenvalue of L. Clearly the cardinality of $\mathcal{S}$ grows with B because we include eigenvectors of L^0 that correspond to increasingly large eigenvalues. The purpose of the construction below is to obtain a crude but helpful in our analysis estimate of the distribution of the eigenvalues of L using the eigenvalues of L^0; in particular to cover possible degeneracy of the eigenvalues of L.

We select $j+1$ values $E^0_{(s_n)}$ from the strictly increasing sequence of eigenvalues (see (3.7)) such that

$$E^0_{(s_0)} := E^0_{(0)} < E^0_{(0)} + M < E^0_{(s_1)} < E^0_{(s_1)} + M < \ldots < E^0_{(s_{j-1})} + M < E^0_{(s_j)} \tag{3.14}$$

where $E^0_{(s_n)} - E^0_{(s_{n-1})} > M$, $n = 1, \ldots, j$. Indeed it is possible to select a subsequence that satisfies these conditions. We know that $E^0_{(s_{n-1})} = \frac{1}{2}(k_1^2 + k_2^2 + \ldots k_d^2)\pi^2$ for a certain $\vec{k}$. The inequality

$$\frac{1}{2}({k'}_1^2 + {k'}_2^2 + \ldots {k'}_m^2)\pi^2 + \frac{1}{2}(k_{m+1}^2 + k_{m+2}^2 + \ldots k_d^2)\pi^2 \geq E^0_{(s_{n-1})} + M$$

is satisfied by selecting m to be a suitable constant and then by selecting $k'_i \geq k_i + \gamma_i$, where γ_i is a suitable positive integer constant, $i = 1, \ldots, m$. For example, after fixing m, we can repeatedly increment each of the k'_i, $i \in \{1, \ldots, m\}$, successively until the desired inequality holds. Iteratively, we define $E^0_{(s_n)} = ({k'}_1^2 + {k'}_2^2 + \ldots {k'}_m^2)\pi^2/2 + (k_{m+1}^2 + k_{m+2}^2 + \ldots k_d^2)\pi^2/2$ for $n = 1, 2, .., j$.

By our construction, the interval $[E^0_{(s_0)}, E^0_{(s_j)}]$ contains at least j distinct eigenvalues of L, since $E^0_{(s_j)} - E^0_{(s_0)} > jM$, and for every i, $E^0_i \leq E_i \leq E^0_i + M$. Moreover, $E^0_{(s_j)} = E^0_0 + c'$, where c' is a constant. Thus, we take $c = 2$ in (3.11) and define the constant B as

$$B := M + E^0_{(s_j)} = M + E^0_0 + c'. \tag{3.15}$$

From (3.13) there exists a $K \in \mathbb{N}$ such that

$$k \geq K \quad \Rightarrow \quad E^0_k > 2M + B = 3M + E^0_{(s_j)}. \tag{3.16}$$

Hence, we construct the *set of trial eigenvectors* $\mathcal{S}$ to be the set of all eigenvectors of L^0 that correspond to eigenvalues less than or equal to $3M + E^0_{(s_j)}$. We bound $|\mathcal{S}|$ next.

The cardinality of $\mathcal{S}$ is the number of tuples $\vec{k} \in \mathbb{N}^d$ such that $(k_1^2 + \cdots + k_d^2)\pi^2/2 \leq 3M + d\pi^2/2 + c'$. Let m be the number of components $k_{i_1}, \ldots, k_{i_m}$ of such a $\vec{k}$ that are greater than 1. Then we have

$$(d-m)\pi^2/2 + (k_{i_1}^2 + \ldots k_{i_m}^2)\pi^2/2 \leq 3M + d\pi^2/2 + c'.$$

Since $k_i \geq 2$ we have

$$3m\pi^2 \leq -m\pi^2 + (k_{i_1}^2 + \ldots k_{i_m}^2)\pi^2 \leq 2(3M + c').$$

Table 3.1: Distribution of eigenvalues of $L^0 = -\frac{1}{2}\Delta$ with respect to the number m of indices $k_i \geq 2$.

m	Combinations	Eigenvalue
0	$\binom{d}{0}$	$d\pi^2/2$
1	$\binom{d}{1}$	$(d-1)\pi^2/2 + k_{i_1}^2\pi^2/2$
2	$\binom{d}{2}$	$(d-2)\pi^2/2 + (k_{i_1}^2 + k_{i_2}^2)\pi^2/2$
...		
$l \leq d$	$\binom{d}{l}$	$(d-l)\pi^2/2 + (k_{i_1}^2 + .. + k_{i_l}^2)\pi^2/2$

Hence, m is $O(1)$. Therefore, in order to construct $\mathcal{S}$ one needs to consider tuples $\vec{k}' \in \mathbb{N}^d$ where at most a constant number of components are greater than 1. The number of such tuples depends on the number of possible combinations by which one can select a constant number of components of $\vec{k}'$ to be greater than or equal to 2. Therefore, this number is polynomial in d.[3]

Table 3.1 shows the eigenvalues of the Laplacian by considering tuples where a constant number m of components exceed 1, assuming that these components are each bounded by a constant N. Observe that in all cases, since $m = O(1)$, the multiplicity of the eigenvalues grows as a polynomial function of d.

Therefore, the cardinality of $\mathcal{S}$ is polynomial in d. As shown in Section 3.4.1, for every eigenvector of L that corresponds to an eigenvalue less than or equal to B, there exists an eigenvector of L^0 in $\mathcal{S}$ such that the two eigenvectors have a nontrivial overlap and it follows from (3.10) with $c = 2$ that the magnitude squared of this projection of the one onto the other will be at least $\frac{3}{4}\frac{1}{|\mathcal{S}|} = \frac{1}{\text{poly}(d)}$, i.e., at worst polynomially small with respect to d, as desired.

3.5.2 Finite Difference Discretization

We obtain a matrix eigenvalue problem by discretizing (3.12) on a grid with mesh size $h = \frac{1}{N+1}$, $N \in \mathbb{N}$, using finite differences [157, 183]. This yields a matrix $M_h := -\frac{1}{2}\Delta_h + V_h$ with size $N^d \times N^d$. The matrix $-\frac{1}{2}\Delta_h$ is obtained using a $2d+1$ stencil for the Laplacian [157, p.60]. It is

[3]This follows immediately for $m = O(1)$ from the bound $\binom{d}{m} \leq \frac{d^m}{m!} = \text{poly}(d)$.

known that the low-order eigenvalues of M_h approximate the corresponding eigenvalues of L. The eigenvalues and eigenvectors of $-\frac{1}{2}\Delta_h$ are known and are given by

$$E^0_{h,\vec{k}} = \frac{2}{h^2}\sum_{i=1}^{d}\sin^2(\pi h k_i/2) \qquad \vec{k} = (k_1, ..., k_d) \quad 1 \le k_i \le N \tag{3.17}$$

$$u^0_{h,\vec{k}} = \bigotimes_{i=1}^{d} v_{k_i}, \tag{3.18}$$

where the vectors $v_{k_i} \in \mathbb{R}^d$ have coordinates

$$v_{k_i,\ell} = \sqrt{2h}\sin(k_i \ell \pi h) \qquad \ell = 1, 2, ..., N \qquad i = 1, 2, ..., d. \tag{3.19}$$

Similarly to (3.7), we index the eigenvalues of $-\frac{1}{2}\Delta_h$ in increasing order ignoring multiplicities to obtain

$$E^0_{h,(0)} < E^0_{h,(1)} < ... < E^0_{h,(i)} < ... \tag{3.20}$$

Then from [244] we have

$$|E^0_{h,(k)} - E^0_{(k)}| \le Cdh^2 \qquad \text{for } k = O(1) \tag{3.21}$$

where $C > 0$ is a constant.

V_h is an $N^d \times N^d$ diagonal matrix which contains evaluations of V at the grid points truncated to $\lceil \log_2 h^{-1} \rceil$ bits of accuracy. Thus M_h is symmetric, positive definite, and sparse. This matrix has been extensively studied in the literature [77, 98, 157]. For V that has bounded first-order partial derivatives and $k = O(1)$, using the results of [244, 245] we have that there exists a matrix eigenvalue $E_{h,k'}$ such that

$$|E_{(k)} - E_{h,k'}| = O(dh) \tag{3.22}$$

as $dh \to 0$, where $E_{(k)}$ is defined in (3.3). We will use the algorithms of Section 3.4 to approximate the low-order eigenvalues of M_h, which as we have seen approximate the low-order eigenvalues of L. For this, we need to construct the set of trial eigenvectors $\mathcal{S}$, and estimate its cardinality. Recall that for the continuous operator, the set of candidate eigenvectors is derived using equation (3.16), and in particular by selecting the eigenvectors of L^0 that correspond to eigenvalues less or equal to $2M + B = 3M + E^0_{(s_j)}$. So for the discretized case we select the eigenvectors of $-\frac{1}{2}\Delta_h$ that correspond to eigenvalues less than or equal to $3M + E^0_{(s_j)} + O(dh^2)$ due to equation (3.21). Since

$dh \to 0$, without loss of generality we slightly modify equation (3.16), to select the eigenvectors of M_h that correspond to eigenvalues less than or equal to

$$2M + B = 3M + E^0_{(s_j)} + 1 \tag{3.23}$$

for sufficiently small h, where this equation effectively redefines B by increasing its value by 1. Thus the cardinality of $\mathcal{S}$ in the case of the matrix M_h follows from the continuous case and remains polynomial in d.

Specifically, we define

$$\mathcal{S} := \{u^0_{h,k} : E^0_{h,k} \le 3M + E^0_{(s_j)} + 1\} \tag{3.24}$$

3.5.3 Algorithm for Excited State Energies

We now give the details of Algorithms 1 & 2 applied to the time-independent Schrödinger equation (3.12). Given ε, the algorithms produce the j eigenvalue estimates $\tilde{E}_0 < ... < \tilde{E}_{j-1}$ of equation (3.4). Algorithm 1 computes $\tilde{E}_0$. For this, QPE [173] is applied repeatedly with its initial state taken to be every single element of the set of trial eigenvectors $\mathcal{S}$. We use repetitions of the procedure to boost the success probability. We remark that our Algorithm 1 computes the ground state energy in a way similar to [182, 183], but under weakened assumptions. Algorithm 2 iterates $j - 1$ times the procedure of Algorithm 1, at each iteration producing the next estimate $\tilde{E}_i$ by taking into account all the previously produced estimates as we will explain below.

Both algorithms use QPE as the main module. The purpose is to compute approximations of the eigenvalues of the matrix M_h of the previous section. Setting $N = 2^{\lceil 2\log_2(d/\varepsilon)\rceil}$, we discretize (3.12) with mesh size $h = \frac{1}{N+1} < \frac{\varepsilon^2}{d^2}$ to obtain the $N^d \times N^d$ matrix M_h, where we have $N^d = O((\frac{d}{\varepsilon})^{2d})$. From (3.22), we obtain that the low-order matrix eigenvalues approximate the low-order eigenvalues of the continuous operator with error proportional to $dh = O(\frac{\varepsilon^2}{d})$. The reason we have taken very small h is because we want to ensure that ε-degenerate eigenvalues of the continuous operator will be approximated by tightly clustered eigenvalues of M_h. As M is a constant, without loss of generality we may assume that $\varepsilon^{-1} \gg M$. Since the largest eigenvalue of $-\frac{1}{2}\Delta_h$ is bounded from above by $2dh^{-2}$, and V is uniformly bounded by M, we obtain that $\|M_h\|$ is bounded from above by $2dh^{-2} + M \ll 3dh^{-2}$, in the sense that $\frac{M}{dh^{-2}} = o(1)$.

Let $R = 3dh^{-2}$ and consider the matrix $W = e^{iM_h/R}$. Its eigenvalues are $e^{iE_h/R} = e^{\frac{2\pi i E_h}{2\pi R}} = e^{2\pi i\phi}$, where E_h is an eigenvalue of M_h and $\phi := \frac{E_h}{2\pi R}$ denotes the corresponding phase.

QPE is used to compute an approximation $\hat{\phi}$ of ϕ with $b = 5\lceil \log_2 \frac{d}{\varepsilon} \rceil + 7$ bits of accuracy, and from this we get $\tilde{E} = 2\pi R\hat{\phi}$ so that

$$|E - \tilde{E}| \leq |E - E_h| + |E_h - \tilde{E}| = O(\varepsilon) \tag{3.25}$$

where E denotes the eigenvalue of L that E_h approximates according to (3.12). QPE uses two registers, the top and the bottom. The size of the top register is related to the accuracy of QPE and its success probability. Recall that QPE succeeds when it produces an estimate with accuracy 2^{-b}. The bottom register is used to hold an (approximate) eigenvector of M_h corresponding to the phase of interest, and therefore has size $d \log_2 N = d \cdot O(\log \frac{d}{\varepsilon})$. The number of qubits in the top register is $t = b + t_0$, so that QPE has accuracy 2^{-b} with probability at least $1 - \frac{1}{2(2^{t_0}-2)}$, assuming that an exact eigenvector is provided as initial state in the bottom register [173, Sec. 5.2]. QPE uses powers of W, namely $W^{2^0}, W^{2^1}, ..., W^{2^{t}-1}$. We will approximate these powers using a splitting formula with error, as we will see below. This reduces the success probability of QPE to at least $p := 1 - \frac{1}{2^{t_0}-2}$. We will set t_0 to be logarithmic in d, and will give all the details later on when dealing with the cost of our algorithm.

Consider an eigenvalue $E_h \leq B + 2M$ (see equations (3.11) and (3.23)) of the matrix M_h and let u_h denote an eigenvector corresponding to E_h. Then QPE with initial state some $u^0_{h,i} \in \mathcal{S}$ succeeds with probability at least $p_{u_h}(i) := |u_h^T u^0_{h,i}|^2 \cdot p = |u_h^T u^0_{h,i}|^2 \cdot (1 - \frac{1}{2^{t_0}-2})$ [3].

Recall that $\mathcal{S}$ contains eigenvectors of L^0 that correspond to eigenvalues $E^0_h \leq B + 2M$ as defined in (3.24), and that the cardinality of $\mathcal{S}$ is polynomial in d. Applying the same approach of Section 3.4.1 for the eigenvectors of M_h, we conclude that for every eigenvector u_h of the matrix M_h that corresponds to an eigenvalue less than B, there exists a vector $u^0_{h,k} \in \mathcal{S}$ such that $|u_h^T u^0_{h,k}|^2 \geq \frac{3}{4|\mathcal{S}|}$, where we have used equation (3.10) with $c = 2$ (since the value of B we are using here leads to $c = 2$ in this case too). Thus, after we run QPE with each element of $\mathcal{S}$ as initial state, the probability that at least one of the outcomes (in principle we do not know which one) will give a good estimate of E_h is at least $p_{u_h}(k) \geq \frac{3}{4|\mathcal{S}|}p$.

We repeat the whole procedure r times to boost the success probability of obtaining an estimate of E_h with accuracy ε. Indeed, the probability that QPE fails with all initial states taken from $\mathcal{S}$ and in all its $r|\mathcal{S}|$ repetitions is

$$\left(\prod_{i=1}^{|\mathcal{S}|}(1-p_{u_h}(i))\right)^r \leq (1-p_{u_h}(k))^r \leq e^{-rp_{u_h}(k)} \leq e^{-r\frac{3}{4|\mathcal{S}|}p} \tag{3.26}$$

Thus, the probability that at least one of the $r|\mathcal{S}|$ outcomes will lead to an approximation of E_h with accuracy $O(\varepsilon)$ is at least

$$1-e^{-r\frac{3}{4|\mathcal{S}|}p} = 1-e^{-r\frac{3}{4|\mathcal{S}|}\cdot(1-\frac{1}{2^{t_0}-2})} \tag{3.27}$$

We can boost this probability to be arbitrarily close to 1 by taking $r = \mathrm{poly}(d)$, since $|\mathcal{S}|$ is polynomial in d.

Observe that Algorithm 1 selects the minimum measurement outcome from all the runs of QPE, and uses it to obtain $\tilde{E}_0$. Let this outcome be $m' \in \{0,\dots,2^t-1\}$. The algorithm converts m' to $m_0 = \lfloor m'2^{-t_0}\rfloor \in \{0,\dots,2^b-1\}$ and uses it to obtain $\tilde{E}_0$, according to the formula

$$\tilde{E}_0 = 2\pi R\hat{\phi}_0 = 2\pi R\frac{m_0}{2^b}. \tag{3.28}$$

Since $|m'/2^t-\phi_0| \leq 2^{-b}$ and $\phi_0, m'/2^t \in [m_0/2^b,(m_0+1)/2^b]$, it follows that $|2\pi R\phi_0 - \tilde{E}_0| = 2\pi R\cdot|\phi_0 - m_0/2^b| \leq 2\pi R/2^b \leq \varepsilon$, which together with (3.22) gives (3.25).

Let $G_0 = \{m : |\frac{m}{2^b}-\phi_0| \leq \frac{1}{2^b}\}$. Algorithm 1 fails either if none of the converted outcomes is an element of G_0, or at least one of the converted outcomes is an element of G_0, but there is another converted outcome (produced by a failure of QPE) smaller than the minimum element of G_0. Thus, we can bound the total probability of failure by

$$\begin{aligned}
\text{Pr(Algorithm 1 fails)} &= \text{Pr(none of the outcomes leads to an element of } G_0)\\
&+ \text{Pr(one of outcomes leads to an element of } G_0\\
&\quad\ \text{but there is at least one smaller converted outcome)}\\
&\leq e^{-r\frac{3}{4|\mathcal{S}|}p} + \text{Pr(QPE failed in at least one of the } |\mathcal{S}| \text{ runs)}\\
&\leq e^{-r\frac{3}{4|\mathcal{S}|}p} + (1-\text{Pr(every run of QPE approximates}\\
&\qquad\qquad\qquad \text{one of the phases with error } 2^{-b}))\\
&\leq e^{-r\frac{3}{4|\mathcal{S}|}p} + (1-p^{r|\mathcal{S}|})\\
&\leq e^{-r\frac{3}{4|\mathcal{S}|}(1-\frac{1}{2^{t_0}-2})} + \left(1-\left(1-\frac{1}{2^{t_0}-2}\right)^{r|\mathcal{S}|}\right) \qquad (3.29)\\
&\leq e^{-r\frac{3}{4|\mathcal{S}|}(1-\frac{1}{2^{t_0}-2})} + \frac{r|\mathcal{S}|}{2^{t_0}-2},
\end{aligned}$$

where the third from last inequality follows from equation (3.30) below. Observe that this bound can be made arbitrarily close to 0 by selecting the number of repetitions r to be a suitable polynomial in d, since $|\mathcal{S}|$ is polynomial in d, and by taking $t_0 = \beta \log d$, where β is an appropriately chosen constant. We have used the fact that if a measurement outcome ℓ fails to estimate any of the phases, i.e., $|\frac{\ell}{2^b} - \phi_s| > 2^{-b}$ for all phases ϕ_s corresponding to eigenvalues of M_h, then

$$\Pr(\ell) = \sum_{s=0}^{N^d-1} |c_s|^2 |\alpha(\ell, \phi_s)|^2 \leq \sum_{s=0}^{N^d-1} |c_s|^2 \frac{1}{2^{t_0}-2} = \frac{1}{2^{t_0}-2} = (1-p). \tag{3.30}$$

Here, the c_s denote the projections of the initial state onto each of the eigenvectors of M_h, and the $|\alpha(\ell, \phi_s)|^2$ denote the probability to get outcome ℓ given the exact eigenvector $u_{h,s}$ as input. We have upper bounds for these quantities from [173, Eq. 5.34]. Therefore, the probability the measurement outcome estimates at least one (or, some) phase is $1 - \Pr(\ell) \geq p$.

Recall that the set $\mathcal{S}$ has been constructed using an upper bound for $E_{(j-1)}$; see equations (3.14) and (3.24). Algorithm 2 essentially repeats Algorithm 1 $(j-1)$ times, but selects the converted measurement outcome in a different way by considering the already selected outcomes. At repetition i, it selects the minimum converted outcome m_i that exceeds the outcome selected at the previous iteration by at least 2, i.e., $m_i \geq m_{i-1} + 2$, where $m_i = \lfloor m_i' 2^{-t_0} \rfloor$ and m_i' is a measurement outcome at the ith run, $i = 1, 2, 3, j-1$; see also equation (3.28). The success probability for both Algorithm 1 and Algorithm 2 follows from (3.29) and is at least

$$\left(1 - \left(e^{-r\frac{3}{4|\mathcal{S}|}(1-\frac{1}{2^{t_0}-2})} + \frac{r|\mathcal{S}|}{2^{t_0}-2}\right)\right)^j, \tag{3.31}$$

which can be made arbitrarily close to 1 by selecting r to be a suitable polynomial in d and taking t_0 to be sufficiently large.

Note that Algorithm 2 computes $\tilde{E}_i = 2\pi R \frac{m_i}{2^b}$, $i = 1, 2, \ldots, j-1$, as estimates of the eigenvalues according to equation (3.4) and the conditions **C1** and **C2** that follow it. If both algorithms are successful with high probability, at the ith run we have that there exists a phase ϕ_i corresponding to an eigenvalue of M_h such that $|2\pi R\phi_i - \tilde{E}_i| \leq \frac{2\pi R}{2^b} \leq \varepsilon$. The condition $m_i \geq m_{i-1} + 2$ in the selection of measurement outcomes guarantees that for any two $i_1 \neq i_2$, the computed matrix eigenvalue approximations satisfy $\tilde{E}_{i_1} \neq \tilde{E}_{i_2}$ and $|\tilde{E}_{i_1} - \tilde{E}_{i_2}| = \Omega(\varepsilon)$ because for the corresponding phases we have $\phi_{i_1} \neq \phi_{i_2}$ as belonging to different intervals; see Figure 3.1. Moreover, the $\tilde{E}_{i_1}$ and $\tilde{E}_{i_2}$ also approximate different eigenvalues $E_{i_1} \neq E_{i_2}$ of the continuous operator because we have used a very fine discretization. Finally, the algorithm does not fail to produce consecutive eigenvalues unless they differ by less than $O(\varepsilon)$ because we always select the minimum outcome that satisfies $m_i \geq m_{i-1} + 2$.

3.5.3.1 Cost of Quantum Phase Estimation

Algorithms 1 & 2 use QPE as a module. The cost of QPE depends on the cost to prepare its initial state, and on the cost to implement the matrix exponentials $W^{2^0}, W^{2^1}, ..W^{2^{t-1}}$, where $W = e^{iM_h/R}$. We approximate these exponentials below using Suzuki-Trotter splitting, the analysis of which proceeds similarly to that of [183, 186]; see also the details given in Chapter 4.

The initial states are taken from $\mathcal{S}$ which contains eigenvectors of $-\frac{1}{2}\Delta_h$ according to (3.24). Each eigenvector can be prepared efficiently using the quantum Fourier transform, which diagonalizes the Laplacian, with a number of quantum operations proportional to $d \cdot \log^2 \frac{d}{\varepsilon}$ and using number of qubits $\log_2 N^d = d \cdot O(\log \frac{d}{\varepsilon})$. We remark that from the tensor product structure of the eigenvectors of $-\frac{1}{2}\Delta_h$, it suffices to prepare eigenvectors of the one-dimensional Laplacian; see e.g. [56, 153, 247].

Now let us turn to the approximation of the matrix exponentials. We simulate the evolution of the Hamiltonian $H = M_h/R$ for times 2^τ, $\tau = 0, 1, \ldots, t-1$, where we have set $t = b + t_0$. Let $H = H_1 + H_2$ where $H_1 = -\Delta_h/2R$ and $H_2 = V_h/R$, where we assume V is given by an oracle.

To simulate quantum evolution by H_1, assuming the known eigenvalues of $-\frac{1}{2}\Delta_h$ are given by a quantum query oracle with $O(\log \frac{1}{\varepsilon})$ bits of accuracy, we again use the quantum Fourier transform to diagonalize H_1 with cost (i.e., a number of quantum operations) bounded by $d \cdot O(\log^2 \frac{d}{\varepsilon})$, and requiring a number of qubits proportional to $d \log \frac{d}{\varepsilon}$. Alternatively, if the eigenvalues of $-\frac{1}{2}\Delta_h$ are implemented explicitly (without an oracle) by the quantum algorithm, then the number of quantum operations required is a low-order polynomial in d and $\log_2 \frac{1}{\varepsilon}$, and so is the number of qubits [56]. For simplicity, we will not pursue this alternative here. The evolution of a system with Hamiltonian H_2 can be implemented using two quantum queries returning the values of V at the grid points, and phase kickback. The queries are similar to those in Grover's algorithm [173] and the function evaluations of V are truncated to $O(\log \frac{1}{\varepsilon})$ bits.

We use a splitting formula S_{2k} of order $2k+1$, $k \geq 1$, to approximate $W^{2^t} = e^{i(H_1+H_2)2^t}$ by a product of the form

$$\prod_{\ell=1}^{N_t} e^{iA_\ell z_\ell}, \tag{3.32}$$

where $A_\ell \in \{H_1, H_2\}$ and suitable z_ℓ that depends on t and k.

The splitting formula S_{2k} is due to Suzuki [214, 215]. It is used to approximate $e^{i(B+C)\Delta t}$, where B and C are Hermitian matrices. This formula is defined recursively by

$$\begin{aligned} S_2(B,C,\Delta t) &= e^{iB\Delta t/2} e^{iC\Delta t} e^{iB\Delta t/2} \\ S_{2k}(B,C,\Delta t) &= [S_{2k-2}(B,C,p_k\Delta t)]^2 S_{2k-2}(B,C,(1-4p_k)\Delta t) \\ &\qquad \times [S_{2k-2}(B,C,p_k\Delta t)]^2, \end{aligned}$$

where $p_k = (4 - 4^{1/(2k-1)})^{-1}$, $k = 2, 3, \ldots$.

Unfolding the recurrence above and combining it with [186, Thm. 1] we obtain that the approximation of W^{2^τ} has the form

$$\widetilde{W}^{2^\tau} = e^{iH_1 a_{\tau,0}} e^{iH_2 b_{\tau,1}} e^{iH_1 a_{\tau,1}} \cdots e^{iH_2 b_{\tau,L_\tau}} e^{iH_1 a_{\tau,L_\tau}}, \tag{3.33}$$

where $a_{\tau,0}, \ldots, a_{\tau,L_\tau}$ and $b_{\tau,1}, \ldots, b_{\tau,L_\tau}$ and L_τ are parameters, $\tau = 0, \ldots, t_0 + b - 1$. The number of exponentials involving H_1 and H_2 in the expression above is $N_\tau = 2L_\tau + 1$. An explicit algorithm for computing each $\widetilde{W}^{2^\tau}$ is given in [183].

Let $\|\cdot\|$ be the matrix norm induced by the Euclidean vector norm. From [186, Thm. 1 & Cor. 1] the number N_t of exponentials needed to approximate W^{2^t} by a splitting formula of order $2k+1$ with error ε_τ, $\tau = 0, \ldots, t_0 + b - 1$, is

$$N_\tau \leq 16e\|H_1\| 2^\tau \left(\frac{25}{3}\right)^{k-1} \left(\frac{8e\, 2^\tau \|H_2\|}{\varepsilon_\tau}\right)^{1/(2k)},$$

for any $k \geq 1$. Since we want to approximate all the W^{2^τ}, $\tau = 0, 1, ..., t_0 + b - 1$, we sum the number of exponentials required to approximate each one of them. Thus the total number of matrix exponentials required by Algorithm 2, $\mathcal{N}_{tot}$, is bounded from above by

$$\begin{aligned} \mathcal{N}_{tot} &= jr|\mathcal{S}| \sum_{\tau=0}^{t_0+b-1} N_\tau \\ &\leq jr|\mathcal{S}| \left(16e\|H_1\| \left(\frac{25}{3}\right)^{k-1} (8e\|H_2\|)^{1/(2k)} \sum_{\tau=0}^{t_0+b-1} 2^\tau \left(\frac{2^\tau}{\varepsilon_\tau}\right)^{1/(2k)}\right). \end{aligned} \tag{3.34}$$

The factor $jr|\mathcal{S}|$ is the number of executions of QPE performed by our algorithms, and the second factor is the cost of a single QPE. Note that j is the number of eigenvalues we wish to estimate, $|\mathcal{S}|$ is the number of eigenvectors we use as initial states, and r is the number of times we repeat QPE per initial state to boost the success probability of getting the desired outcome. We select a polynomial $g(d)$ such that the product $r|\mathcal{S}|/g(d) = o(1)$ (as $d \to \infty$). We then select the error of each exponential to be $\varepsilon_\tau = \frac{2^{\tau+1-(b+t_0)}}{40g(d)}$, $\tau = 0, \dots, t_0+b-1$. It is easy to check that $\sum_{\tau=0}^{t_0+b-1} \varepsilon_\tau \leq \frac{1}{20g(d)}$. Thus the success probability of QPE is reduced by at most twice this amount [173, p. 195], giving $1 - \frac{1}{2(2^{t_0}-2)} - \frac{1}{10g(d)}$. Next we set $t_0 = \lfloor \log_2(5g(d)+2) \rfloor$, to get $p = 1 - \frac{1}{2^{t_0}-2}$ that we used above in deriving equation (3.29). Our choice of $g(d)$ and t_0 aims to make the bound of equation (3.31) arbitrarily close to 1.

The largest eigenvalue of $-\Delta_h$ is $4dh^{-2}\sin^2(\pi N h/2) < 4dh^{-2}$. Since $R = 3dh^{-2}$, $H_1 = -\frac{1}{2}\Delta_h/R = -\frac{1}{2}\frac{1}{3dh^{-2}}\Delta_h$ and we have $\|H_1\| \leq \frac{2dh^{-2}}{3dh^{-2}} = \frac{2}{3}$. Since V is uniformly bounded by M and $H_2 = V_h/R$ we have $\|H_2\| \leq M/3dh^{-2}$. Substituting the value of ε_τ in (3.34), yields that the algorithm uses a number of exponentials of H_1 and H_2 that satisfies

$$\begin{aligned}
\mathcal{N}_{tot} &\leq jr|\mathcal{S}| \left(16e\|H_1\| \left(\frac{25}{3}\right)^{k-1} (8e\|H_2\|)^{1/(2k)}\right) \sum_{\tau=0}^{t_0+b-1} 2^\tau \left(\frac{40g(d)2^\tau}{2^{\tau+1-(b+t_0)}}\right)^{1/(2k)} \\
&\leq jr|\mathcal{S}| \left(16e\|H_1\| \left(\frac{25}{3}\right)^{k-1} (8e\|H_2\|)^{1/(2k)}\right) \left(20g(d)2^{t_0+b}\right)^{1/(2k)} \\
&\leq jr|\mathcal{S}| \left(16e\|H_1\|2^{t_0+b} \left(\frac{25}{3}\right)^{k-1} \left(160e\, 2^{t_0+b}\|H_2\|g(d)\right)^{1/(2k)}\right).
\end{aligned}$$

Using the bounds on $\|H_1\|$ and $\|H_2\|$, we obtain

$$\mathcal{N}_{tot} \leq jr|\mathcal{S}| \left(\frac{32e}{3}2^{t_0+b} \left(\frac{25}{3}\right)^{k-1} \left(160e\, 2^{t_0+b}\frac{Mh^2}{3d}g(d)\right)^{1/(2k)}\right).$$

From $b = 5\lceil \log_2 \frac{d}{\varepsilon} \rceil + 7$, we have $2^b = 2^{5\lceil log_2 \frac{d}{\varepsilon}\rceil+7} \leq 2^{12}\left(\frac{d}{\varepsilon}\right)^5 = O(\frac{d^5}{\varepsilon^5})$. Since $h < \frac{\varepsilon^2}{d^2}$, we have $2^b h^2 \leq 2^{12}\frac{d}{\varepsilon} = O(\frac{d}{\varepsilon})$. Also, $2^{t_0} = 2^{\lfloor \log_2(5g(d)+2) \rfloor} \leq 5g(d)+2$. We obtain

$$\begin{aligned}
\mathcal{N}_{tot} &\leq jr|\mathcal{S}| \left(\frac{32e}{3}(5g(d)+2)\left(2^{12}\left(\frac{d}{\varepsilon}\right)^5\right)\left(\frac{25}{3}\right)^{k-1}\right. \\
&\qquad \left. \times \left(160e\ (5g(d)+2)\left(2^{12}\frac{d}{\varepsilon}\right)\frac{M}{3d}g(d)\right)^{1/(2k)}\right) \\
&\leq jr|\mathcal{S}| \left(\widetilde{C}\,\frac{d^5 g(d)}{\varepsilon^5}\left(\frac{25}{3}\right)^{k-1}\left(\widehat{C}\,\frac{g^2(d)}{\varepsilon}\right)^{1/(2k)}\right), \qquad (3.35)
\end{aligned}$$

for any $k > 0$, where $\widetilde{C}$ and $\widehat{C}$ are suitable constants.

The *optimal* k^*, i.e., the one minimizing the upper bound for $\mathcal{N}_{tot}$ in (3.35), is obtained in [186, Sec. 5] and is given by

$$k^* = \left\lfloor \sqrt{\frac{1}{2}\log_{25/3}\left(\widehat{C}\,\frac{g^2(d)}{\varepsilon}\right)} + \frac{1}{2} \right\rfloor = \bar{C}\sqrt{\ln\frac{d}{\varepsilon}},$$

for a suitable constant $\bar{C}$, since $g(d)$ is a polynomial in d and we are taking its logarithm. With k^* and using again [186, Sec. 5], equation (3.35) yields

$$\mathcal{N}^*_{tot} \le \widetilde{C} jr|\mathcal{S}|\frac{d^5}{\varepsilon^5}\, g(d)\, e^{2\bar{C}\sqrt{\ln\frac{25}{3}\ln\frac{d}{\varepsilon}}} = O\left(g^2(d)\left(\frac{d}{\varepsilon}\right)^{5+\eta}\right) \quad \text{as } d\varepsilon \to 0, \tag{3.36}$$

where we have used $j = O(1)$ and $r|\mathcal{S}| = o(g(d))$, and where the equality above holds asymptotically for arbitrarily small $\eta > 0$.

We remark that of the N^*_{tot} matrix exponentials roughly half involve H_1 and the remaining involve H_2; see (3.33). Since each exponential involving H_2 requires two queries the total number of queries is also of order N^*_{tot}. The cost to prepare the initial state, to diagonalize $-\frac{1}{2}\Delta_h$, and to implement the inverse Fourier transform that is applied prior to measurement in QPE, is proportional to

$$d\log^2\frac{d}{\varepsilon} + (t_0 + b)^2 = O\left(d\log^2\frac{d}{\varepsilon}\right),$$

since $t_0 + b = O(\log\frac{d}{\varepsilon})$. Hence, the total number of quantum operations, excluding queries, is proportional to

$$\mathcal{N}^*_{tot}\cdot d\log^2\frac{d}{\varepsilon}. \tag{3.37}$$

Equations (3.36) and (3.37) yield that the total cost of the algorithm, including the number of queries and the number of all other quantum operations, is proportional to

$$d\, g^2(d)\left(\frac{d}{\varepsilon}\right)^{5+\delta},$$

where $\delta > 0$ is arbitrarily small.

Finally, using equation (3.31) we can select r to be polynomial in d and obtain success probability at least $\frac{3}{4}$, and the cost remains polynomial in $\frac{1}{\varepsilon}$ and d. We summarize our results in the following theorem.

Theorem 1. *Consider the time-independent Schrödinger equation (3.12) on the d-dimensional unit cube with Dirichlet boundary conditions and where the potential V and its first-order derivatives are uniformly bounded. Algorithms 1 & 2 of Section 3.4 compute approximations of $j = O(1)$ low-order eigenvalues as in equation (3.4), each with error $O(\varepsilon)$ and satisfying conditions* **C1** *and* **C2** *of Section 3.2, with overall success probability at least*

$$\left(1 - \left(e^{-r\frac{3}{4|\mathcal{S}|}(1-\frac{1}{2^{t_0}-2})} + \frac{r|\mathcal{S}|}{2^{t_0}-2}\right)\right)^j \geq \frac{3}{4},$$

where r and $|\mathcal{S}|$ are polynomial in d, $t_0 = \lfloor \log_2(5g(d)+2)\rfloor$, and $g(d)$ is a polynomial in d selected such that $r|\mathcal{S}| = o(g(d))$. The algorithms apply QPE with initial state each element of a set of trial eigenvectors $\mathcal{S}$, and repeat this procedure r times. They use a number of queries proportional to

$$\left(\frac{d}{\varepsilon}\right)^{5+\delta} g^2(d) \quad \text{as } d\varepsilon \to 0,$$

and a number of quantum operations excluding queries proportional to

$$\left(\frac{d}{\varepsilon}\right)^{5+\delta} d\ g^2(d) \quad \text{as } d\varepsilon \to 0,$$

where $\delta > 0$ is arbitrarily small. The algorithms use a number of qubits proportional to

$$d\ \log\frac{d}{\varepsilon} + \log g(d).$$

Remark 2. *The 5 in the exponent of $\frac{d}{\varepsilon}$ is due to the fact that for simplicity we have taken $N = 2^{\lceil 2\log_2 \frac{d}{\varepsilon}\rceil}$ in the discretization of the continuous operator and our consequent choice of b, the number of bits of accuracy of QPE. As explained, a fine discretization is needed to ensure that degenerate eigenvalues of the continuous problem are approximated by tightly clustered eigenvalues of the matrix. By taking slightly coarser discretization, it is possible to reduce this exponent. As our goal was to establish an algorithm with cost polynomial in d and ε^{-1} we do not pursue this further.*

Remark 3. *The classical complexity of approximating a constant number of low-order eigenvalues with error ϵ grows as $\left(\frac{1}{\epsilon}\right)^d$ in the deterministic worst case [184]. Since our quantum algorithm for this problem has cost polynomial in d and $\frac{1}{\epsilon}$, it vanquishes the curse of dimensionality.*

3.6 Discussion

There are a number of recent results suggesting that certain eigenvalue problems are very hard, even for quantum computers [44, 61, 148, 205, 243]. On the other hand, obtaining positive results for eigenvalue problems showing where quantum algorithms give advantages over classical algorithms is particularly important towards understanding the power of quantum computers.

We show such a positive result for the approximation of ground and excited state energies on a quantum computer. In summary, general conditions for the efficient approximation of a constant number of low-order excited state energies follow by combining conditions for efficient quantum simulation and for deriving a relatively small set of trial eigenvectors that can be implemented efficiently as quantum states. For quantum algorithms, previous approaches for computing the ground state energy require stronger conditions on V than those we consider here, and these approaches do not extend to computing excited state energies. We have developed an entirely new approach to approximate not only the ground state energy, but also excited state energies, with cost polynomial in d and ε^{-1}. For the special case of the time-independent Schrödinger equation with d degrees of freedom we study, our quantum algorithm vanquishes the curse of dimensionality.

We remark on several open problems. We have assumed that $L = L^0 + V$, but such a partition need not be unique, and different partitions may result in algorithms with significantly different costs. It is possible, that with additional assumptions, one would be able to determine suitable partitions leading to fast algorithms. Such a characterization is an open problem. We have provided a condition for constructing a set of trial eigenvectors $\mathcal{S}$. Improving this condition to minimize the size of the resulting set $\mathcal{S}$ is another open problem. Finally, in the initial investigation of quantum algorithms for eigenvalue problems, strong assumptions on V were considered in order to obtain efficient algorithms. Progressively, culminating with our work, these assumptions have been weakened. It is important to continue working in this direction to further extend the scope of our algorithm, in particular to first-quantized approaches for important problems in physics and chemistry.

Chapter 4

Divide and Conquer Approach to Hamiltonian Simulation

4.1 Introduction

Simulating quantum mechanical systems is a very important yet very difficult problem. The computational cost of the best classical deterministic algorithm known grows exponentially with the system size. In some cases classical randomized algorithms, such as quantum Monte Carlo, have been used to overcome the difficulties, but these algorithms also have limitations. On the other hand, as Feynman proposed [95], quantum computers may be able to carry out the simulation more efficiently than classical computers. This led to a large body of research dealing with quantum algorithms for Hamiltonian simulation [5, 17, 18, 29–34, 40, 59, 62, 63, 65, 146, 162, 186, 193, 194, 197, 210, 246, 248, 249, 257, 258], with efficient algorithms found for simulating many classes of important Hamiltonians. In particular, these algorithms have numerous applications to problems in physics and chemistry [55, 140, 141, 147, 233, 242].

Nevertheless, for certain problems, despite being "efficient," the cost of implementing these quantum algorithms appears prohibitive; i.e., even though they scale polynomially with respect to the problem size, the resources required are formidable for problems of interest in practice. A prototypical example which we will consider in this chapter is the *electronic Hamiltonian*, which encodes the energy level structure of a given molecule, although we emphasize that our results are general and apply to other problems. The (Born-Oppenheimer approximate) electronic Hamiltonian in the second quantized form [126, 217] is given by

$$H = \sum_{p,q=1}^{\mathcal{N}} h_{pq} a_p^\dagger a_q + \frac{1}{2} \sum_{p,q,r,s=1}^{\mathcal{N}} h_{pqrs} a_p^\dagger a_q^\dagger a_r a_s. \tag{4.1}$$

Here $a^\dagger$ and a are fermionic creation and annihilation operators, respectively, and the coefficients $h_{pq}, h_{pqrs} \in \mathbb{R}$ for $p, q, r, s = 1, \ldots, \mathcal{N}$ are provided as input. Combining adjoint pairs of terms as Hamiltonians H_j we may write

$$H = \sum_{j=1}^{m} H_j,$$

where each H_j may be simulated efficiently [246], and their number is $m = \Theta(\mathcal{N}^4)$. The parameter $\mathcal{N}$ is at least the number of electrons, and should be taken larger than this to increase the accuracy in the underlying problem. For important problems in chemistry that are believed to be beyond the capabilities of classical algorithms, where, say, $\mathcal{N} \simeq 100$, the number of Hamiltonian terms m is proportional to 10^8. Hence, even for a quantum algorithm with relatively low polynomial cost dependence on m, say m^2, the simulation cost is immediately prohibitive, independent of its dependence on the other simulation problem parameters (time, accuracy, etc.).

It is thus critical to derive simulation algorithms with reduced cost dependence on m. In this chapter, we show a general *divide and conquer* approach which takes advantage of Hamiltonian structure. By partitioning a Hamiltonian $H = \sum_{j=1}^{m} H_j$ into a number of partial sums, simulating each sum separately, and then recombining the partial results, we obtain refined cost bounds which can lead to faster simulation. In particular, for simulating the electronic Hamiltonian (4.1) we show that, under reasonable assumptions taken from the literature, our algorithm reduces the simulation cost dependence from $\mathcal{N}^8$ to $\mathcal{N}^9$, to between $\mathcal{N}^5$ to $\mathcal{N}^7$, with exponent depending on the particular problem representation and details. Previously, a sequence of papers has argued for the possibility of similar cost improvements based on heuristics or empirical evidence [18, 167, 193, 240]. In contrast, we provide rigorous cost and error bounds for our algorithms. The following table summarizes our cost estimates, which we explain in detail in Section 4.4.

Although the derivation and analysis of our algorithms is complicated, we emphasize that their implementation is relatively straightforward, and they yield an easily computable sequence of basic quantum operations that is similar in form to commonly used Suzuki-Trotter splitting formula approaches. We remark that a recent paper [17], specific to the electronic structure problem, has given

Method	Cost dependence on $\mathcal{N}$
Splitting Formulas [186, 240]	$\mathcal{N}^8 - \mathcal{N}^9$
Truncated Taylor Series [17, Eq. 46]*	$\mathcal{N}^8$
Our algorithms with local basis functions	$\mathcal{N}^5 - \mathcal{N}^7$

Table 4.1: Summary of the cost estimates, expressed in terms of $\mathcal{N}$, for different algorithms for the simulation of the electronic Hamiltonian (4.1). $(^*)$ The cost scaling is improved to $\mathcal{N}^5$ under strong assumptions on the basis functions and the computation of the h_{pq}, h_{pqrs} [17].

a sophisticated algorithm with cost scaling nearly as $\mathcal{N}^5$, but under even stronger assumptions. Our approach seeks to give simple-to-implement quantum algorithms that are generally applicable, and can achieve similar performance improvements for important problems.

We emphasize that our approach is general and applies to simulation problems beyond the electronic Hamiltonian. We derive our algorithms, analysis, and results in terms of a general Hamiltonian simulation problem, and revisit quantum chemistry as an application at the end of the chapter.

4.1.1 Problem Definition and Background

In the Hamiltonian simulation problem one is given a Hamiltonian H acting on q qubits, a time $t \in \mathbb{R}$, and an accuracy demand ε, and the goal is to derive a quantum algorithm that constructs an operator $\widetilde{U}$ which approximates the unitary operator e^{-iHt} with error $\|\widetilde{U} - e^{-iHt}\| \leq \varepsilon$ measured in the spectral norm.[1] When the Hamiltonian is given explicitly, the size of the quantum circuit realizing the algorithm is its cost. In particular, the cost depends on the complexity parameters q, t and ε^{-1}. On the other hand, when the Hamiltonian is given by an oracle, the number of queries (oracle calls) used by the algorithm plays a major role in its cost, in addition to the number of qubits and the other necessary quantum operations. Different types of queries have been considered in the literature.

[1]The spectral-, or two-norm, of a Hermitian matrix H is the magnitude of the maximum eigenvalue of H, and typically written $\|H\|_2$.

Many papers study only the query complexity. For example, [29,186] use splitting formulas[2] of order $2k+1$ to simulate $H = \sum_{j=1}^{m} H_j$, $\|H_1\| \geq \|H_2\| \geq \cdots \geq \|H_m\|$. They approximate e^{-iHt} with error ε by an ordered product of N unitary operators of the form $e^{-iH_{j_\ell}t_\ell}$, $j_\ell \in \{1,\ldots,m\}$, $|t_\ell| \leq t$, $\ell = 1,\ldots,N$. It is assumed that the Hamiltonian H is given by an oracle (a "black-box"), and that H can be decomposed efficiently by a quantum algorithm using oracle calls into a sum of Hamiltonians H_j, $j = 1,\ldots,m$, that individually can be simulated efficiently. This kind of query has been considered in numerous other papers, see e.g., [5,65,162,248]. The cost of the simulation is the total number of oracle calls,[3] which is proportional to the number N of exponentials $e^{-iH_{j_\ell}t_\ell}$. Then [186] shows the number of exponentials is bounded from above by

$$N \leq m^2 \|H_1\| t \left(\frac{4em\|H_2\|t}{\varepsilon} \right)^{1/(2k)} \frac{16e}{3} \left(\frac{25}{3} \right)^{k-1}, \tag{4.2}$$

where $\|\cdot\|$ is the spectral norm. We shall also use the number N as our measure of simulation cost for the algorithms we give later in this chapter.

On the other hand, [31] uses a different type of query to simulate d-sparse Hamiltonians. In particular, one is given access to a d-sparse Hamiltonian H acting on q qubits via a black box that accepts a row index i and a number j between 1 and d, and returns the position and value of the jth nonzero entry of H in row i. The paper shows a clever technique applied in combination with oblivious amplitude amplification to derive an algorithm simulating d-sparse Hamiltonians with a number of queries

$$N = O\left(\frac{\tau \log \tau/\varepsilon}{\log\log \tau/\varepsilon} \right), \quad \tau = d^2 \|H\|_{max} t, \tag{4.3}$$

where $\|\cdot\|_{max}$ is the maximum norm. This is an important result. The dependence of the cost on ε^{-1} is exponentially better in the latter case. However, this fact is not sufficient to conclude that the algorithm is exponentially faster than previously known simulation algorithms, because the size of the other complexity parameters, τ and particularly the Hamiltonian norm $\|H\|_{\max}$, needs to be taken into account as well.

For instance, the spectral and maximum norms are proportional to dh^{-2} in the case where $H = -\Delta_h + V_h$ is a matrix obtained from the discretization of the d-variate Laplacian Δ and a uniformly

[2]Recall that high-order splitting formulas [214,215] were used for Hamiltonian simulation in Algorithms 1 and 2 of Chapter 3. We provide a review of splitting formulas in Appendix D.1.

[3]Since the H_j are obtained by decomposing H by the algorithm, an oracle call to any H_j is simulated by making oracle calls to H; see [29, Sec. 5] for details.

bounded d-variate potential function V on a grid with mesh size h; see [77] for details. In this case, the sparsity of H is $\Theta(d)$. Thus, for univariate functions the sparsity is constant. If we set $h = \varepsilon$,[4] both cost estimates (4.2, 4.3) become polynomial in ε^{-1} and there is no exponential speedup. It is easy to extend this argument to d-variate functions and the situation is more interesting. In this case H is a matrix of size $\varepsilon^{-d} \times \varepsilon^{-d}$. For $k = 1$ the bound (4.2) is proportional to

$$d\varepsilon^{-2.5}t^{1.5}$$

while that of (4.3), modulo polylogarithmic factors, is proportional to

$$d^3\varepsilon^{-2}t.$$

Both query estimates are *low degree polynomials* in each of the complexity parameters. Moreover, polynomial improvements, such as reducing the exponent of d in (4.3) by one, as in [33], hardly make a difference. This situation is typical for matrices obtained from the discretization of ordinary and partial differential equations. We may have an exponential speedup when τ is at most polylogarithmic in ε^{-1}, but this is not typically the case in practice. Indeed [31] does not mention any practical situation where an exponential speedup is realized. These considerations apply to other recent papers also showing polylogarithmic dependence on ε^{-1} of the query complexity [32, 33].

It is interesting to observe that the query complexity might be low and depend on ε^{-1} polylogarithmically as in [32], yet when one considers the total gate count the picture may be quite different. An example can be found in [17, Table 1 & Table 2] which applies [32] to the simulation of the Born-Oppenheimer second-quantized electronic Hamiltonian (4.1). In particular the query complexity is proportional to $t\,\mathcal{N}^4$ times a quantity polylogarithmic in t, $\mathcal{N}$ and ε^{-1}, while the total gate count is proportional to $t\,\mathcal{N}^8$ times a quantity polylogarithmic in t, $\mathcal{N}$ and ε^{-1}. Improvements of the cost are possible under significant assumptions on the class of basis functions used and assumptions about the cost and accuracy in computing the h_{pq} and h_{pqrs} by the quantum algorithm. Moreover, in chemistry the desired accuracy is not arbitrarily small [18] and thus it may impact the cost only by a constant factor. The important parameter is $\mathcal{N}$ which is the number of single-particle

[4]When dealing with partial differential equations, the mesh size h determines the discretization error, which subject to smoothness conditions often is $O(h^\alpha)$, for some $\alpha > 0$. In terms of the partial differential equation, the combination of the discretization error and the simulation error determines the accuracy of the final result. In this sense h and ε are related.

basis functions used in the approximation of the Born-Oppenheimer electronic Hamiltonian. Larger values of $\mathcal{N}$ give more accurate approximations of the Hamiltonian operator.

Although improving exponentially the dependence of the simulation cost on ε^{-1} is very significant, it is not a panacea. We already mentioned that the other complexity parameters may be dominant. There are other issues as well to consider. Queries that require one to have precomputed and stored the positions and values of all nonzero matrix entries of the Hamiltonian, or else have an efficient routine to generate these, can be restrictive. Similar concerns are discussed in [1]. Moreover, simulation algorithms relying on oblivious amplitude amplification are probabilistic. This means that for applications where numerous Hamiltonian simulations need to be carried out, such as in phase estimation, the overall success probability must be further boosted. Making such an algorithm deterministic in practice is a numerical stability consideration.

On the other hand, without being advocates of splitting methods, we cannot avoid recognizing they have some very appealing features. Splitting methods "conserve an important symmetry of the system in problems of quantum dynamics and Hamiltonian dynamics", a "remarkable advantage" according to [124]. Suzuki also remarks that splitting methods are particularly useful for studying quantum coherence [214]. Simulation using splitting methods is deterministic in the sense that any repetition produces the same output with exactly the same accuracy. The simulation methods [31–33] do not have these properties.

4.1.2 Divide and Conquer Simulation

We give a new approach for simulating Hamiltonians of the form

$$H = \sum_{j=1}^{m} H_j. \tag{4.4}$$

Our approach is especially useful when the number m of Hamiltonians H_j is large, and many of the H_j have relatively small norm. Such Hamiltonians are common in physics and chemistry [147, 177, 189, 213, 216, 233, 242]. For example, a system of interacting bodies or particles is described typically by a Hamiltonian of the above form.

Without loss of generality, assume that the H_j are indexed as

$$\|H_1\| \geq \|H_2\| \geq \cdots \geq \|H_m\|. \tag{4.5}$$

For many problems, the norms $\|H_j\|$ vary substantially, and many Hamiltonians may have norm $\|H_j\| \ll \|H_1\|$. Then one can take advantage of the discrepancy between the norm sizes to derive fast simulation algorithms. The main idea is as follows:

0. Partition the Hamiltonians $H_1, H_2, \ldots, H_m$ into groups using the magnitude of their norms. Ideally, Hamiltonians of proportionate norms are grouped together.
1. Approximate e^{-iHt} by a splitting formula applied with respect to the partition into groups, pretending that the sums of Hamiltonians in each group can be simulated exactly.
2. Simulate the sum of the Hamiltonians in each group separately with sufficient accuracy.
3. Combine all the group simulation results to give the overall simulation of H.

A top-level description of the procedure above applied to two groups and utilizing splitting formulas is shown in Figure 4.1 below. Nevertheless, our approach is not limited to splitting formulas.

To motivate this idea consider the bound (4.2) which depends particularly on m, $\|H_1\|$ and $\|H_2\|$, and not on $\|H_3\|, \ldots, \|H_m\|$. For the sake of argument, suppose m is huge and $\|H_2\| \gg \|H_3\|$. Then we can split the Hamiltonians in two groups $\{H_1, H_2\}$ and $\{H_3, \ldots, H_m\}$, simulate $A := H_1 + H_2$ and $B := H_3 + \cdots + H_m$ independently, and then combine the partial simulation results using a splitting formula. Observe that $e^{-iHt} \to e^{-iAt}$ as $\|H_3\| \to 0$ and in the limit the total simulation cost becomes independent of m. Thus in the limit the bound (4.2) holds with m replaced by 2. This suggests that when many Hamiltonians are relatively small in norm one should be able to improve the cost estimate (4.2) by partitioning them into groups and, for instance, using splitting formulas of different orders as we indicate in Figure 4.1, and we will explain in detail later.

An example application with these properties is the simulation of the electronic Hamiltonian (4.1). This problem has been well-studied in the literature; see e.g. [17,18,123,147,154,167,193,240,246]. Recall the number of single-particle basis functions $\mathcal{N}$ is typically chosen to be proportional to the number of particles in a given problem, and the number of Hamiltonians is $m = \Theta(\mathcal{N}^4)$. The best classical algorithms can reasonably solve problem instances with $\mathcal{N}$ in the range $50 - 70$, and it is believed that a quantum computer able to simulate problem instances with $\mathcal{N} \simeq 100$ will solve many important applications ranging from chemical engineering to biology [240]. In these cases, the required modest number of qubits (typically $\Theta(\mathcal{N})$ [15]) makes these very attractive applications for

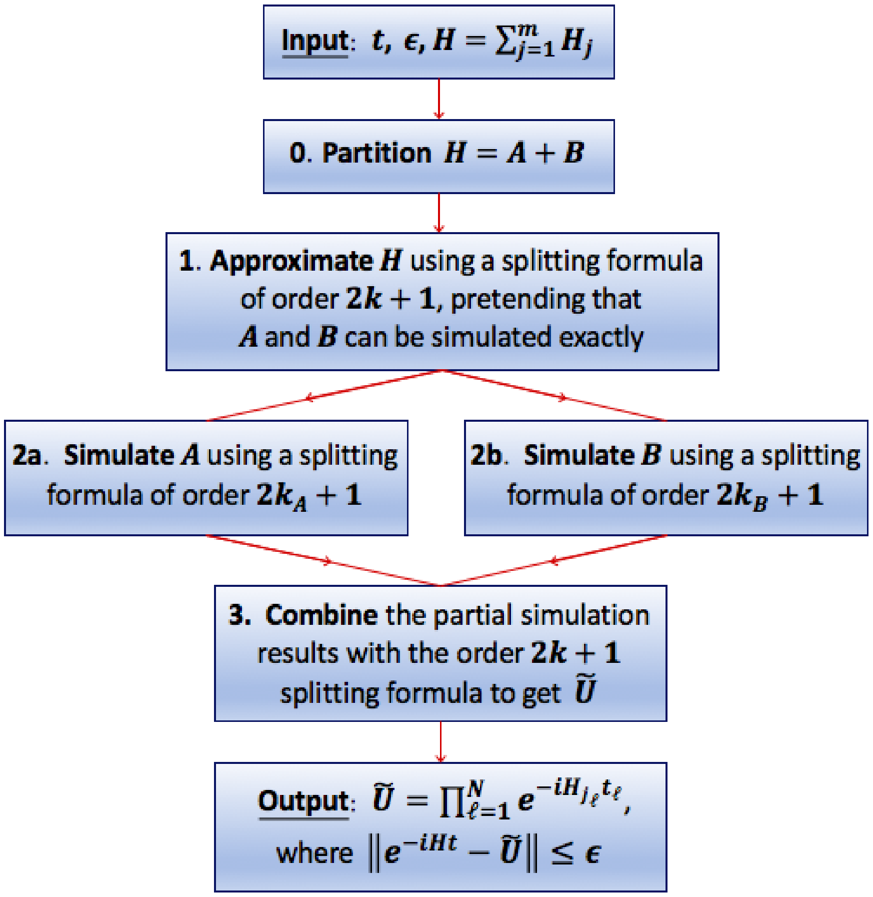

Input: $t,\ \epsilon, H = \sum_{j=1}^{m} H_j$
0. Partition $H = A + B$
1. Approximate H using a splitting formula of order $2k+1$, pretending that A and B can be simulated exactly
2a. Simulate A using a splitting formula of order $2k_A + 1$
2b. Simulate B using a splitting formula of order $2k_B + 1$
3. Combine the partial simulation results with the order $2k+1$ splitting formula to get $\tilde{U}$
Output: $\tilde{U} = \prod_{\ell=1}^{N} e^{-iH_{j_\ell} t_\ell}$, where $\|e^{-iHt} - \tilde{U}\| \le \epsilon$

Fig. 4.1: Divide and conquer simulation using splitting formulas.

early quantum computers. As explained, despite the many recent advances in quantum simulation algorithms, the cost of Hamiltonian simulation remains the primary bottleneck to solving this problem on a quantum computer, Indeed, reducing the simulation cost dependence on m (i.e., on $\mathcal{N}$) for this problem has been the subject of considerable recent effort [17, 18, 123, 167, 193, 240]. Furthermore, in many situations, it has been observed that the Hamiltonian norms vary significantly, and many of them are relatively small [126, 139]. It has been suggested that this could be used in some way to potentially reduce the simulation cost, without any rigorous analysis [18, 123, 139, 167]. In contrast, in this chapter we develop algorithms that use the discrepancy between sizes of Hamiltonian norms to speedup Hamiltonian simulation and we derive their cost in full detail.

4.1.3 Overview of Main Results

For simplicity, we consider here the case where we partition the Hamiltonians in two groups, but the idea extends to many groups, as we show in Section 4.2. Let $H = A + B$, with $A = \sum_{i=1}^{m'} H_i$ and $B = \sum_{i=m'+1}^{m} H_i$, with $m' \ll m$, where again $\|H_1\| \geq \|H_2\| \geq \cdots \geq \|H_m\|$. The bound (4.2) for the number of queries N scales with m as $m^{2+1/2k}$, and our goal is to improve that.

1. Suppose we have two *arbitrary* algorithms $\widetilde{U}_A(\tau) \simeq e^{-iA\tau}$ and $\widetilde{U}_B(\tau) \simeq e^{-iB\tau}$ for approximately simulating the Hamiltonians A and B, respectively, for time $\tau \in \mathbb{R}$. We show how splitting formulas may be used to combine $\widetilde{U}_A$ and $\widetilde{U}_B$ such that an approximation $\widetilde{U}(t)$ to $U(t) = e^{-iHt}$ is achieved. For example, dividing the time t into n intervals of length $\tau := t/n$ and using the Strang splitting formula [186] we get the overall approximation

$$\widetilde{U}(t) := \left(\widetilde{U}_A(\tau/2)\, \widetilde{U}_B(\tau)\, \widetilde{U}_A(\tau/2)\right)^n \simeq \left(e^{-iA\tau/2} e^{-iB\tau} e^{-iA\tau/2}\right)^n \simeq e^{-iHt}. \tag{4.6}$$

   Then to obtain $\|U(t) - \widetilde{U}(t)\| = O(\varepsilon)$ it suffices that $\|e^{-iA\tau/2} - \widetilde{U}_A(\tau/2)\|$ and $\|e^{-iB\tau} - \widetilde{U}_B(\tau)\|$ are each of order ε/n. Higher order splitting formulas may be used instead of the Strang splitting formula such that the error and resulting cost are further reduced.

   In the following items we use splitting formulas to derive $\widetilde{U}_A$ and $\widetilde{U}_B$; however, in principle, different applicable simulation algorithms could be used for each of $\widetilde{U}_A$ and $\widetilde{U}_B$. Moreover, we use the ordering $\|H_1\| \geq \|H_2\| \ldots$ to partition the H_j, though in practice criteria other than the norms may be used to group the Hamiltonians, such as sparsity, commutativity, or unitarity or any other property which may allow one to use an advantageous algorithm for simulating the Hamiltonians in that group.

2. We use splitting formulas (of orders $2k_A + 1$ and $2k_B + 1$, respectively) to obtain the approximations $\widetilde{U}_A(t)$ and $\widetilde{U}_B(t)$, which we combine with an order $2k + 1$ splitting formula. The resulting total number of queries N for simulating $H = A + B$ satisfies

$$N \leq 8m' 5^{k+k_A-2} \max\{n_A, n\} + 4(m - m')5^{k+k_B-2} \max\{n_B, n\}, \tag{4.7}$$

   where

   - $n \geq \|A\| t \, (16e\|B\|t/\varepsilon)^{1/2k} \, \frac{8e}{5} \left(\frac{5}{3}\right)^k$ for $\|A\| \geq \|B\|$,

- $n_A = m'\|H_1\|t\left(\frac{64e}{5}\, m'\|H_2\|t/\varepsilon\right)^{1/2k_A} 7e\left(\frac{5}{3}\right)^{k_A-k}$,
- $n_B = (m-m')\|H_{m'+1}\|t\left(\frac{64e}{5}\,(m-m')\|H_{m'+2}\|t/\varepsilon\right)^{1/2k_B} 14e\left(\frac{5}{3}\right)^{k_B-k}$.

The form of the cost bound (4.7) is similar to that of (4.2), but with refined cost dependence. Roughly speaking, the two terms of the cost bound above correspond to the cost of simulating the Hamiltonians in the two groups forming A and B, respectively, plus some partitioning/recombining overhead that is captured by the maximum function.

The novelty of the algorithm is that it uses substantially fewer exponentials to simulate Hamiltonians of small norm, relative to the number of exponentials required for Hamiltonians of much larger norm, while maintaining the desired accuracy. In this respect, different time slices are chosen adaptively to simulate Hamiltonians in different groups. As a result, it is possible to use few exponentials to simulate a large number of Hamiltonians H_j of relatively small norm for longer time slices, and this reduces the overall simulation cost.

We emphasize that even though the cost bound (4.7) appears complicated, implementing the algorithms achieving this bound is straightforward, with similar implementation details to those of splitting formulas; see e.g. (4.6).

Items 3 and 4 below illustrate the impact of the divide and conquer approach, relative to earlier work, as the number m of terms grows and becomes huge. Item 5 estimates the practical advantage of the divide and conquer approach for simulating the electronic Hamiltonian.

3. For the case $k = k_A = k_B = O(1)$, and assuming that a large number of Hamiltonians have very small norm such that $(m-m')\|H_{m'+1}\| \leq m'\|H_2\|$, we can select n so that $n_A \geq n \geq n_B$ and

$$\begin{aligned} N &= O\left(m'^{\,2+1/2k}\,\|H_1\|t\,(\|H_2\|t/\varepsilon)^{1/2k}\right) \\ &+ O\left((m-m')^{1+1/2k}\,m'\,\|H_1\|t\,(\|H_{m'+1}\|t/\varepsilon)^{1/2k}\right). \end{aligned}$$

In particular, when a relatively small number of H_j form A so that $m' = O(m^a)$, and the remaining H_j have small norms in the sense that $(m-m')\|H_{m'+1}\|/\|H_2\| = O(m^b)$, for $0 \leq b \leq a < 1$, we have a speedup over the number of queries in (4.2) given by

$$\frac{N}{N_{prev}} = O\left(\frac{1}{m^{(1-a)+(1-b)/2k}}\right),$$

independently of t, ε, where N_{prev} denotes the upper bound shown in (4.2) with the same k. Observe that this quantity goes to 0 as $m \to \infty$.

4. In [29, 65, 186], it is shown how for splitting methods, the order of the splitting formula may be selected "optimally" such that the respective cost bounds are minimized. We show how optimal parameters k^*, k_A^*, and k_B^* may be similarly selected for our algorithms. Let N_{prev}^* and N^* be the resulting numbers of queries for the algorithm in [186] and for our algorithm, respectively. We show conditions for a strong speedup over [186] in the sense that
$$\frac{N^*}{N_{prev}^*} \xrightarrow[m\to\infty]{} 0 \qquad \text{for fixed } t, \varepsilon.$$

5. We apply our algorithm to the approximate electronic Hamiltonian (4.1) of quantum chemistry. Let $\mathcal{N}$ be the number of single-particle basis functions. The number of Hamiltonians in (4.1) is $\Theta(\mathcal{N}^4)$. We can assume that the largest Hamiltonian norm in the sum is constant. It is known that in practical cases a large number of terms have very small norm [18, 139, 167]. This allows us to dramatically improve the simulation cost. Table 4.1 presented at the beginning of this chapter illustrates this point by comparing our technique to others. Recall that the important complexity parameter is $\mathcal{N}$; we express the cost with respect to $\mathcal{N}$ in the table, assuming t, ε are fixed.

   Our cost estimates of $\mathcal{N}^5 - \mathcal{N}^7$ are consistent with empirical studies indicating that previous cost and error estimates may be overly conservative for practical applications [193].

   We emphasize that standard circuits implementing the evolution under the individual terms in (4.1) can be incorporated into our algorithm directly to yield its gate level implementation. For example, one can use the circuits in [246].

In the remainder of this chapter, we give our approach and algorithms in detail and derive the above results. Several of the more involved proofs are deferred to Appendix D. The results of this chapter can also be found in [113].

4.2 Preliminary Analysis

Our goal is to take the Hamiltonian simulation problem and partition it into a number of smaller and simpler Hamiltonian simulation problems, then solve each one of them, and combine the results.

The splitting should be customized to take advantage of the properties of each of the subproblems, yielding refined bounds for the overall simulation cost.

In certain applications, for instance in chemistry, Hamiltonians with extremely small norm can be discarded from the sum (4.4) as a preprocessing step, to the extent that this does not affect the desired accuracy. We formalize this idea in the following subsection.

4.2.1 Discarding Small Hamiltonians

Hamiltonians of very small norm relative to the accuracy ε may be discarded, and it suffices to consider the simulation problem for the remaining Hamiltonians. This may substantially reduce the cost, particularly for problems where ε is not arbitrarily small.

Proposition 1. *Let $H = A + B$ where H, A, B are Hamiltonians, $t > 0$, and $\varepsilon > 0$. If*

$$\|B\|t \leq \varepsilon/2, \tag{4.8}$$

and $\widetilde{U}$ is such that $\|e^{-iAt} - \widetilde{U}\| \leq \varepsilon/2$ then $\|e^{-iHt} - \widetilde{U}\| \leq \varepsilon$.

The proof of the proposition is shown in Appendix D. Thus, when the conditions of the proposition are satisfied, simulating A with error $\varepsilon/2$ implies the simulation of H with error ε.

Remark 4. *Equation (4.8) implies that the aggregate norm of the discarded Hamiltonians must be small, not just the norms of the discarded Hamiltonians themselves. Generally, Hamiltonians cannot be discarded without considering how many they are and the magnitudes of the other problem parameters.*

In practical applications a large number of "negligible" Hamiltonians are sometimes discarded, often using heuristics. For example, in quantum chemistry, an ad hoc fixed cut-off parameter, say 10^{-10}, is used [126]. (For applications such as eigenvalue estimation, a relatively large error can be tolerated for Hamiltonian simulation [173].) However, in general the effect of discarding terms must be accounted for in the error analysis.

We will assume that possible discarding of Hamiltonians according to Proposition 1 may have happened as a preprocessing step. Our results and proof techniques do not depend on whether Hamiltonians have been discarded or not. Thus, from this point on m will refer to the total number of Hamiltonians that we consider as input for our algorithms.

4.2.2 Recursive Lie-Trotter Formulas

Suppose the number m of Hamiltonians is large, and we are given a partition as

$$H = A + B := (H_1 + \cdots + H_{m'}) + (H_{m'+1} + \cdots + H_m). \tag{4.9}$$

We consider partitions into two groups to make the ideas of this section clear; it is straightforward to extend to an arbitrary number of groups μ. As A and B are themselves Hamiltonians, we may apply the Lie-Trotter formula (see equation (D.1) in Appendix D) with respect to them to give

$$\lim_{n\to\infty} (e^{-iAt/n} e^{-iBt/n})^n = e^{-iHt}. \tag{4.10}$$

Thus, we see that if we are able to approximate $e^{-iAt/n}$ and $e^{-iBt/n}$ then we should be able to combine the approximations as in (4.10) to approximate e^{-iHt}.

Indeed, we can again apply the Lie-Trotter formula (D.1) to each $e^{-iAt/n}$ and $e^{-iBt/n}$ to yield the *Recursed Lie-Trotter formula*

$$\lim_{\alpha,\beta,n\to\infty} \Big((e^{-iH_1 t/\alpha n} \dots e^{-iH_{m'} t/\alpha n})^\alpha (e^{-iH_{m'+1} t/\beta n} \dots e^{-iH_m t/\beta n})^\beta \Big)^n = e^{-iHt}, \tag{4.11}$$

where the limits may been taken in any order; see Appendix D for the proof.

Compared to (D.1), there are now three parameters n, α, β in (4.11) which reduce the error of the truncated product approximation as they are increased. Suppose $\|H_1\| \gg \|H_\ell\|$ for some $1 \le \ell \ll m$; then, grouping the largest Hamiltonians in A and the remaining Hamiltonians in B, it follows that we may want to take $\alpha > \beta$ as to reduce the overall error, while keeping β relatively small to reduce the overall cost. We will shortly derive divide and conquer simulation algorithms based on splitting formulas which will take three parameters k, k_A, k_B specifying the order of each formula. Thus we may use a high order splitting formula for A and a lower order splitting formula (and also larger time slices) for B, without compromising the error and such that the overall cost is reduced.

We remark that generalizing (4.11) to more than two groups of Hamiltonians gives a Trotter step parameter α_i for each group. Alternatively, this formula could be recursed deeper by further decomposing A and B into subgroups of Hamiltonians and again applying (D.1). Finally, the ideas of this subsection are easily generalized from the Trotter approximation to higher order formulas.

4.2.3 Combining Different Simulation Methods

We now describe our approach generally. Consider a Hamiltonian H as in (4.4, 4.5), and let $U = e^{-iHt}$. Assume the H_j have been partitioned into $\mu = O(1)$ disjoint groups, where we denote by $A_1, \dots, A_\mu$ the sums of the Hamiltonians in the respective groups. We are not concerned with how the partitioning is done at this point. As we will see later, the partitioning can be done adaptively and follows from general cost estimates. In practice, small values of μ may suffice and we'll see such an example in Section 4.4.

Then $H = A_1 + \cdots + A_\mu$. Assume the A_j have been indexed so that $\|A_1\| \geq \|A_2\| \geq \dots \|A_\mu\|$. Suppose we divide the simulation time t into intervals $\Delta t = t/n$, $n \in \mathbb{N}$; we will show how to select n later. Applying a splitting formula of order $2k+1$ with respect to this partition yields the operator

$$\widehat{U} := (S_{2k}(A_1, \dots, A_\mu, t/n))^n = \left(\prod_{\ell=1}^{N_{k,\mu}} e^{-iA_{j_\ell} t_\ell / n} \right)^n, \qquad j_\ell \in \{1, \dots, \mu\}, \quad \sum_{\ell=1}^{N_{k,\mu}} t_\ell = \mu t, \tag{4.12}$$

where $N_{k,\mu} = (2\mu - 1)5^{k-1}$, $|t_\ell| \leq t/n$, and $S_{2k}(A_1, \dots, A_\mu, t/n)$ is given in (D.3). Then, if we have algorithms $\widetilde{U}_{A_j}(\tau)$ to simulate (approximately) each exponential $e^{-iA_j \tau}$ in the right-hand side above, we can substitute them into (4.12) and obtain the approximation

$$\widetilde{U} := (\widetilde{S}_{2k}(A_1, \dots, A_\mu, t/n))^n = \left(\prod_{\ell=1}^{N_{k,\mu}} \widetilde{U}_{A_{j_\ell}}(t_\ell / n) \right)^n, \qquad j_\ell \in \{1, \dots, \mu\}, \quad \sum_{\ell=1}^{N_{k,\mu}} t_\ell = \mu t. \tag{4.13}$$

We emphasize $\widetilde{S}_{2k}(A_1, \dots, A_\mu, t/n)$ is constructed by expanding $S_{2k}(A_1, \dots, A_\mu, t/n)$ as an ordered product of exponentials $e^{-iA_{j_\ell} t_\ell}$ and replacing each $e^{-iA_{j_\ell} t_\ell}$ with $\widetilde{U}_{A_{j_\ell}}(t_\ell)$. The precise ordering of the product and the values t_ℓ are obtained from the particular choice of the splitting formula of order $2k+1$; see [214, 215]. For example, for $k = 1$, this gives

$$\widetilde{S}_2(A_1, \dots, A_\mu, t/n) = \widetilde{U}_{A_1}(t/2n) \dots \widetilde{U}_{A_{\mu-1}}(t/2n) \widetilde{U}_{A_\mu}(t/n) \widetilde{U}_{A_{\mu-1}}(t/2n) \dots \widetilde{U}_{A_1}(t/2n). \tag{4.14}$$

In principle, any available method may be used to implement the approximations $\widetilde{U}_{A_j}$, with the possibility of using different subroutines for different j.

We bound the overall error by

$$\|U - \widetilde{U}\| \leq \|U - \widehat{U}\| + \|\widehat{U} - \widetilde{U}\|. \tag{4.15}$$

We refer to $\|U - \widehat{U}\|$ and $\|\widehat{U} - \widetilde{U}\|$ as the **first-step error** and **second-step error**, respectively. Clearly, if both error terms are $O(\varepsilon)$, then so is the overall error $\|U - \widetilde{U}\|$.

The first-step error depends only on the splitting formula used at the first step, and is independent of the subroutines used to simulate each group at the second step. We have

$$\|U - \widehat{U}\| = \|(e^{-iHt/n})^n - (S_{2k}(A_1, \ldots, A_\mu, t/n))^n\| \le n\|e^{-iHt/n} - S_{2k}(A_1, \ldots, A_\mu, t/n)\|, \tag{4.16}$$

where $\|e^{-iHt/n} - S_{2k}(A_1, \ldots, A_\mu, t/n)\|$ is the error of S_{2k} over a single time slice. Following the approach of [186] (see equation (D.5)) for the simulation of a sum of μ-many Hamiltonians with accuracy $\varepsilon/2$ at the first step, we define the quantity

$$M = \left(\frac{4e\mu t\|A_2\|}{\varepsilon/2}\right)^{1/2k} \frac{4e\mu}{3} \left(\frac{5}{3}\right)^{k-1}, \tag{4.17}$$

which gives the first-step time slice size as $\Delta t := (M\|A_1\|)^{-1}$. The number of first-step time slices is $n = \lceil M\|A_1\|t \rceil$. Observe that the final time slice may be smaller than Δt. With this in mind, for simplicity we assume here that $M\|A_1\|t$ is an integer.

The second-step error is

$$\begin{aligned} \|\widehat{U} - \widetilde{U}\| &= \|S_{2k}(A_1, \ldots, A_\mu, t/n)^n - \widetilde{S}_{2k}(A_1, \ldots, A_\mu, t/n)^n\| \\ &\le n \left\| \prod_{\ell=1}^{N_{k,\mu}} e^{-iA_{j_\ell} t_\ell / n} - \prod_{\ell=1}^{N_{k,\mu}} \widetilde{U}_{A_{j_\ell}}(t_\ell/n) \right\| \\ &\le n \sum_{\ell=1}^{N_{k,\mu}} \|e^{-iA_{j_\ell} t_\ell/n} - \widetilde{U}_{A_{j_\ell}}(t_\ell/n)\| \end{aligned} \tag{4.18}$$

Hence, a sufficient condition for $\|\widehat{U} - \widetilde{U}\| \le \varepsilon/2$ is that the error of each stage satisfies

$$\|e^{-iA_{j_\ell} t_\ell/n} - \widetilde{U}_{A_{j_\ell}}(t_\ell/n)\| \le \frac{\varepsilon}{2N_{k,\mu} n}.$$

Assume the cost $N_{j_\ell} = N(A_{j_\ell}, t_\ell/n)$ of each simulation subroutine $\widetilde{U}_{A_{j_\ell}}(t_\ell/n)$ is expressed in terms of the number of exponentials of the form $e^{-iH_j z}$, where the H_j belong to the group forming A_{j_ℓ}, for suitable values $z \in \mathbb{R}$. The total simulation cost is the number of time slices n times the cost per time slice. The latter is $\sum_{\ell=1}^{N_{k,\mu}} N_{j_\ell}$, and therefore the total simulation cost is

$$N = n \cdot \left(\sum_{\ell=1}^{N_{k,\mu}} N_{j_\ell} \right). \tag{4.19}$$

4.3 Divide and Conquer Splitting Formulas

Consider again a Hamiltonian $H = \sum_{j=1}^{m} H_j$ as in (4.4, 4.5), partitioned into two groups $H = A + B$ as in (4.9). The two algorithms we present are illustrated in Figure 4.1. Algorithm 1 is a special case of Algorithm 2. Both algorithms, like splitting formulas, result in an ordered product of exponentials $\widetilde{U} = e^{-iH_{j_1}t_1}e^{-iH_{j_2}t_2}\dots e^{-iH_{j_N}t_N}$, $|t_\ell| \le t$. The difference between our algorithms is that Algorithm 1 uses $k = 1$ in the first step, while Algorithm 2 considers arbitrary k. Even though this difference might appear minor, the analysis of Algorithm 2 is much more complicated. Algorithm 1 is simpler to understand and implement, while Algorithm 2 is more general, offering one the possibility to reduce the number of exponentials by selecting $k > 1$.

4.3.1 Algorithm 1

Algorithm 1 follows the construction of Section 4.2.3 for the general case, applied to the partition $H = A + B$. At the **first step**, applying the Strang splitting formula ($k = 1$) gives the operators

$$\widehat{U} := (S_2(A, B, \Delta t))^n = (e^{-iA\Delta t/2}e^{-iB\Delta t}e^{-iA\Delta t/2})^n, \tag{4.20}$$

where $\Delta t = t/n$ and we will define n below. For the **second step**, Algorithm 1 approximates the operators $e^{-iA\Delta t/2}$ and $e^{-iB\Delta t}$ using different high-order splitting formulas $\widetilde{U}_A(\Delta t/2)$ and $\widetilde{U}_B(\Delta t)$, of orders $2k_A + 1$ and $2k_B + 1$, respectively. This yields the overall approximation $\widetilde{U}$ of $U = e^{-iHt}$ which is defined by

$$\widetilde{U} \;:=\; (\widetilde{S}_2(A, B, \Delta t))^n = \left(\widetilde{U}_A(\Delta t/2)\widetilde{U}_B(\Delta t)\widetilde{U}_A(\Delta t/2)\right)^n. \tag{4.21}$$

Note that in general $\widetilde{U}_A(\Delta t/2)\widetilde{U}_A(\Delta t/2) \neq \widetilde{U}_A(\Delta t)$.

As in [186] let

$$\mathcal{H}_j := \begin{cases} H_j/\|H_1\| & (1 \le j \le m') \\ H_j/\|H_{m'+1}\| & (m' < j \le m). \end{cases}$$

For splitting formulas, such a rescaling of the Hamiltonian norms is equivalent to a rescaling of the respective group simulation times, i.e., $S_{2k_A}(\mathcal{H}_1, \dots, \mathcal{H}_{m'}, \|H_1\|\tau) = S_{2k_A}(H_1, \dots, H_{m'}, \tau)$ and $S_{2k_B}(\mathcal{H}_{m'+1}, \dots, \mathcal{H}_m, \|H_{m'+1}\|\tau) = S_{2k_B}(H_{m'1}, \dots, H_m, \tau)$. Observe that the Hamiltonians in A and B are rescaled by different quantities, which leads to different simulation times for each.

The time slice sizes for simulating $\widetilde{U}_A(\Delta t/2)$ and $\widetilde{U}_B(\Delta t)$ are $1/M_A$ and $1/M_B$, respectively, where M_A and M_B are defined below. Thus, applying splitting formulas of orders $2k_A+1$ and $2k_B+1$ for U_A and U_B, respectively, gives

$$\widetilde{U}_A(\Delta t/2) := S_{2k_A}(\mathcal{H}_1,\dots,\mathcal{H}_{m'},1/M_A)^{\lfloor M_A\|H_1\|\Delta t/2\rfloor} S_{2k_A}(\mathcal{H}_1,\dots,\mathcal{H}_{m'},\delta_A/M_A), \quad (4.22)$$

$$\widetilde{U}_B(\Delta t) := S_{2k_B}(\mathcal{H}_{m'+1},\dots,\mathcal{H}_m,1/M_B)^{\lfloor M_B\|H_{m'+1}\|\Delta t\rfloor} S_{2k_B}(\mathcal{H}_{m'+1},\dots,\mathcal{H}_m,\delta_B/M_B). \quad (4.23)$$

Since we have effectively rescaled the simulation times by dividing by the respective largest Hamiltonian norms, we are actually subdividing an interval of size $\|H_1\|\Delta t/2$ into $\lceil M_A\|H_1\|\Delta t/2\rceil$ intervals of length at most $1/M_A$ for the simulation of $\widetilde{U}_A(\Delta t/2)$, and into $\lceil M_B\|H_{m'+1}\|\Delta t\rceil$ intervals of length at most $1/M_B$ for $\widetilde{U}_B(\Delta t)$. Clearly, the last of these subintervals in either case may have length less than $1/M_A$ or $1/M_B$, respectively. In such a case, the length of the last subinterval is equal to δ_A/M_A or δ_B/M_B, with $\delta_A := M_A\|H_1\|\Delta t/2 - \lfloor M_A\|H_1\|\Delta t/2\rfloor$ and $\delta_B := M_B\|H_{m'+1}\|\Delta t - \lfloor M_B\|H_{m'+1}\|\Delta t\rfloor$, respectively. That is the reason why we have taken the floors of the exponents in the first factors of (4.22) and (4.23).

For the simulation error, from (4.15) we have $\|U-\widetilde{U}\| \le \|U-\widehat{U}\| + \|\widehat{U}-\widetilde{U}\|$. Thus, to guarantee $\|U-\widetilde{U}\|\le\varepsilon$, we require $\|U-\widehat{U}\|\le\varepsilon/2$ and $\|\widehat{U}-\widetilde{U}\|\le\varepsilon/2$.

We consider each error term separately. The first error term is independent of the algorithms used for $\widetilde{U}_A$ and $\widetilde{U}_B$, and results only from the first-step Strang splitting and time slice size $\Delta t = t/n$, $n\in\mathbb{N}$. From Lemma 5, shown in Appendix D, we have

$$\|U-\widehat{U}\| \le \frac{2}{3}t\Delta t^2\|A\|\|B\|\cdot\max\{\|A\|,\|B\|\}. \quad (4.24)$$

From (4.19) the cost of our algorithm is proportional to n, and therefore we would like to select it to be as small as possible. Setting the right hand side of the equation above to $\varepsilon/2$ we obtain

$$n \ge \sqrt{4t^3\,\|A\|\|B\|\max\{\|A\|,\|B\|\}/3\varepsilon}. \quad (4.25)$$

For instance, when $\|A\|\ge\|B\|$, from the triangle inequality bounds $\|A\|\le m'\|H_1\|$ and $\|B\|\le(m-m')\|H_{m'+1}\|$, to obtain $\|U-\widehat{U}\|\le\varepsilon/2$ it therefore suffices to select n as

$$n := \left\lceil \sqrt{4/3}\,m'\|H_1\|t\sqrt{(m-m')\|H_{m'+1}\|t/\varepsilon}\right\rceil. \quad (4.26)$$

Now consider the second error term. As $\|S_2\| = \|\widetilde{S}_2\| = 1$, we have (cf. eq. (4.18))

$$\begin{aligned}\|\widehat{U} - \widetilde{U}\| &= \|S_2(A,B,\Delta t)^n - \widetilde{S}_2(A,B,\Delta t)^n\| \le n\|S_2(A,B,\Delta t) - \widetilde{S}_2(A,B,\Delta t)\| \\ &\le n\|e^{-iA\Delta t/2}e^{-iB\Delta t}e^{-iA\Delta t/2} - \widetilde{U}_A(\Delta t/2)\widetilde{U}_B(\Delta t)\widetilde{U}_A(\Delta t/2)\| \\ &\le n\left(2\|e^{-iA\Delta t/2} - \widetilde{U}_A(\Delta t/2)\| + \|e^{-iB\Delta t} - \widetilde{U}_B(\Delta t)\|\right),\end{aligned}$$

where the terms $\|e^{-iA\Delta t/2} - \widetilde{U}_A(\Delta t/2)\|$ and $\|e^{-iB\Delta t} - \widetilde{U}_B(\Delta t)\|$ bound the error of each $\widetilde{U}_A(\Delta t/2)$ and $\widetilde{U}_B(\Delta t)$. Hence, to ensure $\|\widehat{U} - \widetilde{U}\| \le \varepsilon/2$, we require

$$\|e^{-iA\Delta t/2} - \widetilde{U}_A(\Delta t/2)\| \le \varepsilon/8n \quad \text{and} \quad \|e^{-iB\Delta t} - \widetilde{U}_B(\Delta t)\| \le \varepsilon/4n. \tag{4.27}$$

The quantity $M_A = M_A(k_A)$ is defined by applying (D.5) to the simulation of A with time $t/2n$ and error at most $\varepsilon/8n$, to obtain

$$M_A := \left(\frac{4em'(t/2n)\|H_2\|}{(\varepsilon/8n)}\right)^{1/2k_A} \frac{4em'}{3}\left(\frac{5}{3}\right)^{k_A-1} = \left(\frac{16em't\|H_2\|}{\varepsilon}\right)^{1/2k_A} \frac{4em'}{3}\left(\frac{5}{3}\right)^{k_A-1}.$$

Remarkably, observe that the factors of n have canceled, i.e., the time interval size for each application of $\widetilde{U}_A(\Delta t/2)$ depends only on the original problem time and error parameters and not on the number of time slices n we subdivided t into. Further note that when $16em't\|H_2\| \le \varepsilon$, then M_A is bounded from above independently of ε. This means that we are dealing with an easy problem for the simulation of A, so the interesting case is when $16em't\|H_2\| > \varepsilon$, and we will consider this case from now on. Similar considerations apply to the simulation of B.

To bound the cost of each $\widetilde{U}_A(\Delta t/2)$, we apply [186, Thm. 1]. Thus, the number of exponentials N_A required for each application of $\widetilde{U}_A(\Delta t/2)$ satisfies

$$N_A \le (2m'-1)5^{k_A-1}\lceil M_A\|H_1\|\Delta t/2\rceil.$$

We have already mentioned that the quantity $\lceil M_A\|H_1\|\Delta t/2\rceil$ gives the number of subintervals of length at most $1/M_A$ that each time slice $\|H_1\|\Delta t/2$ is subdivided. When the ceiling function argument is at most one, no sub-division is necessary. Then it may be possible to reduce the cost further by decreasing k_A.

Now consider $\widetilde{U}_B(\Delta t)$. For the simulation of B for time $\Delta t = t/n$ and error at most $\varepsilon/4n$, we set $M_B = M_B(k_B)$ as in (D.5) to obtain

$$\begin{aligned}M_B &:= \left(\frac{4e(m-m')(t/n)\|H_{m'+2}\|}{(\varepsilon/4n)}\right)^{1/2k_B} \frac{4e(m-m')}{3}\left(\frac{5}{3}\right)^{k_B-1} \\ &= \left(\frac{16e(m-m')t\|H_{m'+2}\|}{\varepsilon}\right)^{1/2k_B} \frac{4e(m-m')}{3}\left(\frac{5}{3}\right)^{k_B-1}.\end{aligned}$$

Once again, M_B is independent of n. As above, the interesting case is $16e(m-m')\|H_{m'+2}\|t > \varepsilon$, because otherwise M_B would be independent of ε and the problem would be easy. Applying again [186, Thm. 1], the number N_B of exponentials for each application of $\widetilde{U}_B(\Delta t)$ satisfies

$$N_B \quad \leq \quad (2(m-m')-1)5^{k_B-1}\lceil M_B\|H_{m'+1}\|\Delta t\rceil .$$

In this case the quantity $\lceil M_B\|H_{m'+1}\|\Delta t\rceil$ gives the number of subintervals of length at most $1/M_B$ that each time interval of size $\|H_{m'+1}\|\Delta t$ is subdivided.

We may now bound the total cost of our algorithm, i.e. bound the number N of exponentials of the form $e^{-H_j t_\ell}$, $j \in \{1,\dots,m\}$, that are used to construct $\widetilde{U}$. From (4.19), we have

$$\begin{aligned} N &= n\cdot(2N_A+N_B) \\ &\leq n\cdot\left(4m'5^{k_A-1}\left\lceil M_A\|H_1\|\frac{t}{2n}\right\rceil + 2(m-m')5^{k_B-1}\left\lceil M_B\|H_{m'+1}\|\frac{t}{n}\right\rceil\right). \end{aligned} \tag{4.28}$$

We summarize the results for Algorithm 1 in the following proposition.

Proposition 2. *Let $H=\sum_{i=1}^m H_i$, $\|H_1\|\geq\|H_2\|\geq\cdots\geq\|H_m\|$, $m\geq 2$, with given partition $H=A+B$, $A=\sum_{i=1}^{m'} H_i$ and $B=\sum_{i=m'+1}^{m} H_i$. Let $t>0$ and $1\geq\varepsilon>0$, and assume $16em'\|H_2\|t\geq\varepsilon$ and $16e(m-m')\|H_{m'+2}\|t\geq\varepsilon$. Let $n\in\mathbb{N}$ such that*

$$n \geq \sqrt{4\,t^3\,\|A\|\|B\|\|C\|/3\varepsilon}, \tag{4.29}$$

where $\|C\| = \max\{\|A\|,\|B\|\}$. For any $k_A,k_B\in\mathbb{N}$, define the quantities

$$M_A = \left(\frac{16em'\|H_2\|t}{\varepsilon}\right)^{1/2k_A}\frac{4em'}{3}\left(\frac{5}{3}\right)^{k_A-1}$$

$$M_B = \left(\frac{16e(m-m')\|H_{m'+2}\|t}{\varepsilon}\right)^{1/2k_B}\frac{4e(m-m')}{3}\left(\frac{5}{3}\right)^{k_B-1},$$

and let $\widetilde{U}$ be defined by (4.21). Then the number N of exponentials for the approximation of e^{-iHt} by $\widetilde{U}$ with accuracy ε is at most

$$N \leq 4m'5^{k_A-1}n\left\lceil M_A\|H_1\|\frac{t}{2n}\right\rceil + 2(m-m')5^{k_B-1}n\left\lceil M_B\|H_{m'+1}\|\frac{t}{n}\right\rceil. \tag{4.30}$$

For $x,y>0$, it is easy to show $x\lceil y/x\rceil \leq \max\{x,2y\}$. Thus we have the following corollary.

Corollary 1. *Let $n_A = M_A\|H_1\|t$ and $n_B = 2M_B\|H_{m'+1}\|t$. The bound to the number of exponentials of Proposition 2, equation (4.30), may be expressed as*

$$N \le 4m'5^{k_A-1} \cdot \max\{n_A, n\} \ + \ 2(m-m')5^{k_B-1} \cdot \max\{n_B, n\}. \tag{4.31}$$

Remark 5. *Observe that if $n_A, n_B \ge n$, then modulo constants (4.31) implies that the cost for simulating $H = A + B$ is upper bounded by the sum of the cost upper bounds for simulating A and B independently.*

Remark 6. *The simulation cost bound (4.31) is minimized with respect to k_A and k_B by selecting optimal values k_A^*, k_B^* such that $1 \le k_A^* \le k_A^{(max)}$ and $1 \le k_B^* \le k_B^{(max)}$, where*

$$\begin{aligned} k_A^{(max)} &= \left\lceil \sqrt{\tfrac{1}{2}\log_{25/3}(16e\, m'\|H_2\|t/\varepsilon)} \right\rceil, \\ k_B^{(max)} &= \left\lceil \sqrt{\tfrac{1}{2}\log_{25/3}(16e\,(m-m')\|H_{m'+2}\|t/\varepsilon)} \right\rceil. \end{aligned}$$

If $n \ge M_A(1)\|H_1\|t$, then k_A is optimally selected as $k_A^* = 1$. Alternatively, if $M_A(k_A^*)\|H_1\|t \ge n$, then $k_A^* = k_A^{(max)}$. Similar remarks apply for k_B, where instead of m', $\|H_2\|$, and M_A we use $(m-m')$, $\|H_{m'+2}\|$, and M_B. We formalize how to optimally select the splitting formula orders for the general case in Section 4.3.3.

It is relatively straightforward to extend Algorithm 1 to a partition of H into $\mu \ge 2$ groups $H = A_1 + \cdots + A_\mu$. We consider this for the more general Algorithm 2 in the next section.

4.3.2 Algorithm 2

Algorithm 2 generalizes Algorithm 1 by applying an arbitrary splitting formulas at its **first step** instead of specifically the Strang splitting formula; see Figure 4.1. The details and analysis of Algorithm 2 are similar to, but more complicated than, those of Algorithm 1. We state the main results, and provide the proofs in Appendix D.

We again consider the simulation of a partitioned Hamiltonian $H = A + B$, with $A = H_1 + \cdots + H_{m'}$, $B = H_{m'+1} + \cdots + H_m$. Just like in Algorithm 1, the **second step** of Algorithm 2 uses splitting formulas of orders $2k_A + 1$ and $2k_B + 1$ for the simulations of A and B, respectively, and combines the partial results using a splitting formula of order $2k + 1$.

Proposition 3. *Let $H = \sum_{i=1}^m H_i$, $\|H_1\| \geq \|H_2\| \geq \cdots \geq \|H_m\|$, $m \geq 2$, with given partition $H = A + B$, $A = \sum_{i=1}^{m'} H_i$ and $B = \sum_{i=m'+1}^m H_i$. Let $\|C\| := \max\{\|A\|, \|B\|\}$ and $\|D\| := \min\{\|A\|, \|B\|\}$. Assume $\|C\|t \geq 1$, $16em'\|H_2\|t \geq \varepsilon$, $16e(m-m')\|H_{m'+2}\|t \geq \varepsilon$, and $16e\|D\|t \geq \varepsilon$. For $k, k_A, k_B \in \mathbb{N}$, define the quantities*

- $n \geq \|C\|t\,(16e\|D\|t/\varepsilon)^{1/2k}\,\frac{8e}{5}\left(\frac{5}{3}\right)^k$,
- $n_A = m'\|H_1\|t\left(\frac{64e}{5}\,m'\|H_2\|t/\varepsilon\right)^{1/2k_A} 7e\left(\frac{5}{3}\right)^{k_A-k}$,
- $n_B = (m-m')\|H_{m'+1}\|t\left(\frac{64e}{5}\,(m-m')\|H_{m'+2}\|t/\varepsilon\right)^{1/2k_B} 14e\left(\frac{5}{3}\right)^{k_B-k}$.

and let $\widetilde{U}$ be defined by (4.13). Then the number N of exponentials for the approximation of e^{-iHt} by $\widetilde{U}$ with accuracy ε is at most

$$N \leq 8m'5^{k+k_A-2}\,\max\{n_A, n\} + 4(m-m')5^{k+k_B-2}\,\max\{n_B, n\} =: \eta(k, k_A, k_B). \quad (4.32)$$

Remark 7. *If $k = 1$, we recover the cost bound (4.31) of Algorithm 1, up to constant factors. Note that in some cases, e.g. when $\|D\|t/\varepsilon$ is large, even though we could use $k = 1$, selecting a value $k > 1$ may yield n that is substantially smaller than that shown in (4.29) in Proposition 2.*

Remark 8. *If any of the conditions $\|C\|t \geq 1$, $16em'\|H_2\|t \geq \varepsilon$, $16e(m-m')\|H_{m'+2}\|t \geq \varepsilon$, or $16e\|D\|t \geq \varepsilon$ are violated, then we end up with an easier simulation problem as ε is relatively large. So, in this sense, these conditions specify the interesting case.*

Algorithm 2 extends to the case where H is partitioned into $\mu \geq 2$ groups $H = A_1 + \cdots + A_\mu$. The algorithm is now specified by $\mu + 1$ parameters $\underline{k} = \{k, k_1, \ldots, k_\mu\} \in \mathbb{N}^{\mu+1}$. The overall approximation $\widetilde{U}$ of $U = e^{-iHt}$ becomes

$$\widetilde{U} \;\; := \;\; \left(\widetilde{S}_{2k}(A_1, A_2, \ldots A_\mu, \Delta t)\right)^n, \quad (4.33)$$

where $\widetilde{S}_{2k}(A_1, A_2, \ldots A_\mu, \Delta t)$ is constructed as in (4.13).

We summarize the results for this case in the following theorem, whose proof can be found in Appendix D. A partition of $H = \sum_{i=1}^m H_i$ to $H = \sum_{j=1}^\mu A_j$, $2 \leq \mu \ll m$, where each A_j is a sum of a subset of the H_i, is *disjoint* if each H_i is contained in a single A_j. Let each A_j contain m_j of the H_i, such that $\sum_{j=1}^\mu m_j = m$. We use $H_{(j,1)}$ to denote the largest Hamiltonian norm in a group

$$H_{(j,1)} = \max_{H_i \in A_j} \|H_i\|,$$

and similarly $H_{(j,2)}$ denotes the second largest Hamiltonian norm.

Theorem 2. *Let $H = \sum_{i=1}^{m} H_i$ be disjointly partitioned as $H = \sum_{j=1}^{\mu} A_j$, labeled such that $\|A_1\| \geq \|A_2\| \geq \cdots \geq \|A_j\|$. Let $t > 0$ and $1 \geq \varepsilon > 0$. Suppose $\mu\|A_1\|t \geq 1$, $\|A_2\|t \geq \varepsilon$, and $\mu m_j \|H_{(j,2)}\|t \geq \varepsilon$. For $k, k_1, \ldots, k_\mu \in \mathbb{N}$, let $n \in \mathbb{N}$ be such that*

$$n(k) \geq \mu\|A_1\|t \left(\frac{8e\mu\|A_2\|t}{\varepsilon}\right)^{1/2k} \frac{4e}{5}\left(\frac{5}{3}\right)^{k}. \tag{4.34}$$

and define the quantities

$$n_{A_j}(k, k_j) = m_j\|H_{(j,1)}\|t\left(\frac{32e}{5}\frac{\mu m_j\|H_{(j,2)}\|t}{\varepsilon}\right)^{1/2k_j} 7e\left(\frac{5}{3}\right)^{k_j - k}, \qquad j = 1, \ldots, \mu. \tag{4.35}$$

Consider $\widetilde{U}$ to be defined by (4.33). The number N of exponentials for the approximation of e^{-iHt} by $\widetilde{U}$ with accuracy ε is at most

$$N \leq 8\sum_{j=1}^{\mu} 5^{k+k_j-2} m_j \max\{n(k), n_{A_j}(k, k_j)\} =: \eta(\underline{k}). \tag{4.36}$$

Remark 9. *The way the Hamiltonians are grouped will influence the upper bound (4.36). Ideally, the formation of the groups should minimize this upper bound. Roughly speaking, Hamiltonians of relatively large norm should be put in groups of relatively small cardinality.*

Remark 10. *The bound for the number of exponentials holds under general conditions and does not depend on how the partitioning of the Hamiltonians into groups is performed. Finding parameters that minimize (4.36) is a separate task, which is to be carried out on a classical computer.*

4.3.3 Selecting the Order of the Splitting Formulas

For any partitioning of the Hamiltonians into μ groups we show how to determine the order of the splitting formulas. The parameters $k_1, \ldots, k_\mu$ allow splitting formulas of different orders to be used for the Hamiltonians in each group. The parameter k determines the order of the splitting formula used in the first algorithm step. Ideally we want to find the optimal parameters $k^*, k_1^*, \ldots, k_\mu^*$ that minimize the simulation cost bound (4.36), which takes the value $\eta^* := \eta(k^*, k_1^*, \ldots, k_\mu^*)$. This expression is complicated and to simplify matters we provide sharp upper bounds $k^{(max)}$, $k_1^{(max)}, \ldots, k_\mu^{(max)}$ to the optimal values. The upper bounds turn out to be turn out to be very slowly growing functions (sublogarithmic in the problem parameters) which means that for all practical instances the parameters $k^*, k_1^*, \ldots, k_\mu^*$ yielding the lowest cost upper bound can be found quickly by exhaustive search.

Proposition 4. *The simulation cost bound (4.36) of Theorem 2 is minimized by integers $k^*, k_1^*, \ldots, k_\mu^*$ satisfying*

$$1 \le k^* \le k^{(max)}, \quad 1 \le k_j^* \le k_j^{(max)} \quad j = 1, \ldots, \mu, \tag{4.37}$$

where

$$k^{(max)} := \max\left\{ \text{round}\left(\sqrt{\frac{1}{2} \log_{25/3} \frac{8e\mu \|A_2\| t}{\varepsilon}} \right), 1 \right\} \tag{4.38}$$

and

$$k_j^{(max)} := \max\left\{ \text{round}\left(\sqrt{\frac{1}{2} \log_{25/3} \frac{32e\mu m_j \|H_{(j,2)}\| t}{5\varepsilon}} \right), 1 \right\}, \qquad j = 1, \ldots, \mu. \tag{4.39}$$

Proof. Let the functions $g(k) := 5^k n(k)$ and $h_j(k_j) := 5^{k+k_j} n_{A_j}(k, k_j)/3^k$. Note that k cancels out in the latter case so $h_j(k_j)$ is a univariate function. Consider minimizing $g(\cdot)$ and $h_j(\cdot)$ independently. For $g(k)$, setting its derivative to zero gives

$$2k^2 \ln \frac{25}{3} - \ln \frac{8e\mu \|A_2\| t}{\varepsilon} = 0,$$

which gives $k^{(max)}$ as in (4.38). Repeating this argument for $h_j(k_j)$ gives $k_j^{(max)}$ as in (4.39). Since $g(\cdot)$ and $h_j(\cdot)$ are log-convex functions [47], the values $k^{(max)}$ and $k_j^{(max)}$ give the respective minima. Next, observe that we may rewrite the right-hand side of (4.36) as

$$\eta(\underline{k}) = \frac{8}{25} \sum_{j=1}^{\mu} m_j \max\{5^{k+k_j} n(k), 5^{k+k_j} n_{A_j}(k, k_j)\} = \frac{8}{25} \sum_{j=1}^{\mu} m_j \max\{5^{k_j} g(k), 3^k h(k_j)\}. \tag{4.40}$$

Now assume $k_1, \ldots, k_\mu$ are arbitrary but fixed. Then $\eta(k, k_1, \ldots, k_\mu) \ge \eta(k^{(\max)}, k_1 \ldots, k_\mu)$ for $k > k^{(\max)}$ since the arguments of the maximum function cannot decrease. By a similar argument, for $k_j > k_j^{(max)}$ $\eta(k, k_1, \ldots, k_j, \ldots k_\mu) \ge \eta(k, k_1 \ldots, k_j^{(\max)}, \ldots k_\mu)$. Therefore, the minimizers $k^*, k_1^*, \ldots, k_\mu^*$ of (4.36) satisfy $k^* \le k^{(max)}$ and $k_j^* \le k_j^{(max)}$, for $j = 1, \ldots, \mu$. □

Remark 11. *Equation (4.39) shows that small group cardinality and small Hamiltonian norms reduce the order of the splitting formula that suffices for the simulation of a given group.*

Indeed, from the arguments of the maximum function in (4.36), we have that if $n(k) \ge n_{A_j}(k, k_j)$ for all k and some j, then $k_j^* = 1$. On the other hand, if $n(k) < n_{A_j}(k, k_j)$ for $k \le k^{(max)}$ and some j, then $k_j^* = k_j^{(\max)}$. Thus, roughly speaking, Hamiltonians of small norm may be grouped and simulated with a low-order splitting formula (e.g., $k_j = 1$), whereas groups of Hamiltonians of large norm in general require higher order formulas to achieve the lowest simulation cost.

4.3.4 Speedup

We illustrate our results by showing the speedup of our algorithms relative to those in [186] for a number of cases. Generally, our approach is preferable when there is a disparity in the Hamiltonian norms and many of them are very small.

From [186], we have the number of exponentials is bounded as

$$N_{prev}(k) = O\left(m^2\|H_1\|t\left(\frac{mt\|H_2\|}{\varepsilon}\right)^{\frac{1}{2k}}\left(\frac{25}{3}\right)^k\right), \tag{4.41}$$

where k is the order of the splitting formula. Selecting $k = k^*$ as in (D.8) this becomes

$$N^*_{prev} = O\left(m^2\|H_1\|t\right) \cdot e^{2\sqrt{\frac{1}{2}\ln\frac{25}{3}\ln\frac{4emt\|H_2\|}{\varepsilon}}}. \tag{4.42}$$

Note that the second factor $e^{2\sqrt{\frac{1}{2}\ln\frac{25}{3}\ln\frac{4emt\|H_2\|}{\varepsilon}}} = O((mt\|H_2\|/\varepsilon)^\delta)$ for any $\delta > 0$. The explicit expressions for (4.41, 4.42) are shown in (D.7, D.9).

For simplicity, we consider $\mu = 2$, i.e., $H = A + B$ with $A = H_1 + H_2 + \cdots + H_{m'}$, $m' < m$, and B equal to the sum of the remaining Hamiltonians. The number of exponentials for Algorithm 2 is then shown in (4.32) in Proposition 3. Assume that $\|A\| \geq \|B\|$ and in addition that

$$(m - m')\|H_{m'+1}\| \leq m'\|H_2\|, \tag{4.43}$$

Note that the left-hand side of the inequality above is an upper bound to $\|B\|$, and the inequality relates that to the number of Hamiltonians forming A times the overall second largest Hamiltonian norm. This condition is easy to check in principle, and it holds especially in cases where the original Hamiltonians have quite disproportionate norms and have been partitioned accordingly.

Clearly, we have $\|A\| \leq m'\|H_1\|$ and $\|B\| \leq (m - m')\|H_{m'+1}\|$, and hence we may select the parameter n of Proposition 3 as

$$n(k) = \left\lceil m'\|H_1\|t\left(\frac{16e(m-m')\|H_{m'+1}\|t}{\varepsilon}\right)^{1/2k}\frac{8e}{5}\left(\frac{5}{3}\right)^k\right\rceil. \tag{4.44}$$

Recall the quantities k^*, k^*_A, k^*_B and $k^{(max)}, k^{(max)}_A, k^{(max)}_B$ shown in Proposition 4. For any algorithm parameters k, k_A, k_B, the cost bound of Proposition 3 satisfies $N \leq \eta^* \leq \eta(k, k_A, k_B)$, where η^* denotes the optimize cost bound $\eta(k^*, k^*_A, k^*_B)$ of Algorithm 2.

For different cases of the algorithm parameters we have the following speedups.

1. Comparison when all splitting formulas have the same order, i.e., $k = k_A = k_B$, $k = O(1)$:

   The cost bound (4.32) has the same dependence on t and ε as that of (4.41), so when we divide the two cost bounds to obtain the speedup the parameters t and ε cancel out. From Proposition 3, (4.43), and (4.44), we have $\max\{n_A, n\} = c_1 n_A$ and $\max\{n_B, n\} = c_2 n$, where $c_1, c_2 \geq 1$ are constants. Hence, (4.32) gives cost

$$\begin{aligned} N \leq \eta(k,k,k) &\leq 8m' 5^{2k-2} c_1 n_A + 4(m-m') 5^{2k-2} c_2 n \\ &\leq C \cdot \big(m'^2 \|H_1\| t \, (m' \|H_2\| t/\varepsilon)^{1/2k} \\ &+ (m-m') m' \|H_1\| t \, ((m-m') \|H_{m'+1}\| t/\varepsilon)^{1/2k} \big), \end{aligned}$$

   where C is a constant, and hence the speedup over [186] (with the same k) is

$$\frac{N}{N_{prev}(k)} = O\left(\left(\frac{m'}{m} \right)^{2+1/2k} \right) + O\left(\frac{m'}{m} \left(\frac{(m-m')\|H_{m'+1}\|}{m\|H_2\|} \right)^{1/2k} \right) \quad (4.45)$$

   for all ε, t. Therefore, the algorithm in [186] is slower than the one in this chapter by a factor proportional to a polynomial in m'/m, the degree of which is in the range $[1, 2.5]$. This is particularly important when $m' \ll m$.

2. Comparison to the cost of [186] with optimally chosen parameter:

   We use the previous case to derive a rough estimate. Observe that, for fixed k we have

$$\frac{N_{prev}(k)}{N^*_{prev}} = O(m\|H_2\|t/\varepsilon)^{1/2k}.$$

   Thus, again considering $k = k_A = k_B$, $k = O(1)$, we have

$$\begin{aligned} \frac{N}{N^*_{prev}} &\leq \frac{\eta(k,k,k)}{N_{prev}(k)} \frac{N_{prev}(k)}{N^*_{prev}} \qquad (4.46) \\ &= O\left(\left(\frac{m'}{m} \right)^2 \left(\frac{m'\|H_2\|t}{\varepsilon} \right)^{1/2k} \right) + O\left(\frac{m'}{m} \left(\frac{(m-m')\|H_{m'+1}\|t}{\varepsilon} \right)^{1/2k} \right), \end{aligned}$$

   which follows from (4.42) and (4.45). Therefore, for fixed $\|H_2\|$, t and ε, the algorithm in [186] with optimally chosen parameters remains slower than Algorithm 2 with arbitrary $k = k_A = k_B$. The speedup depends on a polynomial in m'/m, the degree of which is in the range $[1, 2]$.

   Clearly, optimally selecting k, k_A, and k_B can only improve the speedup.

3. Comparison among optimal methods, i.e., using the respective optimal splitting formulas:

   Assuming that all complexity parameters are fixed, with the exceptions of m and $m' = O(m^{5/6})$, we have
$$\frac{N}{N^*_{prev}} \le \frac{\eta^*}{N^*_{prev}} \xrightarrow[m\to\infty]{} 0, \tag{4.47}$$
   where η^* is the optimized cost bound of Algorithm 2. The proof is given in Appendix D.

   In this sense we achieve a strong speedup over [186].

4. Comparison when a significant number of Hamiltonians have very small norm relative to $\|H_2\|$:

   Recall that we are interested in simulation problems where a significant number of the Hamiltonians H_j are relatively small in norm, where existing simulation methods do not take advantage of this structure.

   We use two parameters $0 \le b \le a < 1$ to describe the relationship of $\|B\|$ and $\|A\|$. This approach has applications to problems such as the simulation of the electronic Hamiltonian, as we will see in the following section. Suppose
$$\|B\| \le (m-m')\|H_{m'+1}\|/\|H_2\| = O((m-m')^b) \qquad \text{for a given } b \in [0,1), \tag{4.48}$$
   i.e., not only do the H_j, $j > m'$, that form B have small norm individually, but $\|B\|$ is relatively small also. For example, we could have $(m-m') = 10^6$ and the Hamiltonians comprising B to have norms at most 10^{-4}, so that $\|B\| \le 10^6 \cdot 10^{-4} = 10^2$, i.e., $b \simeq 1/3$.

   Recall $1 \le m' < m$ because m' is the number of Hamiltonians forming A. Further suppose
$$m' = O(m^a) \qquad \text{for some } a \in [0,1). \tag{4.49}$$
   For the case $k = k_A = k_B = O(1)$ above (where the speedup is independent of ε and t), using these assumptions in (4.45) we obtain
$$\begin{aligned}\frac{\eta(k,k,k)}{N_{prev}(k)} &= O\left(\left(\frac{m^a}{m}\right)^{2+1/2k}\right) + O\left(\frac{m^a}{m}\left(\frac{(m-m')^b}{m}\right)^{1/2k}\right)\\ &= O\left(\frac{1}{m^{(1-a)+(1-b)/2k}}\right).\end{aligned}$$
   Similarly, for the case above where we obtain (4.46), using the new assumptions we obtain
$$\frac{N}{N^*_{prev}} = O\left(\frac{1}{m^{1-a-b/2k}}\right).$$

Therefore, selecting k such that the exponent of the denominator is positive yields a speedup the grows with m. In the next section we will use the parameters a and b to estimate the cost of our algorithms for practical instances of the electronic Hamiltonian.

We emphasize that for the speedup estimates derived in this section we have made many simplifications, and they are hence quite conservative. For practical problem instances, the speedups may be much greater than those indicated here.

4.4 Application to Quantum Chemistry

Solving difficult problems in quantum chemistry is viewed as a primary application of quantum computers. We apply our algorithms to simulate the electronic Hamiltonian, which describes molecular systems. Quantum algorithms for simulating the electronic Hamiltonian have applications to the calculation of electronic energies (i.e., the *electronic structure problem*), and also reaction rates, and other chemical properties [15, 146, 147, 236, 246].

Robust classical algorithms for this simulation exist (e.g., diagonalization), but in general they are intractable as their cost grows exponentially with the number of particles. Thus, large molecules are out of reach for classical computers [246]. On the other hand, quantum algorithms [15, 246] can efficiently simulate the second-quantized electronic Hamiltonian (4.1). There exist quantum algorithms for this simulation problem with cost polynomial in the number m of Hamiltonian terms. Unfortunately, the combination of the size of m and the degree of the polynomial makes the algorithms impractical in many cases of interest [167, 189, 224, 240, 246]. Hence, reducing the cost of Hamiltonian simulation will have a significant impact in chemistry.

4.4.1 Electronic Hamiltonian

Recall the second-quantized Born-Oppenheimer electronic Hamiltonian (4.1), i.e.,[5]

$$H := \sum_{p,q=1}^{\mathcal{N}} h_{pq} a_p^\dagger a_q + \frac{1}{2} \sum_{p,q,r,s=1}^{\mathcal{N}} h_{pqrs} a_p^\dagger a_q^\dagger a_r a_s .$$

[5]Using *atomic units*, where the electron mass, electron charge, Coulomb's constant, and reduced Planck constant $\hbar$ are unity, the electronic Hamiltonian can be written in the given *dimensionless* form; see, e.g., [217, Sec. 2.1.1] for details.

The quantities h_{pq} and h_{pqrs} are obtained by considering $\mathcal{N}$ single-particle basis functions (spin orbitals) taken from a given family of such functions. Particularly, the h_{pq} and h_{pqrs} are one-electron and two-electron integrals, respectively, as defined in [246, Sec. 3]. For our purposes the h_{pq} and h_{pqrs} are problem inputs. The $a_p^\dagger$ and a_p are the creation and annihilation operators for the pth orbital, which encode the fermionic exchange antisymmetry of the problem. The general Hamiltonian form is the same for all molecules with the same number of single particle orbitals $\mathcal{N}$. Therefore, the Hamiltonian of a particular molecule is defined by $\mathcal{N}$ and the h_{pq} and h_{pqrs}.

The Hamiltonian above can be written in the form

$$H = \sum_{j=1}^{m} H_j, \tag{4.50}$$

where $m = \Theta(\mathcal{N}^4)$, and the H_j are Hamiltonians obtained from the terms of (4.1) by combining adjoint pairs; see e.g. [139, 246]. Thus, modulo constant factors, the norms $\|H_j\|$ are given by the $|h_{pq}|$ and $|h_{pqrs}|$. These quantities depend on molecular geometry and the chosen set of basis functions [126,217]. For basis functions that are spatially localized, which are called local basis sets, many of the $|h_{pq}|$ and $|h_{pqrs}|$ are small or very small relative to their largest magnitude [66,69,126]. We use this disparity to partition the Hamiltonians into groups for our algorithms.

For example, [139] considers the quantum simulation of the lithium hydride (LiH) molecule with different choices of basis sets. The authors of [139] consider Slater-type (STO-3G) [125] and triple-zeta (TZVP) [85] basis sets and in both cases they find that a substantial fraction of the H_j have quite small norm. In Table 4.2, we illustrate how one can partition the Hamiltonian using the h_{pq}, h_{pqrs} values shown in [139] to obtain $H = A + B$, where the Hamiltonian B is the sum of the terms for which the corresponding $|h_{pq}|$ and $|h_{pqrs}|$ are less than or equal to different "cutoffs" and A is the sum of the remaining terms. Clearly, different partitions lead to different bounds for the norm of each group, which will be reflected in the cost bounds of the algorithm as shown in Theorem 2. Extending this idea to $\mu > 2$ groups is straightforward.

We digress for a moment to remark that in practical applications of quantum chemistry, the computational cost is often reduced by discarding terms of H that have "negligible" norm relative to some cut-off, say 10^{-10} [69, 126, 139], but this cannot be done in an ad hoc way. Recall that Proposition 1 shows that we may possibly, depending on the particular problem instance, discard certain terms from H, subject to the relationship between the cutoff, t, ε, and the number of

Basis Set	Cutoff	m	m'	$\|A\|$	$\|B\|$
STO-3G	10^{-10}	231	99	10^2	10^{-8}
TZVP	10^{-10}	22155	10315	10^4	10^{-6}
TZVP	10^{-4}	22155	9000	10^4	1
TZVP	10^{-3}	22155	6000	10^4	10
TZVP	10^{-2}	22155	2000	10^3	10^2

Table 4.2: For simulating LiH with bond distance 1.63 Å, in [139] they approximate the Born-Oppenheimer electronic Hamiltonian in two ways. The first approximation uses a minimal basis set (STO-3G) and has $m = 231$. The second approximation uses a more accurate basis set (TZVP) and has $m = 22155$. In each case m is the number of non-zero h_{pq} and h_{pqrs}. The quantity m' is the number of $|h_{pq}|$ and $|h_{pqrs}|$ that are larger than the different cutoff values. The quantities of the leftmost four columns in the first two rows are are taken from [139, Sec. 3.2]. The quantities of the leftmost four columns in the remaining rows are estimated from [139, Fig. 10]. We consider $H = A + B$, where A is the sum of the m' terms corresponding to $|h_{pq}|$, $|h_{pqrs}|$ larger than the cutoff, and B is the sum of the remaining $m - m'$ terms. The norms of A and B are estimated using the triangle inequality, i.e., $\|A\| \le m'\|H_1\|$ and $\|B\| \le (m - m')\|H_{m'+1}\|$, where $\|H_1\| \ge \dots \ge \|H_{m'}\| \ge \dots \|H_m\|$, and for simplicity we assume $\|H_1\| \simeq 1$. Thus, we estimate $\|A\| \simeq m'$ and $\|B\| \simeq (m - m')\times$cutoff, rounded to the nearest power of 10 for simplicity.

terms $(m - m')$ below the cutoff. On the other hand, when the product of the cutoff with $(m - m')$ exceeds ε/t, we cannot arbitrarily discard $(m - m')$ terms, even though individually they may be tiny, because this could introduce truncation error that would exceed the desired simulation accuracy. This is also made particularly clear in the last three rows of Table 4.2, where excluding the terms below the cutoff may introduce error exceeding any reasonable accuracy as evidenced by the respective estimates of $\|B\|$.

4.4.2 Simulation Cost

In chemistry problems the desired simulation accuracy is not arbitrarily small, while $\mathcal{N}$ can be quite large so that (4.50) adequately represents the system of interest [18, 167]. Therefore, the important parameters affecting the simulation cost are the number of single-particle basis functions $\mathcal{N}$, and the magnitudes of the h_{pq} and h_{pqrs}.

In second quantization, i.e., the occupation number representation, states are given by linear combinations of $\mathcal{N}$-bit strings, where a 1/0 indicates which orbitals are occupied/unoccupied by electrons, respectively [126, 217]. Thus H acts on $\mathcal{N}$ qubits. Each Hamiltonian H_j in (4.50) can be represented by tensor products of Pauli matrices through the Jordan-Wigner transformation, and can be simulated efficiently using $O(\mathcal{N})$ standard quantum gates [246]. Alternatives to the Jordan-Wigner transformation have been proposed, such as the Bravyi-Kitaev transformation [50, 206] which improves the gate count for simulating the individual H_j to $O(\log \mathcal{N})$. Hence, our cost bounds for the number of queries (exponentials) immediately translate to bounds for the total gate count through multiplication. Thus, using [50, 206], modulo polylogarithmic factors, the total gate count is proportional to the number of queries. This is what we will consider for our algorithms.

Now consider the simulation of H using splitting formulas. For $\mathcal{N}$ spin-orbitals, the number of terms in (4.50) is $m = \Theta(\mathcal{N}^4)$. Naively applying an order $2k+1$ splitting formula (D.6) yields a number of queries (i.e., a number of exponentials)

$$O\left(\mathcal{N}^{8+2/k}\|H_1\|t\;(\|H_2\|t/\varepsilon)^{1/2k}(25/3)^k\right).$$

Thus, for arbitrary k the cost grows with $\mathcal{N}$ at least as $\mathcal{N}^8$. In particular, if we use the Strang splitting formula ($k=1$), the number of queries is proportional to $\mathcal{N}^{10}$. Hence, a straightforward application of splitting formulas yields a number of queries in the range $\mathcal{N}^8 - \mathcal{N}^{10}$, which clearly becomes impractical even for moderate $\mathcal{N}$ (e.g., $\mathcal{N} = 100$).

Improving this cost bound is critical for quantum computers to have an impact in quantum chemistry applications. A sequence of papers [17,18,123,167,193,240] describe the recent progress. They show both analytic and empirical results. Some of them perform gate-level optimizations across queries, and are thereby specific to the particular problem instance. In [240, Table I], the number of queries using the Strang splitting formula is shown to be proportional to $\mathcal{N}^{10}$, which corresponds to the one that follows from [186] shown above. It is also shown in [240, App. B] that the number of queries can be reduced to become $\mathcal{N}^9$, and it is conjectured that the proof leading to this reduction in the case $k=1$ could be extended to high-order splitting formulas ($k>1$). The paper also considers the implementation of the queries using the Jordan-Wigner transformation. Thus, the total gate count becomes proportional to $\mathcal{N}^{10}$, but allowing parallel gate execution the circuit depth becomes proportional to $\mathcal{N}^9$. Moreover, the authors of the paper carried out numerical tests of molecules from a random ensemble suggesting a number of queries proportional to $\mathcal{N}^8$ as shown in [240, Table I]. Gate-level optimizations on the entire circuit are considered in [123]. In particular, using the Jordan-Wigner transformation for implementing the queries, the authors of that paper conclude that their optimizations make the total gate count proportional to the total number of queries. Therefore, for the Strang formula as presented in [240], the total gate count is proportional to $\mathcal{N}^9$, and allowing parallel execution in conjunction with gate-level optimization leads to a circuit with depth proportional to $\mathcal{N}^7$. In [18, 167], it was argued using empirical evidence that similar improvements on the number of queries are possible for certain restricted but commonly used basis function sets, and this may lead to a number of queries proportional to $\mathcal{N}^{5.5} - \mathcal{N}^7$, while [193] reports even better empirical query estimates in the range $\mathcal{N}^{5.5} - \mathcal{N}^{6.5}$. Finally, a recent paper [17] that uses the simulation method of [32] with different queries than the matrix exponentials used in splitting formulas, obtains a total gate count proportional to $\mathcal{N}^8$, modulo polylogarithmic factors. Furthermore, in a special case they are able to obtain a total gate count proportional to $\mathcal{N}^5$ (up to polylogarithmic factors), under strong assumptions on the basis functions and the computation of the h_{pq}, h_{pqrs} and the resulting accuracy and cost. However, we point out that other authors consider the computation of these quantities to be "complicated business" in general [126, sec. 9.9.5].

Further note that the possibility of using problem specific information in quantum chemistry (e.g., simulating different Hamiltonians for different amounts of time, or simulating them in a certain order) to improve the simulation cost was suggested in [18, 123, 139, 193, 240] without presenting an algorithm or a rigorous analysis exhibiting error and cost bounds. Our goal is obtain rigorous simulation cost improvements under fairly general conditions.

Divide and conquer simulation

It is well known that the locality of physical interactions can be exploited to give substantial advantages for classical algorithms [104], yet only recently considered in detail for quantum algorithms [167]. By utilizing local basis functions, which are localized near atomic centers and have mutual overlap which is typically exponentially decaying with their separation [167], there generally exists a characteristic distance between atomic centers beyond which the corresponding integrals will be negligible. In particular, molecules with large *physical* size, for which a large fraction of atomic orbitals are sufficiently distant from each other, will have many Hamiltonian terms with very small norms. Similarly, another example is the Hartree-Fock basis (which is accurate and commonly used for small systems), where many off-diagonal Hamiltonians are very small [240]. Therefore, in many applications of interest we expect the number of Hamiltonian terms with substantial norm to scale with $\mathcal{N}$ much lower than $\mathcal{N}^4$ [9, 69, 126, 212]. We take advantage of this property to derive faster quantum algorithms.

Consider a fixed set of local basis functions. In general, the number of "non-negligible" $|h_{pq}|$ and $|h_{pqrs}|$ is significantly less than $\mathcal{N}^4$. In [126, Sec. 9.12.2], it is argued that for sufficiently large molecules this number is of order $\mathcal{N}^2$. In [167], the authors claim that this number can scale even as $\mathcal{N}$ using local basis functions. Also, [18] has found this number to be $O(\mathcal{N})$ modulo logarithmic factors. Using these estimates, we obtain $a = 1/2$ or $a = 1/4$, where we have assumed $A = \sum_{j=1}^{m'} H_j$ with $m' = O(m^a)$ as in (4.49), and $B = \sum_{j=m'+1}^{m} H_j$ with $\|B\| \leq (m - m')\|H_{m'+1}\| = O(m^b)$ as in (4.48), with $0 \leq b \leq a$. Taking $\|H_1\| = O(1)$ then further implies $\|H_{\mathcal{N}}\| = O(m^a)$. Observe that our assumptions are consistent with the situation depicted in Table 4.2.

Now consider Algorithm 2 with $H = A + B$. Assume t and ε are arbitrary but fixed, and let us study the simulation cost with respect to $\mathcal{N}$. Even if we do not select the optimal values for k, k_A, and k_B, and we simply assume they are $O(1)$, we obtain a simulation cost improvement. The quantities of Proposition 3 become $\|C\| = O(m^a)$, $\|D\| = O(m^b)$, and hence $n = O(m^{a+b/2k})$, $n_A = O(m^{a+a/2k_A})$, and $n_B = O(m^{b+b/2k_B})$. Using these quantities and (4.32) the simulation cost is bounded from above by

$$c_1 m^{2a} \max\{m^{b/2k}, m^{a/2k_A}\} + c_2 m \max\{m^{a+b/2k}, m^{b+b/2k_B}\},$$

where $c_1, c_2 > 0$ are constants and t, ε are fixed. Since $0 \leq b \leq a \leq 1/2$ the previous expression is bounded by a quantity proportional to

$$m \ \max\{m^{a+b/2k}, m^{b+b/2k_B}\}.$$

Taking $k \leq k_B$ yields that the simulation cost is proportional to

$$m^{1+a+b/2k} = O(\mathcal{N}^{4+4a+2b/k}). \tag{4.51}$$

Recall that using the Bravyi-Kitaev representation [50] for implementing the terms of (4.50), the number of queries of our algorithms, modulo polylogarithmic factors, is proportional to the total gate count. We compare our results to those from the literature in Table 4.3 (as summarized in Table 4.1 previously).

Remark 12. *Using the estimates $a = 1/2$ and $a = 1/4$ for local basis functions from [18, 126, 167] Table 4.3 shows that the number of queries of Algorithm 2 scales as $\mathcal{N}^5 - \mathcal{N}^7$, $0 \leq b \leq a$. This is consistent with the empirical results in [18, 193].*

Method	Cost Dependence on $\mathcal{N}$
Suzuki-Trotter splitting formulas [186]	$\mathcal{N}^8 - \mathcal{N}^{10}$
Improved Strang splitting for the electronic Hamiltonian [240]	$\mathcal{N}^9$
Empirical scaling of random "real" molecules [240]	$\mathcal{N}^8$
Truncated Taylor series [17] (# gates)	$\mathcal{N}^8$
Improved empirical scaling [18, 167, 193]	$\mathcal{N}^{5.5} - \mathcal{N}^7$
On-the-fly algorithm [17] (# gates)	$\mathcal{N}^5$
Algorithm 2 : $(a, b) = (3/4, 0)$	$\mathcal{N}^7$
Algorithm 2 : $(a, b) = (1/2, 1/2)$	$\mathcal{N}^6 - \mathcal{N}^7$
Algorithm 2 : $(a, b) = (1/2, 1/4)$	$\mathcal{N}^6 - \mathcal{N}^{6.5}$
Algorithm 2 : $(a, b) = (1/2, 0)$	$\mathcal{N}^6$
Algorithm 2 : $(a, b) = (1/4, 1/4)$	$\mathcal{N}^5 - \mathcal{N}^{5.5}$
Algorithm 2 : $(a, b) = (1/4, 0)$	$\mathcal{N}^5$

Table 4.3: Comparison of empirical and analytic cost bounds with respect to the number $\mathcal{N}$ of single-particle basis functions for the simulation of the electronic Hamiltonian. The top half of the table are estimates taken from the literature, ignoring any polylogarithmic factors. The bottom half of the table is the estimated scaling for Algorithm 2, where the parameters a and b have been estimated; see the text for details. We give a range for the cost dependence in cases where it varies with some of the algorithm parameters, or when the cost is obtained empirically. All cost estimates refer to the number of queries, except in the case of [17] which does not use splitting formula and presents the total gate count. For all estimates concerning queries, the transition from queries to gate counts involves a multiplication by a $O(\log \mathcal{N})$ factor in the most favorable case.

Remark 13. *If $a, b \to 0$, the cost of Algorithm 2 tends to $O(\mathcal{N}^4)$, which is a lower bound to the simulation cost since the input size is $\Theta(\mathcal{N}^4)$. In contrast, a naive application of an order $2k+1$ splitting formula without partitioning the Hamiltonian would still have cost proportional to $\mathcal{N}^{8+2/k}$.*

Thus, our algorithms exhibit speedup for simulation of the electronic Hamiltonian comparable to the empirical predictions discussed above. Characterizing classes of basis functions and molecules where tighter bounds on the distribution of Hamiltonians of large norm and their number is an important open problem.

Our algorithms take advantage of problem structure in terms of the Hamiltonian norms, without relying on other domain-specific information or implementation-level assumptions. As part of future work, it would be interesting to study how gate-level optimizations and other information specific to chemistry could further improve the performance of our algorithms. Furthermore, partitioning the Hamiltonian into $\mu > 2$ groups may lead to further cost improvements in applications.

We conclude this section by emphasizing that the advantages of our approach may extend to problems beyond quantum chemistry.

4.5 Discussion

Splitting formulas and similar approaches have many advantages for Hamiltonian simulation, and are especially important for near-term quantum computing. Our algorithms take advantage of the problem structure without relying on heavy assumptions and are as simple to implement as standard splitting formulas, but can lead to significantly lower cost. Just like splitting formulas, our algorithms succeed deterministically and therefore they can be used as subroutines that are called numerous times in other quantum algorithms without this affecting the overall success probability. The reduced cost of our algorithms may make them especially suitable for applications in near-term quantum computing devices which will likely have limited resources available.

We emphasize that our results are general, and may be improved given further structural information for a given problem. In particular, our error and cost estimates are worst-case, and hence may be overly pessimistic for application to real-world problem instances. Nevertheless, the rigorous cost and error bounds we derive, allow our algorithms to be used as well-characterized subroutines, which is critically important for the development and deployment of future quantum algorithms.

Finally, we remark that it is possible to extend our algorithms to the case of time-dependent Hamiltonians $H = H(t) = \sum_{j=1}^{m} H_j(t)$, using similar techniques as those applied for splitting formulas in [194, 248, 249]; we leave this as an area for future investigation.

Chapter 5

Quantifying the Performance of Low-Depth QAOA

5.1 Introduction

As small quantum computers begin to emerge in the near future, practitioners will be empowered to experiment with and analyze a new frontier of quantum algorithms. One promising area of application is to approximately solve challenging optimization problems. Indeed, for many such problems, classical algorithms finding the optimal solution require a number of steps that is exponential in the input size in the worst case, so we often must settle for algorithms that produce approximate solutions (in a polynomial number of steps). Hence, an important natural question to explore is whether or not quantum computers offer advantages for approximate optimization. It is thus important to develop new algorithmic techniques and derive new methods of analysis towards resolving this question.

Recently, Farhi et al. [86] proposed a new class of quantum algorithms and heuristics, the Quantum Approximate Optimization Algorithm (QAOA), to tackle challenging approximate optimization problems on gate model quantum computers. In QAOA, the problem Hamiltonian, which encodes the objective function of the given optimization problem, and the mixing Hamiltonian, which transfers probability amplitude between different basis states encoding problem solutions, are applied in alternation p times each to a suitable initial state. Then a computational basis measurement is performed, which returns an approximate solution. The process is repeated a number of times and the best outcome kept. A handful of recent papers suggest such circuits may be powerful for different types of computational problems [87, 91, 114, 138, 241, 254].

The QAOA algorithm and resulting quantum circuits are parameterized by the times (angles) for which the problem Hamiltonian and the mixing Hamiltonian are applied at each iteration. A level-p (depth-p) algorithm has $2p$ parameters. The success of QAOA relies on being able to find a good set of parameters. For QAOA$_p$ of a fixed depth p, straightforward sampling of the search space was proposed [86], but this is practical in general only for small p; as the level increases the parameter optimization becomes inefficient due to the curse of dimensionality [109, 238]. Elegant analytical tools for specific problems can provide parameter values for $p \gg 1$ that give near optimal performance, e.g., for searching an unstructured database [138], but for general problems practically efficient search strategies are needed. Here, we analytically study the performance of QAOA applied to the Maximum Cut (MaxCut) problem, and characterize the optimal algorithm parameters.

In Ref. [86], Farhi et al. investigated low-depth QAOA for MaxCut for specific (bounded-degree) graphs, and provided numerical results for a few special cases. We extend the known results for MaxCut by deriving a sequence of analytic expressions for the expected performance of the algorithm for $p = 1$ on both arbitrary and special classes of graphs, which can be solved to obtain the optimal algorithm parameters. We apply these results to bound the approximation ratio achieved on certain classes of graphs. In particular, for MaxCut on a triangle-free graph with maximum vertex degree D, we show that QAOA$_1$ achieves a solution within a factor of

$$\frac{1}{2} + \Omega\left(\frac{1}{\sqrt{D}}\right)$$

of the optimal value. A similar expressions with a correction term proportional to $1/D$ is obtained for general graphs.

The proofs of our results rely on the algebraic properties of the Pauli matrices. A similar analysis quickly becomes cumbersome as the number p of QAOA iterations increases, as the amount of terms involved in the analysis grows exponentially with p. (Indeed, optimizing QAOA in general appears to suffer from the curse of dimensionality, i.e., the number possible combinations of angles grows exponentially with p.) Nevertheless, for $p = 2$ we solve for the performance of QAOA on a toy problem, called the ring of disagrees [86], producing a particularly complicated intermediate result which exemplifies the difficulty of deriving similar expressions for higher p or for more general graphs. Similarly, we apply our technique to the Directed MaxCut problem, which even for $p = 1$ results in complicated expressions for the expected algorithm output.

We use MaxCut as a case study to demonstrate our approach to quantifying the performance of QAOA. In Section 5.4.2, we propose the *Pauli Solver* algorithm, which is a classical method for computing the expected output of a QAOA circuit. The primary advantage of the Pauli solver is that it doesn't require computing the QAOA state explicitly, avoiding any large matrix-vector multiplications. Applying this algorithm as an analytic procedure is the primary technique we use for proving the main results of this chapter, though it may also be implemented numerically for instance-wise optimization of the QAOA parameters. Moreover, this algorithm is general, and we explain how it may be applied to QAOA applications beyond MaxCut.

Many of the results of this chapter can also be found in [238].

5.2 Background

We briefly review the rich field of approximation algorithms, and provide the details of the Quantum Approximate Optimization Algorithm (QAOA). This material will also be utilized in Chapter 6.

5.2.1 Approximate Optimization

In a *combinatorial optimization problem*, we seek to maximize (or minimize) an *objective function*[1] $f : \{0,1\}^n \to \mathbb{R}$. , We are interested in problems where finding the optimal value is computationally difficult, i.e., the best classical algorithms known have costs that scale exponentially with the input size n. Hence, for such problems we must settle for an *approximation algorithm* or *heuristic*, a procedure that for all n terminates in an amount of time polynomially bounded in n and produces as good a solution as possible. This has led to rich theories of approximation algorithms, hardness of approximation, and approximation complexity, in addition to a variety of heuristic approaches for different problems. We provide a brief outline of some important results; see, e.g., [14, 16, 134, 226, 229, 251] for comprehensive overviews of these subjects.

For some problems, we have results showing that no polynomial time classical algorithm exists

[1]In Chapter 6, we consider more general domains than $\{0,1\}^n$, which result from feasibility constraints and choice of problem encoding.

that always produces an approximation within some factor of the optimal value (unless something believed to be unlikely such as P=NP is true), and, moreover, in some cases we have algorithms that achieve these optimal bounds. We emphasize that such results are worst-case. Indeed, in some cases, heuristics are known that perform much better than any such algorithm on *practical instances*, i.e., return a solution much closer to optimal than the worst-case hardness bounds would suggest. These heuristics are typically evaluated empirically, and may or may not have general performance guarantees, or may require exponentially large running time on some problem instances. On the other hand, for some problems there are significant gaps between the performance of the best algorithm known and the best hardness result. Such problems are natural targets for heuristics and improved algorithms, and in particular quantum approaches, to provide better performance. Hence, it is important for scientists and engineers dealing with such problems to have at their disposal a variety of tools and algorithms.

We are primarily interested in *NP-optimization* (NPO) problems, where, informally, the corresponding decision problem (Given k, does there exist an x such that $f(x) \geq k$?) is in NP (typically, NP-complete). Hence, given a witness solution x', we can efficiently compute $f(x')$ to verify that $f(x') \geq k$. In this chapter we study an important example, the MaxCut problem, and we will see many more examples of such problems in Chapter 6.

For a maximization problem, let $f^* = \max_x f(x)$ denote the optimal value, with optimal solution $x^* = \arg\max_x f(x)$. Note that we are not concerned whether x^* is unique. A solution x is an *R-approximation* if $f(x)/f^* \geq R$, with $0 \leq R \leq 1$. We call R the *approximation ratio*. Similarly, an algorithm $\mathcal{A}$ achieves approximation ratio R if it satisfies $f(\mathcal{A}(y))/f^* \geq R$ for all problem instances y (i.e., in the worst case), where $A(y) \in \{0,1\}^n$ denotes the solution returned by $\mathcal{A}$. If $\mathcal{A}$ runs in time polynomial in the problem size, then we call it an *R-approximation*.

For a minimization problem, the approximation ratio for a solution x may be similarly defined, but with $R \geq 1$ and $f(x)/f^* \leq R$, where f^* denotes the minimal value, and again similarly defined for an approximation algorithm $\mathcal{A}$. Unfortunately, there are multiple conventions in the literature, with the approximation ratio defined to be either $R \geq 1$ or $0 \leq R \leq 1$, for both maximization or minimization problems. In either convention, the ratio is simply the inverse $1/R$ of the other convention, so for a fixed problem and given R there is no ambiguity as to which convention is being used. For convenience, we will primarily consider approximation ratios $R \leq 1$.

We remark that approximation schemes utilizing randomness are also common in the literature. As is standard practice [105], we use *R-approximate algorithm* to describe randomized polynomial time algorithms that output solutions with expected value at least R times the optimal value.

It turns out that different problems may possess very different properties with respect to efficient approximation. As with decision problems, this has led to rich theories of approximation complexity, and natural grouping of problems into approximation complexity classes. We mention three of the most important classes, though there naturally exists others. The complexity class APX contains problems which can be efficiently approximated (on a classical computer) to within some constant factor. Similarly, PTAS is the subset of problems which can be efficiently approximated (with respect to the input size n) to within any fixed ε, and FPTAS is the further subset of problems where the approximation is also efficient with respect to ε^{-1} (as $\varepsilon \to 0$). As problems in FPTAS can be efficiently solved to arbitrary accuracy and in this sense are effectively solvable, primary targets for quantum algorithms are problems in PTAS or APX, in particular problems where the best classical algorithms are not known to be optimal and hence new algorithms could offer significant improvement. (We will see various examples of problems in these classes in Chapter 6.)

Moreover, *complete* problems can be also defined for many approximation complexity classes, which gives a notion of the hardest problems in each class. This requires computational reductions between problems, in particular, reductions preserving approximate solutions in some sense, of which a richer variety exists than as compared to decision problems [72]. See, e.g., [16] for details.

Finally, many important results have been obtained showing that particular problems cannot be approximated better than some factor, known as the *hardness of approximation*. Typically, such results show an approximation ratio for a given problem such that an efficient algorithm beating this ratio could be used to solve an NP-hard problem, and hence such an algorithm is impossible unless P=NP. For certain problems, these results imply the best approximation algorithms known are essentially optimal. For example, as we discuss below, if P≠NP then MaxCut cannot be approximated better than 0.941, whereas, under a weaker complexity theoretic conjecture, we similarly have that the 0.8785 ratio achieved by the Goemans-Williamson algorithm is optimal.

These difficulties in solving optimization problems efficiently have led to much excitement about the possibility of using quantum information processing devices for approximation. Indeed, currently available quantum devices such as the D-WAVE quantum annealers may be appropriately viewed as heuristic solvers, since generally we do not have performance guarantees, and these devices must be characterized empirically; see e.g. [168]. It remains an important open problem whether or not such devices can provide an advantage for real-world optimization problems. We will instead focus here on quantum gate model algorithms, which offer greater design flexibility and several implementation advantages as we will discuss further in Chapter 6.

5.2.2 Quantum Approximate Optimization Algorithm (QAOA)

The Quantum Approximate Optimization Algorithm [86] seeks to approximately solve hard optimization problems on a quantum computer. This family of parameterized algorithms builds off of both quantum annealing and earlier proposals for gate model quantum optimization [131]. We give an overview of the details of QAOA and some important related results.

Consider the problem of maximizing an *objective function* $f(x)$ acting on n-bit strings x. We may always map such a function to a Hamiltonian H_f with eigenvalues that encode the 2^n values $f(x)$ takes over all possible inputs. Thus, finding the optimal value of f corresponds to a special case of the problem of finding extremal eigenvalues for H_f.

We say a Hamiltonian H_f *represents* a function $f : \{0,1\}^n \to \mathbb{R}$ if its eigenvalues satisfy

$$H_f|x\rangle = f(x)|x\rangle \qquad \text{for all } \; x \in \{0,1\}^n.$$

Given such a Hamiltonian, if we can compute its expansion in terms of Pauli Z operators

$$H_f = c_0 I + \sum_{j=1}^{n} c_1 Z_j + \sum_{j<k} c_{jk} Z_j Z_k + \dots \qquad c_\alpha \in \mathbb{R}, \tag{5.1}$$

then we can easily simulate H_f, i.e., implement the operation $U_f(t) = e^{-iH_f t}$, $t \in [0, 2\pi]$, using controlled-NOT and Z-rotation gates to simulate each term in the sum independently, and without requiring any additional ancilla qubits; such a quantum circuit is shown in Figure 6.1. If the number of terms in the sum is not too many, i.e., bounded by a polynomial in the number of variables n, then this simulation is efficient. On the other hand, if f is efficiently computable classically, then it is well known that we may efficiently simulate H_f using ancilla qubits (to compute and store $f(x)$) and controlled Z-rotation gates to then apply the proportional phase shift; see e.g. [59, 173]. We provide a design toolkit elaborating on the construction and simulation of Hamiltonians of the form (5.1) in Section 6.3, along with several related results, with the details given in Appendix B.

The primary application of QAOA we consider is to solving *constraint satisfaction problems*, where the objective function f to maximize is given by a (possibly weighted) sum of m many *soft constraints* (Boolean clauses) f_j to be satisfied, $f(x) = \sum_{j=1}^{m} f_j(x)$. In many important cases such problems are NP-hard to solve optimally, and therefore we must settle for approximate solutions. When the constraints involve a constant number of variables, e.g. as in Max-k-SAT, then the number of terms in the resulting Hamiltonian is bounded by a constant times m, so such Hamiltonians can be efficiently constructed and simulated with polynomial cost.

For a given problem instance on n binary variables $x \in \{0,1\}^n$ with objective function $f(x)$, a QAOA mapping consists of three components:

1. An **initial state**, which we take to be the equal superposition state
$$|s\rangle = |+\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle.$$

2. A family of **phase separation operators**
$$U_P(\gamma) = e^{-i\gamma H_f},$$
where $\gamma \in [0, 2\pi)$ and H_f acts on basis states $|x\rangle$ as $H_f|x\rangle = f(x)|x\rangle$.

3. A famly of **mixing operators**
$$U_M(\beta) = e^{-i\beta B},$$
where $\beta \in [0, 2\pi)$ and $B = \sum_{j=1}^{n} X_j$.

The circuit depth parameter $p \in \mathbb{N}$ and the $2p$ angles $\gamma_1, \dots, \gamma_p, \beta_1, \dots, \beta_p$ then suffice to specify a QAOA circuit, which we denote QAOA$_p$. More precisely, a QAOA mapping for a given problem is a classical procedure which uniformly and efficiently maps each problem instance to a corresponding quantum circuit that depends on the parameter values selected for γ and β. Note that the same mapping applies whether the goal is maximization or minimization of the objective function f, up to the sign of H_f. We consider more general operators and initial states for QAOA in Chapter 6.

The interval $[0, 2\pi)$ for the angles γ, β is chosen arbitrarily, and different intervals may be used. Thus, the angles for QAOA are real numbers. To avoid complicating details, we do not deal here with issues of precision regarding how such angles may be represented. Generally, the constructions we give in the remainder of this chapter and in Chapter 6 will result in quantum circuits consisting of fixed quantum gates such as Hadamard and CNOT gates, and single-qubit rotation gates that depend on γ, β. Standard techniques are known for efficiently approximating such rotations with a number of quantum gates taken from a universal set that scales polylogarithmically with the desired accuracy; see the discussion in Appendix A and the related details in [173].

Given a QAOA mapping and set of angles, the algorithm consists of applying the phase and mixing operators in alternation to create the QAOA$_p$ state

$$|\boldsymbol{\gamma\beta}\rangle = U_M(\beta_p)U_P(\gamma_p)\dots U_M(\beta_1)U_P(\gamma_1)|s\rangle, \tag{5.2}$$

followed by a computational basis measurement. This process is repeated a number of times, and the best solution kept after evaluating f at each of the measurement outcomes. A high-level QAOA$_p$ circuit is shown in Figure 5.1. We remark that the description of the problem instance is encoded in the QAOA operators themselves, and is not otherwise input into the quantum algorithm; therefore, each QAOA circuit is problem instance dependent.

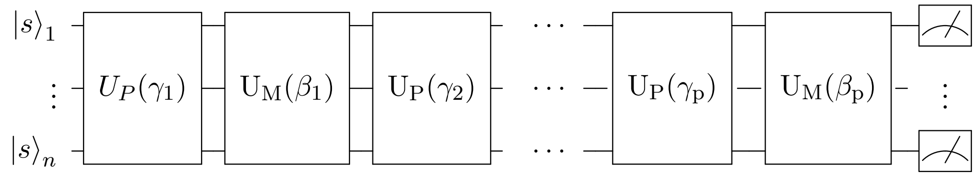


Fig. 5.1: The Quantum Approximate Optimization Algorithm (QAOA$_p$).

Then, if somehow we can determine "good" angles γ_j, β_j such that the state $|\boldsymbol{\gamma\beta}\rangle$ has significant projection (i.e., probability amplitude) onto the subspace of states corresponding to good approximate solutions, then repeated preparation and measurement of $|\boldsymbol{\gamma\beta}\rangle$ will yield such a solution with high probability. Indeed, if such a QAOA$_p$ state can be found with probability amplitude at least $1/\text{poly}(n)$ for a given solution state $|x'\rangle$, then repeating this process a polynomial number of times will yield the solution x' with high probability.

Clearly, the success of this approach depends on the number p of QAOA rounds, and the angles $\gamma_1, \dots, \gamma_p, \beta_1, \dots, \beta_p$. When $p = O(1)$, QAOA is hoped to be especially suitable for small quantum computers [89]. Indeed, promising results have been shown even for $p = 1$ [87]. The angles can either be determined in advance through analysis and numerical testing, or searched for as part of the algorithm. In general, $|\boldsymbol{\gamma\beta}\rangle$ is a vector exponentially large in n, so deriving or finding parameters producing a "good" state is a difficult task. Moreover, knowing which solutions are "good" may not be meaningful without knowing the optimal solution itself. Hence, we follow the approach of [86] and consider the value of the objective function output by the algorithm in expectation (i.e., we take the average of the function evaluation on the measurement outcomes). Bounds on this quantity will lead to useful performance bounds to the approximation ratio achieved by QAOA.

For a fixed set of angles, each preparation and measurement of $|\boldsymbol{\gamma\beta}\rangle$ returns a solution x with some probability. Repeating this process, the expected value of the objective function is

$$\langle f\rangle_{\text{QAOA}_p} = \langle\boldsymbol{\gamma\beta}|H_f|\boldsymbol{\gamma\beta}\rangle =: \langle H_f\rangle.$$

Thus, for fixed p the best the algorithm can do with respect to the expected solution output is

$$F_p := \max_{\gamma_1,\ldots,\beta_p} \langle\boldsymbol{\gamma\beta}|H_f|\boldsymbol{\gamma\beta}\rangle. \tag{5.3}$$

We refer to angles $\gamma_1^*,\ldots,\beta_p^*$ maximizing the right-hand side of (5.3) as *optimal*. Selecting such angles and repeating the algorithm a polynomial number of times produces with high probability[2] a solution x such that $f(x)$ is close to F_p. For a constraint satisfaction problem with m constraints, i.e., $0 \le f(x) \le f^* \le m$ for optimal value f^*, the expected approximation ratio with optimally selected angles then satisfies

$$\max_{\gamma\beta}\langle R\rangle \ge F_p/m. \tag{5.4}$$

Note that for some problems, a better upper bound than m to the optimal solution f^* is known, which given F_p yields a sharper lower bound to $\langle R\rangle$.

We typically consider QAOA$_p$ for fixed p. Clearly, the set of QAOA$_p$ states contains the QAOA$_q$ states for $q < p$, so as p increases the performance of QAOA$_p$, with respect to optimally selected angles, can only improve. The trade-off of course is increased implementation cost, and increased difficulty to find such optimal or near-optimal angles. Clearly, the low p implementations are the most suitable for early quantum computers. We show below that for the MaxCut problem, the QAOA$_p$ state can be implemented using $O(p(n+m))$ basic quantum gates, and $O(pn^2)$ basic gates suffice for more general quadratic unconstrained optimization problems.

[2]Fom the Chebyshev inequality, an outcome of at least $F_p - 1$ will be obtained with probability at least $1 - 1/m$ after $O(m^2)$ repetitions In [86], it is argued that $O(m \log m)$ repetitions suffice for MaxCut.

Henceforth, by *basic quantum gates* we mean controlled-not (CNOT) gates and arbitrary single-qubit gates; see Appendix A for a review of some important quantum gates.

We remark that there are close connections between QAOA and quantum annealing. In particular, standard techniques (i.e., Suzuki-Trotter splitting formulas) to approximate the latter with discrete steps naturally yields a quantum state of the form (5.2). Hence, from the adiabatic theorem, it follows that

$$\lim_{p\to\infty} F_p = f^*, \tag{5.5}$$

i.e., for p sufficiently large there exists optimal angles with expected value arbitrarily close to the optimal solution. This was shown in [86] for the MaxCut problem under minor assumptions, but is easily seen to also apply to many other optimization problems. Thus, the QAOA method is sound in the sense that it in principle can solve optimization problems optimally. Unfortunately, this result does not tell us how large p must be selected to obtain a certain quality of approximation, or how the algorithm behaves for constant or polynomially bounded p, and how to select the optimal angles.

Remark 14. *For a given class of problem instances, if we can show for QAOA$_p$ there exists angles such that $\langle H_f \rangle \geq \alpha$, then from standard probabilistic arguments, there must exist a solution x such that $\alpha \leq f(x) \leq f^*$, where f^* is the optimal value.*

Since their initial proposal [86], QAOA circuits have sparked considerable interest in the research community. As mentioned, a QAOA$_1$ algorithm was shown to beat the best classical approximation algorithm known for the problem Max-E3Lin2, only to subsequently inspire a slightly better classical algorithm [22,87]. The performance of QAOA$_p$ for Max-E3Lin2 with $p > 1$ has yet to be determined, which is the truly interesting case.

Generally, characterizing the power of QAOA$_p$ circuits remains the most important open problem. The lowest depth version QAOA$_1$ has provable performance guarantees for certain problems, though there exist better classical approximation algorithms. However, the class of QAOA$_1$ circuits turns out to be surprisingly powerful. In a recent paper, Farhi and Harrow [91] proved that, under reasonable complexity assumptions, the output distribution of QAOA$_1$ circuits cannot be efficiently approximated by a classical algorithm.[3] QAOA circuits are therefore among the most

[3]More precisely, it was shown that if one could efficiently classically sample from the output of a QAOA$_1$ circuit, then the Polynomial Hierarchy would collapse to its third level, which is believed to be unlikely; see [91] for details.

promising candidates for early demonstrations of "quantum supremacy" [41, 196], i.e., the physical demonstration on a quantum computer of a computational task which cannot be reproduced classically without an exponentially scaling amount of resources. However, it remains an open question whether QAOA circuits provide a quantum advantage for approximate optimization, whether in terms of better approximation algorithms, or heuristics that empirically outperform existing methods on practical problem instances.

Ultimately, the success of the QAOA approach will depend on finding effective parameter-setting strategies. For fixed p, and problems where the objective function is given by a sum of locally acting terms, the optimal angles can be computed in time polynomial in the number of qubits [86]. With increasing p, however, exhaustive search of the QAOA parameters becomes inefficient due to the curse of dimensionality; if we discretize so that each angle can take on L different values, searching to find the optimal angles (without any further structural information) takes a number of steps at least $\Omega(L^p)$ in the worst case. Hence, we will study approaches to bounding the expected performance of QAOA, but with an eye to techniques for parameter setting. We will focus on the MaxCut problem, originally considered in [86], as a prototype for the application of QAOA to other problems

Finally, we remark that QAOA circuits have also been considered for applications to exact optimization [138, 241] and sampling [91]. In particular, the authors of [138] show that QAOA circuits are powerful enough to achieve the $O(\sqrt{2^n})$ query complexity of Grover's quantum algorithm for unstructured search problems, showing the first definitive quantum advantage for QAOA circuits with finite p. Nevertheless, in this chapter we restrict our attention to applications of QAOA to approximate optimization problems.

5.3 Unconstrained Optimization

We first consider *unconstrained* optimization problems, where all strings $x \in \{0,1\}^n$ encode valid solutions. With the natural mapping to n qubits, where the jth qubit encodes the jth binary variable x_j, such problems are especially well-suited for QAOA, typically resulting in simple initial states, phase operators, and mixing operators, and yielding especially low-resource constructions. These implementation advantages make unconstrained problems particularly promising target applications for quantum approximation on early quantum computers.

For unconstrained problems, it is relatively straightforward to implement QAOA$_p$ using basic quantum gates. For the initial state, we may use the equal superposition state $|s\rangle = |+\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle$. This state may be easily prepared from $|0\rangle^{\otimes n} = |0^n\rangle$ using n Hadamard (H) gates in depth 1 (i.e., applied in parallel), as follows from the identity $|s\rangle = \mathrm{H}^{\otimes n}|0^n\rangle$. Similarly, the standard mixing operator $U_M(\beta) = e^{-i\beta \sum_{j=1}^n X_j} = \prod_{j=1}^n e^{-i\beta X_i}$ may be used, and implemented with n X-rotation gates $R_X(\beta/2)$ in depth 1. Hence, for problems where the phase operators can also be implemented relatively inexpensively, QAOA$_p$ can be applied with relatively low resources.

We next quantify the required resources for the general class of *quadratic* unconstrained binary optimization problems. In the following section we consider in detail a prototypical problem from this class, the MaxCut problem.

5.3.1 Quadratic Unconstrained Binary Optimization

A general and important class of pseudo-Boolean optimization problems are *quadratic unconstrained binary optimization* (QUBO) problems [168], where we seek to minimize a function

$$f(x) = a + \sum_{j=1}^{n} c_j x_j + \sum_{j<k} d_{jk} x_j x_k, \tag{5.6}$$

with $a, c_j, d_{jk} \in \mathbb{R}$, $x_j \in \{0,1\}$. Indeed, this is the class of problems (ideally) implementable on a quantum annealing device such as, for example, the D-WAVE 2X, where the qubit interactions are themselves quadratic [37, 168]. The QUBO class also contains many problems, via polynomial reductions, which at first sight are not quadratic, sometimes requiring extra variables; indeed, the natural QUBO decision problem is NP-complete [101]. A variety of exact and approximate classical algorithms and heuristics have been developed for QUBO problems; see e.g. [46, 222] and the references therein.

We remark that although (5.6) contains only positive variables x_j, as $\overline{x}_j = 1 - x_j$, it is without loss of generality. For example, the clause $x_1 \oplus x_2$ may be equivalently written as $x_1 + x_2 - 2x_1x_2$, which is of the same form as (5.6). Similarly, other common Boolean clauses may be mapped to Hamiltonians using the rules given explicitly in Section 6.3 (and their derivations in Appendix B). Applying these results, we have the following lemma.

Lemma 1. *The QUBO function (5.6) maps to a Hamiltonian given as a quadratic sum of Pauli Z operators, with size (number of terms) at most $1 + n/2 + n^2/2$. Explicitly, we have*

$$H_f = (a + c + d)I - \frac{1}{2}\sum_{j=1}^{n}(c_j + d_j)Z_j + \frac{1}{4}\sum_{j<k} d_{jk}Z_jZ_k, \tag{5.7}$$

where we have defined $c = \frac{1}{2}\sum_{j=1}^{n} c_j$, $d = \frac{1}{4}\sum_{j<k} d_{jk}$, and $d_j = \frac{1}{2}\sum_{k\neq j} d_{jk}$ with $d_{jk} = d_{kj}$.

Proof. The proof follows directly from applying Theorem 11 of Appendix B to (5.6). □

The terms in (5.7) mutually commute. Thus, we can simulate H_f, i.e., implement the QAOA phase operator $U_P(\gamma) = e^{-i\gamma H_f}$, using at most n-many R_Z gates and $\binom{n}{2}$-many R_{ZZ} gates. As shown in Figure 6.1, each R_{ZZ} can be simulated with 2 CNOT gates and a R_Z gate (up to an irrelevant global phase), hence $U_P(\gamma)$ can be implemented with $\frac{3}{2}n^2 - \frac{3}{2}n$ basic quantum gates. For the initial state $|s\rangle = |+\rangle^{\otimes n}$ and the mixing operator $U_M(\beta) = e^{-i\beta\sum_{j=1}^{n} X_j}$, we obtain the following bound to the total gate cost.

Theorem 3. *For a QUBO problem on n variables, we can prepare the QAOA$_p$ state using at most*

$$\frac{3}{2}pn^2 - \frac{1}{2}pn + n \tag{5.8}$$

CNOT and single qubit gates (R_X, R_Z, H), with depth $O(pn)$, and n qubits.

Proof. The theorem follows from Lemma 1 and the discussion of the general case above. □

We remark that for a given instance of a particular problem, the implementation cost may be much lower. For example, in the next section we will see that the QAOA$_p$ construction for the Max-Cut problem on bounded degree graphs requires a number of gates scaling as pn rather than pn^2.

We may similarly construct QAOA circuits for objective functions of order higher than quadratic. For maximum degree $d = O(1)$, the phase operator can be implemented with $O(n^d)$ basic gates, and the QAOA$_p$ state can be implemented with $O(pn^d)$ basic gates. It is straightforward to construct such phase operators and to explicitly bound their implementation costs using the results of Appendix B, so we do not explore this in detail here. In the remainder of this chapter we focus on strategies for analyzing the performance of QAOA and finding good algorithm parameters.

5.4 Maximum Cut

We consider QAOA applied to the MaxCut optimization problem, as studied in [86, 238]. In an instance of MaxCut, we are given a graph, and we seek to "cut" as many edges as possible by dividing the vertices into two sets. This problem may be naturally represented as a QUBO. We summarize the QAOA implementation of MaxCut [86]. We then derive analytic formulas for the performance of QAOA$_1$ for MaxCut, reproducing and significantly extending the results of [86], where numerical results were found for limited classes of graphs.

Problem: Given a graph $G = (V, E)$, with $|V| = n$ vertices and $|E| = m$ edges, partition V into two sets V' and $V \setminus V'$ such that the number of edges crossing the cut, i.e., edges with one endpoint in V and the other in $V \setminus V'$, is maximized.

The MaxCut problem has many applications in various fields, including circuit layout design and statistical physics [21, 81]. The corresponding decision problem, deciding if there exists a cut of size at least k, is NP-complete [101], so we cannot hope to efficiently find the optimal solution in general unless P=NP. For approximation, MaxCut is APX-complete [150, 179], which means it has no polynomial-time approximation scheme (PTAS) unless P=NP. Thus, the best we can do is a constant-factor approximation. Indeed, it is NP-hard to approximate MaxCut better than 0.941 [122]. Using semidefinite programming and randomization, the Goemans-Williamson algorithm [105] achieves an approximation ratio of 0.8785, the best classical algorithm known. It has been shown that if the unique games conjecture is true (a weaker complexity theoretic conjecture than P$\neq$NP), then this value is optimal [151].

The problem remains hard to approximate on graphs of bounded degree. For a graph with maximum vertex degree $D_G \geq 3$, MaxCut can be efficiently approximated to within $0.8785 + O(D_G)$ [93]. In particular, for $D_G = 3$ this gives a 0.921 approximation, yet this case remains APX-complete [179]. Indeed, the MaxCut decision problem remains NP-complete for $D_G = 3$ [255]. On the other hand, MaxCut is known to be solvable in polynomial time when restricted to certain simple classes of graphs, for example, planar graphs, toroidal graphs, or graphs not contractable to the complete graph with five vertices K_5 [20, 116]. For the application of QAOA to early quantum computers, we are particularly interested in subclasses of problems, such as MaxCut on bounded degree graphs, where fewer implementation resources are required than the general case, yet the problem remains hard to approximate.

We emphasize that the stated complexity and hardness results concern the worst-case performance of efficient algorithms approximately solving MaxCut. Different randomized heuristics are known for MaxCut, which may produce much better solutions for practical instances [94].

Construction. The QAOA construction for MaxCut follows directly from that of the general QUBO case discussed above, resulting in the same construction as the one given in [86]. For an n-vertex graph, states are represented with n qubits, with the 2^n computational basis states encoding all 2^n possible partitions of V. Each solution x specifies a unique $S' \subset V$. Explicitly, basis states encode the n binary indicator variables $x_1, \ldots, x_n$, with $x_j = 1$ indicating that vertex j is included in S'. All states are feasible, so the problem is unconstrained. We consider the the initial state $|s\rangle = |+\rangle^{\otimes n}$ and mixing Hamiltonian $B = \sum_{v \in V} X_v$ as in [86], though other initial states or mixing operators are possible. As explained, $|s\rangle$ can be prepared, and the mixing operator $U_M(\beta) = e^{-i\beta B}$ can be implemented, with n single-qubit gates each.

The objective function counts the number of edges crossing the partition, and is represented by the quadratic Hamiltonian

$$C = \sum_{(uv) \in E} C_{uv}, \qquad C_{uv} = \frac{1}{2}(I - Z_u Z_v), \tag{5.9}$$

where each C_{uv} encodes the predicate $x_u \oplus x_v$, which is true when the edge (uv) is properly colored. For convenience, here we use C for the objective Hamiltonian instead of H_f, as well as for the objective function $C(x)$, which is consistent with the notation of [86]. Thus, ignoring the global phase term, the phase operator $U_P(\gamma) = e^{-i\gamma C}$ can be implemented with at most $3m$ basic gates.

Hence, the QAOA$_p$ state $|\gamma\beta\rangle$ for MaxCut can be prepared with $n + p(n + 3m)$ basic quantum gates. In particular, for bounded degree graphs ($D_G = O(1)$), only $O(pn)$ basic gates are required.

5.4.1 Performance

We turn to the performance of QAOA for MaxCut. For $p = 1$, [86] derives approximation ratio bounds for 2-regular and 3-regular graphs based on numerical results. We generalize these bounds to arbitrary graphs by deriving an exact formula for $\langle C \rangle = \langle \gamma\beta | C | \gamma\beta \rangle$, which is used to bound the expected approximation ratio $\langle R \rangle$. Unfortunately, deriving similar results for higher p, which is the true question of interest, appears to be a difficult problem. We derive a complicated expression for $p = 2$ on a particularly simple family of graphs which exemplifies this difficulty.

Consider an arbitrary edge $(uv) \in E$, and let $d = d_u = \deg(u) - 1$ and $e = e_u = \deg(v) - 1$. Let $f = f_{uv}$ be the number of triangles in the graph containing (uv), i.e., $f = |\mathrm{nbhd}(u) \cap \mathrm{nbhd}(v)|$, where the *neighbourhood* function $\mathrm{nbhd}(v)$ gives the set of vertices adjacent to v. The following theorem shows that the $p = 1$ expectation value $\langle C_{uv}\rangle$ depends only on the angles (γ, β), and the *neighbourhood parameters* (d, e, f) of the edge (uv), i.e., the local structure of the subgraph containing (uv) and its adjacent vertices. Hence, the overall expectation value $\langle C\rangle$ reduces to a sum over triplets (d, e, f), weighted by the number of times an edge with these parameters appears in G.

For MaxCut, the optimal solution $C^* = C(x^*)$ is at most m, so the expected approximation ratio satisfies $\langle R\rangle \geq \langle C\rangle/C^* \geq \langle C\rangle/m$. The case $C^* = m$ occurs if G is a bipartite graph, which can be checked (and if so, MaxCut solved) in linear time [71]. Hence, we emphasize that the lower bound $\langle R\rangle \geq \langle C\rangle/m$ is quite conservative, and the algorithm may perform much better on instances occurring in practice. (For fixed angles, $\langle R\rangle$ may be significantly larger than $\langle C\rangle/m$ if $C^* < m$.)

Theorem 4. *Consider the QAOA$_1$ state $|\gamma\beta\rangle$ for MaxCut on a graph G.*

- *For each edge (uv),*

$$\begin{aligned}\langle\gamma\beta|C_{uv}|\gamma\beta\rangle &= \frac{1}{2} + \frac{1}{4}\sin(4\beta)\sin\gamma\,(\cos^d\gamma + cos^e\gamma) \\ &- \frac{1}{4}\sin^2(2\beta)\cos^{d+e-2f}\gamma\,(1-\cos^f(2\gamma)) \quad =: \langle C_{uv}\rangle(d,e,f)\end{aligned} \tag{5.10}$$

  where $d = deg(u) - 1$, $e = deg(v) - 1$, and f is the number of triangles in the graph containing (uv).

- *The overall expectation value is*

$$\langle C\rangle = \langle\gamma\beta|C|\gamma\beta\rangle = \sum_{(d,e,f)} \langle C_{uv}\rangle(d,e,f)\,\chi(d,e,f), \tag{5.11}$$

  where $\chi(d, e, f)$ gives the number of edges in G with neighbourhood parameters (d, e, f).

We prove the theorem using the Pauli Solver algorithm which we introduce in the next section.

For a *fixed* arbitrary graph, the expectation value $\langle C\rangle$ for QAOA$_1$ may thus be efficiently classically computed, *analytically in closed form*, for any angles γ, β, and hence efficiently optimized.

Note that, recalling Remark 14, these results imply that there exists a partition that cuts at least $\lceil \max_{\gamma,\beta} \langle C \rangle \rceil$ edges. However, QAOA (or some other algorithm) is still required to *find* a bit string realizing such an approximation or better.

For graphs with structure or symmetry, we may significantly simplify the result (5.11). To prove the theorem, we first show a general procedure for computing expectation values for QAOA$_p$. We then consider the special cases of triangle free and regular graphs as corollaries.

5.4.2 Pauli Solver Algorithm

We describe a high-level classical procedure for deriving $\langle C \rangle$ using the properties of the Pauli matrices and their exponentials. (See Appendix A for a review of the most important properties.) We then apply this approach explicitly to prove Theorem 4. This procedure is general and may be applied to different mappings, initial states, or to other problems beyond MaxCut. Moreover, this approach works in principle for arbitrary QAOA depth p.

Observe that, for a general Hamiltonian H expanded in the basis of tensor products of Pauli matrices (see equation (A.5)), only the terms in the sum that are strictly composed of X and I operators will have non-zero expectation value for the state $|s\rangle = |+\rangle^{\otimes n}$, and hence only these terms will contribute to $\langle s|H|s\rangle$. Thus, in the spirit of the *Heisenberg representation* of quantum mechanics [204], instead of evolving the state $|s\rangle$, to compute $\langle C \rangle$ we (equivalently) evolve the observable C itself. For a unitary evolution U, this corresponds to the transformation $C \to U^\dagger C U$, called *conjugation* of C by U.

Hence, for a general objective function $C = \sum_{\ell=1}^m C_\ell$ (and corresponding Hamiltonians C, C_ℓ), define the operator $Q = U_M(\beta)U_P(\gamma)$ for QAOA$_1$. Suppose we can compute, somehow, the Pauli expansions of each C_ℓ conjugated by Q,

$$Q^\dagger C_\ell Q = a_0 I + \sum_{j=1}^{n} \sum_{\sigma=X,Y,Z} a_{j\sigma}\sigma_j + \sum_{j\neq k} \sum_{\sigma,\lambda=X,Y,Z} a_{j\sigma\lambda}\sigma_j\lambda_k + \dots, \tag{5.12}$$

$a_\alpha \in \mathbb{R}$. Then, using $\langle +|I|+\rangle = \langle +|X|+\rangle = 1$ and $\langle +|Y|+\rangle = \langle +|Z|+\rangle = 0$, $\langle C_\ell \rangle$ is given by

$$\langle \gamma\beta|C_\ell|\gamma\beta\rangle = \langle s|(Q^\dagger C_\ell Q)|s\rangle = a_0 + \sum_{j=1}^{n} a_{jX} + \sum_{j\neq k} a_{jkXX} + \dots,$$

i.e., the sum of the coefficients of the terms containing only X or I factors in (5.12). Hence, computing the Pauli coefficients a_α of $Q^\dagger C_\ell Q$ gives $\langle C_\ell \rangle$ directly. This suggests the following general (classical) procedure for computing $\langle C\rangle$.

Algorithm computing $\langle C\rangle$:

1. Compute $Q^\dagger C_\ell Q$ as a sum of Pauli operators as in (5.12).
2. Discard all terms in the sum containing a Y or a Z factor (i.e., keep the terms containing strictly X and I factors).
3. Set $\langle C_\ell\rangle$ as the sum of the remaining coefficients.
4. Apply Steps $1-3$ for each edge $\ell = 1, \dots, m$ and return the overall sum $\langle C\rangle = \sum_\ell \langle C_\ell\rangle$.

The same steps apply for higher p by replacing Q with $Q_p = U_M(\beta_p)U_P(\gamma_p)\dots U_M(\beta_1)U_P(\gamma_1)$. However, the difficulty is that, in general, the number of terms to deal with in the sum (5.12) grows exponentially with p in this case. By implementing the algorithm in Python, in Sec. 5.4.5 we derive an expression for $\langle C_\ell\rangle$ for $p = 2$ in a particular case. For $p = 1$, the number of terms may be few enough that it is possible to derive general results such as those of Theorem 4, whereas this becomes cumbersome even for $p = 2$. However, $\langle C\rangle$ can always be solved for using the algorithm, and the angles optimized, on an instance-by-instance basis.

The advantage of the Pauli Solver algorithm is that $\langle C\rangle$ can often be computed much more efficiently by conjugating C to $Q^\dagger C Q$ and computing the coefficients than by computing (and storing) the state $|\gamma\beta\rangle$ explicitly. This, in general, depends on the given problem and the form of the phase and mixing operators. In particular, $Q^\dagger C_\ell Q$ can often be computed directly using the algebraic properties of the Pauli matrices, avoiding any large matrix-vector multiplications. For MaxCut, using the locality of the clause Hamiltonians C_{uv}, and the product structure of the phase and mixing operators, in computing $\langle C_{uv}\rangle$ we can ignore many of the terms in $U_M(\beta)$ and $U_P(\gamma)$ which substantially simplifies the computation. These ideas will be made clear in the proof of Theorem 4 below.

The Pauli Solver algorithm is also suitable for different initial states $|s\rangle$. In this case, Steps 2 and 3 must be appropriately modified to keep only the terms in the Pauli expansion that contribute to $\langle s|\cdot|s\rangle$, which will depend on the particular $|s\rangle$.

We now use this procedure to prove Theorem 4. We show the steps explicitly, with the goal that similar techniques may be used to characterize the performance of QAOA for other problems.

Proof of Theorem 4. Consider QAOA$_1$ applied to MaxCut with $Q = e^{-i\beta B}e^{-i\gamma C}$. Observe that $\langle C\rangle = \langle s|Q^\dagger CQ|s\rangle = \sum_{(uv)\in E}\langle C_{uv}\rangle = \frac{m}{2} - \frac{1}{2}\sum_{(uv)\in E}\langle s|Q^\dagger Z_u Z_v Q|s\rangle$. Hence, to compute $\langle C\rangle$ it suffices to compute the quantities $\langle Z_u Z_v\rangle$.

Fix an edge $(uv) \in E$. Recall that each Pauli matrix Λ satisfies $e^{-i\theta\Lambda} = \cos(\theta)I - i\sin(\theta)\Lambda$. Let $c = \cos 2\beta$ and $s = \sin 2\beta$. Consider the action of Q on C_{uv} as first a conjugation by the mixing operator, followed by a conjugation by the phase operator. From the commutation properties of the Pauli matrices, most of the terms in the mixing operator $e^{-i\beta B} = \prod_{j=1}^n e^{-i\beta X_j}$ will commute through and cancel, and we have

$$e^{i\beta B}Z_u Z_v e^{-i\beta B} = e^{2i\beta X_u}e^{2i\beta X_v}Z_u Z_v = c^2 Z_u Z_v + sc(Y_u Z_v + Z_u Y_v) + s^2 Y_u Y_v. \tag{5.13}$$

Similarly, for the subsequent application of the phase operator, terms corresponding to edges not containing u or v also commute through and cancel. The first term on the right, $c^2 Z_u Z_v$, commutes with $e^{-i\gamma C}$ and thus does not contribute to the expectation value. We conjugate each remaining term in (5.13) separately by $e^{-i\gamma C}$. Let $c' = \cos\gamma$ and $s' = \sin\gamma$. We have

$$\begin{aligned}
\langle s|e^{i\gamma C}Y_u Z_v e^{-i\gamma C}|s\rangle &= \langle s|e^{2i\gamma C_{uv}}e^{2i\gamma C_u}Y_u Z_v|s\rangle \\
&= \langle s|e^{-i\gamma Z_u Z_v}e^{-i\gamma\sum_{w\in \mathrm{nbhd}(u)\backslash v} Z_u Z_w}Y_u Z_v|s\rangle \\
&= \langle s|(Ic' - is'Z_u Z_v)\prod_{i=1}^{d}(Ic' - is'Z_u Z_{w_i})Y_u Z_v|s\rangle,
\end{aligned}$$

where $C_u = \sum_{w\in\mathrm{nbhd}(u)\backslash v} C_{uw}$ and $C_v = \sum_{w\in\mathrm{nbhd}(v)\backslash w} C_{vw}$. Recall that $d = |\mathrm{nbhd}(u)\setminus v|$ and $e = |\mathrm{nbhd}(v)\setminus u|$. Expanding the product on the right hand side above gives a sum of tensor products of Pauli operators. Clearly, the only term that can contribute to the expectation value is the one proportional to $Z_u Z_v * I^{\otimes d} * Y_u Z_v = -iX_u$. Thus, we have

$$\langle s|e^{i\gamma C}Y_u Z_v e^{-i\gamma C}|s\rangle = \langle s| - is'c'^d(-iX_u)|s\rangle = -s'c'^d.$$

By symmetry, this implies $\langle s|e^{i\gamma C}Z_u Y_v e^{-i\gamma C}|s\rangle = -s'c'^e$. Observe that these terms depend only on the numbers of neighbours d and e.

The last term in (5.13) becomes

$$\begin{aligned}
\langle s|e^{i\gamma C}Y_u Y_v e^{-i\gamma C}|s\rangle &= \langle s|e^{2i\gamma C_u}e^{2i\gamma C_v}Y_u Y_v|s\rangle \\
&= \langle s|\prod_{i=1}^{d}(c'I - is'Z_u Z_{w_i})\prod_{j=1}^{e}(c'I - is'Z_v Z_{w_j})Y_u Y_v|s\rangle.
\end{aligned}$$

In this case, there are many terms in the product which can contribute to the expectation value. The simplest terms that contribute are $\langle s|(c'I)^{d+e-2}(-is'Z_uZ_w)(-is'Z_vZ_w)Y_uY_v|s\rangle = c'^{d+e-2}s'^2$, of which there are f many, corresponding to the f triangles containing (uv). As $Z_uZ_{w_i} * Z_uZ_{w_i} = I$, if $f > 2$ then higher order terms will contribute. The next higher-order terms result from three different pairs $(Z_uZ_{w_i}, Z_vZ_{w_i})$ in the product, and hence their contribution is proportional to s'^6. There are $\binom{f}{3}$ many such terms. Thus, for the general case, we have

$$\begin{aligned}\langle s|e^{i\gamma C}Y_uY_ve^{-i\gamma C}|s\rangle &= \binom{f}{1}c'^{d+e-2}s'^2 + \binom{f}{3}c'^{d+e-6}s'^6 + \binom{f}{5}c'^{d+e-10}s'^{10} + \dots \\ &= c'^{d+e-2f}\sum_{i=1,3,5,\dots}^{f}\binom{f}{i}(c'^2)^{f-i}(s'^2)^i. \end{aligned} \tag{5.14}$$

To sum this series, we twice apply the binomial theorem to yield

$$\sum_{i=1,3,\dots}^{f}\binom{f}{i}a^{f-i}b^i = \frac{1}{2}((a+b)^f - (a-b)^f).$$

Thus the above sum becomes

$$\sum_{i=1,3,\dots}^{f}\binom{f}{i}(c'^2)^{f-i}(s'^2)^i = \frac{1}{2}(c'^2+s'^2)^f - (c'^2-s'^2)^f = \frac{1}{2}(1-\cos^f 2\gamma),$$

which gives

$$\langle s|e^{i\gamma C}Y_uY_ve^{-i\gamma C}|s\rangle = \frac{1}{2}c'^{d+e-2f}(1-\cos^f 2\gamma). \tag{5.15}$$

Combining the above results with basic trigonometric identities gives (5.10).

Finally, summing $\langle C_{uv}\rangle$ over the set of edges gives (5.11). □

5.4.3 Triangle-Free Graphs

For triangle-free graphs, Theorem 4 yields simple results for $\langle C\rangle$ that are particularly amenable to further analysis. We first consider the case of graphs of fixed vertex degree.

Corollary 2. *For a D-regular triangle-free graph, for QAOA$_1$ we have*

$$\langle C\rangle = \frac{m}{2} + \frac{m}{2}\sin 4\beta\sin\gamma\cos^{D-1}\gamma, \tag{5.16}$$

with maximum value

$$\max_{\gamma,\beta}\langle C\rangle = \frac{m}{2} + \frac{m}{2}\frac{1}{\sqrt{D}}\left(\frac{D-1}{D}\right)^{(D-1)/2} =: C^{reg}_{max}(D), \tag{5.17}$$

and approximation ratio satisfying

$$\max_{\gamma,\beta} \langle R\rangle > \frac{1}{2} + \frac{1}{2\sqrt{e}}\frac{1}{\sqrt{D}} = \frac{1}{2} + \Omega(\frac{1}{\sqrt{D}}). \tag{5.18}$$

Proof. Theorem 4 gives the first equation directly, and the second equation then follows applying simple calculus. (The optimal angles are given below in Theorem 5.) For the third equation we use the bounds $\langle R\rangle \geq \langle C\rangle/m$ and $\left(\frac{d}{d+1}\right)^d > 1/e$. □

In particular, (5.17) gives $\max_{\gamma,\beta}\langle R\rangle \geq 0.75,\ 0.69245,\ 0.66238, 0.64310$ for $D = 2, 3, 4, 5$, respectively. The cases $D = 2, 3$ reproduce results found by numerical simulation in [86].

We next consider arbitrary triangle-free graphs. As any edges with a degree one vertex can always be trivially cut, we may assume the minimal vertex degree is at least two. Observing that (5.16) can only decrease with D for fixed γ, β, we have the following corollary.

Corollary 3. *For an arbitrary triangle-free graph with maximum degree D_G, and n_D vertices of degree D, $D = 2, 3, \ldots, D_G$, for QAOA$_1$ we have*

$$\langle C\rangle = \frac{m}{2} + \frac{1}{4}\sin 4\beta \sin\gamma \sum_D D\, n_D \cos^{D-1}\gamma, \tag{5.19}$$

for which the previous corollary gives the lower bound

$$\max_{\alpha,\beta}\langle C\rangle \geq C^{reg}_{max}(D_G) > \frac{m}{2} + \frac{m}{2\sqrt{eD_G}}, \tag{5.20}$$

and hence the expected approximation ratio satisfies

$$\max_{\gamma,\beta}\langle R\rangle > \frac{1}{2} + \frac{1}{2\sqrt{e}}\frac{1}{\sqrt{D_G}}. \tag{5.21}$$

Thus, $\max_{\alpha,\beta}\langle C\rangle > \frac{m}{2}$, so QAOA always beats random guessing on triangle-free graphs, i.e., there always exist angles γ, β with expected approximation ratio strictly greater than $\frac{1}{2}$. The result (5.21) follows from (5.20) using $\langle R\rangle \geq \langle C\rangle/m$, which is a relatively crude lower bound to the expected approximate ratio. Indeed, there exist families of triangle-free graphs with m edges such that the best possible cut contains only $C^* = \frac{m}{2} + \Theta(m^{4/5})$ edges [11]; clearly, the approximation ratio will be much higher on such instances. (On the other hand, an m-edge triangle-free graph could be bipartite or nearly bipartite, so we can have graphs with $C^* \simeq m$ in the worst case.)

Using the corollaries, we classify the optimal QAOA$_1$ angles for MaxCut on triangle free graphs.

Optimal Angles

Recall a pair of angles (γ^*, β^*) is *optimal* if they maximize the lower bound to the expected approximation ratio $\langle R \rangle$. For our purposes here we consider angles optimal if they maximize $\langle C \rangle / m$.

Theorem 5. *For QAOA$_1$ applied to MaxCut on any triangle-free graph, the optimal angles maximizing $\langle C \rangle$ (or, equivalently, maximizing $\langle C \rangle / m$) satisfy the following:*

- *For a D-regular triangle-free graph, the unique smallest positive optimal pair of angles is*

$$(\gamma^*,\ \beta^*) := (\arctan \frac{1}{\sqrt{D-1}}, \pi/8), \tag{5.22}$$

  for $D \geq 2$ (i.e., no other optimal pair (γ, β) exists with $0 \leq \gamma \leq \gamma^*$ or $0 \leq \beta \leq \beta^*$).

  All optimal angles are periodic in γ, β with periodicity depending on D:

  – *If D is even, there is a second independent pair of optimal angles given by $(-\gamma^*, -\beta^*)$, independent in the sense that all optimal angles are generated from these two pairs as*

$$(\gamma^* + a\pi,\ \beta^* + b\frac{\pi}{2}), \qquad (-\gamma^* + c\pi,\ -\beta^* + d\frac{\pi}{2}), \qquad a, b, c, d \in \mathbb{Z}. \tag{5.23}$$

  – *Else if D is odd, there are four independent pairs of optimal angles (γ^*, β^*), $(-\gamma^*, -\beta^*)$, $(\pi - \gamma^*, \beta^*)$, and $(\pi + \gamma^*, -\beta^*)$, and all optimal angles are generated from one of these pairs, denoted (γ', β'), as*

$$(\gamma' + a2\pi,\ \beta' + b\frac{\pi}{2}), \qquad a, b \in \mathbb{Z}. \tag{5.24}$$

- *For an arbitrary triangle-free graph with maximum vertex degree D_G and minimum vertex degree D_{min}, the smallest positive optimal angles $\gamma^*, \beta^* \in [0, \pi/2]$ satisfy*

$$\arctan \frac{1}{\sqrt{D_G - 1}} \ \leq\ \gamma^* \ \leq\ \arctan \frac{1}{\sqrt{D_{min} - 1}}, \qquad \beta^* = \frac{\pi}{8}. \tag{5.25}$$

  Given such a pair, the angles $(-\gamma^*, -\beta^*)$ are also optimal, and both pairs are 2π-periodic in the first argument and $\pi/2$-periodic in the second, with respect to optimality.

The proof is given in Appendix E. The theorem implies the smallest optimal angles are $(\pi/4, \pi/8)$ and $(0.6155, \pi/8)$ for 2-regular and 3-regular triangle-free graphs, respectively, which reproduces results found numerically in [86]. Plugging the angles (5.22) into (5.16) gives (5.17).

5.4.4 General Graphs

For a general graph with bounded degree, combining previous results we have the following lemma.

Lemma 2. *For QAOA$_1$ applied to MaxCut on a graph G with maximum vertex degree D_G, and containing F triangles, in addition to* $\max_{\gamma,\beta}\langle C\rangle \geq \frac{m}{2}$ *we have*

$$\max_{\gamma,\beta}\langle C\rangle \geq \frac{m}{2} + \frac{m}{2\sqrt{e}}\frac{1}{\sqrt{D_G}} - O\left(\frac{F}{D_G}\right). \tag{5.26}$$

Proof. Clearly, $\max_{\gamma,\beta}\langle C\rangle \geq \langle\gamma^*\beta^*|C|\gamma^*\beta^*\rangle$, where $\gamma^* = \arctan\frac{1}{D_G-1}, \beta^* = \pi/8$ are taken from (5.25), i.e., pretending the graph had no triangles. Plugging γ^*, β^* into (5.10), it is straightforward to derive (5.26). Note that the right-hand side may become less than $m/2$ as the number of triangles F becomes large; this shows that for such cases the angles γ^*, β^* are no longer good choices. Indeed, setting $\gamma = 0 = \beta$, we can always obtain $\langle C\rangle = \frac{m}{2}$ for any graph. □

We remark that there exist families of m-edge graphs such that the best possible cut contains $C^* = \frac{m}{2} + \sqrt{\frac{m}{8}} + O(m^{1/4})$ edges [11,12]. Clearly, the expected approximation ratio will be much higher on such instances than $\langle C\rangle/m$.

Thus, for general graphs, we have shown that the performance of QAOA depends strongly on the graph topology of a given instance. Using Lemma 2 it is straightforward to derive the following lower bound on the performance of QAOA on general graphs.

Theorem 6. *For QAOA$_1$ applied to MaxCut on a graph G with bounded maximum vertex degree $D_G = O(1)$, we have*

$$\max_{\gamma,\beta}\langle R\rangle \geq \frac{1}{2} + \frac{1}{2\sqrt{e}}\frac{1}{\sqrt{D_G}} - O\left(\frac{1}{D_G}\right). \tag{5.27}$$

Proof. The proof follows from Lemma 2 using the bound $\langle R\rangle \geq \langle C\rangle/m$ and the fact that for graphs with bounded degree $D_G = O(1)$, the number of triangles F in the graph is $O(m)$. □

Remark 15. *It is worthwhile to elaborate on the expected approximation ratio lower bounds (5.18), (5.21), and (5.27). For graphs with large maximum degree D_G, these bounds become close to $1/2$, which is the approximation ratio obtained for MaxCut by random guessing. These bounds are not competitive with the best classical algorithm known, the Goemans-Williamson algorithm [105] based on semidefinite programming, which achieves an approximation ratio of 0.8785, independently of D_G. Moreover, under a plausible conjecture from computational complexity theory, no polynomial-time classical algorithm can do better than this in general.*

Indeed, the Goemans-Williamson algorithm, published in 1995, was a huge breakthrough for the MaxCut problem, after 20 years of relative standstill. Previously, a (deterministic) $1/2$*-approximation had been found [203], which despite much effort, led to a sequence of algorithms with relatively minor improvements (i.e., no improvement to the* $1/2$ *factor), yielding approximation ratios* $R = \frac{1}{2} + \frac{1}{2m}$, $R = \frac{1}{2} + \frac{1}{2n}$, $R = \frac{1}{2} + \frac{n-1}{4m}$, *and* $R = \frac{1}{2} + \frac{1}{D_G}$, *for graphs with* n *vertices and* m *edges [117,130,191,234]. (See [105] for an insightful discussion on the history of approximation algorithms for MaxCut.) Thus, we take the results (5.18), (5.21) and (5.27) as important positive indicators that quantum computers may be useful for approximating hard optimization problems such as MaxCut. Whether QAOA, or another quantum approximation algorithm, can improve upon the result of [105] remains a tantalizing open problem.*

Remark 16. *For each class of graphs studied, the obtained lower bounds to* $\langle R \rangle$ *decrease as the maximum vertex degree* D_G *increases. This suggests that as* D_G *increases, we may need to take higher* p *for QAOA*$_p$ *to obtain the same performance as on graphs with smaller* D_G.

5.4.5 Depth-Two QAOA for the Ring of Disagrees

While it is in principle straightforward to extend our results for MaxCut to QAOA$_p$ with $p > 1$, the number of terms in the analysis quickly becomes prohibitive for direct calculation. The expectation value $\langle C_{uv} \rangle$ for each edge (uv) will now depend on its p-local graph topology (i.e., the subgraph induced from the set of vertices within edge-distance p of u or v), which becomes difficult to succinctly characterize as p increases. We show here how even for $p = 2$, and for one of the simplest possible graphs, the number of terms is daunting.

Consider MaxCut on a 2-regular connected graph, called the *ring of disagrees*, which is a useful toy problem for QAOA studied in [86, 238]. In physics, it is equivalently described as the closed one-dimensional chain of spin-1/2 particles with nearest-neighbour antiferromagnetic couplings. Assume n is even, so the optimal cut size is trivially seen to be $C^* = m$. For a given choice of angles γ, β, the expected approximation ratio is given by $\langle R \rangle = \langle C \rangle / m$. In [86], the optimal expected approximation ratios for this problem were found numerically to be $3/4$ and $5/6$ for QAOA$_1$ and QAOA$_2$, respectively.

In [238], we study MaxCut on the ring of disagrees using a different approach inspired from physics. By reformulating the QAOA mapping for this problem to a fermionic representation using the Jordan-Wigner transformation [160], we show that the parameterized QAOA evolution translates equivalently into the quantum control of an ensemble of independent spins, significantly simplifying the analysis. This, in principle, allows for the optimal expected approximation ratio to be computed for arbitrary p; see [238] for details. Furthermore, we show how symmetries satisfied by the optimal angles can make finding such angles much easier. Unfortunately, it is unclear whether or not this fermionic approach can be extended to derive performance bounds or parameter setting strategies for MaxCut on general graphs. Here, we again apply our Pauli Solver algorithm, but with $p = 2$.

For QAOA$_1$, Corollary 2 and Theorem 5 reproduce the optimal expected approximation ratio of $3/4$ for the ring of disagrees, with the optimal angle pairs for $\gamma, \beta \in [0, \pi)$ given by

$$(\gamma^*, \beta^*) = (\frac{\pi}{4}, \frac{\pi}{8}),\ (\frac{\pi}{4}, \frac{5\pi}{8}),\ (\frac{3\pi}{4}, \frac{3\pi}{8}),\ (\frac{3\pi}{4}, \frac{7\pi}{8}).$$

For QAOA$_2$ on the ring of disagrees, we consider maximization of the quantity

$$\langle C\rangle := \langle \boldsymbol{\gamma}, \boldsymbol{\beta}|C|\boldsymbol{\gamma}\boldsymbol{\beta}\rangle = \sum_{(uv)\in E} \langle s|e^{i\gamma_1 C}e^{i\beta_1 B}e^{i\gamma_2 C}e^{i\beta_2 B}C_{uv}e^{-i\beta_2 B}e^{-i\gamma_2 C}e^{-i\beta_1 B}e^{-i\gamma_1 C}|s\rangle.$$

Using the Pauli Solver algorithm and some trigonometric simplifications, the expected approximation ratio is found to be

$$\langle R\rangle = \frac{1}{2} + f(\gamma_1, \beta_1, \gamma_2, \beta_2), \tag{5.28}$$

where the function $f(\gamma_1, \beta_1, \gamma_2, \beta_2)$ is given by

$$\begin{aligned}
f(\gamma_1, \beta_1, \gamma_2, \beta_2) \;=\;& \frac{1}{128} * \Big(4\cos(2(\gamma_1 - 2\beta_1)) + 4\cos(2(\gamma_2 - 2\beta_1)) - 4\cos(2(\gamma_1 + 2\beta_1)) \\
-\;& 4\cos(2(\gamma_1 + \gamma_2 - 2\beta_1)) - 4\cos(2(\gamma_2 + 2\beta_1)) + 4\cos(2(\gamma_1 + \gamma_2 + 2\beta_1)) \\
-\;& 6\cos(2(\gamma_1 - \gamma_2 - 2\beta_2)) + 4\cos(2(\gamma_2 - 2\beta_2)) + 6\cos(2(\gamma_1 + \gamma_2 - 2\beta_2)) \\
-\;& 6\cos(2(\gamma_1 + 2\beta_1 - 2\beta_2)) + 3\cos(2(\gamma_1 - \gamma_2 + 2\beta_1 - 2\beta_2)) \\
+\;& 3\cos(2(\gamma_1 + \gamma_2 + 2\beta_1 - 2\beta_2)) + 6\cos(2(\gamma_1 - \gamma_2 + 2\beta_2)) \\
-\;& 4\cos(2(\gamma_2 + 2\beta_2)) - 6\cos(2(\gamma_1 + \gamma_2 + 2\beta_2)) + 6\cos(2(\gamma_1 - 2\beta_1 + 2\beta_2)) \\
-\;& 3\cos(2(\gamma_1 - \gamma_2 - 2\beta_1 + 2\beta_2)) - 3\cos(2(\gamma_1 + \gamma_2 - 2\beta_1 + 2\beta_2)) \\
+\;& 6\cos(2(\gamma_1 - 2(\beta_1 + \beta_2))) + 3\cos(2(\gamma_1 - \gamma_2 - 2(\beta_1 + \beta_2))) \\
-\;& 4\cos(2(\gamma_2 - 2(\beta_1 + \beta_2))) + 7\cos(2(\gamma_1 + \gamma_2 - 2(\beta_1 + \beta_2))) \\
-\;& 6\cos(2(\gamma_1 + 2(\beta_1 + \beta_2))) - 3\cos(2(\gamma_1 - \gamma_2 + 2(\beta_1 + \beta_2))) \\
+\;& 4\cos(2(\gamma_2 + 2(\beta_1 + \beta_2))) - 7\cos(2(\gamma_1 + \gamma_2 + 2(\beta_1 + \beta_2)))\Big).
\end{aligned}$$

Thus. even for a simple graph such as the ring of disagrees, the expected output of QAOA$_2$ depends on the angles $\gamma_1, \gamma_2, \beta_1, \beta_2$ in a relatively complicated way. Optimizing this function gives

$$\max_{\gamma_1,\gamma_2,\beta_1,\beta_2} \langle R\rangle = 0.8333,$$

confirming the value found numerically in [86]. A particular set of optimal angles is $(\gamma_1, \gamma_2, \beta_1, \beta_2) = (0.655871, 0.62143, 1.24286, 0.327935)$, with this choice satisfying $\gamma_2 = 2 * \beta_1$, $\gamma_1 = 2 * \beta_2$, and $\gamma_2 + \beta_2 = \pi/2$ (or, equivalently, $\gamma_1 + 4\beta_1 = \pi$). These symmetry conditions may be explicitly enforced to yield a simpler expression for $f(\gamma_1, \beta_1, \gamma_2, \beta_2)$.

The optimal angles in general can again be seen to obey periodicity conditions. Suppose $(\gamma_1^*, \beta_1^*, \gamma_2^*, \beta_2^*)$ is a set of optimal angles. Then the angles $(\gamma_1^*+a\pi, \beta_1^*+b\pi/2, \gamma_2^*+c\pi, \beta_2^*+d\pi/2)$ are also optimal for $a, c, b, d \in \mathbb{Z}$.

Clearly, the result (5.28) indicates that bounding the performance of QAOA$_2$ for MaxCut on general graphs, or even relatively simple subclasses of graphs, is a nontrivial task. We leave this important question as an open problem for future research.

5.4.6 Weighted Maximum Cut

A natural generalization is *Weighted MaxCut* where we are given a weight w_{uv} for each edge and seek a partition such that the sum of the weights of edges crossing the partition is maximized. The corresponding weighted decision problem was one of Karp's original NP-complete problems [145] (shown via a reduction from the number partitioning problem).

The QAOA construction is the same as MaxCut, up to the multiplicative weights in the objective Hamiltonians, which become

$$C = \sum_{(uv)\in E} C_{uv}, \qquad C_{uv} = \frac{w_{uv}}{2}(I - Z_u Z_v).$$

The case $w_{uv} = 1$ gives MaxCut. Generalizing our previous approach gives the following result.

Theorem 7. *Consider QAOA$_1$ applied to Weighted MaxCut on a triangle-free graph. Letting $\gamma_{uv} := \gamma w_{uv}$ and $W := \sum_{(uv)\in E} w_{uv}$, the overall expectation value $\langle\gamma\beta|C|\gamma\beta\rangle$ is*

$$\frac{W}{2} + \frac{\sin 4\beta}{4} \sum_{(uv)\in E} w_{uv} \sin\gamma_{uv} \left(\prod_{t\in \mathrm{nbhd}(u)\backslash\{v\}} \cos\gamma_{ut} + \prod_{w\in \mathrm{nbhd}(v)\backslash\{u\}} \cos\gamma_{vw} \right). \qquad (5.29)$$

The proof of Theorem 7 follows similarly to the proof of Theorem 4, so we omit its details. We remark that it is also possible to extend Theorem 7 to general graphs. However, in this case, the term in $\langle C_{uv}\rangle$ which is non-zero for triangles (i.e., analogous to (5.14) in the proof of Theorem 4) will now depend on the graph weights, and can no longer be collapsed in general to a succinct result after summing over the edges as in (5.15). This results in a substantially more complicated formula for $\langle C\rangle$. As a simple example, we consider the ring of disagrees with weights.

Corollary 4. *For the weighted ring of disagrees, for an edge (uv) with adjacent edges (tu) and (vx) we have*

$$\langle\gamma\beta|C_{uv}|\gamma\beta\rangle = \frac{w_{uv}}{2} + \frac{w_{uv}}{2}\sin 4\beta \sin\gamma w_{uv} \cos\frac{\gamma(w_{tu}+w_{vx})}{2}\cos\frac{\gamma(w_{tu}-w_{vx})}{2}.$$

Importantly, for a given a problem instance, (5.29) may be efficiently maximized classically, with optimal value and angles depending on the particular problem graph and weights.

Finally, given further information such as the distribution of weights w_{uv} over a given class of problem instances, it may be possible to use Theorem 7 to obtain further quantities of interest, e.g., the expected value $\langle C\rangle$ taken with respect to this distribution.

5.4.7 Directed Maximum Cut

Another natural generalization is MaxCut on directed graphs. In Directed MaxCut (MaxDiCut), we are given a *directed* graph $G = (V, E)$ and we seek to find a subset of the vertices $L \subset V$, $R = V \setminus L$, such that the number of directed edges $|uv)$ with $u \in L$ and $v \in R$ is maximized. Here we use $|uv)$ to indicate the directed edge from u to v.

We emphasize that only directed edges from L to R, and not those from R to L, are counted by the objective function. Furthermore, the usual problem formulation considers weighted edges, and we seek to maximize the total edge weight from L to R. For simplicity, for the remainder of the section we take all weights to be 1. It is relatively straightforward to obtain a weighted version of the theorem below, similar to (undirected) Weighted MaxCut.

Suppose each vertex $u \in V$ is the left endpoint of ℓ_u-many edges and the right endpoint of r_u-many edges, and define $k_u = \ell_u - r_u$. Then $\sum_u \ell_u = \sum_u r_u = m$, and $\sum_u k_u = 0$.

Let the indicator variable x_u be 1 if vertex u is assigned to R. The possible vertex partitions are again encoded with n qubits. Recall that for MaxCut, the objective function for each edge

$x_u \oplus x_v = \bar{x}_u x_v + x_u \bar{x}_v$ was encoded as the Hamiltonian $\frac{1}{2}(I - Z_u Z_v)$. For MaxDiCut, the objective function counts the number of edges strictly leaving L, which for an edge $|uv)$ becomes

$$C_{uv} = \bar{x}_u x_v = \frac{1}{4}(I + Z_u - Z_v - Z_u Z_v).$$

Observe that if both $|uv)$ and $|vu)$ are in the graph, then $C_{uv} + C_{vu} = \frac{1}{2}(I - Z_u Z_v)$, i.e., the combination $C_{uv} + C_{vu}$ acts exactly as an undirected edge in the sense of (5.9). (However, in this case at most one of $|uv)$ or $|vu)$ can be cut.) We use this to simplify the objective Hamiltonian.

Partition the edge set E into the sets we call the *directed* and *undirected* edges

$$D := \{|uv) \in E : \ |vu) \notin E\}, \qquad U := \{(uv) : \ |uv) \in E \text{ and } |vu) \in E\},$$

where a pair of edges $|uv), |vu)$ is included as a single element $(uv) \in U$. Note that $|D| + 2|U| = |E| = m$. Then the objective Hamiltonian $C = \sum_{|uv)\in E} C_{uv}$ becomes

$$C = \frac{m}{4} I + \frac{1}{4} \sum_{u \in V} k_u Z_u - \frac{1}{4} \sum_{|uv)\in D} Z_u Z_v - \frac{1}{2} \sum_{(uv)\in U} Z_u Z_v. \tag{5.30}$$

Observe that the ZZ terms are symmetric with respect to flipping the edge directions, and all information about the direction of each edge lies in the $\frac{1}{4}\sum_{u\in V} k_u Z_u$ term.

The analysis of QAOA$_1$ for MaxDiCut is similar to that of MaxCut, but considerably more complicated. We give results for oriented graphs and triangle-free directed graphs, which are two more manageable cases; it is relatively straightforward but complicated to extend the proof of the theorem below to arbitrary graphs. A graph is *oriented* if it contains no symmetric edge pairs $|uv)$ and $|vu)$, i.e., $U = \emptyset$ and $D = E$. A *triangle* in a directed graph is defined to be any subset of three edges forming a cycle when the direction of each edge is ignored.

For each $u \in V$, let $D_u \subset D$ be the edges in D containing u, let $U_u \subset U$ be the edges in U containing u, and define $d_u = |D_u|$ and $e_u = |U_u|$.

Theorem 8. *Consider QAOA$_1$ applied to MaxDiCut, with the quantities d_u, e_u, and k_u for each vertex u defined as above.*

- *For an oriented graph, let f_{uv} be the number of triangles containing an edge $|uv)$. Then*

$$\langle\gamma\beta|C|\gamma\beta\rangle = \frac{m}{4} + \frac{1}{4}\sum_{u\in V} k_u \mathcal{K}_u + \frac{1}{8}\sum_{|uv)\in D} (\mathcal{D}_u + \mathcal{D}_v - \mathcal{T}_{uv}), \tag{5.31}$$

where we have the quantities

$$\begin{aligned}
\mathcal{K}_u &= \sin(2\beta)\sin(\gamma k_u/2)\cos^{d_u}(\gamma/2),\\
\mathcal{D}_u &= \sin(4\beta)\sin(\gamma/2)\cos(\gamma k_u/2)\cos^{d_u-1}(\gamma/2),\\
\mathcal{T}_{uv} &= \sin^2(2\beta)\cos^{d_u+d_v-2-2f_{uv}}\left(\tfrac{\gamma}{2}\right)\Big(\cos\left(\tfrac{\gamma}{2}(k_u-k_v)\right)-\cos^{f_{uv}}(\gamma)\cos\left(\tfrac{\gamma}{2}(k_u+k_v)\right)\Big).
\end{aligned}$$

- *For a triangle-free directed graph, the overall expectation value is*

$$\langle\gamma\beta|C|\gamma\beta\rangle = \frac{m}{4}+\frac{1}{4}\sum_{u\in V}k_uK'_u+\frac{1}{8}\sum_{|uv)\in D}\mathcal{D}_{uv}+\frac{1}{4}\sum_{(uv)\in U}\mathcal{U}_{uv}, \tag{5.32}$$

where we now have the quantities $K'_u = \cos^{e_u}(\gamma)K_u$,

$$\begin{aligned}
\mathcal{D}_{uv} &= \mathcal{D}_u\cos^{e_u}(\gamma)+\mathcal{D}_v\cos^{e_v}(\gamma)\\
&- \sin^2(2\beta)\sin(\tfrac{\gamma k_u}{2})\sin(\tfrac{\gamma k_v}{2})\cos^{d_u+d_v-2}(\tfrac{\gamma}{2})\cos^{e_u+e_v}(\gamma),\\
U_{uv} &= \sin(4\beta)\sin(\gamma)\Big(\cos(\tfrac{\gamma k_u}{2})\cos^{d_u}(\tfrac{\gamma}{2})\cos^{e_u-1}(\gamma)+\cos(\tfrac{\gamma k_v}{2})\cos^{d_v}(\tfrac{\gamma}{2})\cos^{e_v-1}(\gamma)\Big)\\
&- \sin^2(2\beta)\sin(\tfrac{\gamma k_u}{2})\sin(\tfrac{\gamma k_v}{2})\cos^{d_u+d_v}(\tfrac{\gamma}{2})\cos^{e_u+e_v-2}(\gamma).
\end{aligned}$$

The proof is again based on the Pauli Solver approach and is similar to but more involved than that of Theorem 4. The proof details are given in Appendix E.

In principle, further results for MaxDiCut may be derived using the theorem, as was done for MaxCut. However, the same difficulties apply in analyzing the performance for $p > 1$. We leave further investigation of the performance of QAOA for MaxDiCut as a direction of future research.

5.5 Discussion

Using quantum computers to approximately solve hard optimization problems is an exciting new theoretical direction. Unfortunately, the most important problems remain open, and, as we have demonstrated, appear to elude simple resolutions.

The results in this chapter are important first steps towards a fuller understanding of the QAOA algorithm. While we have not yet answered the most critical question, namely to characterize the performance of QAOA$_p$ generally, we have made important progress, improving significantly on known results for QAOA$_1$ applied to MaxCut. In particular, we have shown exact analytic formulas for several results given in [86] that were found numerically. Moreover, we have presented the *Pauli Solver* algorithm, which can in principle be used to derive performance bounds for a wider variety of problems, and can potentially assist in finding good QAOA parameters.

As gate model quantum computers begin to come online over the next several years, we expect experimentation and empirical analysis to enable a significant expansion of our knowledge of quantum approximation algorithms and heuristics. In particular, smaller quantum computers may be useful for characterizing the performance of QAOA on larger quantum computers. For example, for problems consisting of clauses C_ℓ each acting on a bounded number of variables $k \ll n$, such as MaxCut, a relatively small quantum computer (requiring much fewer than n qubits) could be used to compute the quantities $\langle C_\ell \rangle$ for QAOA$_p$ with fixed p. These quantities then in turn could be used to characterize the expected QAOA output for the objective function $C = \sum_\ell C_\ell$ on n variables, i.e., the performance of QAOA$_p$ on a much larger problem instance (to be executed on a larger n qubit quantum computer).

Generally, the performance of QAOA in practice will be highly dependent on the ability to find good angles. These may be found in advance through analysis similar to our demonstrated techniques, or found on an instance-by-instance basis by incorporating searching over angles as part of the QAOA algorithm. Finding further techniques for reducing the cost of this search is important towards improving the efficacy of the algorithm. The most important question is how QAOA performs (i.e., how the optimized approximation ratio scales) for $p > 1$. In particular, results for $p = O(1)$ are enticing for application to early quantum computers, and results for, say, $p = O(\log(n))$ or $p = \text{poly}(n)$ are important for characterizing the power of QAOA itself. Unfortunately, we give evidence in this chapter that deriving performance bounds for $p > 1$ is a difficult problem. We are, however, optimistic that the techniques of this chapter may be used to study MaxCut further, or to analyze the application of QAOA to other problems.

In the next chapter, we give a generalization of QAOA that is particularly suitable to low-resource implementations for *constrained* optimization problems. The techniques presented in this chapter similarly apply to the analysis of constrained problems; however, for these cases, the QAOA constructions themselves, and likewise the details of the analysis, become further complicated due to the additional constraints. A major breakthrough we leave for future investigation is to find new approaches to analyzing the performance of QAOA that are generally applicable.

Chapter 6

Quantum Approximate Optimization with Hard and Soft Constraints

6.1 Introduction

While some small-scale exploration of quantum algorithms and heuristics for approximate optimization beyond quantum annealing has been possible through classical simulation, the exponential overhead in such simulations has greatly limited their usefulness. The next decade will see a blossoming of quantum algorithms as a broader and more flexible array of quantum computational hardware becomes available. The immediate question is: which algorithms should we prioritize that will give us insight into the power and utility of quantum computers? One leading candidate is the Quantum Approximate Optimization Algorithm (QAOA), for which a number of tantalizing related results have been obtained [87, 91, 138, 231, 238, 241, 254]. As discussed in Chapter 5, QAOA facilitates low-resource implementations for unconstrained optimization problems, although the performance of QAOA for these problems remains open. It is important to derive constructions for even more general classes of problems, where we may not have good classical approximation algorithms at all, that in particular also exhibit low or modest resource requirements. Indeed, implementing such QAOA constructions to find approximate solutions may lead to the first examples of experimental quantum computers performing truly *practically useful* computations.

In this chapter, we formally describe the Quantum Alternating Operator Ansatz (QAOA), extending the approach of Farhi et al. [86] to encompass alternation between more general families of operators.[1] The essence of this extension is to consider the alternation of operators drawn from general parameterized families of unitaries, rather than only those that correspond to the time-evolution of a fixed local Hamiltonian with the time specified by the parameter. Thus, this ansatz supports the representation of a larger and potentially more useful set of states than the original formulation. For cases that call for mixing only within a feasible subspace, refocusing on unitary operators rather than Hamiltonians leads to a variety of possible mixing operators, many of which are much simpler and can be implemented more efficiently than those of the original framework. Such mixers are particularly useful for optimization problems with *hard constraints* that must always be satisfied, defining a feasible subset of solutions, and *soft constraints* which we seek to satisfy as many of as possible. Simple and efficient implementations are especially important towards enabling earlier experimental exploration of quantum alternating operator approaches to a wide variety of potential applications, including approximate optimization, exact optimization, and sampling problems.

We specify a framework for this ansatz, laying out design criteria for constructing initial states, phase operators, and mixing operators. We then detail QAOA mappings of several important optimization problems, including Maximum Independent Set, three graph coloring optimization problems, and the Traveling Salesman problem. The constructions described serve as prototypes for many other optimization problems; a compendium of mappings is included in [114].

For each problem we show how a variety of different mixing operators may be constructed by combining local Hamiltonians and unitaries. In particular, we describe *sequential* (ordered product) mixers, which are similar in form to a Trotterization step, and we show explicit circuits and bound their costs. Our constructions utilizing sequential mixers are simple by design and exhibit relatively low resource scaling, as desired for early quantum hardware. We summarize these implementation results in Table 6.1 below. We emphasize that our ansatz encompasses even more general mixing operators, generally requiring higher implementation cost; investigating the trade-off between increased resource requirements and algorithm performance is an important direction for future work.

It is worthwhile to remark on the relation between these mappings and those for quantum annealing and adiabatic quantum optimization. Because current quantum annealers have a fixed driver Hamiltonian (which is similar to the mixing Hamiltonian in the QAOA setting), all problem depen-

[1] As the Quantum Alternating Operator Ansatz generalizes the Quantum Approximate Optimization Algorithm, by design we use the same acronym *QAOA* for both.

dence must be captured in the objective Hamiltonian on such devices. Hence, to deal with hard constraints, the typical strategy is to add extra terms to the objective Hamiltonian which penalize states encoding infeasible solutions such that these states are avoided; see, e.g., [38, 165, 201]. But this approach means that the algorithm must search a much larger space than would be necessary if the evolution was somehow restricted to feasible configurations. This issue, and other drawbacks, led Hen & Spedalieri [128] and Hen & Sarandy [127] to suggest a different approach for adiabatic quantum optimization in which the standard driver Hamiltonian is replaced by an alternative one that, given a feasible initial state, confines the evolution to the feasible subspace. We apply similar ideas to the quantum circuit model, leading to a much more general approach with wider applicability. Many of the results of this chapter can also be found in [111, 114, 115].

QAOA$_p$ Problem	# of Qubits	# of Basic Gates
Quadratic Unconstrained Binary Optimization	n	$O(p(m+n))$
Max Independent Set	$n+1$	$O(p(m+n))$
Max k-Colorability (Max k-Cut)	kn	$O(pk(m+n))$
Max k-Colorable Induced Subgraph	$(k+1)n+1$	$O(p(km+n))$
Min Chromatic Number $(k = D_G + O(1))$	$(n+1)k+1$	$O(p(k^2m+kn))$
Traveling Salesman	n^2	$O(pn^3)$
Single Machine Scheduling (Min Total Tardiness)	nP	$O(pn^2P)$

Table 6.1: Summary of implementation costs for creating a QAOA$_p$ state for the indicated problems with n variables. The initial state $|s\rangle$, phase operator $U_P(\gamma)$, and mixing operator $U_M(\beta)$ for each problem are given in the following sections. In each case, the mixing operator may be replaced by $U_M^r(\beta)$, $r = O(1)$, without affecting the scaling of the cost estimates. Quadratic Unconstrained Binary Optimization is addressed in Chapter 5, with m specifying the number of quadratic terms. For the last problem, P is the sum of the processing times for each job. Basic gates are defined to consist of CNOT and arbitrary single-qubit gates.

6.2 The Quantum Alternating Operator Ansatz

We formally describe the *Quantum Alternating Operator Ansatz* (QAOA), generalizing the approach of Farhi et al. [86]. QAOA, in our sense, encompasses a more general class of quantum states that may be algorithmically accessible and useful.

We consider here QAOA for approximate optimization problems, though it may also have other applications, such as, for example, exact optimization or sampling problems [91, 138, 241].

An instance of an *optimization problem* is a pair (F, f), where F is the *domain* (set of valid solutions) and $f : F \to \mathbb{R}$ is the *objective function* to be optimized. Recall from Section 5.2.2 that a QAOA mapping is the same between the maximization or minimization versions of a given problem, up to trivial sign flips, and possibly a different choice of initial state. Hence, in this chapter we we will generally not be concerned with which is the case and we consider optimization problems generally.

Earlier in Chapter 5 we considered *Hamiltonian-based QAOA* (H-QAOA) [86], the subclass of QAOA circuits in which both the phase operators $U_{\mathrm{P}}(\gamma) = e^{-i\gamma H_{\mathrm{P}}}$ and the mixing operators $U_{\mathrm{M}}(\beta) = e^{-i\beta H_{\mathrm{M}}}$ correspond to time-evolution under Hamiltonians H_{P} and H_{M}, respectively. The quantum approximate optimization algorithm as originally proposed fits within this paradigm, whereas our construction below encompasses much more general operators; we will see a variety of explicit examples in the subsequent sections.

Let $\mathcal{F}$ be the Hilbert space of dimension $|F|$, whose standard basis we take to be $\{|\mathbf{x}\rangle : \mathbf{x} \in F\}$. A general QAOA circuit is defined by two parameterized families of operators on $\mathcal{F}$:

- a family of *phase separation operators* $U_{\mathrm{P}}(\gamma)$ that depends on the *objective function* f, and
- a family of *mixing operators* $U_{\mathrm{M}}(\beta)$ that depends on the domain and its structure,

where γ and β are real parameters. A QAOA$_p$ circuit consists of p alternating applications of operators from these two families,

$$Q_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) = U_{\mathrm{M}}(\beta_p)U_{\mathrm{P}}(\gamma_p) \cdots U_{\mathrm{M}}(\beta_1)U_{\mathrm{P}}(\gamma_1). \tag{6.1}$$

The Quantum Alternating Operator Ansatz then consists of the states that can be represented (i.e., generated) by the application of such a circuit to a suitably simple initial state $|s\rangle$,

$$|\boldsymbol{\gamma\beta}\rangle = Q_p(\boldsymbol{\gamma}, \boldsymbol{\beta})|s\rangle. \tag{6.2}$$

Hence, for a given optimization problem, a *QAOA mapping* consists of a family of phase separation operators, a family of mixing operators, and a starting state. Once a problem encoding onto qubits is selected, the QAOA mapping can be used to compile directly to a gate-level quantum circuit. Note that different problem encodings lead to different gate and qubit costs.

Constrained optimization problems require optimization over *feasible* solutions, generally a subset of a *configuration space* (such as $\{0,1\}^n$) that is often specified by a set of Boolean predicates called *feasibility* (or *hard*) *constraints*, which are satisfied by feasible solutions. Hard constraints result both from the problem itself and how it is encoded, often specifying nontrivial subsets. For implementation on quantum hardware, it is typically easier to encode the entire configuration space onto qubits, with the problem domain subsumed by the (more general) *feasible subspace*, which results from both the natural structure of the domain and how the configuration space is encoded. For convenience, we will use the terms domain and feasible subspace interchangeably. Quantum states (generally, superpositions) lying entirely in the feasible subspace are called *feasible states*.

For a given problem, our goal is to design families of mixing operators that preserve feasibility; then, given a feasible initial state, the QAOA state will remain feasible always (for all possible algorithm parameters and p), and, in particular, the final QAOA state will be feasible, so any computational basis measurement performed is guaranteed to produce a feasible solution. This avoids all difficulties related to dealing with infeasible states directly; in particular, without this property many measurement outcomes could yield infeasible solutions, which would have to be carefully accounted for in analyzing the success probability and performance of the algorithm, or dealt with by some other means. We formalize this idea as design criteria in Section 6.2.1 below.

For an objective function f we define H_f to be the Hamiltonian that acts as f on basis states

$$H_f|\mathbf{x}\rangle = f(\mathbf{x})|\mathbf{x}\rangle. \tag{6.3}$$

In prior work, the domain F is the set of all n-bit strings, $U_{\mathrm{P}}(\gamma) = e^{-i\gamma H_f}$, and $U_{\mathrm{M}}(\beta) = e^{-i\gamma H_M}$, where, with just one exception, the mixing Hamiltonian is $H_M = \sum_{j=1}^{n} X_j$. Recall X_j denotes the Pauli matrix X acting on the jth qubit, and similarly for Y_j and Z_j. The one exception is Section VIII of [86], which discusses a variant for the Max Independent Set problem, in which F is the set of bit strings encoding independent sets, and the mixing operator is $U_{\mathrm{M}}(\beta) = e^{-i\gamma H_M}$ where

$$\langle \mathbf{x}|H_M|\mathbf{y}\rangle = \begin{cases} 1 & \mathbf{x}, \mathbf{y} \in F \text{ and } \mathrm{Ham}(\mathbf{x}, \mathbf{y}) = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{6.4}$$

The Hamiltonian H_M connects feasible states with Hamming distance one (i.e., independent sets differing only by a single vertex). However, Section VIII of [86] does not discuss the implementability of $U_{\mathrm{M}}(\beta)$. We consider this problem in Section 6.4 and derive the implementation cost shown in Table 6.1 using a closely related mixing operator.

We extend this approach to applying QAOA to constrained optimization problems, with a focus towards implementability, both in the short and long terms. We also build on the ideas developed for adiabatic quantum optimization (AQO) by Hen and Spedalieri [128] and Hen and Sarandy [127], though the gate-model setting of QAOA leads to different implementation considerations than those for AQO. For example, Hen et al. identified Hamiltonians of the form $H_{\mathrm{M}} = \sum H_{j,k}$, where $H_{j,k} = X_j X_k + Y_j Y_k$, as useful in the AQO setting for restricting state evolution to the feasible subspace for certain problems. By incorporating this restriction directly into the mixing operator itself, it was found that the resources required for implementation could be substantially reduced as compared to different approaches for ensuring feasibility, such as the standard approach of including extra terms in the objective Hamiltonian to penalize infeasible states. Analogously, the mixing unitary $U_{\mathrm{M}}(\beta) = e^{-i\beta H_{\mathrm{M}}}$ meets our design criteria, specified in the next section, for applications of QAOA to a number of optimization problems including many of those considered in [127, 128]. Since the Hamiltonians $H_{j,k}$ and $H_{i,l}$ do not commute in general, compiling $U_{\mathrm{M}}(\beta)$ to two-qubit gates is nontrivial. One could Trotterize, or use higher-order splitting formulas, to approximately implement $U_{\mathrm{M}}(\beta)$ in terms of exponentials of individual $H_{j,k}$, which are each efficiently simulatable; recall the discussion of Hamiltonian simulation in Chapter 4. Alternatively, $U_{\mathrm{M}}(\beta)$ can be implemented efficiently using techniques related to the quantum Fourier transform [232]. Instead, in particular, we will propose alternative mixing operators such as

$$U_{\mathrm{M}}(\beta) = e^{-i\beta H_{S_\ell}} \cdots e^{-i\beta H_{S_2}} e^{-i\beta H_{S_1}},$$

where the $H_{j,k}$ have been partitioned into ℓ subsets (partial sums) $S_1, \ldots, S_\ell$ containing only mutually commuting pairs. Such mixing operators may often be selected by design to be much simpler to implement than $e^{-i\beta H_M}$, motivating in part our more general ansatz.

We remark that, clearly, there are obvious further generalizations in which U_{P} and U_{M} are taken from families parameterized by more than a single parameter. Such operator families could be designed to take advantage of specific quantum hardware. For example, in [89] a different free parameter for every term in the mixing Hamiltonian is considered. In this chapter, we consider one-dimensional families of operators, given that this is already a rich area of study, with the task of finding good parameters $\gamma_1, \ldots, \gamma_p$ and $\beta_1, \ldots, \beta_p$ already challenging enough due to the curse of dimensionality [238]. A larger parameter space may support more effective circuits, but further increases the difficulty of finding good parameters.

6.2.1 Design Criteria

Here, we specify design criteria for the three components of a QAOA mapping of a problem, namely, the initial state, the phase operators, and the mixing operators.

Initial State. We require that the initial state $|s\rangle$ be feasible, and moreover it must be *trivial* to implement, by which we mean that it can be created by a constant-depth quantum circuit from the $|0\ldots0\rangle$ state. The standard initial state $|+\cdots+\rangle$ from [86] may be obtained from the $|0\ldots0\rangle$ state by a depth-1 circuit applying a Hadamard gate H to each qubit. For our purposes, it is often convenient to select the initial state to be a single feasible solution $|\mathbf{x}\rangle$, $\mathbf{x} \in F \subset \{0,1\}^n$, which can be prepared by a depth-1 circuit consisting of up to n single-qubit bit-flip operations X. In such cases, the initial phase operator applies a global phase and can be discarded, and hence we may reindex to consider QAOA as starting with a single mixing operator $U_{\mathrm{M}}(\beta_0)$ applied to the basis state $|\mathbf{x}\rangle$. This $0th$ round of QAOA creates a superposition state

$$|s\rangle = U_{\mathrm{M}}(\beta_0)|\mathbf{x}\rangle. \tag{6.5}$$

We remark that the constant-depth criterion could be relaxed to logarithmic depth if needed. It should not be relaxed too much: relaxing the criterion to polynomial depth would obviate the usefulness of the ansatz as a model for a strict subset of states producible via polynomially-sized quantum circuits. Algorithms with more complicated initial states may be considered hybrid algorithms, with an initialization part and a QAOA part.

Mixing unitaries ("Mixers"). We require the family of mixing operators $U_{\mathrm{M}}(\beta)$ to

- *preserve the feasible subspace*: for all values of the parameter β the resulting unitary takes feasible states to feasible states, and
- *explore the feasible subspace*: provide possible transitions between all feasible solutions. More concretely, for any pair of feasible computational basis states $\mathbf{x}, \mathbf{y} \in F$, there is some parameter value β^* and some positive integer r such that the corresponding mixer *connects* those two states: $|\langle \mathbf{x}|U_{\mathrm{M}}^r(\beta^*)|\mathbf{y}\rangle| > 0$. (Note that $U_{\mathrm{M}}^r(\beta)$ denotes $(U_{\mathrm{M}}(\beta))^r$.)

We remark that these criteria are intentionally not overly restrictive, facilitating the design of a variety of mixing operators with different trade-offs. In particular, given a general mixing operator $U_{\mathrm{M}}(\beta)$, applying it r times gives the operator $U_{\mathrm{M}}^r(\beta) \neq U_{\mathrm{M}}(r\beta)$, which may provide transitions between states not connected by any $U_{\mathrm{M}}(\beta)$ alone. If $r = O(1)$, then the increased overhead to implement $U_{\mathrm{M}}^r(\beta)$ is relatively small. Note that for the special case of H-QAOA, the mixing operator $U_{\mathrm{M}}(\beta) = e^{-i\beta H_M}$ satisfies $U_{\mathrm{M}}^r(\beta) = U_{\mathrm{M}}(r\beta)$, so repetitions of the mixing operator do not give any advantages in this case.

Phase separation unitaries. We require the family of phase separation operators $U_{\mathrm{P}}(\gamma)$ to be diagonal in the computational basis. We take

$$U_{\mathrm{P}}(\gamma) = e^{-i\gamma H_f}, \tag{6.6}$$

up to trivial global phase terms which act as $e^{-i\gamma a}I$, $a \in \mathbb{R}$, and may be ignored. In the constructions of this chapter we consider only phase separators where H_f represents the classical objective function f, though more general types of phase separators may be considered (e.g., using $H_{\tilde{f}}$ where $\tilde{f}$ is a simpler to implement approximation of f).

Together, these criteria *restrict state evolution to the feasible subspace*. In particular, all computational basis measurements are guaranteed to return a feasible string. We remark that the restriction to feasible states often allows for substantial simplification of the phase separation operator, reducing its implementation cost, as we shall see in several of the problem constructions we study.

6.2.2 Simultaneous and Sequential Mixers

The implementation of diagonal phase operators of the form (6.6) was addressed in Chapter 5, and we give several more general results in Section 6.3 below. Here, we consider the construction of mixing operators. By deriving simple transformations between states that preserve feasibility, we can map these transformations to Hamiltonians B_j, and then combine the Hamiltonians and their corresponding unitaries $\exp(-i\beta B_j)$ to yield mixing operators satisfying the design criteria.

The original formulation of QAOA considered the domain $\{0,1\}^n$ of all bit strings and used the mixing operator $U_M(\beta) = \exp(-i\beta B)$, with $B = \sum_{j=1}^n X_j$. (For the remainder of this chapter it will be convenient to use B instead of H_M to denote mixing Hamiltonians.) As the X_j mutually commute and each acts on a single qubit, we have $U_M(\beta) = \prod_{j=1}^n e^{-i\beta X_j}$ which may be implemented with n many X-rotation (R_X) gates and depth 1. In the subsequent sections, we give constructions for problems with nontrivial domains. This will result, generally, in mixing Hamiltonians of the form

$$B = \sum_{j=1}^{\ell} B_j,$$

where each B_j acts on a subset of the qubits and $\|B_j\| = O(1)$ (typically, $\|B_j\| = 1$). However, $[B_j, B_k] \neq 0$ in general, so $e^{-i\beta B} \neq \prod_{j=1}^{\ell} e^{-i\beta B_j}$, and more sophisticated Hamiltonian simulation techniques are required to implement $\exp(-i\beta B)$. We refer to Hamiltonian-based mixers of the form $\exp(-i\beta B)$ as *simultaneous mixers*.

Indeed, suppose the decomposition $B = \sum_{j=1}^{\ell} B_j$ satisfies the following properties:

- for each j and for any β, the exponential $\exp(-i\beta B_j)$ can be efficiently implemented, and
- for each j, B_j maps feasible states to feasible states.

Then, for any permutation $j_1, \dots, j_\ell$ of $1, \dots, \ell$, the *sequential mixer* defined as the ordered product

$$U_M(\beta) = e^{-i\beta B_{j_\ell}} \dots e^{-i\beta B_{j_2}} e^{-i\beta B_{j_1}} \tag{6.7}$$

also preserves feasibility. The cost of implementing $U_M(\beta)$ is the cost of implementing the ℓ many $e^{-i\beta B_j}$ operations.

Importantly, different orderings of the exponentials $e^{-i\beta B_j}$ in the product (6.7) result in inequivalent operators. We may associate a sequential mixer to each of the $\ell!$ possible orderings of the $e^{-i\beta B_j}$, some of which result in equivalent operators. If two operators $e^{-i\beta B_j}$ and $e^{-i\beta B_k}$, $j \neq k$, act on disjoint sets of qubits, they may be implemented in parallel, and moreover $e^{-i\beta B_j}e^{-i\beta B_k} = e^{-i\beta(B_j+B_k)}$. Therefore, selecting an ordering where many such pairs are adjacent can significantly reduce the resulting circuit depth. Generally, given a disjoint partition $P = \{P_1, P_2, \ldots, P_\alpha\}$, $\alpha \leq \ell$, of $[\ell] = \{1, \ldots, \ell\}$ (i.e., $\cup_j P_j = [\ell]$ and $P_i \cap P_j = \emptyset$ for $i \neq j$), we define the the *partitioned sequential* mixer to be

$$U_M^{(P)} = \prod_{i=1}^{\alpha} \prod_{j \in P_i} \exp(-i\beta B_j). \tag{6.8}$$

Then, if each P_i contains B_j that act on disjoint sets of qubits, it may be possible to implement $U_M^{(P)}$ with much lower circuit depth than that of an arbitrary partition (i.e., an arbitrary ordering).

On the other hand, it is easy to see that the implementation costs of the sequential and simultaneous mixers are polynomially related. Indeed, using the Strang ($k = 1$) splitting formula, from [186] (cf. equation (4.2)) the cost of simulating $\exp(-i\beta B)$, with $B = \sum_{j=1}^{\ell} B_j$ and $\|B_j\| = O(1)$, is at most N times the maximal cost of any $\exp(-i\beta B_j)$, where the number N of such exponentials is at most

$$N = O(\ell^2\beta(\ell\beta/\varepsilon)^{1/2}).$$

Using higher order splitting formulas ($k > 1$) reduces the exponent above from $1/2$ to $1/2k$. Thus for $\ell = \text{poly}(n)$, $\beta = O(1)$, and accuracy $\varepsilon = 1/\text{poly}(n)$, the cost is polynomial in n. Still more sophisticated Hamiltonian simulation algorithms could be used to reduce the cost dependence on ε to $O(\log(1/\varepsilon))$, though potentially requiring more complicated implementations.

As we are particularly interested in applications to early quantum computers, we will focus on sequential mixers similar to (6.7) and (6.8) for deriving implementation cost estimates in our constructions to follow. Generally, these mixers will be defined up to the order of exponentials in the product, or equivalently, a corresponding partition. As partitions may be selected on an instance-by-instance basis, and, moreover, optimized for compilation to specific gate sets, we will not consider them in detail here; possible partitions and their selection are discussed in [114]. Moreover, as mentioned, each mixer $U_M(\beta)$ we propose can always be replaced by $U_M^r(\beta)$, $r = O(1)$, with r times the implementation cost, for potentially more rapid mixing. With these caveats in mind, it suffices for each of our constructions to specify a single sequential mixer.

Our main technique will be to construct mixers based on *local mixing rules*, which correspond to Hamiltonians B_j acting on a small number of qubits, possibly controlled by a number of other qubits. In each case, the B_j will themselves preserve feasibility, and the operators $\exp(-i\beta B_j)$ will be efficiently implementable. In the remainder of this chapter we demonstrate the ideas of this section by deriving explicit constructions for a variety of problems.

In future applications, as more powerful quantum computing devices come online, simultaneous (i.e., Hamiltonian-based) mixers may be more appealing. It is an important future research problem to quantify the performance of QAOA for given problems with respect to simultaneous or the various possible sequential mixers, and the trade-offs between performance and implementation cost.

6.2.3 Constraint Satisfaction via Commuting Operators

We show here how mixing operators that preserve feasibility may be derived from the commutation properties of the Hamiltonians for a problem and its QAOA construction.

Suppose that, for a given problem, the feasible subspace is specified exactly as the ground state (minimal eigenvalue) eigenspace of a Hamiltonian A, which typically encodes a suitable function, i.e., the hard constraints. For example, we may have $H_A = \sum_j H_{g_j}$, where the H_{g_j} encode Boolean functions g_j such that $\bigvee_j g_j(x) = 0$ if and only if x is feasible. As H_A is diagonal, it trivially commutes with the objective Hamiltonian $[H_A, H_f] = 0$.

Now consider a Hamiltonian-based mixing operator $U_M(\beta) = e^{-i\beta B}$. We require $[B, H_f] \neq 0$, else, clearly, the QAOA dynamics will be trivial. However, if we can select, somehow, B such that $[B, H_A] = 0$, then evolution under linear combinations of B and H_f is guaranteed to not mix between the eigenspaces of H_A. Thus, if the initial state is feasible, the QAOA evolution using such Hamiltonians B and H_f preserves feasibility at all times.

This was observed in [127, 128] for adiabatic quantum optimization, where it was shown that mixing Hamiltonians satisfying these properties could be used instead of penalty Hamiltonian terms, with several advantages in that setting including reduced resource requirements for implementation. We extend these ideas to the quantum gate model with more general unitaries, which provides a useful tool for finding mixing operators satisfying our design criteria.

For a Hamiltonian H_A on n qubits, we say that a unitary operator U *preserves eigenspaces of* H_A if for any eigenvector ψ of H_A with eigenvalue a, $U\psi$ is also an eigenvector of H_A with eigenvalue a. Clearly, this is a stronger condition than preserving feasibility.

Proposition 5. *A unitary operator U preserves eigenspaces of H_A if and only if $[U, H_A] = 0$.*

Proof. Suppose $[U, H_A] = 0$, which is trivially equivalent to $UH_AU^{-1} = H_A$. Then for ψ such that $H_A\psi = a\psi$, we have $H_A(U\psi) = UH_A\psi = a(U\psi)$ as desired. For the other direction, suppose U preserves eigenspaces and consider an arbitrary eigenvector ψ of H_A. Then $[U, H_A]\psi = UH_A\psi - H_AU\psi = aU\psi - aU\psi = 0$. As the eigenvectors (including degeneracy) of a Hermitian operator A give a basis for the Hilbert space of n qubits, this suffices to show $[U, H_A] = 0$. $\square$

Proposition 6. *Consider unitary operators $U = U_\ell U_{\ell-1} \dots U_1$, with $U_j = e^{-i\alpha_j H_j}$, $\alpha_j \in \mathbb{R}$. Then $[H_j, H_A] = 0$ for $j = 1, 2, \dots, \ell$ is a sufficient but not necessary condition for U to preserve eigenspaces of H_A for all α_j.*

Proof. The condition $[H_j, H_A] = 0$ implies $[U_j, H_A] = 0$ from the series expansion of $e^{-i\alpha_j H_j}$, and hence $[U, H_A] = 0$, so U preserves eigenvalues by Proposition 5. This argument holds for any α_j. To see that the conditions are not necessary, consider a single qubit with $H_A = Z$ and $B = X$. Then $U = e^{-i2\pi B} = I$ satisfies $[U, H_A] = 0$, but $[B, H_A] = [X, Z] = -2iY \neq 0$. $\square$

These propositions are general and apply to quantum algorithms beyond QAOA or quantum annealing. For QAOA, Proposition 6 implies that for a mixer $U_M(\beta) = e^{-i\beta H_1} e^{-i\beta H_2} \dots e^{-i\beta H_m}$, a sufficient condition for the quantum state to remain feasible is that the initial state is feasible and $[H_j, H_A] = 0$ for $j = 1, \dots, m$. Thus it suffices to consider Hamiltonian commutators to ensure our mixing operators satisfy the desired design criteria.

We elaborate on how to select and construct such Hamiltonians and give several examples in the remainder of the chapter.

6.3 Design Toolkit for Quantum Optimization

A basic requirement of many quantum algorithms is the ability to translate between mathematical functions acting on a domain, typically a string of bits, and a quantum Hamiltonian operator acting on qubits. Indeed, mapping Boolean and real functions to diagonal Hamiltonians has many important applications in quantum computing, in particular, for algorithms solving decision or optimization problems such as quantum annealing and adiabatic quantum optimization [88, 90, 142], or QAOA. See [114, 165] for a variety of problem mappings.

In this section, we summarize several results which are particularly useful for the QAOA constructions of the remainder of the chapter. Their proof and details are deferred to Appendix B. An expanded presentation of these results appears in [111]. We emphasize that these results have applications to quantum algorithms beyond QAOA. This section is self-contained and the remainder of the chapter may be read independently.

Boolean Functions

Boolean functions can be represented as diagonal Hamiltonians. We show how every such function naturally maps to a Hamiltonian expressed as a linear combination of Pauli Z operators, with terms corresponding to the Fourier expansion of the function. For the (faithful) representation on n qubits, this mapping is unique.

Proposition 7. *For a Boolean function $f : \{0,1\}^n \to \{0,1\}$, the unique Hamiltonian on n-qubits satisfying $H_f|x\rangle = f(x)|x\rangle$ for each computational basis state $|x\rangle$ is*

$$H_f = \sum_{S\subset[n]} \widehat{f}(S) \prod_{j\in S} Z_j = \widehat{f}(\emptyset)I + \sum_{j=1}^{n} \widehat{f}(\{j\})Z_j + \sum_{j<k} \widehat{f}(\{j,k\})Z_jZ_k + \dots \tag{6.9}$$

where the Fourier coefficients $\widehat{f}(S) = \frac{1}{2^n}\sum_{x\in\{0,1\}^n} f(x)(-1)^{S\cdot x} \in [-1,1]$ satisfy

$$\sum_{S\subset[n]} \widehat{f}(S)^2 = \frac{1}{2^n} \sum_{x\in\{0,1\}^n} f(x) = \widehat{f}(\emptyset). \tag{6.10}$$

Here we have used the standard notation $S\cdot x := \sum_{j\in S} x_j$. The proof of the proposition follows from Theorem 10 which is shown in Appendix B.

Thus, computing the Hamiltonian representation (6.9) of a Boolean function is equivalent to computing its Fourier expansion. By considering functions corresponding to NP-hard decision problems, from (6.10) we have the following corollary, which is analogous to the well-known result that deciding equations of Boolean algebra is NP-hard [70].

Corollary 5. *Computing the identity coefficient $\widehat{f}(\emptyset)$ of the Hamiltonian H_f representing an n-variable Boolean satisfiability (SAT) formula f (given in conjunctive normal form and described by* $\text{poly}(n)$ *bits) is #P-hard. Deciding if $\widehat{f}(\emptyset) = 0$ is equivalent to deciding if f is unsatisfiable, in which case H_f is identically (reducible to) the* 0 *matrix.*

Note that the quantity $\widehat{f}(\emptyset)$ is proportional to the trace of H_f and hence is basis independent.

We emphasize that even if we could compute the value of each Fourier coefficient, a Hamiltonian H_f representing a general Boolean function f may require an exponential (with respect to n) number of Pauli Z terms in the sum (6.9). We define the size of H_f, size(H_f), to be the number of (non-zero) terms in the sum (6.9), and the degree $\deg(H_f) = \deg(f)$ to be the maximum locality (number of qubits acted on) of any such term. We say that a function f_n on n-bits is *efficiently representable* as the Hamiltonian H_{f_n} if $\text{size}(H_{f_n})$ is $\text{poly}(n)$ and so is the cost for computing the nonzero coefficients.

Pseudo-Boolean functions

We are particularly interested in real functions f given as weighted sums of Boolean functions f_j,

$$f(x) = \sum_{j=1}^{m} w_j f_j(x) \qquad w_j \in \mathbb{R},$$

where f acts on n bits and $m = \text{poly}(n)$. The objective functions for constraint satisfaction problems, considered in QAOA, are typically expressed in this form. A different example is the penalty term approach of quantum annealing, where the objective function is augmented with a number of high-weight penalty terms which perform local checks to see if a state is feasible.

Note that we do not deal with issues of how the real numbers w_j may be represented and stored. The problems considered in the remainder of this chapter will typically have bounded integer weights, in which case this issue relates to the precision of the QAOA angles.

We have the following useful general result, the proof of which can be found in Appendix B.

Proposition 8. *For an n-bit real function f given as $f(x) = \sum_{j=1}^{m} w_j f_j(x)$, $w_j \in \mathbb{R}$, where the f_j are Boolean functions, the unique Hamiltonian on n-qubits satisfying $H_f|x\rangle = f(x)|x\rangle$ is*

$$H_f = \sum_{S\subset[n]} \widehat{f}(S) \prod_{j\in S} Z_j = \sum_{j=1}^{m} w_j H_{f_j}, \tag{6.11}$$

with Fourier coefficients $\widehat{f}(S) = \frac{1}{2^n}\sum_{x\in\{0,1\}^n} f(x)(-1)^{S\cdot x} = \sum_j \widehat{f}_j(S) \in \mathbb{R}$, where the H_{f_j} are defined as in (6.9).

In particular, $\deg(H_f) \leq \max_j \deg(f_j) =: d$ *and* $\text{size}(H_f) \leq \min\{\sum_j \text{size}(H_{f_j}), 1+(e/d)^{d-1}n^d\}$.

Constructing Hamiltonians

The construction of Hamiltonians representing standard Boolean functions follows directly from Proposition 7. We summarize mappings of important basic clauses in Table 6.2 below.

We derive formal rules for obtaining Hamiltonians representing more complicated expressions such as Boolean formulas or circuits. Together with the basic clauses, Hamiltonians for a large variety of functions can be easily derived.

Theorem 9 (Composition rules)**.** *Let f, g be Boolean functions represented by Hamiltonians H_f, H_g. Then Hamiltonians representing basic operations on f and g are given by*

- $H_{\neg f} = H_{\overline{f}} = I - H_f$
- $H_{f \wedge g} = H_{fg} = H_f H_g$
- $H_{f \oplus g} = H_f + H_g - 2H_f H_g$
- $H_{f \Rightarrow g} = I - H_f + H_f H_g$
- $H_{f \vee g} = H_f + H_g - H_f H_g$
- $H_{af+bg} = aH_f + bH_g \quad a, b \in \mathbb{R}.$

The proof of this theorem is given in Appendix B. Note that the rules of the theorem hold regardless of whether f and g act on the same or independent variables (which correspond to overlapping or independent sets of qubits).

Using these results, Hamiltonians representing objective functions for many important optimization problems can be easily constructed. In particular, we use these results to design phase operators for our QAOA constructions in the remainder of the chapter.

$f(x)$	H_f	$f(x)$	H_f
x	$\frac{1}{2}I - \frac{1}{2}Z$	$\overline{x}$	$\frac{1}{2}I + \frac{1}{2}Z$
$x_1 \wedge x_2$	$\frac{1}{4}I - \frac{1}{4}(Z_1 + Z_2 - Z_1Z_2)$	$\bigwedge_{j=1}^k x_j$	$\frac{1}{2^k}\prod_j(1 - Z_j)$
$x_1 \vee x_2$	$\frac{3}{4}I - \frac{1}{4}(Z_1 + Z_2 + Z_1Z_2)$	$\bigvee_{j=1}^k x_j$	$1 - \frac{1}{2^k}\prod_j(1 + Z_j)$
$x_1 \oplus x_2$	$\frac{1}{2}I - \frac{1}{2}Z_1Z_2$	$x_1 \Rightarrow x_2$	$\frac{3}{4}I + \frac{1}{4}(Z_1 - Z_2 + Z_1Z_2)$

Table 6.2: Hamiltonians representing basic Boolean clauses.

Simulating Diagonal Hamiltonians

It is well known that if a function f can be efficiently computed classically, and if ancilla qubits are available, then the Hamiltonian H_f can be simulated efficiently by computing f in a register and performing controlled rotations; see, e.g., [59]. These methods avoid computing H_f explicitly. On the other hand, there exist applications where a Hamiltonian-based implementation is desirable, such as quantum annealing, or cases where we wish to minimize the need for ancilla qubits, such as, for example, low-resource applications of QAOA.

Efficient circuits simulating products of Pauli Z operators are known as shown in Figure 6.1.

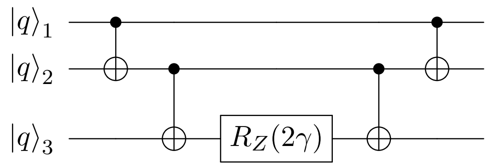


Fig. 6.1: Quantum circuit performing the operation $U = exp(-i\gamma Z_1 Z_2 Z_3)$ on three qubits labeled 1, 2, and 3. The middle operator is a Z-rotation gate, and the other gates are controlled-NOT gates with a black circle indicating the control qubit and cross indicating the target. By similar circuits, $U = exp(-i\gamma Z_1 Z_2 \dots Z_\ell)$ can be implemented with $2(\ell - 1)$ CNOT gates and one R_Z gate. Different circuit compilations are possible, including compilation to different gate sets.

Thus, a Hamiltonian H_f representing a general Boolean or pseudo-Boolean function may be simulated efficiently if the number of Pauli Z-terms in the sums (6.9) or (6.11) are not too many. Recall that we define *basic quantum gates* to be the universal set of CNOT and single-qubit gates.

Corollary 6. *A Hamiltonian H_f as in (6.9) or (6.11) can be simulated (i.e., the operation $exp(-i\gamma H_f)$ implemented for $\gamma = O(1)$) with n qubits and $O(\deg(H_f) \cdot size(H_f))$ basic quantum gates. Thus, if $size(H_f)$ is upper bounded by a polynomial in n, then H_f can be simulated efficiently.*

In particular, Hamiltonians H_f with bounded maximum degree $d = O(1)$ can be simulated with $O(n^d)$ basic gates.

The corollary follows from the above results, the circuits indicated in Figure 6.1, and simple counting arguments. We remark that the Hamiltonian simulation considered in the corollary is exact in the sense that if each of the basic gates is implemented exactly, then so is $exp(-i\gamma H_f)$. The approximation of quantum gates is an important topic but we do not deal with it here; see, e.g., [173]. Moreover, no ancilla qubits are necessary for the simulation.

Controlled Hamiltonians and Unitaries

In many applications we require controlled unitary operations, or, in particular, controlled Hamiltonian simulations. Consider two quantum registers of $k+n$ qubits. Given a k-bit Boolean function $f(y)$ and a unitary operator U acting on n qubits, we define the $(k+n)$-qubit f-controlled unitary operator $\Lambda_f(U)$ by its action on basis states

$$\Lambda_f(U)\,|y\rangle|x\rangle = \begin{cases} |y\rangle|x\rangle & f(y)=0 \\ |y\rangle U|x\rangle & f(y)=1. \end{cases}$$

Proposition 9. *Let f be a Boolean function represented by a k-qubit Hamiltonian H_f, and let H be an arbitrary Hamiltonian acting on n qubits. Then the $(k+n)$-qubit Hamiltonian*

$$\widetilde{H} = H_f \otimes H \tag{6.12}$$

corresponds to f-controlled evolution under H, i.e., satisfies $e^{-i\widetilde{H}t} = \Lambda_f(e^{-iHt})$.

The proof follows from exponentiating (6.12) directly. We will use this result many times in the remainder of the chapter to construct controlled mixing operators for QAOA for constrained optimization problems with feasibility constraints. These operators implement evolution under a local mixing Hamiltonian H only if a Boolean function f is true, where for each basis state f checks that the action of the mixing Hamiltonian will preserve feasibility.

6.4 Mappings on Bits

We first consider problems where the configuration space is naturally expressed as the set of n-bit strings. Recall that *unconstrained* problems, where every string is feasible, were considered in Section 5.3, in particular for the case of quadratic objective functions. Here we consider *constrained* binary optimization, and show how a suitable generalization of the mixing operator facilitates the application of QAOA to such problems.

6.4.1 Max Independent Set

Problem: Given a graph $G = (V, E)$, with $|V| = n$ and $|E| = m$, find the largest cardinality subset $V' \subset V$ of mutually nonadjacent vertices.

No polynomial-time classical algorithm exists for Max Independent Set unless P=NP [226], and the best algorithms known for general graphs give approximations within a polynomial factor.[2] On bounded degree graphs with maximum degree $D_G \geq 3$, Max Independent Set can be approximated to $(D_G + 2)/3$ [23], but remains APX-complete [179].

Our construction generalizes that of Sec. VII of [86]. The configuration space is the set of n-bit strings $x = x_1 x_2 \ldots x_n$ representing subsets of vertices $V' \subset V$, where $i \in V'$ if and only if the indicator variable $x_i = 1$. The domain is the subset of n-bit strings corresponding to independent sets of G. Note that the domain is dependent on the problem instance.

The objective function may be written $f(x) = \sum_{j=1}^{n} x_j$, which counts the number of vertices in V', and maps to the Hamiltonian

$$H_f = \sum_{u \in V} \frac{1}{2}(I - Z_u) = \frac{n}{2} I - \frac{1}{2} \sum_{u \in V} Z_u. \tag{6.13}$$

Dropping the constant (identity matrix) term, which affects the algorithm dynamics only trivially via a global phase, the phase operator e^{-iH_f} becomes

$$U_{\mathrm{P}}(\gamma) = e^{i\frac{\gamma}{2}\sum_{u \in V} Z_u} = \prod_{u \in V} e^{i\frac{\gamma}{2} Z_u} = \prod_{u \in V} R_{Z_u}(-\gamma). \tag{6.14}$$

Clearly $U_{\mathrm{P}}(\gamma)$ can be implemented with n single-qubit (Z-rotation) R_Z gates and depth 1 (the gates do not overlap so can be implemented simultaneously).

We remark that H_f gives the correct value of $f(x)$ on the feasible subspace of states representing independent sets, but gives erroneous values on infeasible states. Typical methods [127,128,165] for dealing with infeasible states require additional complicated Hamiltonian terms to be added to H_f, whereas our approach avoids this. Hence, restricting state evolution to the feasible subspace allows for simple low-cost phase operators.

As an initial state, we may take $|s\rangle = |0\rangle^{\otimes n}$ which encodes the empty set and is assumed to be trivial to prepare. Alternatively, suppose we used a classical algorithm or heuristic to find an

[2] Max Independent set is in fact complete for Poly-APX [23], the class of problems efficiently approximable to within a poly(n) factor.

approximate solution y. Then the initial state $|s\rangle = |y\rangle$ could be used, with cost at most n many X-gates and depth 1, in addition to the cost of the classical preprocessing. Both of these states are clearly feasible, and the latter is problem instance dependent.

Following our design criteria of Sections 6.2.1 and 6.2.2, we define two mixing operators. In order to preserve feasibility, we will utilize controlled quantum operations.

Observe that given an independent set V', adding a vertex $w \notin V'$ to V' preserves feasibility only if none of the neighbours (adjacent vertices) of w' are already in V'. On the other hand, we can always remove any vertex $w \in V'$ without affecting feasibility. Combining these properties in a reversible way gives the following feasibility-preserving transformation rule.

Mixing Rule: flip the bit x_w if and only if $\bar{x}_{v_1}\bar{x}_{v_2}\dots\bar{x}_{v_\ell} = 1$, where $v_1, \dots, v_\ell$ are the vertices adjacent to w. Using the results of Section 6.3, we encode this rule as the Hamiltonian

$$B_u = (\bar{x}_{v_1}\bar{x}_{v_2}\dots\bar{x}_{v_\ell}) \cdot X_u = \frac{1}{2^\ell} X_u \prod_{j=1}^{\ell}(I + Z_{v_j}). \tag{6.15}$$

Exponentials of such Hamiltonians correspond to controlled unitaries. Writing the control predicate as $f_u = \prod_{v\in \mathrm{nbhd}(u)} \bar{x}_v$, we define the operator

$$U_{M,u}(\beta) = e^{-i\beta B_u} = \Lambda_{f_u}(e^{-i\beta X_u}), \tag{6.16}$$

which is a multiqubit controlled rotation that applies an X_u-rotation to a basis state only if the control condition f_u is true. Explicitly, for a basis state $|x\rangle$, $x \in \{0,1\}^n$, $U_{M,u}$ acts as

$$U_{M,u}(\beta)|x\rangle = (f_u(x)\cos\beta + \overline{f_u}(x))|x\rangle - i f_u(x)\sin\beta|x_1 \dots x_{u-1}\overline{x}_u x_{u+1}\dots x_n\rangle, \tag{6.17}$$

so clearly each $U_{M,u}(\beta)$ preserves feasibility. Furthermore, clearly a sequence of such transformations connects every independent set to the empty set, and vice versa, so there exists a sequence connecting any two feasible states.

From the Hamiltonian $B = \sum_u B_u$, we define two related but inequivalent mixers:

- the *simultaneous* (Hamiltonian-based) controlled-X mixer $U_M^{(H)}(\beta) = e^{-i\beta B}$,
- the *sequential* (partitioned) controlled-X mixer $U_{\mathrm{M}}(\beta) = \prod_{u\in V} U_{M,u}(\beta)$.

The simultaneous mixer, while consistent with the original proposal of [86], is nontrivial to implement because the Hamiltonians B_u in B do not mutually commute in general. Indeed, the primary advantage of the sequential mixer is that it results in much simpler quantum circuits.

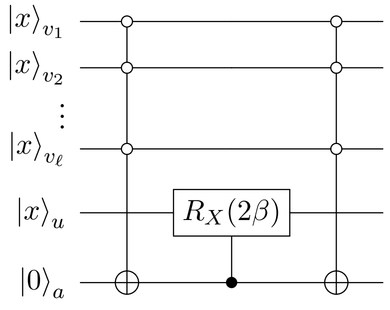


Fig. 6.2: Quantum circuit implementing the mixing operator $U_{M,u}(\beta) = \Lambda_{\overline{x}_{v_1}\overline{x}_{v_2}\dots\overline{x}_{v_\ell}}(X_u)$ for Maximum Independent Set. The circuit consists of two $(\ell+1)$-bit Toffoli gates, for $v_1,\dots,v_\ell$ the neighbours of vertex u, and a controlled X-rotation gate. The ancilla qubit is initialized and returned to $|0\rangle$. The circuit can be implemented with $O(\ell)$ basic gates.

As explained, there is freedom to select the ordering of the product defining the sequential mixer, with some orderings yielding implementation advantages. In particular, for bounded-degree graphs with maximum degree $D_G = O(1)$, many of the $U_{M,u}$ will act on disjoint sets of qubits and hence can be implemented in parallel. Thus, by partitioning the $U_{M,u}$ into groups of terms acting on disjoint qubits, U_M may be implemented with depth $O(D_G) = O(1)$, which is significantly less than n. We further elaborate on possible partitions for mixing operators in [114].

The sequential mixer $U_{\mathrm{M}}(\beta)$ consists of multiqubit controlled X-rotations $\Lambda_{f_u}(e^{-i\beta X_u})$, which may each be implemented efficiently using basic quantum gates, as we now explain. Appending an ancilla qubit labeled a and initialized to $|0\rangle$, which we use to store the value $f_u(x)$, we may implement the action of $\Lambda_{f_u}(e^{-i\beta X_u})$ using the operator

$$\Lambda_{f_u}(X_a)\ \Lambda_{x_a}(e^{-i\beta X_u})\ \Lambda_{f_u}(X_a), \tag{6.18}$$

shown in Figure 6.2. The first operator $\Lambda_{f_u}(X_a)$ is a multi-controlled X gate, i.e., a multiqubit Toffoli gate, which computes f_u in the ancilla register, taking each basis state $|x\rangle|0\rangle$ to $|x\rangle|0 \oplus f_u(x)\rangle$. An important result for our purposes is that any such multi-controlled Toffoli $\Lambda_{f_u}(X_a)$ acting on $\ell+1$ qubits can be implemented with $O(\ell)$ basic gates and using one additional ancilla qubit [173, 230]. (Note that the white circles in Fig. 6.2, which indicate negated control variables, are immaterial; such a Toffoli can be implemented from a regular Toffoli with 2 additional X gates per control line.) The second operator in (6.18) is an X-rotation controlled by the value of $f_u(x)$ in the ancilla qubits, and the final operator $\Lambda_{f_u}(X_a)$ uncomputes (clears) the ancilla qubit a for reuse.

Using the constructions of [230], we may implement each $\Lambda_{f_u}(e^{-i\beta X_u})$ using $O(D_u)$ CNOT and single-qubit gates, i.e., *basic gates*. Here D_u is the degree of the vertex u, and hence the number of control bits in f_u. Therefore, $U_{\mathrm{M}}(\beta)$ may be implemented using at most $O(\sum_u D_u) = O(m)$ basic gates, and a single ancilla qubit. It is tedious but straightforward to derive estimates of the constants in the implementation cost, but this is not our concern here. Similarly, different compilations, in particular to different gate sets, are possible.

Implementation Cost: The initial state $|0\dots0\rangle$ is trivial to prepare. Each application of the QAOA operator $Q = U_{\mathrm{M}}(\beta)U_P(\gamma)$ requires at most $O(m+n)$ basic quantum gates, and $n+1$ qubits. Thus the QAOA$_p$ state can be created with $O(p(m+n))$ basic quantum gates.

We emphasize that different gate sets, compilation choices, and optimizations are possible, with varying cost trade-offs and suitability for a given architecture. In particular, the mixing operator $U_M(\beta)$ may be replaced with $U_M^r(\beta)$, $r = O(1)$, affecting the cost estimates only by a constant.

6.4.1.1 Applications to Other Problems

Our construction for Max Independent Set extends to the related problems Max Clique and Min Vertex Cover, which we now summarize. It is interesting to observe that while these three problems have very similar QAOA constructions, each has quite different properties concerning the best classical algorithms known and hardness of approximation. Exploring these connections is an interesting future research direction.

Note that it is straightforward to extend all three problem constructions to vertex-weighted problem variants via the modification $H_f = \sum_{i=1}^n w_i x_i$.

Max Clique

Problem: Given a graph $G = (V, E)$, find the largest cardinality clique (a subset $V' \subset V$ of mutually-adjacent vertices).

The Maximum Clique decision problem is NP-hard [145], and the optimization problem cannot be approximated better than $O(n^{1-\varepsilon})$ for any ε (as the graph becomes large) unless P=NP [260]. The best algorithm for general graphs achieves a $O(n\frac{(\log\log n)^2}{(\log n)^3})$-approximation [92]. The problem of finding cliques of fixed size was considered for adiabatic quantum optimization in [60].

Observe that every clique in $G = (V, E)$ is an independent set in the complement graph

$G^c = (V, \binom{V}{2} \setminus E)$. Thus, Maximum Clique can be approximated with the above construction for Maximum Independent Set applied to G^c. Note that is this case, feasible states now encode valid cliques of G. The details follow as for Max Independent Set, and the same compilation costs apply (with the parameters of G^c). The details follow as above using the parameters of G^c.

Min Vertex Cover

Problem: Given $G = (V, E)$, minimize the size of a subset $V' \subset V$ that covers V (i.e., for every $(uv) \in E$, $u \in V'$ or $v \in V'$).

The Minimum Vertex Cover problem is APX-complete [179]. It has a $(2 - \Theta(1/\sqrt{\log n}))$-approximation [144], but cannot be approximated better than 1.3606 unless P=NP [82].

We again reduce the problem to Maximum Independent Set, though as approximation problems they are not equivalent [226]. A subset $V' \subset V$ is a vertex cover if and only if $V \setminus V'$ is an independent set, so the problem of finding a minimum vertex cover is equivalent to that of finding a maximum independent set. Hence, we can use the same mapping as for Max Independent Set with each $\bar{x}_v$ replaced by x_v. The resource counts are the same as for Max Independent Set.

6.5 Mappings on k-Dits

In this section, we consider constructions for problems with configuration space $[k]^n = \{1, \dots, k\}^n$ for $k > 2$, i.e., strings of n-many k-dits. We show how encoding each k-dit in unary (i.e., using k-qubits), as opposed to binary, gives implementation advantages for certain problems, and we give explicit constructions for three graph coloring optimization problems.

Graph-k-Coloring ($k \geq 2$) is an important NP-complete problem with many applications, such as scheduling [155, 200] and memory allocation [57]. Given an undirected graph $G = (V, E)$, Graph-k-Coloring asks whether there exists an assignment of one of k colors to each vertex such that every edge is *properly colored* (connects two vertices of different colors). If such an assignment exists, the graph is said to be *k-colorable*. Note that every graph is trivially $(D_G + 1)$-colorable, where D_G is the maximum degree of any vertex in G.

Several optimization variants of Graph-k-Coloring are known. We first consider the Max k-Colorability problem of maximizing the number of properly colored edges [10, 190]. As a coloring specifies a partition, this problem naturally generalizes MaxCut, and is also known as Max k-Cut [99, 151], typically for the case of weighted edges. We then consider the approximation problems of finding the largest k-colorable induced subgraph, and determining a graph's chromatic number.

The constructions for the three problems are related and each extends the previous.

6.5.1 Max k-Colorability

This optimization version of graph coloring relaxes the hard constraint that every edge be properly colored, and instead we try to maximize the number of such edges. This problem is equivalent to MaxCut for $k = 2$.

Problem: Given a graph $G = (V, E)$ with n vertices and m edges, and k colors, find a k-color assignment maximizing the number of properly colored edges.

A random coloring properly colors a fraction $1 - 1/k$ of edges in expectation. For $k > 2$, semidefinite programming gives a $(1-1/k+(2 + o(k))\,\frac{\ln k}{k^2})-$approximation [99], which is optimal up to the $o(k)$ factor under the unique games conjecture [151]. This problem is APX-complete for $k \geq 2$ [179] with no PTAS unless P=NP [190].

The domain F is the set of colorings $\mathbf{x}$ of G, an assignment of a color to each vertex. (Note that here and throughout, the term "colorings" includes *improper* colorings.) The domain F can be represented as the set of length n strings over an alphabet of k characters, $\mathbf{x} = x_1 x_2 \ldots x_n$, where $x_i \in [k]$. The objective function $f : [k]^n \to \mathbb{N}$ counts the number of properly colored edges in a coloring

$$f(\mathbf{x}) = \sum_{(uv)\in E} \text{NotEqual}(x_u, x_v). \tag{6.19}$$

There are different ways to represent this problem on a quantum computer, with various trade-offs. We employ a unary *one-hot* encoding for each k-dit, consisting of the k-many Hamming weight 1 bit strings on k bits, $\{100\ldots, 010\ldots, \ldots\}$, in which the position of the single 1 indicates the assigned color. This encoding, which requires k qubits per vertex, has been used in quantum annealing [127, 128, 165, 200]. Thus, overall color assignments are encoded using kn binary variables $x_{v,i}$, with $x_{v,i} = 1$ indicating that vertex v has been assigned color i. For each vertex $v \in V$, a hard constraint is that it be assigned exactly one color,

$$\sum_{i=1}^{k} x_{v,i} = 1, \tag{6.20}$$

i.e., the allowed states of the k variables $x_{v,1}, \dots, x_{v,k}$ encode the k-dit x_v. Feasible strings are then Hamming weight n strings such that these n constraints are satisfied. The initial state $|s\rangle$ may be taken to be any feasible state, which requires n many X-gates to prepare in depth 1.

On feasible strings x, the cost function may be written

$$f(x) = m - \sum_{(uv)\in E} \sum_{i=1}^{k} x_{u,i} x_{v,i}, \tag{6.21}$$

which subtracts one for every improperly colored edge. Substituting $\frac{1}{2}(I - Z)$ for each binary variable yields a quadratic Hamiltonian of the same form as (5.7). Furthermore, (6.20) implies that each operator $\sum_{i=1}^{k} Z_{v,i}$ acts as a constant multiple of the identity, simplifying the Hamiltonian to

$$H_f = \frac{km}{4} I - \frac{1}{4} \sum_{(uv)\in E} \sum_{i=1}^{k} Z_{u,i} Z_{v,i} \ . \tag{6.22}$$

Dropping again the identity term from $e^{-i\gamma H_f}$, we define the phase operator

$$U_P(\gamma) = \prod_{(uv)\in E} \prod_{i=1}^{k} e^{i\gamma \frac{1}{4} Z_{u,i} Z_{v,i}}, \tag{6.23}$$

which consists of km many R_{ZZ} operations. The R_{ZZ} mutually commute and can be applied in any order. Each R_{ZZ} can be implemented with two CNOT gates and one R_Z gate; see Figure 6.1.

Turning to the mixing operator, we seek a mixing Hamiltonian that meets the criteria laid out in Sec. 6.2.1, keeping the evolution within the feasible subspace. Observe that the hard constraints (6.20) each depend only on a single vertex. Thus it suffices to define mixing Hamiltonians which preserve feasibility locally for each vertex. In particular, the constraints (6.20) can be written as $(1 - \sum_{i=1}^{k} x_{v,i})^2 = 0$, so that the corresponding Hamiltonian

$$H_A = -\frac{1}{4} \sum_{v} \left(2(k-2) \sum_{i=1}^{k} Z_{v,i} - \sum_{i\neq j}^{k} Z_{v,i} Z_{v,j} \right) \tag{6.24}$$

admits the feasible subspace exactly as its ground subspace.

For each $v \in V$, we define

$$B_v = \sum_{i=1}^{k} X_{v,i}X_{v,i+1} + Y_{v,i}Y_{v,i+1}, \tag{6.25}$$

with indices taken modulo k. This Hamiltonian is known in physics as the *XY Model on a ring* [160]. The B_v each act on disjoint sets of qubits and hence mutually commute, so for the Hamiltonian $B = \sum_v B_v$, the corresponding simultaneous mixer is

$$U_M^{(H)}(\beta) = e^{-i\beta B} = \bigotimes_{v\in V} e^{-i\beta B_v}. \tag{6.26}$$

It is easy to check that $[B_v, H_A] = 0$ for each $v \in V$. Note that this requires the presence of both the XX and YY terms in (6.25). Thus, Propositions 5 and 6 imply that $U_M^{(H)}(\beta)$ preserves feasibility.

It is insightful to elaborate on how the operators $e^{-i\beta B_v}$ act on basis states. We will use related operators in our subsequent problem constructions. For a vertex v assigned a color j, indicated by the state $|j\rangle_v$, $1 \le j \le k$, we have

$$B_v|j\rangle_v = |j+1 \pmod{k}\rangle_v + |j-1 \pmod{k}\rangle_v.$$

Thus we may identify $B_v = L_v + R_v$, where L_v, R_v are left and right circular shift operators. From the series expansion, the operator $e^{-i\beta B_v}$ is easily seen to be given by a weighted linear combination of the identity (the zero shift), single left/right shift operators, double shifts, and so on.

Each operator $e^{-i\beta B_v}$ can be efficiently compiled to a quantum circuit using the quantum Fourier transform [232]. On the other hand, we can derive a simple mixing operator with low implementation cost. It is easy to check that the Hamiltonians

$$B_{v,i,j} = X_{v,i}X_{v,j} + Y_{v,i}Y_{v,j}, \tag{6.27}$$

$v \in V$, $i, j \in [k]$, themselves preserve feasibility. Thus, from (6.25) we define the *sequential* mixer

$$U_{\mathrm{M}}(\beta) = \prod_{v\in V, j\in[k]} e^{-i\beta B_{v,j,j+1}}, \tag{6.28}$$

where the order of the product is arbitrary and may be selected aas desired. Each $B_{v,j,j+1}$ commutes with all but 2 of the other $B_{v',j',j'+1}$, so $U_{\mathrm{M}}(\beta)$ can be implemented in depth 2 with respect to the $e^{-i\beta B_{v,j}}$.

As $[X \otimes X, Y \otimes Y] = 0$, each $e^{-i\beta B_{v,j}}$ can be decomposed into an XX-rotation and a YY-rotation as $e^{-i\beta B_{v,j}} = e^{-i\beta X_{v,j}X_{v,j+1}} e^{-i\beta Y_{v,j}Y_{v,j+1}}$. The XX-rotation can be implemented with 4 Hadamard (H) gates, 2 CNOT gates, and an R_Z gate, as shown in Figure 6.3. A similar construction holds for each YY-rotation with the Hadamards replaced by $R_X(\pi/2)$ gates; see e.g. [246]. Thus, $U_{\mathrm{M}}(\beta)$ can be implemented with $O(nk)$ basic gates.

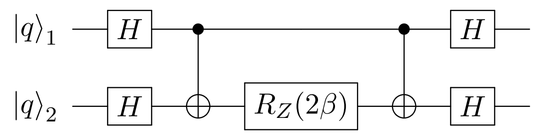


Fig. 6.3: Quantum circuit performing $R_{X_1X_2}(2\beta) = \exp(-i\beta X_1 X_2)$ on two qubits labeled $1, 2$.

Implementation Cost: Our construction uses kn qubits. Any feasible basis state may be used as the initial state, and prepared using at most n many X-gates. The operators $U_P(\gamma)$ and $U_{\mathrm{M}}(\beta)$ require, respectively, $O(km)$ and $O(kn)$ basic quantum gates. Thus the QAOA$_p$ state can be created with $O(pk(m+n))$ basic quantum gates.

It is easy to see that for a suitable $U_M(\beta)$, say $\beta = \pi/4$, that $r = \lfloor k/2 \rfloor$ repetitions of $U_M(\beta)$ suffice to connect any two states in the sense of our criteria of Section 6.2.1. Thus, setting $r = O(k)$, the *repeated mixing operator* $U_M^r(\beta)$ can be used instead, now requiring $O(krn) = O(k^2 n)$ basic quantum gates. Note that, typically, $k = O(1)$. Alternatively, the *fully-connected XY model Hamiltonian* $B_v^{(fc)} = \sum_{i<j} X_{v,i}X_{v,j} + Y_{v,i}Y_{v,j}$, also satisfies $[B_v^{(fc)}, H_A] = 0$ for all $v \in V$, and hence preserves feasibility. The corresponding sequential mixer $U_M^{(fc)}(\beta) = \prod_{v\in V}\prod_{i<j} e^{-i\beta B_{v,i,j}}$ similarly requires $O(nk^2)$ basic quantum gates.

With either of these alternative mixers, we would hope to obtain better performance in exchange for the higher implementation costs. Analyzing this trade-off between the efficacy of different mixing operators and their implementation costs is an important direction of future work.

The next two graph coloring problems we consider will use similar unary one-hot encodings for the color state of each vertex. It is worthwhile to remark on the trade-offs between this encoding and the binary alternative. Firstly, the unary encoding uses kn qubits, whereas the binary encoding requires $\lceil \log_2 k \rceil n$ qubits; as typically $k \ll n$ (e.g., $k = O(1)$), our approach does not add unreasonable overhead. Secondly, for the unary encoding, 2-local interactions are sufficient to compute the objective function, as is evident from (6.23); in contrast, a $2\lceil \log_2 k \rceil$-local interaction may be necessary in general to compute this in binary, as every bit position must be compared to determine if two color labels are the same, leading to increased resource requirements. Furthermore, if k is not a power of two, then some of the one-hot states are redundant or not used and must be dealt with somehow, which may be nontrivial.

6.5.2 Max k-Colorable Induced Subgraph

The *induced subgraph* of a graph $G = (V, E)$ for a subset of vertices $W \subset V$ is the graph $H = (W, E_W)$, where E_W is the subset of edges in E with both endpoints in W.

Problem: Given a graph $G = (V, E)$ with n vertices and m edges, find the largest induced subgraph (largest number of vertices) that can be properly k-colored.

This problem is as easy and as hard to approximate as Max Independent Set [16, 178]. For $k = 1$, the two problems are equivalent. On bounded degree graphs, Max k-Colorable Induced Subgraph can be approximated to $(D_G/k + 1)/2$, but remains APX-complete [119].

We represent colorings as in the previous section with variables $x_{v,1}, \ldots, x_{v,k}$, but with one additional variable $x_{v,0}$ per vertex to represent an "uncolored" vertex, indicating that the vertex v is not included in the induced subgraph. The state of each vertex thus corresponds to a $(k + 1)$-dit.

In this case, feasible strings, in addition to having each vertex uniquely colored or assigned as uncolored, correspond to proper colorings on the subgraph induced by the colored vertices. Thus, the mixing operator will be more complicated, essentially incorporating information that was in the cost function of the previous problem.

On the feasible subspace, the cost function takes an especially simple form,

$$f = m - \sum_v x_{v,0}, \tag{6.29}$$

which counts the number of included vertices. The corresponding objective Hamiltonian, after dropping the identity terms, is

$$H_f = \frac{1}{2} \sum_v Z_{v,0} \tag{6.30}$$

Hence, the phase separation operator $U_P = e^{-i\gamma H_f}$ can again be implemented with n many Z-rotation gates and depth one.

To design mixing operators, consider the allowed transitions between feasible states. A given vertex can be feasibly colored i only if none of its adjacent vertices are also colored i. Thus, the transition rule at each vertex must depend on the local graph topology and assigned colors. Consider the controlled operation

$$(\bar{x}_{v_1,i}\bar{x}_{v_2,i}\dots\bar{x}_{v_{D_u},i})\cdot \mathrm{SWAP}(x_{u,0},x_{u,i}), \tag{6.31}$$

where $v_1,\dots,v_{D_u}$ are the neighbours of vertex u. This operation swaps the color of vertex u between the uncolored state and color i if and only if none of the neighbours of u are already colored i. The corresponding Hamiltonian term (after dropping the terms for the SWAP that have no effect) is

$$B_{u,i} = \frac{1}{2^{D_u}}(X_{u,0}X_{u,i} + Y_{u,0}Y_{u,i})\prod_{j=1}^{D_u}(I + Z_{v_j,i}). \tag{6.32}$$

The overall mixing Hamiltonian is $B = \sum_u \sum_i B_{u,i}$. Since B contains the means to color a vertex with color i if none of its neighbours are colored with color i, and a means to uncolor a vertex (as long as none of its neighbours share its current color, which is always the case in the feasible subspace), the mixing term enables exploration of the full feasible subspace starting from any state in that subspace. A simple initial state is $|s\rangle = |0\rangle^{\otimes n}$ in which all vertices are uncolored.

Exponentials of the Hamiltonians $B_{u,i}$ again give controlled unitaries. For the control predicate $f_{u,i} = \prod_{v\in \mathrm{nbhd}(u)} \bar{x}_{v,i}$, we define

$$U_{M,u,i}(\beta) = e^{-i\beta B_{u,i}} = \Lambda_{f_{u,i}}(e^{-i\beta(X_{u,0}X_{u,i}+Y_{u,0}Y_{u,i})}), \tag{6.33}$$

which is a multiqubit controlled $(XX+YY)$-rotation.

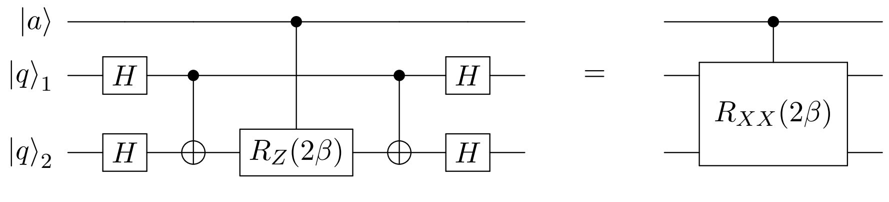


Fig. 6.4: Quantum circuit performing the operation controlled XX-rotation $U = \Lambda_a(e^{-i\beta X_1X_2})$. Replacing each H gate with a $R_X(\pi/2)$ gate gives instead $U = \Lambda_a(e^{-i\beta Y_1Y_2})$.

From the Hamiltonian $B = \sum_u \sum_i B_{u,i}$, we define the mixers:

- the *simultaneous* (Hamiltonian-based) controlled-X mixer $U_M^{(B)}(\beta) = e^{-i\beta B}$,
- the *sequential* (partitioned) controlled-X mixer $U_{\mathrm{M}}(\beta) = \prod_{u\in V}\prod_{i=1}^{k} U_{M,u,i}(\beta)$.

The sequential mixer requires nk application of $U_{M,u,i}(\beta)$. Similar to the previous constructions, using an ancilla qubit each $U_{M,u,i}(\beta)$ can be implemented with two (D_u+1)-qubit Toffoli gates, a controlled XX-rotation, and a controlled YY-rotation. The circuits are shown in Figures 6.4 and 6.5. Thus, by a similar argument as for Max Independent Set, we have that $U_M(\beta)$ can be implemented with $O(km)$ basic gates.

Implementation Cost: Our construction uses $(k+1)n+1$ qubits. The initial state $|0\dots0\rangle$ is trivial to prepare. Each application of the QAOA operator $Q = U_{\mathrm{M}}(\beta)U_P(\gamma)$ requires $O(km+n)$ basic gates. Thus the QAOA$_p$ state can be created with $O(p(km+n))$ basic quantum gates.

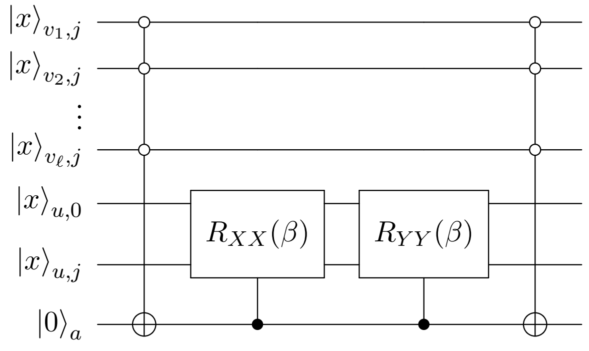


Fig. 6.5: Quantum circuit implementing the mixing operator $U_{M,u,j}(\beta)$. The ancilla qubit is initialized and returned to $|0\rangle$. The circuit can be implemented with $O(\ell)$ basic quantum gates.

6.5.3 Min Chromatic Number

A graph G that can be κ-colored but not $(\kappa-1)$-colored is said to have *chromatic number κ*.

Problem: For a graph $G=(V,E)$, minimize the number of colors needed to properly color it.

The Min Chromatic Number problem has important applications to scheduling [75] and to physics [227]. The best classical algorithm [118] achieves an approximation ratio of $O\left(n\frac{(\log\log n)^2}{(\log n)^3}\right)$, and we cannot do better than $n^{1-\varepsilon}$ for any $\varepsilon > 0$ in polynomial time unless P = NP [260].

Any graph can be properly (D_G+1)-colored, where recall $D_G \leq n-1$ is the maximum vertex degree in G. We seek a proper coloring (i.e., a certificate) showing that fewer colors suffice.

For the mixing operators we consider below, $k = D_G + 2$ colors suffice to allow transitions between any two feasible states. This follows because any coloring using at most $D_G + 2$ colors can be transformed into any other coloring using at most $D_G + 1$ colors by a series of moves that changes the color of one vertex at a time while maintaining a proper coloring at each step. Moreover, a $(D_G + 2)$-coloring can be trivially constructed for any graph.

Thus, we may use $k = D_G+2$ qubits to encode the k possible colors of each vertex in the unary one-hot encoding. We define the feasible domain to be the subset of states encoding proper graph colorings, many of which may use fewer than k colors.

For a given coloring x, the function $y_j(x) = \bigwedge_{u\in V} \bar{x}_{u,j}$ gives 1 only if no vertex is colored j. Thus, we seek to maximize the number of unused colors, encoded by the n-local objective function

$$f(x) = \sum_{j=1}^{D+2} y_j = \sum_{j=1}^{D+2} \bigwedge_{u\in V} \bar{x}_{u,j}. \tag{6.34}$$

The corresponding problem Hamiltonian is

$$H_f = \frac{1}{2^n} \sum_{j=1}^{D+2} \prod_{u\in V} (I + Z_{u,j}). \tag{6.35}$$

Expanding the right-hand side gives a sum of $\Omega(2^n)$ terms with locality up to n, which renders our previous approach to implementing $U_P(\gamma) = e^{-i\gamma H_f}$ inefficient (the number of Z-rotation gates required is exponential in n).

However, we can implement $U_P(\gamma)$ efficiently with the help of an ancilla register. We append k ancilla qubits $a_1, \dots, a_k$ to our state, initialized to $|00\dots0\rangle$. The ancilla a_j is used to store $y_j(x)$. For each color j, define the $(n+1)$-local unitary operator to be the multi-controlled CNOT

$$U_j = \Lambda_{y_j}(X_{a_j}).$$

Using an ancilla qubit, each U_j can be implemented with $O(n)$ basic gates. The U_j act nontrivially on disjoint sets of qubits and mutually commute. Thus the operator

$$U_a := U_1 U_2 \dots U_k$$

maps the basis state $|x\rangle|00\dots0\rangle$ to the state $|x\rangle|y_1 y_2 \dots y_k\rangle$. As $U_i^2 = I$, a second application of U_a uncomputes each ancilla.

After computing the $y_j(x)$, the phase operator may be implemented by applying a Z-rotation gate to each ancilla qubit. Summing the bits $y_j(x)$ gives the number of unused colors. Dropping the identity term, the corresponding Hamiltonian is given by

$$H_g = -\frac{1}{2}\sum_{j=1}^{k} Z_{a_j}, \tag{6.36}$$

for which $e^{-i\gamma H_g}$ can be implemented with k many R_Z gates.

The overall phase operator is given by

$$U_P(\gamma) = U_a e^{-i\gamma H_g} U_a = U_1U_2\dots U_k R_{Z_{a1}}(\gamma)\dots R_{Z_{ak}}(\gamma) U_1U_2\dots U_k, \tag{6.37}$$

which can be implemented using k ancilla qubits and $O(kn)$ basic quantum gates in constant depth.

For the mixing operator, we use a controlled operation similar to (6.33). In this case, there is no uncolored state. A vertex u can be recolored i only if that would produce no conflicts with its neighbours, i.e., the coloring remains proper. Thus, the color bits x_i and x_j for u may be safely swapped if none of the D_u neighbours of u are colored either i or j. This gives the Hamiltonian

$$B_{uij} = (\bar{x}_{v_1 i}\bar{x}_{v_2 i}\dots\bar{x}_{v_{D_u} i})(\bar{x}_{v_1 j}\bar{x}_{v_2 j}\dots\bar{x}_{v_{D_u} j})\cdot \mathrm{SWAP}(ui, uj)$$

for $i, j = 1, \dots, \ell$, which after dropping the terms having no effect reduces to

$$B_{uij} = \frac{1}{2^{2D_u+1}}(X_{ui}X_{uj} + Y_{ui}Y_{uj})\prod_{a=1}^{D_u}(I + Z_{v_a i})\prod_{b=1}^{D_u}(I + Z_{v_b i}).$$

Exponentials of the Hamiltonians B_{uij} give unitaries controlled by the Boolean predicate $f_{uij} = \prod_{v\in \mathrm{nbhd}(u)} \bar{x}_{vi}\bar{x}_{vj}$. We define the phase operator

$$U_{M,u,i,j}(\beta) = e^{-i\beta B_{uij}} = \Lambda_{f_{uij}}(e^{-i\beta(X_{ui}X_{uj}+Y_{ui}Y_{uj})}), \tag{6.38}$$

which by the previous argument can be implemented with $O(D_u)$ basic gates.

For $B = \sum_u \sum_{i<j} B_{u,i,j}$, we define the Hamiltonian-based mixer $U_M^{(B)}(\beta) = e^{-i\beta B}$, and the *sequential* (partitioned) controlled-X mixer

$$U_{\mathrm{M}}(\beta) = \prod_{u\in V}\prod_{i<j} U_{M,u,i,j}(\beta). \tag{6.39}$$

We can implement $U_M(\beta)$ using $O(mk^2)$ basic gates.

Implementation Cost: Our construction uses $nk + k + 1$ qubits, with $k \geq D_G + 2$. A proper $(D_G + 2)$-coloring can be prepared as an initial state using n many X gates. Each application of the QAOA operator $Q = U_{\mathrm{M}}(\beta)U_P(\gamma)$ requires at most $O(k^2m + nk)$ basic quantum gates, and $nk + k + 1$ qubits. Thus the QAOA$_p$ state can be created with $O(p(k^2m + nk))$ basic quantum gates.

6.6 Mappings on Permutations

Many important but challenging computational problems have a configuration space that is the set of orderings or schedules of some number of items or events. Here, we introduce the machinery for mapping such problems to QAOA, using the Traveling Salesman Problem (TSP) and a Single-Machine Scheduling (SMS) problem as illustrative examples.

The constructions of this sections are applicable to many other problems. In particular, in [114] we show constructions for two other single machine scheduling problem variants.

6.6.1 Traveling Salesman Problem

A *vertex tour* of a complete graph $G = (V, E)$ is a simple cycle that contains all $|V| = n$ vertices, i.e., gives a route for the salesman to visit each 'city' exactly once and finish where they started, and similarly for directed graphs. For both cases, up to symmetries, the set of possible tours is isomorphic to the set of possible orderings of the n vertices.

Problem: Given a complete graph $G = (V, E)$ and distances $d_{u,v} \in \mathbb{R}$, find the shortest tour.

The TSP problem is NPO-complete [176]. MetricTSP, where the distances satisfy the triangle inequality, is APX-complete [180] and has a $3/2$-approximation [67]. The corresponding MaxTSP problem is approximable within $7/5$ for symmetric distance, and $63/38$ if asymmetric. The TSP has previously been considered for quantum annealing [165, 166].

We represent vertex tours with n^2 binary variables $\{x_{vj}\}$ indicating whether vertex v is visited at the jth stop of the tour, $j = 1, \ldots, n$, which we represent using n^2 qubits. Feasible states are those that encode valid tours, i.e. valid orderings, expressed as the hard constraints that for each v we have $\sum_{j=1}^{n} x_{v,j} = 1$ (each v visited exactly once), and for each position j we have $\sum_{v \in V} x_{v,j} = 1$ (a single vertex visited at each stop).

The objective function is the tour length, which may be written

$$f(x) = \sum_{\{u,v\}\in E} d_{u,v} \sum_{j=1}^{n} \left(x_{u,j}x_{v,j+1} + x_{v,j}x_{u,j+1}\right). \tag{6.40}$$

Mapping each term to a Hamiltonian and simplifying using the hard constraints (which alleviates the need for single Z terms) yields the phase operator

$$U_P(\gamma) = \prod_{\{u,v\}\in E} \prod_{j=1}^{n} e^{-\gamma d_{uv} Z_{u,j} Z_{v,j+1}} e^{-\gamma d_{uv} Z_{u,j+1} Z_{v,j}}, \tag{6.41}$$

where we have again dropped the terms contributing only global phase. The number of ZZ-rotations in $U_P(\gamma)$ is $n^3 - n^2$, or half this amount for directed graphs (where the second rotation in (6.41) is not needed). Thus $U_P(\gamma)$ can be implemented with $n^3 - n^2$ R_Z gates and $2(n^3 - n^2)$ CNOT gates.

For the mixing operator, it is useful to view feasible states as $n \times n$ matrices with a single 1 in every column or row. A mixing Hamiltonian may be constructed as a sum of row swaps $B = \sum_{u=1}^{n} B_{u,u+1}$ or $B = \sum_{u<v} B_{u,v}$, where

$$B_{u,v} = \prod_{j=1}^{n} \text{SWAP}((u,j),(v,j)) \tag{6.42}$$

clearly preserves feasibility. The Hamiltonians $B_{u,v}$ contain $2n$-local interactions and are nontrivial to implement.

Alternatively, we can mix feasible states with the 4-local Hamiltonian

$$\begin{aligned} H_{M,u,v,i,j} &= |0_{u,i}1_{u,j}1_{v,i}0_{v,j}\rangle\langle 1_{u,i}0_{u,j}0_{v,i}1_{v,j}| + |1_{u,i}0_{u,j}0_{v,i}1_{v,j}\rangle\langle 0_{u,i}1_{u,j}1_{v,i}0_{v,j}| \\ &= S^-_{u,i}S^+_{u,j}S^+_{v,i}S^-_{v,j} + S^+_{u,i}S^-_{u,j}S^-_{v,i}S^+_{v,j}, \end{aligned} \tag{6.43}$$

where in the second line we have introduced the *spin*[3] creation and annihilation operators $S^+ = \frac{1}{2}(X - iY) = |1\rangle\langle 0|$ and $S^- = \frac{1}{2}(X + iY) = |0\rangle\langle 1|$. Similar mixing operations have been considered previously for quantum annealing [166].

We define the unitaries

$$U_{M,u,v,i,j}(\beta) = e^{-iH_{M,u,v,i,j}}, \tag{6.44}$$

[3]In contrast to the *fermionic* creation and annihilation operators a^+, a^- considered in Section 4.4 for the electronic Hamiltonian, the operators S^+ and S^- satisfy $[S^-, S^+] = Z$ and $\{S^-, S^+\} := S^-S^+ + S^+S^- = I$.

which can each be implemented using basic gates. Indeed, substituting Pauli matrices and expanding yields $H_{M,u,v,i,j}$ as a sum of 8 terms given by (suppressing the qubit indices)

$$\frac{1}{8}(XXXX-YYXX+XYXY+YXYX+XYYX+YXXY-XXYY+YYYY). \quad (6.45)$$

The terms in the sum (6.45) mutually commute, so $U_{M,u,v,i,j}(\beta)$ can be implemented with eight applications of the circuit in Figure 6.6. These circuits are similar to, but simpler than, those for the simulation of the electronic Hamiltonian [246]. Thus each $U_{M,u,v,i,j}(\beta)$ can be implemented with $O(1)$ basic gates. We define the sequential mixing operator

$$U_M(\beta) = \prod_{u<v}\prod_{j=1}^{n} U_{M,u,v,j,j+1}(\beta), \quad (6.46)$$

which consists of $O(n^3)$ applications of $U_{M,u,v,j,j+1}(\beta)$. It is easy to see $U_M^r(\pi/4)$ generates all possible basis state transitions for $r = n-1$, so $U_M(\beta)$ satisfies our design criteria.

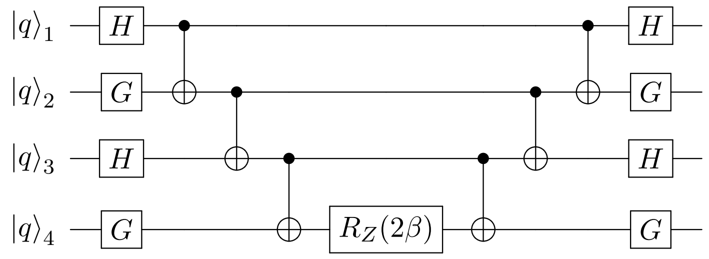


Fig. 6.6: Quantum circuit performing the operation $R_{X_1Y_2X_3Y_4}(2\beta) = \exp(-i\beta X_1Y_2X_3Y_4)$ on four qubits labeled 1 to 4. Generally, the exponential of any tensor product of four X and Y operators can be implemented by a similar circuit where Hadamard H and $G = R_X(\pi/2)$ gates have been substituted appropriately, corresponding to which of X or Y acts on each qubit.

Implementation Cost: States are represented with n^2 qubits. Any vertex ordering suffices as an initial state which can be prepared with n many X gates. Each application of the QAOA operator $Q = U_{\mathrm{M}}(\beta)U_P(\gamma)$ requires at most $O(n^3)$ basic quantum gates. Thus the QAOA$_p$ state can be created with $O(pn^3)$ basic quantum gates.

We remark that it is possible to instead define the mixing operator as $\prod_{u<v}\prod_{i<j}^{n} U_{M,u,v,i,j}(\beta)$, now containing $O(n^4)$ terms and requiring proportionately more resources to implement. The $O(n^3)$ cost scaling of the proposed $U_M(\beta)$ operator matches that of the phase operator above.

6.6.2 Single Machine Scheduling

We consider scheduling jobs on a single machine as to minimize the total (weighted) tardiness.

Problem: Given n jobs to run on a single machine, each with a running time p_j, a deadline $d_j \geq p_j$, and a weight w_j, find a schedule that minimizes the total weighted tardiness $T = \sum_{j=1}^{n} w_j T_j$, where $T_j \geq 0$ gives the amount of time job j is late. All times are taken to be integers.

In scheduling notation [53] this problem is denoted $(1|d_j|\sum w_j T_j)$. There exists an $(n-1)$-approximate algorithm [58]. The corresponding decision problem is strongly NP-hard [156].

Assuming a job is always running, job schedules are equivalent to job orderings. As in the previous construction, an encoding using $O(n^2)$ qubits may be used to represent the possible schedules. However, computing the tardiness for the phase operator is then relatively nontrivial. We propose a different encoding which enables a novel but relatively simple phase operator construction. We will, however, still make use of the equivalence to orderings in the design of the mixing operator.

Clearly, all jobs will finish by time $P = \sum_j p_j$, so the last job will start by time $\tau = P - \min_i p_i$. For each schedule, let the binary variables $x_{j,t}$ denote if job j starts at time t, where $t = 0, 1, \ldots \tau$. We represent these variables with $n\tau$ qubits. Feasible strings are those for which each job is assigned a single start time, i.e. Hamming weight n strings satisfying $\sum_t x_{j,t} = 1$, and for which there are no overlapping jobs scheduled. Given an ordering, i.e., a sechule, it is easy to compute the $x_{j,t}$, so an arbitrary ordering can be used as the initial state and prepared using n-many X gates.

Suppose job j starts at time s_j; its tardiness is then defined as $T_j = \max\{0, s_j + p_j - d_j\}$. Generally, a maximum function is nontrivial to implement as a Hamiltonian. However, T_j is simple to implement in the $x_{j,t}$ variables by restricting to times t where job j is tardy. The cost function, i.e., the weighted total tardiness, then becomes

$$f(x) = \sum_j w_j \sum_{t=d_j-p_j}^{\tau} x_{j,t}(t + p_j - d_j), \tag{6.47}$$

which maps to a Hamiltonian that is a sum of single-qubit Z operators. Similarly to our previous constructions, we define the phase operator

$$U_P(\gamma) = \prod_{j=1}^{n} \prod_{t=d_j-p_j}^{\tau} e^{i w_j (t+p_j-d_j) Z_{j,t}/2}, \tag{6.48}$$

which can be implemented with at most nP many R_Z gates.

We construct a mixing operator similar to the previous problem by considering pairwise swaps of jobs in the schedule. It is easy to see sequences of such swaps mix between all possible schedules. For consecutive jobs j starting at t and j' starting at $t + p_j$, swapping their order results in job j' starting at time t and job j starting at $t + p_{j'}$. This exchange is realized by the Hamiltonian $S^-_{j,t}S^-_{j',t+p_j}S^+_{j,t+p_{j'}}S^+_{j',t} + S^+_{j,t}S^+_{j',t+p_j}S^-_{j,t+p_{j'}}S^-_{j',t}$. Therefore, the mixing Hamiltonian is

$$B = \sum_{j<j'}\sum_{t=0}^{\tau} S^-_{j,t}S^-_{j',t+p_j}S^+_{j,t+p_{j'}}S^+_{j',t} + S^+_{j,t}S^+_{j',t+p_j}S^-_{j,t+p_{j'}}S^-_{j',t}. \tag{6.49}$$

We thus define the operator

$$U_{M,i,j,t} = e^{-i\beta(S^-_{j,t}S^-_{j',t+p_j}S^+_{j,t+p_{j'}}S^+_{j',t}+S^+_{j,t}S^+_{j',t+p_j}S^-_{j,t+p_{j'}}S^-_{j',t})}, \tag{6.50}$$

which can be efficiently implemented as described in the previous section; see Figure 6.6. Hence, we define the mixing operator to be

$$U_M(\beta) = \prod_{i<j}\prod_{t=0}^{\tau} U_{M,i,j,t}, \tag{6.51}$$

which can be implemented using $O(n^2\tau)$ basic quantum gates.

Implementation Cost: States are represented with $n\tau$ qubits, where $\tau = P - \min_i p_i \leq P$. Any vertex ordering suffices as initial state which can be prepared with n gates. Each application of the QAOA operator $Q = U_{\mathrm{M}}(\beta)U_P(\gamma)$ requires at most $O(n^2\tau + n\tau)$ basic quantum gates. Thus the QAOA$_p$ state can be created with $O(pn^2P)$ basic quantum gates.

We remark that many important variants of scheduling problems exist, with other parameters (e.g., release dates) and objective functions (e.g., completion time) [53]. Our construction serves as a prototype for these problems. We consider several other scheduling problems in [114].

6.7 Discussion and Concluding Remarks

We introduced the Quantum Alternating Operator Ansatz (QAOA), a generalization of the Quantum Approximate Optimization Algorithm, and showed how to apply the ansatz to a variety of hard optimization problems. The essence of this extension is the consideration of general parameterized families of unitaries, rather than only those corresponding to the time evolution of a local Hamiltonian, which allows the representation of a larger and potentially more useful set of states than the original formulation. Refocusing on unitaries rather than Hamiltonians in the specification leads to a variety of efficiently implementable mixing operators with relatively low resource requirements in terms of the number of qubits and basic gates required for implementation. Hence, our constructions provide evidence that QAOA may be an especially suitable application for near-term quantum computers. Furthermore, the constructions we outline cover a range of problem domains, and may serve as prototypes for mapping other problems of interest. We include a compendium of additional mappings in [114].

For each of the problems we consider, our constructions preserve the feasible subspace. This requires more sophisticated quantum circuitry than the original QAOA proposal. Hence, if we start with a feasible state and create a QAOA$_p$ state, a computational basis measurement is guaranteed to give a feasible solution. The feasibility property allows for simplifications to the phase separation operator, typically reducing its implementation cost. The mixing operations, however, are nontrivial to derive and implement. We derive mixing Hamiltonian terms controlled by Boolean predicates. We further show how these terms can be combined in different ways to generate a variety of mixing operators, in particular, products of simple controlled unitaries which can be implemented efficiently. When the controlled unitaries are local, say for bounded degree problems, they may be implemented with depth much less than their number.

On the other hand, for some optimization problems, it is NP-hard to decide if a feasible state exists, let alone find such an initial state; see e.g. the problems in [259]. Designing mixing operators in such cases is problematic as there is no obvious way to efficiently ensure that the mixing operations preserve feasibility. Clearly, our approach is not generally efficient for such problems.

The biggest open question is to characterize the performance of QAOA. Can QAOA be used to beat classical algorithms for certain problems, either in terms of giving a rigorous approximation algorithm that beats all possible classical algorithms, or outperforming (say, empirically) all known classical algorithms and heuristics? If so, and QAOA does indeed yield practical advantages, then approximate optimization could turn out to be a very important "killer application" for quantum computers, especially near-term ones. As evident from the results of Chapter 5, obtaining similar results for more general problem constructions such as those of this chapter appears to be a difficult task. While obtaining further analytic results may be possible in some cases using our techniques, in general this remains an open research direction. Improved techniques for classically simulating quantum circuits, potentially including approaches tailored to QAOA, may provide some insight, but the ultimate test may prove to be experimentation on quantum hardware itself.

While we have successfully shown basic design criteria and example constructions for QAOA initial states, mixing operators, and phase separation operators, we have barely scratched the surface in terms of which possibilities perform better than others. For most of the example problems, we discussed multiple mixers, coming from different partitions and orderings of simpler partial mixers. Analytic, numerical, and ultimately experimental work is required to understand which of these mixers are most effective, in particular with respect to the trade-offs between performance and implementation costs. For example, for a near-term quantum device, it may be possible to implement QAOA$_p$ for much larger p with a mixer requiring n gates, than when using a more complicated mixer requiring n^2 gates; however, it is not at all clear which option would ultimately lead to better performance. Furthermore, work is required to investigate other trade-offs such as the difficulty of finding good algorithm parameters, and lower level concerns such as robustness to noise or control error. Clearly, similar questions arise with respect to choosing an initial state.

Effective parameter setting also remains a critical, but mostly open, area of research. While brute-force search was considered for fixed p in [86], it is practical only for small p, suffering from the curse of dimensionality as p increases; see [109]. In certain simple [238] or highly symmetric [138] cases, some insights into parameter setting for $p > 1$ have been obtained, but even in the simplest cases, understanding good choices of parameters seems nontrivial [238]. Improved parameter setting protocols may come from adapting techniques from existing control theory and parameter optimization methods, and by using insights gained from classical simulation of quantum circuits and experimentation on quantum hardware as it becomes available. In particular, for problem constructions with local Hamiltonians, early small-scale quantum computers may be used to help characterize the performance of QAOA on much larger problem instances [86].

To run on near-term quantum hardware, further compilation will be required. We have primarily considered compilation to CNOT and general single-qubit gates. For other gate sets and architectures, for example where quantum error correction is used, these gates will need to be further compiled (and optimized). Furthermore, near-term hardware will have additional restrictions, including which qubits each gate can be applied to, duration and fidelity of the gates, and cross-talk, among others. This necessitates additional compilation, especially to optimize success probability on pre-fault-tolerance devices. Other architectures, e.g. ones based on higher-dimensional qudits, may prompt other sorts of compilations as well. Low-level compilation and optimization of quantum circuits is a rich topic outside the scope of this thesis; see [231] for a recent approach.

Going forward, we expect some fruitful cross-fertilization between research on QAOA and research on quantum annealing. A promising direction is to build on the results of Yang et al. [254], who used Pontryagin's minimization principle to show that for quantum annealing, a "bang-bang" schedule similar to the form of QAOA is (essentially) always optimal. Unfortunately, their argument does not seem to provide an efficient means to find such a schedule for a QAOA. Similarly, exploiting certain structural commonalities with variational quantum algorithms such as the variational quantum eigensolver (VQE) [189] may be fruitful. Indeed, as suggested in [89], it may be possible to take advantage of the set of natively available quantum gates and to use essentially a VQE approach to optimize the algorithm parameters. We are optimistic that tools and results from other aspects of quantum computation may prove useful towards a better understanding of the power of QAOA circuits.

Generally, a fundamental question is whether or not quantum computers provide advantages over classical algorithms for approximate optimization. Insight into this deep question would have important implications to computational complexity theory. We do not generally believe that quantum computers can efficiently solve NP-hard decision problems. However, it remains open whether or not there exists an NP-hard optimization problem such that a polynomial time quantum algorithm can give a better approximation (in the worst case) than any classical algorithms. For problems where we have tight classical algorithms and hardness of approximation results, this appears unlikely. On the other hand, for many problems there is a gap between the best known algorithm and complexity lower bound, and quantum computers may find utility here. Moreover, even if it turns out that quantum computers can only provide a quadratic speedup for an optimization problem, meaning finding the same quality solution as a classical algorithm but in reduced time, such a result may nevertheless be very important for solving large problem instances. Of further interest still is the performance of quantum computers for approximation in the average-case setting, and whether advantages can be realized through quantum heuristics generally. While we do not attempt to settle these important but difficult questions here, the results of this thesis are first steps towards this goal.

Chapter 7

Conclusions and Future Work

In this thesis we have studied five problems related to quantum algorithms for scientific computing and approximate optimization.

We first described a modular approach to scientific computing on quantum computers. We showed quantum algorithms and circuits for computing square roots, logarithms, and arbitrary fractional powers, and derived worst-case error and cost bounds. In order for future quantum computers to have impact for scientific problems, it will be important to develop numerical standards and libraries. Our work represents important first steps in this direction. A natural next step is to derive efficient quantum algorithms for additional useful numerical functions. Of particular interest are functions where the range of the input may be much smaller than that of the output, or vice versa; two examples are the exponential and arctangent functions. New techniques may be required for such functions to ensure that the input and output are represented efficiently while the error remains under control. Furthermore, it is important to explore specific applications of our circuits, as subroutines of larger quantum algorithms, to problems where quantum computers give advantages over their classical counterparts.

We then considered quantum algorithms for approximating ground and excited state energies, i.e., Hamiltonian eigenvalues. This problem suffers from the curse of dimensionality; for an ℓ-particle system, the cost of the best classical algorithms grows exponentially with ℓ. We showed a general quantum algorithm for approximating a constant number of low-order Hamiltonian eigenvalues using a perturbation approach. We then applied this algorithm to a computationally difficult special case of the Schrödinger equation and showed that our algorithm succeeds with high probability, and with cost polynomial in the number of degrees of freedom and the reciprocal of the desired accuracy. Our results significantly extend earlier work showing quantum computers can break the curse of dimensionality for this problem. It is important to continue working in this direction, to further weaken the assumptions if possible, and to extend the scope of our algorithm and its applications, in particular, to first-quantized approaches to important problems in physics and chemistry.

We next considered quantum algorithms for the simulation of quantum mechanical systems. We showed a novel divide and conquer approach for Hamiltonian simulation that takes advantage of the Hamiltonian structure to yield faster simulation algorithms. We illustrated our results by applying our approach to the the electronic structure problem of quantum chemistry, and showed significantly improved cost estimates under very mild assumptions. A next step for chemistry applications is to investigate particular classes of molecules and single-particle basis functions where our approach is particularly advantageous over current methods. Generally, it is important to further investigate the power and limitations of quantum algorithms for Hamiltonian simulation. An open problem is whether or not there exist Hamiltonian simulation algorithms for important applications with cost that scales polynomially in $\log \|H\| t$, $\log \varepsilon^{-1}$ and n, where n is the number of qubits the Hamiltonian H acts on. Finally, it remains open whether further improvements can be obtained by using our divide and conquer approach in combination with other Hamilton simulation algorithms as subroutines in place of splitting formulas.

Quantum algorithms for approximate optimization are relatively unexplored, with many basic problems open. Indeed, the fundamental question remains far from resolved: do quantum computers provide advantages for the approximation of classically hard combinatorial problems? We studied the application of the recently proposed quantum approximate optimization algorithm (QAOA) to the Maximum Cut problem. We showed a general technique, the Pauli Solver algorithm, which we applied to derive analytic performance bounds for the lowest depth realization of the algorithm. The details and proof of our results exemplify the difficulty of obtaining similar performance bounds for other problems or deeper circuits. Indeed, characterizing the performance of QAOA for depth $p > 1$, particularly how the performance improves as a function of p, remains the most important open problem for QAOA. It is important to find ways to further improve our techniques towards obtaining such results. Furthermore, for QAOA of arbitrary fixed depth, it remains open to classify, or give an efficient general procedure for finding, sufficiently good algorithm parameters, which is critical for QAOA to be effective in practice.

We then showed a generalization of QAOA to wider classes of quantum operators and states, the Quantum Alternating Operator Ansatz. Our approach is especially suitable to optimization problems with feasibility constraints. After specifying design criteria and a design toolkit, we applied our approach to yield efficient constructions for a variety of prototypical optimization problems. We derived explicit cost estimates for these constructions, in each case showing appealing resource scaling indicative of suitability for early quantum computers. Our ansatz allows freedom in the selection of operators and initial states. An important future direction is to investigate the trade-off between cost and performance in this selection; e.g., is a more costly mixing operator preferable to a less costly one, in terms of performance, if it means we can only afford to implement fewer rounds of the algorithm? Moreover, can we derive criteria specifying the best possible initial states and mixing operators? Quantum algorithms for approximate optimization remain at an early stage, so many open questions remain.

Finally, a primary future goal is to implement the algorithms of this thesis on physical quantum computers. As such devices become available, we are optimistic that experimentation, analysis, and testing will empower algorithm designers to discover a variety of new and improved impactful applications of quantum computing.

Bibliography

[1] AARONSON, S. Read the fine print. *Nature Physics 11*, 4 (2015), 291–293.

[2] ABRAMS, D. S., AND LLOYD, S. Simulation of many-body Fermi systems on a universal quantum computer. *Phys. Rev. Lett. 79*, 13 (1997), 2586.

[3] ABRAMS, D. S., AND LLOYD, S. Quantum algorithm providing exponential speed increase for finding eigenvalues and eigenvectors. *Phys. Rev. Lett. 83* (Dec 1999), 5162–5165.

[4] AHARONOV, D., AND NAVEH, T. Quantum NP – a survey. *arXiv preprint quant-ph/0210077* (2002).

[5] AHARONOV, D., AND TA-SHMA, A. Adiabatic quantum state generation and statistical zero knowledge. In *Proc. 35th ACM Symposium on Theory of Computing* (2003), ACM, pp. 20–29.

[6] AHARONOV, D., VAN DAM, W., KEMPE, J., LANDAU, Z., LLOYD, S., AND REGEV, O. Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM Review 50*, 4 (2008), 755–787.

[7] AHO, A. V., HOPCROFT, J. E., AND ULLMAN, J. *The Design and Analysis of Computer Algorithms*, 1st ed. Addison-Wesley, Boston, MA, USA, 1974.

[8] ALBASH, T., AND LIDAR, D. A. Adiabatic quantum computing. *arXiv preprint arXiv:1611.04471* (2016).

[9] ALMLÖF, J., FAEGRI, K., AND KORSELL, K. Principles for a direct SCF approach to LICAO–MO ab-initio calculations. *J. Comput. Chem. 3*, 3 (1982), 385–399.

[10] ALON, N. The algorithmic aspects of the regularity lemma. In *Proc. 33rd IEEE Symposium on Foundations of Computer Science* (1992), IEEE, pp. 473–481.

[11] ALON, N. Bipartite subgraphs. *Combinatorica 16*, 3 (1996), 301–311.

[12] ALON, N., AND HALPERIN, E. Bipartite subgraphs of integer weighted graphs. *Discrete Mathematics 181*, 1-3 (1998), 19–29.

[13] ALVAREZ-SANCHEZ, J. J., ALVAREZ-BRAVO, J. V., AND NIETO, L. M. A quantum architecture for multiplying signed integers. *Journal of Physics: Conference Series 128*, 1 (2008), 012013.

[14] ARORA, S., AND BARAK, B. *Computational Complexity - A Modern Approach.* Cambridge University Press, 2009.

[15] ASPURU-GUZIK, A., DUTOI, A. D., LOVE, P. J., AND HEAD-GORDON, M. Simulated quantum computation of molecular energies. *Science 309*, 5741 (2005), 1704–1707.

[16] AUSIELLO, G., CRESCENZI, P., GAMBOSI, G., KANN, V., MARCHETTI-SPACCAMELA, A., AND PROTASI, M. *Complexity and approximation: Combinatorial optimization problems and their approximability properties.* Springer Science & Business Media, 2012.

[17] BABBUSH, R., BERRY, D. W., KIVLICHAN, I. D., WEI, A. Y., LOVE, P. J., AND ASPURU-GUZIK, A. Exponentially more precise quantum simulation of fermions in second quantization. *New Journal of Physics 18*, 3 (2016), 033032.

[18] BABBUSH, R., MCCLEAN, J., WECKER, D., ASPURU-GUZIK, A., AND WIEBE, N. Chemical basis of Trotter-Suzuki errors in quantum chemistry simulation. *Phys. Rev. A 91*, 2 (2015), 022311.

[19] BABUŠKA, I., AND OSBORN, J. Eigenvalue problems. *Handbook of numerical analysis 2* (1991), 641–787.

[20] BARAHONA, F. The Max–Cut problem on graphs not contractible to K5. *Operations Research Letters 2*, 3 (1983), 107–111.

[21] BARAHONA, F., GRÖTSCHEL, M., JÜNGER, M., AND REINELT, G. An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research 36*, 3 (1988), 493–513.

[22] BARAK, B., MOITRA, A., O'DONNELL, R., RAGHAVENDRA, P., REGEV, O., STEURER, D., TREVISAN, L., VIJAYARAGHAVAN, A., WITMER, D., AND WRIGHT, J. Beating the random assignment on constraint satisfaction problems of bounded degree. *arXiv preprint arXiv:1505.03424* (2015).

[23] BAZGAN, C., ESCOFFIER, B., AND PASCHOS, V. T. Completeness in standard and differential approximation classes: Poly-(D) APX-and (D) PTAS-completeness. *Theoretical Computer Science 339*, 2-3 (2005), 272–292.

[24] BEALS, R., BUHRMAN, H., CLEVE, R., MOSCA, M., AND DE WOLF, R. Quantum lower bounds by polynomials. *Journal of the ACM (JACM) 48*, 4 (2001), 778–797.

[25] BECKMAN, D., CHARI, A. N., DEVABHAKTUNI, S., AND PRESKILL, J. Efficient networks for quantum factoring. *Phys. Rev. A 54* (Aug 1996), 1034–1063.

[26] BEIGEL, R. The polynomial method in circuit complexity. In *Proc. 8th Structure in Complexity Theory Conference* (1993), IEEE, pp. 82–95.

[27] BENNETT, C. H. Time/space trade-offs for reversible computation. *SIAM Journal on Computing 18*, 4 (1989), 766–776.

[28] BERNSTEIN, E., AND VAZIRANI, U. Quantum complexity theory. *SIAM Journal on Computing 26*, 5 (1997), 1411–1473.

[29] BERRY, D. W., AHOKAS, G., CLEVE, R., AND SANDERS, B. C. Efficient quantum algorithms for simulating sparse Hamiltonians. *Communications in Mathematical Physics 270*, 2 (2007), 359–371.

[30] BERRY, D. W., AND CHILDS, A. M. Black-box Hamiltonian simulation and unitary implementation. *Quantum Information & Computation 12*, 1-2 (2012), 29–62.

[31] BERRY, D. W., CHILDS, A. M., CLEVE, R., KOTHARI, R., AND SOMMA, R. D. Exponential improvement in precision for simulating sparse Hamiltonians. In *Proc. 46th ACM Symposium on Theory of Computing* (2014), ACM, pp. 283–292.

[32] BERRY, D. W., CHILDS, A. M., CLEVE, R., KOTHARI, R., AND SOMMA, R. D. Simulating Hamiltonian dynamics with a truncated Taylor series. *Phys. Rev. Lett. 114*, 9 (2015), 090502.

[33] BERRY, D. W., CHILDS, A. M., AND KOTHARI, R. Hamiltonian simulation with nearly optimal dependence on all parameters. In *Proc. 56th IEEE Symposium on Foundations of Computer Science* (2015), IEEE, pp. 792–809.

[34] BERRY, D. W., CLEVE, R., AND SOMMA, R. D. Exponential improvement in precision for Hamiltonian-evolution simulation. *arXiv preprint arXiv:1308.5424* (2013).

[35] BHASKAR, M. K., HADFIELD, S., PAPAGEORGIOU, A., AND PETRAS, I. Quantum algorithms and circuits for scientific computing. *Quantum Information & Computation 16*, 3-4 (2016), 197–236.

[36] BIAMONTE, J., WITTEK, P., PANCOTTI, N., REBENTROST, P., WIEBE, N., AND LLOYD, S. Quantum machine learning. *arXiv preprint arXiv:1611.09347* (2016).

[37] BIAN, Z., CHUDAK, F., MACREADY, W. G., AND ROSE, G. The Ising model: teaching an old problem new tricks. Tech. rep., D-Wave Systems, 2010.

[38] BISWAS, R., JIANG, Z., KECHEZHI, K., KNYSH, S., MANDRÀ, S., O'GORMAN, B., PERDOMO-ORTIZ, A., PETUKHOV, A., REALPE-GÓMEZ, J., RIEFFEL, E., ET AL. A NASA perspective on quantum computing: opportunities and challenges. *Parallel Computing 64* (2017), 81–98.

[39] BOCCHIERI, E. Fixed-point arithmetic. In *Automatic Speech Recognition on Mobile Devices and over Communication Networks*. Springer, 2008, pp. 255–275.

[40] BOGHOSIAN, B. M., AND TAYLOR, W. Simulating quantum mechanics on a quantum computer. *Physica D: Nonlinear Phenomena 120*, 1 (1998), 30–42.

[41] BOIXO, S., ISAKOV, S. V., SMELYANSKIY, V. N., BABBUSH, R., DING, N., JIANG, Z., MARTINIS, J. M., AND NEVEN, H. Characterizing quantum supremacy in near-term devices. *arXiv:1608.00263* (July 2016).

[42] BOIXO, S., RØNNOW, T. F., ISAKOV, S. V., WANG, Z., WECKER, D., LIDAR, D. A., MARTINIS, J. M., AND TROYER, M. Evidence for quantum annealing with more than one hundred qubits. *Nature Physics 10*, 3 (2014), 218.

[43] BOIXO, S., SMELYANSKIY, V. N., SHABANI, A., ISAKOV, S. V., DYKMAN, M., DENCHEV, V. S., AMIN, M. H., SMIRNOV, A. Y., MOHSENI, M., AND NEVEN, H. Computational multiqubit tunnelling in programmable quantum annealers. *Nature communications 7* (2016).

[44] BOOKATZ, A. D. QMA-complete problems. *Quantum Information & Computation 14*, 5&6 (2014), 361–383.

[45] BOROS, E., AND HAMMER, P. L. Pseudo-Boolean optimization. *Discrete applied mathematics 123*, 1 (2002), 155–225.

[46] BOROS, E., HAMMER, P. L., AND TAVARES, G. Local search heuristics for quadratic unconstrained binary optimization (QUBO). *Journal of Heuristics 13*, 2 (2007), 99–132.

[47] BOYD, S., AND VANDENBERGHE, L. *Convex Optimization*. Cambridge university press, 2004.

[48] BRASSARD, G., HOYER, P., MOSCA, M., AND TAPP, A. Quantum amplitude amplification and estimation. *Contemporary Mathematics 305* (2002), 53–74.

[49] BRAVYI, S., DIVINCENZO., D. P., OLIVEIRA, R. I., AND TERHAL, B. M. The complexity of stoquastic local Hamiltonian problems. *Quantum Information & Computation 8*, 5 (2008), 361–385.

[50] BRAVYI, S. B., AND KITAEV, A. Y. Fermionic quantum computation. *Annals of Physics 298*, 1 (2002), 210–226.

[51] BRIEGEL, H. J., BROWNE, D. E., DÜR, W., RAUSSENDORF, R., AND VAN DEN NEST, M. Measurement-based quantum computation. *Nature Physics 5*, 1 (2009), 19–26.

[52] BROADBENT, A., AND SCHAFFNER, C. Quantum cryptography beyond quantum key distribution. *Designs, Codes and Cryptography 78*, 1 (2016), 351–382.

[53] BRUCKER, P. *Scheduling Algorithms*, vol. 5. Springer, 2007.

[54] BULUTA, I., AND NORI, F. Quantum simulators. *Science 326*, 5949 (2009), 108–111.

[55] BYRNES, T., AND YAMAMOTO, Y. Simulating lattice gauge theories on a quantum computer. *Phys. Rev. A 73*, 2 (2006), 022328.

[56] CAO, Y., PAPAGEORGIOU, A., PETRAS, I., TRAUB, J. F., AND KAIS, S. Quantum algorithm and circuit design solving the Poisson equation. *New Journal of Physics 15* (2013), 013021.

[57] CHAITIN, G. J. Register allocation & spilling via graph coloring. In *ACM Sigplan Notices* (1982), vol. 17, ACM, pp. 98–105.

[58] CHENG, T. E., NG, C., YUAN, J., AND LIU, Z. Single machine scheduling to minimize total weighted tardiness. *European Journal of Operational Research 165*, 2 (2005), 423–443.

[59] CHILDS, A. M. *Quantum information processing in continuous time*. PhD thesis, Massachusetts Institute of Technology, 2004.

[60] CHILDS, A. M., FARHI, E., GOLDSTONE, J., AND GUTMANN, S. Finding cliques by quantum adiabatic evolution. *Quantum Information & Computation 2*, 3 (2002), 181–191.

[61] CHILDS, A. M., GOSSET, D., AND WEBB, Z. The Bose-Hubbard model is QMA-complete. In *Automata, Languages, and Programming*, J. Esparza, P. Fraigniaud, T. Husfeldt, and E. Koutsoupias, Eds., vol. 8572 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2014, pp. 308–319.

[62] CHILDS, A. M., AND KOTHARI, R. Limitations on the simulation of non-sparse Hamiltonians. *Quantum Information & Computation 10*, 10 (2010), 669–684.

[63] CHILDS, A. M., AND KOTHARI, R. Simulating sparse Hamiltonians with star decompositions. In *Theory of Quantum Computation, Communication, and Cryptography*. Springer, 2010, pp. 94–103.

[64] CHILDS, A. M., AND VAN DAM, W. Quantum algorithms for algebraic problems. *Reviews of Modern Physics 82*, 1 (2010), 1.

[65] CHILDS, A. M., AND WIEBE, N. Hamiltonian simulation using linear combinations of unitary operations. *Quantum Information & Computation 12*, 12 (2012), 901–924.

[66] CHRISTOFFERSEN, R. E. Ab initio calculations on large molecules. *Advances in Quantum Chemistry 6* (1972), 333–393.

[67] CHRISTOFIDES, N. Worst-case analysis of a new heuristic for the travelling salesman problem. Tech. rep., Carnegie-Mellon University Management Sciences Research Group, 1976.

[68] CHUANG, I., AND MODHA, D. S. Reversible arithmetic coding for quantum data compression. *IEEE Transactions on Information Theory 46* (2000), 1104–1116.

[69] CLEMENTI, E. Computation of large molecules with the Hartree-Fock model. *Proceedings of the National Academy of Sciences of the United States of America 69*, 10 (1972), 2942.

[70] COOK, S. A. The complexity of theorem-proving procedures. In *Proc. 3rd ACM Symposium on Theory of Computing* (1971), ACM, pp. 151–158.

[71] CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., AND STEIN, C. *Introduction to Algorithms*. MIT press, 2009.

[72] CRESCENZI, P. A short guide to approximation preserving reductions. In *Computational Complexity, 1997. Proceedings., Twelfth Annual IEEE Conference on (Formerly: Structure in Complexity Theory Conference)* (1997), IEEE, pp. 262–273.

[73] CUCCARO, S. A., DRAPER, T. G., KUTIN, S. A., AND MOULTON, D. P. A new quantum ripple-carry addition circuit. *8th Workshop on Quantum Information Processing* (2004).

[74] CULLUM, J. K., AND WILLOUGHBY, R. A. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations: Vol. 1: Theory*, vol. 41. SIAM, 2002.

[75] DÁNIEL, M. Graph colouring problems and their applications in scheduling. *Periodica Polytech., Electr. Eng 48*, 1-2 (2004), 11–16.

[76] DE WOLF, R. A brief introduction to Fourier analysis on the Boolean cube. *Theory of Computing, Graduate Surveys 1* (2008), 1–20.

[77] DEMMEL, J. W. *Applied Numerical Linear Algebra.* SIAM, Philadelphia, PA, 1997.

[78] DENCHEV, V. S., BOIXO, S., ISAKOV, S. V., DING, N., BABBUSH, R., SMELYANSKIY, V., MARTINIS, J., AND NEVEN, H. What is the computational value of finite-range tunneling? *Phys. Rev. X 6*, 3 (2016), 031015.

[79] DEUTSCH, D. Quantum theory, the Church-Turing principle and the universal quantum computer. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* (1985), vol. 400, The Royal Society, pp. 97–117.

[80] DEUTSCH, D. Quantum computational networks. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* (1989), vol. 425, The Royal Society, pp. 73–90.

[81] DEZA, M., AND LAURENT, M. Applications of cut polyhedra–ii. *Journal of Computational and Applied Mathematics 55*, 2 (1994), 217–247.

[82] DINUR, I., AND SAFRA, S. On the hardness of approximating minimum vertex cover. *Annals of mathematics* (2005), 439–485.

[83] DRAPER, T. G. Addition on a quantum computer. *arXiv preprint quant-ph/0008033* (2000).

[84] DRAPER, T. G., KUTIN, S. A., RAINS, E. M., AND SVORE, K. M. A logarithmic-depth quantum carry-lookahead adder. *Quantum Information & Computation 6*, 4 (July 2006), 351–369.

[85] DUNNING, T., ET AL. Gaussian basis functions for use in molecular calculations. iii. contraction of (10s6p) atomic basis sets for the first-row atoms. *Journal of Chemical Physics 55* (1971), 716–723.

[86] FARHI, E., GOLDSTONE, J., AND GUTMANN, S. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028* (2014).

[87] FARHI, E., GOLDSTONE, J., AND GUTMANN, S. A quantum approximate optimization algorithm applied to a bounded occurrence constraint problem. *arXiv preprint arXiv:1412.6062* (2014).

[88] FARHI, E., GOLDSTONE, J., GUTMANN, S., LAPAN, J., LUNDGREN, A., AND PREDA, D. A quantum adiabatic evolution algorithm applied to random instances of an NP–complete problem. *Science 292*, 5516 (2001), 472–475.

[89] FARHI, E., GOLDSTONE, J., GUTMANN, S., AND NEVEN, H. Quantum algorithms for fixed qubit architectures. *arXiv preprint arXiv:1703.06199* (2017).

[90] FARHI, E., GOLDSTONE, J., GUTMANN, S., AND SIPSER, M. Quantum computation by adiabatic evolution. *arXiv preprint quant-ph/0001106* (2000).

[91] FARHI, E., AND HARROW, A. W. Quantum supremacy through the quantum approximate optimization algorithm. *arXiv preprint arXiv:1602.07674* (2016).

[92] FEIGE, U. Approximating maximum clique by removing subgraphs. *SIAM Journal on Discrete Mathematics 18*, 2 (2004), 219–225.

[93] FEIGE, U., KARPINSKI, M., AND LANGBERG, M. Improved approximation of max-cut on graphs of bounded degree. *Journal of Algorithms 43*, 2 (2002), 201–219.

[94] FESTA, P., PARDALOS, P. M., RESENDE, M. G., AND RIBEIRO, C. C. Randomized heuristics for the max-cut problem. *Optimization methods and software 17*, 6 (2002), 1033–1058.

[95] FEYNMAN, R. Simulating physics with computers. *SIAM Journal on Computing 26* (1982), 1484–1509.

[96] FEYNMAN, R. P., LEIGHTON, R. B., AND SANDS, M. Lectures on physics, vol. iii, 1965.

[97] FOLLAND, G. B. *Real Analysis: Modern Techniques and Their Applications*. Wiley Inter-Science, 1999.

[98] FORSYTHE, G. E., AND WASOW, W. R. *Finite–Difference Methods for Partial Differential Equations*. Dover, New York, 2004.

[99] FRIEZE, A., AND JERRUM, M. Improved approximation algorithms for max–k–cut and max bisection. *Algorithmica 18*, 1 (1997), 67–81.

[100] FURCHE, F., AND RAPPOPORT, D. *Density functional methods for excited states: equilibrium structure and electronic spectra*, vol. 16 of *Theoretical and Computational Chemistry*. Elsevier, Amsterdam, 2005, pp. 93–128.

[101] GAREY, M. R., AND JOHNSON, D. S. *Computers and Intractability: A Guide to the Theory of NP–Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[102] GEORGESCU, I., ASHHAB, S., AND NORI, F. Quantum simulation. *Reviews of Modern Physics 86*, 1 (2014), 153.

[103] GIVANT, S., AND HALMOS, P. *Introduction to Boolean Algebras*. Springer Science & Business Media, 2008.

[104] GOEDECKER, S. Linear scaling electronic structure methods. *Reviews of Modern Physics 71*, 4 (1999), 1085.

[105] GOEMANS, M. X., AND WILLIAMSON, D. P. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM) 42*, 6 (1995), 1115–1145.

[106] GOLUB, G. H., AND VAN LOAN, C. F. *Matrix Computations*. JHU Press, 2012.

[107] GRIFFITHS, D. J. *Introduction to Quantum Mechanics*. Cambridge University Press, 2016.

[108] GROVER, L. K. Quantum mechanics helps in searching for a needle in a haystack. *Phys. Rev. Lett. 79*, 2 (1997), 325.

[109] GUERRESCHI, G. G., AND SMELYANSKIY, M. Practical optimization for hybrid quantum-classical algorithms. *arXiv preprint arXiv:1701.01450* (2017).

[110] GUSTAFSON, S. J., AND SIGAL, I. M. *Mathematical Concepts of Quantum Mechanics*. Universitext. Springer, 2011.

[111] HADFIELD, S. On the representation of Boolean and real functions as Hamiltonians for quantum computing. *arXiv preprint arXiv:1804.09130* (2018).

[112] HADFIELD, S., AND PAPAGEORGIOU, A. Approximating ground and excited state energies on a quantum computer. *Quantum Information Processing 14*, 4 (2015), 1151–1178.

[113] HADFIELD, S., AND PAPAGEORGIOU, A. Divide and conquer approach to quantum hamiltonian simulation. *New Journal of Physics 20*, 4 (2018), 043003.

[114] HADFIELD, S., WANG, Z., O'GORMAN, B., RIEFFEL, E. G., VENTURELLI, D., AND BISWAS, R. From the quantum approximate optimization algorithm to a quantum alternating operator ansatz. *arXiv preprint arXiv:1709.03489* (2017).

[115] HADFIELD, S., WANG, Z., RIEFFEL, E. G., O'GORMAN, B., VENTURELLI, D., AND BISWAS, R. Quantum approximate optimization with hard and soft constraints. In *Proceedings of the Second International Workshop on Post Moores Era Supercomputing* (New York, NY, USA, 2017), PMES'17, ACM, pp. 15–21.

[116] HADLOCK, F. Finding a maximum cut of a planar graph in polynomial time. *SIAM Journal on Computing 4*, 3 (1975), 221–225.

[117] HAGLIN, D. J., AND VENKATESAN, S. M. Approximation and intractability results for the maximum cut problem and its variants. *IEEE Transactions on Computers 40*, 1 (1991), 110–113.

[118] HALLDÓRSSON, M. M. A still better performance guarantee for approximate graph coloring. *Information Processing Letters 45*, 1 (1993), 19–23.

[119] HALLDÓRSSON, M. M. Approximating discrete collections via local improvements. In *SODA* (1995), vol. 95, pp. 160–169.

[120] HAMMER, P. L., AND RUDEANU, S. *Boolean Methods in Operations Research and Related Areas*, vol. 7. Springer Science & Business Media, 2012.

[121] HARROW, A. W., HASSIDIM, A., AND LLOYD, S. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett. 103* (Oct 2009), 150502.

[122] HÅSTAD, J. Some optimal inapproximability results. *Journal of the ACM (JACM) 48*, 4 (2001), 798–859.

[123] HASTINGS, M. B., WECKER, D., BAUER, B., AND TROYER, M. Improving quantum algorithms for quantum chemistry. *Quantum Information & Computation 15*, 1-2 (2015), 1–21.

[124] HATANO, N., AND SUZUKI, M. Finding exponential product formulas of higher orders. In *Quantum annealing and other optimization methods*. Springer, 2005, pp. 37–68.

[125] HEHRE, W. J., STEWART, R. F., AND POPLE, J. A. Self-consistent molecular-orbital methods. i. Use of Gaussian expansions of Slater-type atomic orbitals. *Journal of Chemical Physics 51*, 6 (1969), 2657–2664.

[126] HELGAKER, T., JORGENSEN, P., AND OLSEN, J. *Molecular Electronic-Structure Theory*. John Wiley & Sons, 2014.

[127] HEN, I., AND SARANDY, M. S. Driver Hamiltonians for constrained optimization in quantum annealing. *Phys. Rev. A 93*, 6 (2016), 062312.

[128] HEN, I., AND SPEDALIERI, F. M. Quantum annealing for constrained optimization. *Phys. Rev. Appl. 5*, 3 (2016), 034007.

[129] HISLOP, P. D., AND SIGAL, I. M. *Introduction to Spectral Theory: With Applications to Schrödinger Operators*. No. v. 113 in Applied Mathematical Sciences Series. Springer Verlag, New York, 1996.

[130] HOFMEISTER, T., AND LEFMANN, H. A combinatorial design approach to maxcut. *STACS 96* (1996), 439–452.

[131] HOGG, T., AND PORTNOV, D. Quantum optimization. *Information Sciences 128*, 3-4 (2000), 181–197.

[132] HOHENBERG, P., AND KOHN, W. Inhomogeneous electron gas. *Phys. Rev. 136*, 3B (1964), B864.

[133] HORN, R. A., AND JOHNSON, C. R. *Matrix Analysis*. Cambridge university press, 2012.

[134] HROMKOVIC, J. Algorithmics for hard problems: Introduction to combinatorial optimization. *Randomization, Approximation, and Heuristics* (2002), 238–248.

[135] IBM. IBM Q and Quantum Computing. https://www.research.ibm.com/ibm-q/, 2017. Accessed: 2017-09-01.

[136] IEEE standard for Floating-Point Arithmetic. *IEEE Computer Society Std 754-2008* (Aug 2008), 1–70.

[137] JAHNKE, T., AND LUBICH, C. Error bounds for exponential operator splittings. *BIT Numerical Mathematics 40*, 4 (2000), 735–744.

[138] JIANG, Z., RIEFFEL, E. G., AND WANG, Z. Near-optimal quantum circuit for Grover's unstructured search using a transverse field. *Phys. Rev. A 95*, 6 (2017), 062317.

[139] JONES, N. C., WHITFIELD, J. D., MCMAHON, P. L., YUNG, M.-H., VAN METER, R., ASPURU-GUZIK, A., AND YAMAMOTO, Y. Faster quantum chemistry simulation on fault-tolerant quantum computers. *New Journal of Physics 14*, 11 (2012), 115023.

[140] JORDAN, S. P., LEE, K. S., AND PRESKILL, J. Quantum algorithms for fermionic quantum field theories. *arXiv preprint arXiv:1404.7115* (2014).

[141] JORDAN, S. P., LEE, K. S., AND PRESKILL, J. Quantum computation of scattering in scalar quantum field theories. *Quantum Information & Computation 14*, 11-12 (2014), 1014–1080.

[142] KADOWAKI, T., AND NISHIMORI, H. Quantum annealing in the transverse Ising model. *Phys. Rev. E 58*, 5 (1998), 5355.

[143] KAHN, J., KALAI, G., AND LINIAL, N. The influence of variables on Boolean functions. In *Proc. 29th IEEE Symposium on Foundations of Computer Science* (1988), IEEE, pp. 68–80.

[144] KARAKOSTAS, G. A better approximation ratio for the vertex cover problem. *ACM Transactions on Algorithms (TALG) 5*, 4 (2009), 41.

[145] KARP, R. M. Reducibility among combinatorial problems. In *Complexity of computer computations*. Springer, 1972, pp. 85–103.

[146] KASSAL, I., JORDAN, S. P., LOVE, P. J., MOHSENI, M., AND ASPURU-GUZIK, A. Polynomial-time quantum algorithm for the simulation of chemical dynamics. *Proceedings of the National Academy of Sciences 105*, 48 (2008), 18681–18686.

[147] KASSAL, I., WHITFIELD, J. D., PERDOMO-ORTIZ, A., YUNG, M.-H., AND ASPURU-GUZIK, A. Simulating chemistry using quantum computers. *Annual review of physical chemistry 62* (2011), 185–207.

[148] KEMPE, J., KITAEV, A., AND REGEV, O. The complexity of the local Hamiltonian problem. *SIAM J. Comput. 35*, 5 (2006), 1070–1097.

[149] KEPLEY, S., AND STEINWANDT, R. Quantum circuits for $\mathbb{F}_{2^n}$-multiplication with subquadratic gate count. *Quantum Information Processing 11* (2015), 2373–2386.

[150] KHANNA, S., MOTWANI, R., SUDAN, M., AND VAZIRANI, U. On syntactic versus computational views of approximability. *SIAM Journal on Computing 28*, 1 (1998), 164–191.

[151] KHOT, S., KINDLER, G., MOSSEL, E., AND O'DONNELL, R. Optimal inapproximability results for max-cut and other 2-variable csps. *SIAM Journal on Computing 37*, 1 (2007), 319–357.

[152] KITAEV, A. Y., SHEN, A., AND VYALYI, M. N. *Classical and Quantum Computation*, vol. 47. American Mathematical Society, 2002.

[153] KLAPPENECKER, A., AND RÖTTELER, M. Discrete cosine transforms on quantum computers. In *Proceedings of the 2nd International Symposium on Image and Signal Processing and Analysis* (2001), pp. 464–468.

[154] LANYON, B. P., WHITFIELD, J. D., GILLETT, G. G., GOGGIN, M. E., ALMEIDA, M. P., KASSAL, I., BIAMONTE, J. D., MOHSENI, M., POWELL, B. J., BARBIERI, M., ASPURU-GUZIK, A., AND WHITE, A. G. Towards quantum chemistry on a quantum computer. *Nature Chemistry 2* (2010), 106–111.

[155] LEIGHTON, F. T. A graph coloring algorithm for large scheduling problems. *Journal of research of the national bureau of standards 84*, 6 (1979), 489–506.

[156] LENSTRA, J. K., KAN, A. R., AND BRUCKER, P. Complexity of machine scheduling problems. *Annals of discrete mathematics 1* (1977), 343–362.

[157] LEVEQUE, R. J. *Finite Difference Methods for Ordinary and Partial Differential Equations*. SIAM, Philadelphia, PA., 2007.

[158] LEVINE, R. Y., AND SHERMAN, A. T. A note on Bennett's time-space tradeoff for reversible computation. *SIAM Journal on Computing 19*, 4 (1990), 673–677.

[159] LIDAR, D. A., AND BRUN, T. A. *Quantum Error Correction*. Cambridge University Press, 2013.

[160] LIEB, E., SCHULTZ, T., AND MATTIS, D. Two soluble models of an antiferromagnetic chain. In *Condensed Matter Physics and Exactly Soluble Models*. Springer, 2004, pp. 543–601.

[161] LINIAL, N., MANSOUR, Y., AND NISAN, N. Constant depth circuits, Fourier transform, and learnability. *Journal of the ACM (JACM) 40*, 3 (1993), 607–620.

[162] LLOYD, S. Universal quantum simulators. *Science 23 273*, 5278 (1996), 1073–1078.

[163] LLOYD, S., AND BRAUNSTEIN, S. L. Quantum computation over continuous variables. *Phys. Rev. Lett. 82*, 8 (1999), 1784.

[164] LOVE, P. J. *Back to the Future: A roadmap for quantum simulation from vintage quantum chemistry*, vol. 154 of *Advances in Chemical Physics*. Wiley, Hoboken, NJ, 2014, pp. 39–66.

[165] LUCAS, A. Ising formulations of many NP problems. *Frontiers in Physics 2*, 5 (2014), 1–15.

[166] MARTOŇÁK, R., SANTORO, G. E., AND TOSATTI, E. Quantum annealing of the traveling-salesman problem. *Phys. Rev. E 70*, 5 (2004), 057701.

[167] MCCLEAN, J. R., BABBUSH, R., LOVE, P. J., AND ASPURU-GUZIK, A. Exploiting locality in quantum computation for quantum chemistry. *The Journal of Physical Chemistry Letters 5*, 24 (2014), 4368–4380.

[168] MCGEOCH, C. C. Adiabatic quantum computation and quantum annealing: theory and practice. *Synthesis Lectures on Quantum Computing 5*, 2 (2014), 1–93.

[169] MOHSENI, M., READ, P., NEVEN, H., BOIXO, S., DENCHEV, V., BABBUSH, R., FOWLER, A., SMELYANSKIY, V., AND MARTINIS, J. Commercialize early quantum technologies. *Nature 543* (2017), 171–174.

[170] MONTANARO, A. Quantum algorithms: an overview. *npj Quantum Information 2* (2016), 15023.

[171] MONTANARO, A., AND OSBORNE, T. J. Quantum Boolean functions. *arXiv preprint arXiv:0810.2435* (2008).

[172] MOSCA, M. Quantum algorithms. In *Computational Complexity*. Springer, 2012, pp. 2303–2333.

[173] NIELSEN, M., AND CHUANG, I. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge UK, 2000.

[174] NISAN, N., AND SZEGEDY, M. On the degree of Boolean functions as real polynomials. *Computational complexity 4*, 4 (1994), 301–313.

[175] O'DONNELL, R. *Analysis of Boolean functions*. Cambridge University Press, 2014.

[176] ORPONEN, P., AND MANNILA, H. On approximation preserving reductions: Complete problems and robust measures (revised version). *Department of Computer Science, University of Helsinki* (1990).

[177] ORTIZ, G., GUBERNATIS, J., KNILL, E., AND LAFLAMME, R. Quantum algorithms for fermionic simulations. *Phys. Rev. A 64*, 2 (2001), 022319.

[178] PANCONESI, A., AND RANJAN, D. Quantifiers and approximation. In *Proc. 22nd ACM Symposium on Theory of Computing* (1990), ACM, pp. 446–456.

[179] PAPADIMITRIOU, C. H., AND YANNAKAKIS, M. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences 43* (1991), 425–440.

[180] PAPADIMITRIOU, C. H., AND YANNAKAKIS, M. The traveling salesman problem with distances one and two. *Mathematics of Operations Research 18*, 1 (1993), 1–11.

[181] PAPAGEORGIOU, A. On the complexity of the multivariate Sturm–Liouville eigenvalue problem. *J. Complexity 23*, 4-6 (2007), 802–827.

[182] PAPAGEORGIOU, A., AND PETRAS, I. Estimating the ground state energy of the Schrödinger equation for convex potentials. *J. Complexity 30* (2014), 469–494.

[183] PAPAGEORGIOU, A., PETRAS, I., TRAUB, J. F., AND ZHANG, C. A fast algorithm for approximating the ground state energy on a quantum computer. *Mathematics of Computation 82*, 284 (2014), 2293–2304.

[184] PAPAGEORGIOU, A., AND TRAUB, J. F. Measures of quantum computing speedup. *Phys. Rev. A 88*, 2 (2013), 022316.

[185] PAPAGEORGIOU, A., AND TRAUB, J. F. *Quantum algorithms for continuous problems and their applications*, vol. 154 of *Advances in Chemical Physics*. Wiley, Hoboken, NJ, 2014, pp. 151–178.

[186] PAPAGEORGIOU, A., AND ZHANG, C. On the efficiency of quantum algorithms for Hamiltonian simulation. *Quantum Information Processing 11* (2012), 541–561.

[187] PARENT, A., ROETTELER, M., AND SVORE, K. M. Reversible circuit compilation with space constraints. *arXiv preprint arXiv:1510.00377* (2015).

[188] PARLETT, B. N. *The Symmetric Eigenvalue Problem*, vol. 7. SIAM, 1980.

[189] PERUZZO, A., MCCLEAN, J., SHADBOLT, P., YUNG, M.-H., ZHOU, X.-Q., LOVE, P. J., ASPURU-GUZIK, A., AND O'BRIEN, J. L. A variational eigenvalue solver on a photonic quantum processor. *Nature communications 5* (2014), 4213.

[190] PETRANK, E. The hardness of approximation: gap location. *Computational Complexity 4*, 2 (1994), 133–157.

[191] POLJAK, S., AND TURZIK, D. A polynomial algorithm for constructing a large bipartite subgraph, with an application to a satisfiability problem. *Can. J. Math 34*, 3 (1982), 519–524.

[192] PORTUGAL, R., AND FIGUEIREDO, C. M. H. Reversible Karatsuba's algorithm. *Journal of Universal Computer Science 12*, 5 (2006), 499–511.

[193] POULIN, D., HASTINGS, M. B., WECKER, D., WIEBE, N., DOHERTY, A. C., AND TROYER, M. The Trotter step size required for accurate quantum simulation of quantum chemistry. *Quantum Information & Computation 15*, 5-6 (2015), 361–384.

[194] POULIN, D., QARRY, A., SOMMA, R., AND VERSTRAETE, F. Quantum simulation of time-dependent Hamiltonians and the convenient illusion of Hilbert space. *Phys. Rev. Lett. 106*, 17 (2011), 170501.

[195] PRESKILL, J. Lecture notes for physics 229: Quantum information and computation. *California Institute of Technology 16* (1998).

[196] PRESKILL, J. Quantum computing and the entanglement frontier. *arXiv:1203.5813* (Mar. 2012).

[197] RAEISI, S., WIEBE, N., AND SANDERS, B. C. Quantum-circuit design for efficient simulations of many-body quantum dynamics. *New Journal of Physics 14*, 10 (2012), 103017.

[198] REIHER, M., WIEBE, N., SVORE, K. M., WECKER, D., AND TROYER, M. Elucidating reaction mechanisms on quantum computers. *Proceedings of the National Academy of Sciences* (2017), 201619152.

[199] RIEFFEL, E. G., AND POLAK, W. H. *Quantum Computing: A Gentle Introduction.* MIT Press, 2011.

[200] RIEFFEL, E. G., VENTURELLI, D., DO, M., HEN, I., AND FRANK, J. Parametrized Families of Hard Planning Problems from Phase Transitions. In *AAAI* (2014), pp. 2337–2343.

[201] RIEFFEL, E. G., VENTURELLI, D., O'GORMAN, B., DO, M. B., PRYSTAY, E. M., AND SMELYANSKIY, V. N. A case study in programming a quantum annealer for hard operational planning problems. *Quantum Information Processing 14*, 1 (2015), 1–36.

[202] SAEEDI, M., AND MARKOV, I. L. Synthesis and optimization of reversible circuits - a survey. *ACM Computing Surveys (CSUR) 45*, 2 (2013), 21.

[203] SAHNI, S., AND GONZALEZ, T. P-complete approximation problems. *Journal of the ACM (JACM) 23*, 3 (1976), 555–565.

[204] SAKURAI, J. J. *Modern Quantum Mechanics (revised edition).* Addison Wesley, 1995.

[205] SCHUCH, N., AND VERSTRAETE, F. Computational complexity of interacting electrons and fundamental limitations of density functional theory. *Nature Physics 5*, 10 (2009), 732–735.

[206] SEELEY, J. T., RICHARD, M. J., AND LOVE, P. J. The Bravyi-Kitaev transformation for quantum computation of electronic structure. *The Journal of chemical physics 137*, 22 (2012), 224109.

[207] SETE, E. A., ZENG, W. J., AND RIGETTI, C. T. A functional architecture for scalable quantum computing. In *2016 IEEE International Conference on Rebooting Computing (ICRC)* (Oct 2016), pp. 1–6.

[208] SHANKAR, R. *Principles of Quantum Mechanics*. Springer Science & Business Media, 2012.

[209] SHOR, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput. 26*, 5 (Oct. 1997), 1484–1509.

[210] SOMMA, R., ORTIZ, G., GUBERNATIS, J. E., KNILL, E., AND LAFLAMME, R. Simulating physical phenomena by quantum networks. *Phys. Rev. A 65*, 4 (2002), 042323.

[211] STRANG, G., AND FIX, G. J. *An Analysis of the Finite Element Method*, 2 ed. Wellesley-Cambridge Press, 2008.

[212] STROUT, D. L., AND SCUSERIA, G. E. A quantitative study of the scaling properties of the Hartree-Fock method. *The Journal of chemical physics 102*, 21 (1995), 8448–8452.

[213] SUZUKI, M. 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 (1976), 183–190.

[214] SUZUKI, M. Fractal decomposition of exponential operators with applications to many-body theories and Monte Carlo simulations. *Phys. Letters A 146*, 6 (1990), 319–323.

[215] SUZUKI, M. General theory of fractal path integrals with application to many-body theories and statistical physics. *J. Math. Phys. 32* (1991), 400–407.

[216] SUZUKI, S., INOUE, J.-I., AND CHAKRABARTI, B. K. *Quantum Ising Phases and Transitions in Transverse Ising Models*, vol. 862. Springer, 2012.

[217] SZABO, A., AND OSTLUND, N. S. *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory*. Dover, 1996.

[218] TA-SHMA, A. Inverting well conditioned matrices in quantum logspace. *Proc. 45th ACM Symposium on Theory of Computing* (2013), 881–890.

[219] TAKAHASHI, Y., AND KUNIHIRO, N. A linear-size quantum circuit for addition with no ancillary qubits. *Quantum Information & Computation 5*, 6 (Sept. 2005), 440–448.

[220] TAKAHASHI, Y., AND KUNIHIRO, N. A fast quantum circuit for addition with few qubits. *Quantum Information & Computation 8*, 6 (2008), 636–649.

[221] TAKAHASHI, Y., TANI, S., AND KUNIHIRO, N. Quantum addition circuits and unbounded fan-out. *Quantum Information & Computation 10*, 9&10 (2010), 0872–0890.

[222] TAVARES, G. *New algorithms for quadratic unconstrained binary optimization (QUBO) with applications in engineering and social sciences*. Rutgers The State University of New Jersey-New Brunswick, 2008.

[223] TITCHMARSH, E. *Eigenfunction Expansions: Associated with Second-Order Differential Equations*. Eigenfunction Expansions Associated with Second-order Differential Equations. Oxford University Press, Oxford, UK, 1962.

[224] TOLOUI, B., AND LOVE, P. J. Quantum algorithms for quantum chemistry based on the sparsity of the CI-matrix. *arXiv preprint arXiv:1312.2579* (2013).

[225] TRAUB, J. F. *Iterative Methods for the Solution of Equations*. American Mathematical Soc., 1982.

[226] TREVISAN, L. Inapproximability of combinatorial optimization problems. *arXiv preprint cs/0409043* (2004).

[227] VAISMAN, R., ROUGHAN, M., AND KROESE, D. P. The multilevel splitting algorithm for graph colouring with application to the potts model. *Philosophical Magazine 97*, 19 (2017), 1646–1673.

[228] VAN METER, R., AND ITOH, K. M. Fast quantum modular exponentiation. *Phys. Rev. A 71*, 5 (2005), 052320.

[229] VAZIRANI, V. V. *Approximation Algorithms*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[230] VEDRAL, V., BARENCO, A., AND EKERT, A. Quantum networks for elementary arithmetic operations. *Phys. Rev. A 54* (Jul 1996), 147–153.

[231] VENTURELLI, D., DO, M., RIEFFEL, E., AND FRANK, J. Compiling quantum circuits to realistic hardware architectures using temporal planners. *Quantum Science and Technology* (2017).

[232] VERSTRAETE, F., CIRAC, J. I., AND LATORRE, J. I. Quantum circuits for strongly correlated quantum systems. *Phys. Rev. A 79*, 3 (2009), 032316.

[233] VIŠŇÁK, J. Quantum algorithms for computational nuclear physics. In *EPJ Web of Conferences* (2015), vol. 100, EDP Sciences, p. 01008.

[234] VITÁNYI, P. M. How well can a graph be n-colored? *Discrete mathematics 34*, 1 (1981), 69–80.

[235] VON NEUMANN, J., AND RÉDEI, M. *John von Neumann: selected letters*, vol. 27. American Mathematical Soc., 2005.

[236] WANG, H., KAIS, S., ASPURU-GUZIK, A., AND HOFFMANN, M. R. Quantum algorithm for obtaining the energy spectrum of molecular systems. *Physical Chemistry Chemical Physics 10*, 35 (2008), 5388–5393.

[237] WANG, Z. *Topological Quantum Computation*. No. 112. American Mathematical Society, 2010.

[238] WANG, Z., HADFIELD, S., JIANG, Z., AND RIEFFEL, E. G. Quantum approximate optimization algorithm for MaxCut: A fermionic view. *Physical Review A 97*, 2 (2018), 022304.

[239] WATROUS, J. Quantum computational complexity. In *Encyclopedia of complexity and systems science*. Springer, 2009, pp. 7174–7201.

[240] WECKER, D., BAUER, B., CLARK, B. K., HASTINGS, M. B., AND TROYER, M. Gate-count estimates for performing quantum chemistry on small quantum computers. *Phys. Rev. A 90*, 2 (2014), 022305.

[241] WECKER, D., HASTINGS, M. B., AND TROYER, M. Training a quantum optimizer. *Phys. Rev. A 94*, 2 (2016), 022309.

[242] WECKER, D., HASTINGS, M. B., WIEBE, N., CLARK, B. K., NAYAK, C., AND TROYER, M. Solving strongly correlated electron models on a quantum computer. *Phys. Rev. A 92*, 6 (2015), 062318.

[243] WEI, T.-C., MOSCA, M., AND NAYAK, A. Interacting boson problems can be QMA hard. *Phys. Rev. Lett. 104*, 4 (2010), 040501.

[244] WEINBERGER, H. F. Upper and lower bounds for eigenvalues by finite difference methods. *Communications on Pure and Applied Mathematics 9*, 3 (1956), 613–623.

[245] WEINBERGER, H. F. Lower bounds for higher eigenvalues by finite difference methods. *Pacific J. Math 8*, 2 (1958), 339–368.

[246] WHITFIELD, J. D., BIAMONTE, J., AND ASPURU-GUZIK, A. Simulation of electronic structure Hamiltonians using quantum computers. *Molecular Physics 109*, 5 (2011), 735–750.

[247] WICKERHAUSER, M. V. *Adapted Wavelet Analysis from Theory to Software*. A.K. Peters, Wellesley, MA, 1994.

[248] WIEBE, N., BERRY, D., HØYER, P., AND SANDERS, B. C. Higher order decompositions of ordered operator exponentials. *Journal of Physics A: Mathematical and Theoretical 43*, 6 (2010), 065203.

[249] WIEBE, N., BERRY, D. W., HØYER, P., AND SANDERS, B. C. Simulating quantum dynamics on a quantum computer. *Journal of Physics A: Mathematical and Theoretical 44*, 44 (2011), 445308.

[250] WIEBE, N., AND ROETTELER, M. Quantum arithmetic and numerical analysis using repeat-until-success circuits. *arXiv preprint arXiv:1406.2040* (2014).

[251] WILLIAMSON, D. P., AND SHMOYS, D. B. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.

[252] WOCJAN, P., AND ZHANG, S. Several natural BQP-complete problems. *arXiv preprint quant-ph/0606179* (2006).

[253] WOIT, P. *Quantum Theory, Groups and Representations: An Introduction*. Springer, 2017.

[254] YANG, Z.-C., RAHMANI, A., SHABANI, A., NEVEN, H., AND CHAMON, C. Optimizing variational quantum algorithms using Pontryagin's minimum principle. *arXiv preprint arXiv:1607.06473* (2016).

[255] YANNAKAKIS, M. Node-and edge-deletion NP–complete problems. In *Proc. 10th ACM Symposium on Theory of Computing* (1978), ACM, pp. 253–264.

[256] YAO, A. C.-C. Quantum circuit complexity. In *Proc. 34th IEEE Symposium on Foundations of Computer Science* (1993), IEEE, pp. 352–361.

[257] ZALKA, C. Efficient simulation of quantum systems by quantum computers. *Fortschritte der Physik 46*, 6-8 (1998), 877–879.

[258] ZALKA, C. Simulating quantum systems on a quantum computer. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences 454*, 1969 (1998), 313–322.

[259] ZUCKERMAN, D. On unapproximable versions of NP–complete problems. *SIAM Journal on Computing 25*, 6 (1996), 1293–1304.

[260] ZUCKERMAN, D. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proc. 38th ACM Symposium on Theory of Computing* (2006), ACM, pp. 681–690.

Appendix A

A Brief Overview of Quantum Computation

Quantum computation lives at the intersection of quantum physics, computer science, mathematics, and engineering. In this appendix we attempt to give a very brief overview of the key concepts, in particular the most relevant for the results of this thesis, and to explain the notation we will use throughout. Many excellent comprehensive introductions to quantum computation exist, notably [152, 173, 195, 199]. Similarly, see [107, 110, 204, 208, 253] for detailed overviews of quantum mechanics and its mathematical structure.

The idea of using quantum mechanics for computation is generally attributed to Feynman [95] who suggested that *universal quantum simulators* may provide computational advantages for tasks which appear to require exponential resources classically, such as simulating quantum systems themselves. Foundational work by Deutsch [79, 80] and subsequently Yao [256] developed computational models based on quantum counterparts to classical Turing machines and Boolean circuits. Interest in quantum computation grew dramatically in 1994 when Shor [209] showed a quantum algorithm for prime factoring (decomposing a number N into its prime factors) requiring only poly$(\log N)$ quantum operations, an exponential speedup over all known classical algorithms. The physical realization of such an algorithm could be used to break certain public-key cryptography systems such as RSA, whose security is based on the assumption that prime factoring is computationally intractible. A second paramount result was the algorithm of Grover [108], which showed that an N element unstructured search problem could be solved a quantum computer with only $O(\sqrt{N})$ queries to the list, whereas any classical algorithm requires $\Omega(N)$ queries.

Subsequently, quantum algorithms have been developed for a wide variety of discrete and continuous problems. In some cases, significant speedups have been shown, yet it remains open whether or not quantum devices are truly more powerful than classical computers. We do not attempt a summary of applications here; see for example [64, 154, 170, 173, 185] for overviews of quantum algorithms.

A.1 Quantum Mechanics of Quantum Computation

The fundamental equation of quantum mechanics, the *Schrödinger equation*

$$\frac{d}{dt}\psi(t) = -iH\,\psi(t), \tag{A.1}$$

says that quantum systems evolve *unitarily* in time, governed by the *Hamiltonian operator* H which encodes the system energy levels.[1] States $\psi(t)$ are *complex* vectors which encode the probability distributions of possible measurement outcomes on the given physical system. Thus, as we shall outline, quantum states evolve in a way fundamentally different from classical probabilistic (stochastic) processes. Hence, the foundational question of quantum computing is whether or not the "strange" behaviour of quantum systems can be used to give computational advantages over classical computers.

For the purposes of quantum computation, finite dimensional quantum mechanics suffices, which is much simpler and more well-behaved than the fully general theory. Hence, quantum computation reduces to a subset of *matrix mechanics* (i.e., complex linear algebra). We remark that alternative quantum information processing schemes such as continuous-variable quantum computation [163] have been proposed; however, these models are not considered here.

Analogous to a classical discrete bit $x \in \{0,1\}$, a *qubit* is defined to be a two-dimensional quantum system, which is described in general by a vector in a complex Hilbert space $|\psi\rangle \in \mathbb{C}^2/\{\mathbf{0}\}$. In the (orthonormal) *computational basis*, which we label $|0\rangle, |1\rangle$, the general state of a qubit may be written

$$|\psi\rangle = a|0\rangle + b|1\rangle. \qquad \||\psi\rangle\|^2 = |a|^2 + |b|^2 = 1. \tag{A.2}$$

[1] We use standard *natural units* where the *reduced Planck constant* $\hbar = 1$, making (A.1) dimensionless; see e.g. [253].

All quantum states are physically equivalent under multiplication by a complex scalar (i.e., up to *normalization* and *global phase*), so without loss of generality we assume states are normalized with $\||\psi\rangle\| = 1$ (we use $\|\cdot\|$ to denote the Euclidean norm). The standard computational basis is taken to be the eigenvectors of the Pauli Z operator, which is typically experimentally convenient for facilitating bit readout. It acts as

$$Z|0\rangle = |0\rangle \qquad Z|1\rangle = -|1\rangle,$$

so we may write $Z = |0\rangle\langle 0| - |1\rangle\langle 1|$. A prototypical example for a systems of electron spins is the identification of $|0\rangle$ with an electron spin-down state, and $|1\rangle$ with the spin-up state.

If we *measure* $|\psi\rangle$ in the Z basis, we obtain outcome '0' with probability $|a|^2$ or outcome '1' with probability $|b|^2 = 1 - |a|^2$, and accordingly the state $|\psi\rangle$ *collapses* to $|0\rangle$ or $|1\rangle$, respectively,

$$|\psi\rangle \xrightarrow{\text{Measure Z}} \begin{cases} |0\rangle & \text{with probability } |a|^2 \\ |1\rangle & \text{with probability } |b|^2. \end{cases}$$

Thus prior to measurement $|\psi\rangle$ gives a probability distribution over the two possible outcomes, whereas immediately after measurement the state is determined, given by the basis vector corresponding to the observed measurement outcome.

Indeed, every possible qubit basis corresponds to a *measurement observable*, which is a self-adjoint matrix M constructed so that the two basis vectors are its non-degenerate eigenvectors. Measurement of the observable M on a qubit probabilistically returns one of the eigenvalues of M, which indicates a '0' or '1' bit value, and collapses the state to the corresponding eigenvector. Therefore, despite the number of qubit states being uncountable, any qubit measurement still only returns one of two possible outcomes $\{0, 1\}$, i.e. a classical bit of information.

It is important to elaborate on our notation. We use the standard *bra-ket* notation, where a *ket* $|j\rangle$ is the vector labeled j, and the *bra* $\langle k|$ is the adjoint linear functional[2] corresponding to the vector labeled k, equivalently represented as the complex-conjugated row vector $\langle k| = |k\rangle^\dagger$. The complex inner product is then compactly represented as the *braket*, $\langle k|j\rangle := (|k\rangle, |j\rangle) = |k\rangle^\dagger \cdot |j\rangle$. For matrices A acting on qubits we write $\langle\phi|A|\psi\rangle := (|\phi\rangle, A|\psi\rangle) = (A^\dagger|\phi\rangle, |\psi\rangle)$. The computational

[2]For finite dimensional Hilbert spaces, we have a natural isomorphism between $\mathcal{H}$ and its *dual space* $\mathcal{H}^* \simeq \mathcal{H}$, so we can naturally identify a *bra* $\langle\psi| \in \mathcal{H}^*$ with the map $(|\psi\rangle, \cdot) : \mathcal{H} \to \mathbb{C}$. See for example [97, 110].

basis orthonormality conditions may be written $\langle 0|0\rangle = 1 = \langle 1|1\rangle$ and $\langle 0|1\rangle = 0$, and a general (normalized) state $|\psi\rangle$ then satisfies $\langle\psi|\psi\rangle = \|\psi\|^2 = 1$.

The *quantum computational state* of n *qubits* is a vector $|\psi\rangle$ in the tensor product Hilbert space $(\mathbb{C}^2)^{\otimes n} \simeq \mathbb{C}^{2^n}$, with $2^n - 1$ complex degrees of freedom (states remain unique up to normalization and overall phase). Thus, a general quantum state is described by a number of coordinates exponential in the number of qubits. The natural n-qubit computational basis is given by the tensor products of the single qubit basis states, which we write as, e.g., $|0\rangle \otimes |0\rangle \otimes |1\rangle \otimes |0\rangle = |0010\rangle$.

A primary difference with classical mechanics is that arbitrary linear combinations of quantum states also give states, known as the *quantum superposition* principle. Furthermore, *quantum entanglement* is the property that an arbitrary state $|\psi\rangle \in \mathbb{C}^{2^n}$ cannot be factored as $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle$, with each $|\psi_i\rangle \in \mathbb{C}^2$; if this was possible generally then every quantum state would have an efficient classical description. We remark that both of these nonclassical properties follow directly from the tensor product structure of the underlying state space.

For n-qubits, it is useful to identify the computational basis vectors $|00\ldots0\rangle, |00\ldots1\rangle, \ldots |11\ldots1\rangle$ with the unsigned integers $|0\rangle, |1\rangle, \ldots, |2^n-1\rangle$. (In Chapter 2 we similarly consider more general signed numbers with fractional parts). Thus a general n qubit state may be written

$$|\psi\rangle = \sum_{j=0}^{2^n-1} c_j |j\rangle,$$

where we normalize $\|\psi\|^2 = \sum_j |c_j|^2 = 1$. Measurement of such a state in the computational basis gives outcome j with probability $|c_j|^2$. Thus, quantum states encode the probability of measurement outcomes in the *probability amplitudes* c_j. We may write $|c_j|^2 = \langle\psi|P_j|\psi\rangle$, where $P_j = |j\rangle\langle j|$ is the projector onto a computational basis state $|j\rangle$.

More generally, a measurement observable M is a self-adjoint operator acting on one or more qubits. Measuring M on $|\psi\rangle$ returns an single eigenvalue λ with probability $\langle\psi|P_\lambda|\psi\rangle$, where P_λ is the projector onto the λ-eigenspace of M. For example, measuring the observable Z_1Z_2 returns 1 for states where the first two bits are equal and -1 for states otherwise; this measurement does not reveal the particular value of the first or second or remaining qubits.

Turning finally to dynamics, for quantum computation, Hamiltonians H are given by $2^n \times 2^n$ self-adjoint (Hermitian) matrices. When H is time-independent, equation (A.1) is solved by

$$|\psi(t)\rangle = e^{-iHt}|\psi(0)\rangle, \tag{A.3}$$

where the unitary operator $U = \exp(-iHt)$ preserves state normalization. We say such a quantum system $|\psi\rangle$ *evolves* under the Hamiltonian H for time t, or equivalently is evolved under the Hamiltonian Ht. For qubits, any unitary transformation may be written as $\exp(-iH)$ for some Hamiltonian H. (A measurement, on the other hand, is a non-unitary transformation.)

Pauli Matrices

A matrix A that is both self-adjoint ($A^\dagger = A$) and unitary ($A^\dagger A = I$) is a square root of the identity, $A^2 = I$. The exponential of such a matrix is given by $e^{-i\gamma A} = \cos(\gamma)I - i\sin(\gamma)A$. A particularly useful set of such matrices are the matrices

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \tag{A.4}$$

and their tensor products, which we collectively call the *Pauli matrices*. The defining relation of the matrices X, YZ is $[X, Y] = 2iZ$, and its cyclic permutations, where the *matrix commutator* is defined $[A, B] := AB - BA$. We write X_j to indicate the matrix X acting on the jth qubit, and do not write identity factors explicitly, e.g. $I \otimes Z \otimes Y \otimes I \cdots \otimes I$ is written as $Z_2 Y_3$.

The Pauli matrices give a basis for the vector space of n-qubit Hamiltonians. Hence, any Hamiltonian H may be expanded as a linear sum

$$H = a_0 I + \sum_{j=1}^{n} \sum_{\sigma=X,Y,Z} a_{j\sigma}\sigma_j + \sum_{j\neq k} \sum_{\sigma=X,Y,Z} \sum_{\lambda=X,Y,Z} a_{jk\sigma\lambda}\sigma_j\lambda_k + \dots, \tag{A.5}$$

with $a_\alpha \in \mathbb{R}$. Thus, an arbitrary Hamiltonian is specified by 4^n real coefficients. Moreover, up to pesky factors of i, Hamiltonians give a representation of the Lie algebra $iu(2^n)$, i.e., they are closed under (i times) the commutator. Thus, exponentials e^{-iH}, and in particular evolution under sums of Pauli operators, gives the full group $U(2^n)$ of unitary transformations on qubits.

We remark that in the computational basis, the operator X acts as $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$. Hence we identify X as the *bit-flip* operator, i.e. the NOT operation. The eigenstates of X are denoted $|+\rangle$ and $|-\rangle$, with $|\pm\rangle = (|0\rangle \pm |1\rangle)/\sqrt{2}$, and related to the computational basis by the *Hadamard gate*

$$\mathrm{H} = \frac{1}{\sqrt{2}}(X + Z) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{A.6}$$

which transforms between qubit bases $\mathrm{H}|0\rangle = |+\rangle$, $\mathrm{H}|1\rangle = |-\rangle$, and is self-inverse $\mathrm{H}^2 = I$.

A.2 Quantum Computational Model

In this thesis we consider the standard *quantum circuit model*. We emphasize that our computational model is abstract; we are generally not overly concerned with the underlying physics. Many alternative quantum computational models have been proposed, such as the quantum Turing machine [79], and adiabatic [8], measurement-based [51], and topological [237] quantum computation, among others. Typically, these models are equivalent to the quantum circuit model and to each other under efficient computational reductions [173], i.e., they can simulate one another with polynomial overhead. Such models are said to be *universal* for quantum computation, an analogy to the many equivalent models of classical computation. This naturally leads to the *Quantum Church-Turing thesis*, which asserts that *a quantum Turing machine can efficiently simulate any realistic model of computation*.

A *quantum algorithm* is defined to be the product of an ordered sequence of *unitary operations* $U_1, U_2, \ldots, U_\ell$, i.e,. algorithmic steps, which maps an input state $|s\rangle$ to the output state

$$|\psi\rangle = U_\ell \ldots U_2 U_1 |s\rangle. \tag{A.7}$$

For some algorithms $U = U_\ell \ldots U_2 U_1$ the goal is to create a state ε-close to some target state $|\phi\rangle$ in some norm, $\| U|s\rangle - |\phi\rangle \| \leq \varepsilon$, or more generally, to ε-approximate a target operator V, $\|U - V\| \leq \varepsilon$. Example include Hamiltonian simulation, which we study in Chapter 4, or distribution sampling problems. For other algorithms, such as classical decision or function problems, the generation of $|\psi\rangle$ is followed by a measurement which reveals some classical bit string x with some probability p_x. Without loss of generality, all intermediate measurements may be deferred until the end of a given quantum computation [173].

The qubits for a quantum algorithm are partitioned into *quantum registers*. Different registers may be used for storing different parts of the computation such as the input, output, or intermediate results. Many computations can be simplified with additional *ancilla* qubits which are often used as scratchpad for temporary results. Qubits are an important computational resource, and we say a quantum algorithm is *space efficient* if it requires a number of qubits that is bounded by a polynomial function of its input size.

Like classical algorithms, a quantum algorithm must be compiled down to a sequence of physically implementable basic operations. Each operation U_j may be further decomposed as $U_j =$

$U_{j\ell_j} \dots U_{j2}U_{j1}$, where the unitaries U_{jk} are each from a set of primitive unitaries which act locally on a small number of qubits, called *quantum gates*. From unitarity, each gate has fan-out equal to fan-in. Consecutive gates which act on disjoint sets of qubits are commuting operators, and may be applied simultaneously, i.e., in parallel. We may draw 'wires', i.e. edge-disjoint paths in a directed acyclic graph, which indicate for each qubit the operations applied to it at each step of the computation. We call this representation a *quantum circuit*; see Chapter 2 for many example circuits. A gate set is said to be *universal* for quantum computation if any arbitrary n-qubit unitary operation can be approximated to any desired accuracy using gates from the set [173]. A fundamental result in the quantum gate model is that universal (finite) sets of one-qubit and two-qubit gates exist; these correspond to operations believed experimentally implementable.

The matrices X, Y, Z given in (A.4) are important examples of single-qubit quantum gates, as are their exponentials the *rotation gates*

$$R_Z(\gamma) = e^{-\frac{i\gamma}{2}Z} = \begin{pmatrix} e^{-i\gamma/2} & 0 \\ 0 & e^{i\gamma/2} \end{pmatrix}, \quad R_X(\beta) = e^{-\frac{i\beta}{2}X} = \begin{pmatrix} \cos(\beta/2) & -i\sin(\beta/2) \\ -i\sin(\beta/2) & \cos(\beta/2) \end{pmatrix},$$

and $R_Y(\gamma) = e^{-i\gamma Y/2}$. These satisfy the useful identity $e^{-i\gamma\Sigma/2} = \cos(\gamma/2)I - i\sin(\gamma/2)\Sigma$, where $\Sigma = I, X, Y, Z$. Furthermore, up to global phase, every single-qubit unitary transformation U can be written $U = R_Z(\alpha)R_Y(\beta)R_Z(\gamma)$ for some $\alpha, \beta, \gamma \in \mathbb{R}$. Other important single-qubit quantum gates include the Hadamard gate H given in (A.6), the T gate $T = R_Z(\pi/4)$, and the S gate $S = R_Z(\pi/8)$; see [173, Ch. 4] for details.

For a gate set to be universal a multiqubit *entangling gate* is required. An important such gate in the controlled-NOT (equivalently controlled-X or *controlled bit-flip*) gate $CNOT = \Lambda_1(X_2) := \frac{1}{2}X_2 - \frac{1}{2}Z_1X_2$, which is drawn as indicated in Fig. 6.1. On computational basis states $\Lambda_1(X_2)$ flips the second *target* bit only if the first *control* qubit is 1, i.e., takes $|a\rangle|b\rangle \to |a\rangle|b \oplus a\rangle$ for $a, b \in \{0, 1\}$. Two important examples of universal gate sets are CNOT with arbitrary single qubit gates, and the particular set $\mathcal{G} = \{H, T, CNOT\}$. A fundamental result is the *Solovay-Kitaev theorem*, which implies that any quantum circuit consisting of ℓ CNOT and single qubit gates can be approximated to within accuracy ε using $\ell \cdot \text{poly}\log(\ell/\varepsilon)$ gates from $\mathcal{G}$ [173]. Hence, without loss of generality we will consider circuits drawn from either gate set, as gate counts will be the same up to polylogarithmic factors; we call such gates *basic*.

For computation, operators controlled by the values of particular qubits are especially useful.

Generalizing the CNOT gate, the *controlled unitary* $\Lambda_1(U_2)$ applies the operator U to the second qubit on basis states where the first bit is 1, and otherwise acts as the identity. We can implement any $\Lambda_1(U_2)$ using $O(1)$ basic gates. A related family of gates, which are particularly useful for intermediate representations of multiqubit gates, are the *multi-controlled Toffoli gates* $\Lambda_\ell(X)$, $\ell \in \mathbb{N}$. These gates act on $(\ell+1)$-qubit basis states as $\Lambda_\ell(X)|x\rangle|a\rangle = |x\rangle\left|a \oplus \wedge_{j=1}^{\ell} x_j\right\rangle$. Recall, e.g., the circuits of Figures 2.4 and 6.1. The $\ell = 1$ gate $\Lambda_1(X)$ is the CNOT gate. For $\ell = 2$, we have the controlled-controlled not gate $\Lambda_2(X)$, known as the Toffoli gate. The Toffoli gate is universal for classical reversible computation, and naturally important for quantum computation. A constant number of basic gates suffices to implement a Toffoli gate [173]. More generally, a $\Lambda_\ell(X)$ gate can be implemented with $O(\ell)$ basic gates, and use of a single temporary ancilla qubit [230]; different compilations are possible. We can further combine Toffoli and $\Lambda_1(U)$ gates to create multi-controlled unitaries $\Lambda_\ell(U)$; see e.g. [230] for details.

For a fixed gate set, the cost of a quantum algorithm U is the minimal number of gates it can be decomposed (compiled) into, or alternatively the depth of such a decomposition, in addition to the number of qubits required. There typically exist time-space trade-offs in the cost, generalizing those for classical reversible circuits. For example, in some proposed architectures, certain gates may be much more 'expensive' than others; this could result from a lower-level quantum error correcting code used to encode the logical qubits, meaning that each quantum gate must typically be further compiled to even lower-level operations. Quantum error correction is a rich topic we do not explore here; see e.g. [159, 173, 195]. We remark that in some cases the cost of a quantum algorithm is taken to be the number of higher-level operations, which provides an indication of cost independent from a specific gate set. For example, in our algorithms of Chapter 2, the cost is taken to be the number of required addition and multiplication operations. A variety of schemes exist in the literature for these basic arithmetic operations, with different trade-offs themselves. Furthermore, in some applications certain operators U_j implement *oracle calls*, i.e., access to an unknown black-box operation, in which case the appropriate cost metric is the separate numbers of oracle and non-oracle operations.

We say a quantum algorithm for a given problem is *(time) efficient* if its cost is polynomially bounded with respect to the problem input size n. More precisely, the quantum circuit for each problem size must be *uniformly* generated; there must exist a classical deterministic Turing machine which given the input string $1^n = 11\dots1$ generates the quantum circuit description in polynomial time. As is this case with classical circuits, non-uniform quantum circuits appear to be artificially powerful [14].

For a decision problem, a quantum algorithm seeks to output a single bit with probability $p > 1/2 + c$ for some constant c. If this is achieved, then the problem can be solved by repeating the algorithm and taking the majority vote. For other problems, the output will be a bit string x, which solves the problem with some probability p. Using amplitude amplification [48], the success probability can be boosted close to unity with $O(1/\sqrt{p})$ repetitions, a quadratic improvement over the classical case. Thus, if p is only polynomially small, i.e., $1/p$ is bounded by a polynomial in n, then a polynomial number of repetitions suffice.

We may define complexity classes for quantum computation analogous to those for classical computation. The class of decision problems efficiently decidable by such a procedure with success probability at least $1/2$ plus a constant is known as BQP, and naturally contains its classical probabilistic analog BPP. Similarly, the class QMA gives the quantum analog of the classical complexity class MA, which may further be seen as the probabilistic analog of the class NP. Quantum complexity classes also have complete problems; as is the case for NP, a variety of QMA-complete problems have been discovered; see [4, 44] for surveys. Moreover, complete problems are also know for the complexity class BQP; see [252]. (In contrast, BPP is not believed to have complete problems unless P=BPP.) Furthermore, natural quantum extensions of *qubit (space), query, circuit* and *communication* complexity can be defined. Indeed, quantum complexity theory is a rich area which we do not explore further here; see e.g. [28, 239] for details.

Finally, we emphasize that quantum computation subsumes classical computation. As classical computation is no more powerful than classical reversible computation, with polynomial overhead any classical circuit can be converted into a reversible circuit and subsequently efficiently simulated by a quantum circuit [27, 173].

Appendix B

Design Toolkit for Quantum Optimization

In this appendix we motivate, derive, and extend the results of Section 6.3, the design toolkit for quantum optimization. The goal is to provide a suite of basic results which can be used by experts and laymen alike to design quantum algorithms. These results can also be found in [111]. We emphasize that our results are general and have applications beyond QAOA or quantum annealing; see [111] for several examples.

B.1 Representing n-bit Functions as Diagonal Hamiltonians

Many important problems involve Boolean predicates. We show how the representation of such functions as Hamiltonians follows naturally from the Fourier analysis of Boolean functions.

Fourier analysis of Boolean functions has many applications in computer science and related fields such as operations research [26,120,143,161,174], and is also useful a useful tool for quantum computation [24,171]; see [76,175] for comprehensive introductions to the subject. Many important combinatoric properties of a given function can be "read off" from its Fourier coefficients [175]. However, this presents an obstruction to computing the Fourier representation of a general n-bit Boolean functions, or equivalently, to computing its Hamiltonian representation explicitly. Indeed, Proposition 7 shows that computing the first Fourier coefficient $\widehat{f}(\emptyset)$ is as hard as counting the the number of inputs $x \in \{0,1\}^n$ such that $f(x) = 1$. For example, if f is an instance of CNF-SAT, then this is $\#P$-hard; see Corollary 5. Hence, for arbitrary Boolean functions, in particular, functions corresponding to instances of NP-hard decision problems, we cannot hope to efficiently obtain their explicit Hamiltonian representations (in the form given in Proposition 7).

Nevertheless, there is no such difficulty for Boolean functions f_j when f_j acts on a constant number of bits. Hence we can efficiently construct Hamiltonians representing *pseudo-Boolean* functions of the form $f(x) = \sum_{j=1}^{m} f_j(x)$, $m = \text{poly}(n)$, which we typically seek to minimize or maximize. For such a pseudo-Boolean function, its Fourier coefficients do not explicitly reveal its optimal value, so the Hamiltonian representation can often be computed efficiently; see Theorem 11 below. For example, solving the optimization problem MaxSAT also solves the decision problem SAT; for m clauses, a string can be found optimally satisfying all m clauses if and only if the conjunction of the clauses is satisfiable. Hence, one approach to solving SAT is to instead try to solve MaxSAT. If the clauses each contain at most $k = O(1)$ variables, and there are poly(n) many clauses, then we can efficiently represent the MaxSat instance as a Hamiltonian (in the sense of Theorem 10 below). This avoids the described difficulty, and is a commonly used approach in quantum annealing to implicitly encode Boolean functions; see, e.g., [165].

Boolean functions are often encountered as a formula in a *normal form*. For example, SAT formulas are given in conjunctive normal form (CNF). Many normal forms exist such as disjunctive (maxterms), algebraic ($\oplus$), etc. [103]. Note that while logically equivalent, the different forms are often (very much) inequivalent for computational purposes. For each form there corresponds a notion of size (which directly relates to the number of bits needed to describe a function in such a form). We give explicit Hamiltonian representations of basic clauses, and composition rules which can be used to construct Hamiltonians for most normal forms. This typically allows for easier construction than by working with the Fourier expansion directly. Our results may also be applied to other common representations such as Boolean circuits.

B.1.1 Boolean Functions

The class of Boolean functions on n-bits is defined as $\mathcal{B}_n := \{f : \{0,1\}^n \to \{0,1\}\}$. As a vector spaces (over $\mathbb{R}$), for each n they give a give a basis for the real functions $\mathcal{R}_n = \{f : \{0,1\}^n \to \mathbb{R}\}$. Moreover, each $\mathcal{R}_n$ is isomorphic to the vector space of diagonal Hamiltonians acting on n-qubits, or, equivalently, the space of $2^n \times 2^n$ diagonal real matrices. Thus, diagonal Hamiltonians naturally encode large classes of functions.

We say a Hamiltonian *represents* a function f if in the computational basis it acts as the corresponding multiplication operator, i.e. it satisfies the eigenvalue equations

$$\forall x \in \{0,1\}^n \quad H_f|x\rangle = f(x)|x\rangle. \tag{B.1}$$

On n qubits, this condition specifies H_f uniquely. Equivalently, we may write $H_f = \sum_x f(x)|x\rangle\langle x|$, which in the case of Boolean functions becomes

$$H_f = \sum_{x:f(x)=1} |x\rangle\langle x|. \tag{B.2}$$

As Boolean functions are idempotent, both $f^2 = f$ and $H_f^2 = H_f$, so H_f is a projector[1] of rank $r = \#f := |\{x : f(x) = 1\}| = \sum_x f(x)$. Hence, determining if f is satisfiable is equivalent to determining if H_f is not identically 0, and determining H_f explicitly in this form is as hard as counting the number of satisfying assignments.

Such a representation is unique (up to change of computational basis). Recall that without loss of generality we consider the standard computational basis of eigenstates of Pauli Z operators, defined by the relations $Z|0\rangle = |0\rangle$ and $Z|1\rangle = -|1\rangle$. Recall $Z_j = I \otimes \dots I \otimes Z \otimes I \cdots \otimes I$ denotes Z acting on the jth qubit. Products of Z_j over a set of qubits act as

$$\prod_{j\in S} Z_j|x\rangle = \chi_S(x)|x\rangle, \tag{B.3}$$

where the *parity function* $\chi_S(x) : \{0,1\}^n \to \{-1,+1\}$ gives the parity of the bits of x in the subset $S \subset [n]$. Identifying each S with its characteristic vector $S \in \{0,1\}^n$ we have $\chi_S(x) = (-1)^{S\cdot x}$. Thus $Z_S := \prod_{j\in S} Z_j$ represents the function $\chi_S(x)$ in the sense of (B.1).

The set of parity functions on n-bits also give a basis for the real functions $\mathcal{R}_n$. This basis is orthonormal with respect to the inner product

$$\langle f, g\rangle := \frac{1}{2^n} \sum_{x\in\{0,1\}^n} f(x)g(x). \tag{B.4}$$

[1] In typical constructions [175], Boolean functions $g \to \{-1,1\}$ are considered, in which case $H_g^2 = I$. The analog of Proposition 7 yields $\sum \widehat{g}(S)^2 = 1$, which does not depend on the structure of g. In this case, multiplication corresponds to the bitwise parity operation, whereas for $f \to \{0,1\}$ it corresponds to bitwise AND.

In particular, every Boolean function $f \in \mathcal{B}_n$ may be written

$$f(x) = \sum_{S \subset [n]} \widehat{f}(S) \chi_S(x), \tag{B.5}$$

called the *Fourier expansion*, with *Fourier coefficients* given by the inner products

$$\widehat{f}(S) = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x) \chi_S(x) = \langle f, \chi_S \rangle. \tag{B.6}$$

The *degree* of f, $\deg(f)$, is defined to be largest $|S|$ such that $\widehat{f}(S)$ is non-zero. Note that if f depends on only $k \leq n$ variables, then $\deg(f) \leq k$. We refer to the mapping from $f(x)$ to $\widehat{f}(S)$ as the *Fourier transform* of f.

Hence, an arbitrary Boolean function f is represented as a Hamiltonian H_f given by a linear combination of tensor products of Z_j operators using the Fourier expansion and the identification $\chi_S = \bigotimes_{j \in S} Z_j$. We define the degree of such a Hamiltonian H_f, $\deg(H_f)$, to be the largest number of qubits acted on by any term in this sum, and the size of H_f, $\mathrm{size}(H_f)$, to be the number of terms.[2] By definition, $\deg(H_f) = \deg(f)$. Applying Parseval's identity and some further Fourier analysis gives the following theorem, which generalizes Propositon 7.

Theorem 10. *For an n-bit Boolean function $f \in \mathcal{B}_n$ of degree $d = \deg(f)$, the unique n-qubit Hamiltonian satisfying $H_f|x\rangle = f(x)|x\rangle$ in the computational basis is*

$$\begin{aligned} H_f &= \sum_{S \subset [n], |S| \leq d} \widehat{f}(S) \prod_{j \in S} Z_j \\ &= \widehat{f}(\emptyset) I + \sum_{j=1}^{n} \widehat{f}(\{j\}) \, Z_j + \cdots + \sum_{j_1 < j_2 < \cdots < j_d} \widehat{f}(\{j_1, j_2, \ldots, j_d\}) \, Z_{j_1} \ldots Z_{j_d} \end{aligned}$$

where the Fourier coefficient $\widehat{f}(S) = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x)(-1)^{S \cdot x}$, with $\deg(H_f) = d$ and $\mathrm{size}(H_f) \leq 1 + (e/d)^{d-1} n^d$. The coefficients satisfy

$$0 \leq \sum_{S \subset [n]} \widehat{f}(S)^2 = \widehat{f}(\emptyset) = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x) = \frac{1}{2^n} \mathrm{tr}(H_f) \ \leq 1 \tag{B.7}$$

and in particular $\sum_{S \subset [n]} \widehat{f}(S) = f(0^n)$, where 0^n denotes the input string of all 0s.

[2]For Boolean functions some authors [45] define size as the sum of $|S|$ over all subsets such that $\widehat{f} \neq 0$, which is larger than our size quantity by at most a multiplicative factor of $\deg(f)$. Our notion of size is often called *sparsity*.

Here, $\mathrm{tr}(H_f)$ is the *trace* of the matrix H_f, i.e., the sum of its diagonal elements, which in particular is a basis-independent linear function [133]. The bound $\mathrm{size}(H_f) \leq 1 + (e/d)^{d-1} n^d$ follows from counting arguments and standard bounds for binomial coefficients. If H_f acts on a constant number $d = O(1)$ of qubits, then its size is polynomially bounded in the number of qubits, $\mathrm{size}(H_f) = O(n^d)$. We say that a function f_n on n-bits is *efficiently representable* as the Hamiltonian H_{f_n} if $\mathrm{size}(H_{f_n}) = \mathrm{poly}(n)$ with increasing n.

Remark 17. *The Hamiltonian coefficients $\widehat{f}(S)$ depend only on the function values $f(x)$, and are independent of how such a function may be represented as input (e.g. formula, circuit, truth table, etc.). Many typical representations of Boolean functions as computational input such as Boolean formulas (e.g. CNF, DNF, etc.) or Boolean circuits can be directly transformed to Hamiltonians using composition rules we derive below; see Theorem 9.*

Consider a Boolean function f given as a formula in *conjunctive normal form*, the AND of clauses containing ORs of variables and their negations. The satisfiability problem (SAT) is to decide if there exists a satisfying assignment for f. It is NP-hard to decide this for an arbitrary such function, even if clauses are restricted to at most 3 literals (3-SAT). Theorem 10 implies that computing the single Fourier coefficient $\widehat{f}(\emptyset)$ is equivalent to computing the number of satisfying assignments, which is believed to be a much harder problem. (In fact, this problem #SAT is complete for the counting complexity class #P; see, e.g., [14].) Thus, the problem of deciding if $\widehat{f}(\emptyset) > 0$ is NP-hard. This result is stated as Corollary 5 above. Moreover, arbitrary Boolean functions may have size exponential in n, in which case, even if we know somehow its Hamiltonian representation, we cannot implement or simulate this Hamiltonian efficiently with the usual approaches.

As remarked, pseudo-Boolean functions, in particular, objective functions for constraint satisfaction problems, often avoid these difficulties. We are particularly interested in functions composed of a number $m = \mathrm{poly}(n)$ of clauses C_j, where each clause acts on $k = O(1)$ bits, and hence has size $O(1)$ and degree $O(1)$ (e.g., Max-k-Sat). In such cases, we have a useful lemma, which follows directly from [174, Thm. 1 & 2].

Lemma 3. *For a function $f \in \mathcal{B}_n$ that depends on $k \leq n$ variables, represented as a Hamiltonian H_f acting on n qubits its degree satisfies*

$$k \geq D(f) \geq \deg(H_f) \geq \log_2 k - O(\log\log k)$$

where $D(f)$ is the decision tree complexity of f and $D(f) = \mathrm{poly}(\deg(H_f))$.

B.1.1.1 Composition Rules

It is often possible to construct a Hamiltonian representing a Boolean function much more efficiently than by evaluating the Fourier coefficients explicitly. In general, this depends on the *input format* of the given function. For example, for a function given as a disjunction of clauses, the Hamiltonian can be constructed by computing the Hamiltonian for each clause separately and combining them using composition rules. These composition rules follow directly from the properties of the Fourier transform. Results for several important basic operations are given in Theorem 9.

Proof of Thm. 9. The logical values 1 and 0 (i.e., *true* and *false*) are represented as the identity matrix I and the zero matrix 0, respectively. Each result follows from the natural embedding of $f, g \in \mathcal{B}_n$ into $\mathcal{R}_n(+,\cdot)$, the real vector space of real functions on n bits. By linearity of the Fourier transform, we immediately have $H_{af+bg} = aH_f + bH_g$ for $a, b \in \mathbb{R}$. Using standard logical identities, Boolean operations $(\cdot, \vee, \oplus, \dots)$ on f, g can be translated into $(\cdot, +)$ formulas, i.e., linear combinations of f and g. Linearity then gives the resulting Hamiltonian in terms of H_f and H_g.

Explicitly, for the complement of a function $\overline{f}$, as $\overline{f} = 1 - f$ we have $H_{\overline{f}} = I - H_f$. Similarly, the identities $f \wedge g = fg$, $f \vee g = f + g - fg$, $f \oplus g = f + g - 2fg$, and $f \Rightarrow g = \overline{f} + fg$, respectively, imply the remainder of the theorem. □

It is straightforward to extend Theorem 9 to other operations, on any number of Boolean functions, using the same technique of the proof.

Remark 18. *The Hamiltonians representing basic clauses given in Table 6.2 follow directly from the Fourier coefficients as in Theorem 10, or equivalently, from the composition rules in Theorem 9. The latter approach is generally much easier for constructing Hamiltonians.*

The rules of Theorem 9 may be applied *recursively*, as desired, to construct Hamiltonians representing more complicated Boolean functions, corresponding to, e.g., nested parentheses in logical formulas or wires in Boolean circuits. For example, the Hamiltonian representing the Boolean clause $f \vee g \vee h = f \vee (g \vee h)$ is given by $H_{f \vee g \vee h} = H_f + H_{g \vee h} - H_f H_{g \vee h}$, which simplifies to $H_{f \vee g \vee h} = H_f + H_g + H_h - H_f H_g - H_f H_h - H_g H_h + H_f H_g H_h$.

Some typical examples of Boolean functions on three variables are the Majority (MAJ), Not-All-Equal (NAE), and 1-in-3 functions, which are represented as the Hamiltonians

- $H_{MAJ(x_1,x_2,x_3)} \;=\; \frac{1}{2}I - \frac{1}{4}(Z_1 + Z_2 + Z_3 - Z_1Z_2Z_3)$
- $H_{NAE(x_1,x_2,x_3)} \;=\; \frac{3}{4}I - \frac{1}{4}(Z_1Z_2 + Z_1Z_3 + Z_2Z_3)$
- $H_{1in3(x_1,x_2,x_3)} \;=\; \frac{1}{8}(3I + Z_1 + Z_2 + Z_3 - Z_1Z_2 - Z_2Z_3 - Z_1Z_3 - 3Z_1Z_2Z_3)$.

The higher-order functions $\bigvee_{j=1}^{k} x_j$ and $\bigwedge_{j=1}^{k} x_j$ are represented by Hamiltonians of size 2^k. This is analogous to the well-known fact that formulas in conjunctive or disjunctive normal form that compute the parity function on k bits have sizes exponential in k.

Finally, observe that for each of the rules of Theorem 9 we may define the right-hand sides as binary operators on Hamiltonians, e.g., $OR(H_f, H_g) := H_f + H_g - H_fH_g$. These rules clearly give homomorphisms (in fact, isomorphisms) between Boolean algebra($\wedge, \vee$) and Boolean ring($\cdot, \oplus$) elements. Thus, $2^n \times 2^n$ diagonal Hamiltonians equipped with these operators faithfully represent the n-element Boolean algebra and Boolean ring; see, e.g., [103].

B.1.2 Pseudo-Boolean Functions and Constraint Satisfaction Problems

Real functions on n-bits are similarly represented as Hamiltionans via the Fourier transform. Every such function $f \in \mathcal{R}_n$ may be expanded (non-uniquely) as a weighted sum of Boolean functions, possibly of exponential size. By linearity of the Fourier transform, the Hamiltonian H_f is given precisely by the corresponding weighted sum of the Hamiltonians representing the Boolean functions. Moreover, the Hamiltonian H_f is unique, so different expansions of f as sums of Boolean functions must all result in the same H_f.

The Fourier coefficients are again given by the inner product (B.4) with the parity functions χ_S,

$$\widehat{f}(S) = \langle f, \chi_S \rangle = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x)\chi_S(x), \tag{B.8}$$

and these coefficients again lead directly to the Hamiltonian representation. We are particularly interested in optimization problems with *objective functions* $f : \{0,1\}^n \to \mathbb{R}$ of the form

$$f(x) = \sum_{j=1}^{m} w_j f_j(x), \tag{B.9}$$

where $f_j \in \mathcal{B}_n$, $w_j \in \mathbb{R}$, and $m = \text{poly}(n)$. (We call such a function $f \in \mathcal{R}_n$ a *pseudo-Boolean* function, generally.) In particular, in a *constraint satisfaction problem*, typically all $w_j = 1$ and hence $f(x)$ gives the number of satisfied constraints (i.e., clauses). We will see many examples of such problems in Chapters 5 and 6.

We have the following theorem which extends the previous results for Boolean functions.

Theorem 11. *An n-bit real function $f : \{0,1\}^n \to \mathbb{R}$ is represented as the Hamiltonian*

$$H_f = \sum_{S \subset [n]} \widehat{f}(S) \prod_{j \in S} Z_j, \qquad \widehat{f}(S) = \langle f, \chi_s \rangle \in \mathbb{R}.$$

In particular, a pseudo-Boolean function $f = \sum_j w_j f_j$, $w_j \in \mathbb{R}$, $f_j \in \mathbb{B}_n$, is represented as

$$H_f = \sum_j w_j H_{f_j},$$

with $\deg(H_f) \leq \max_j \deg(f_j)$ *and* $\text{size}(H_f) \leq \min\{\sum_j \text{size}(H_{f_j}), 1 + (e/d)^{d-1} n^d\}$.

The theorem follows from Theorem 10 and the linearity of the Fourier expansion. Proposition 8 in Section 6.3 then follows directly from Theorems 10 and 11.

Remark 19. *In contrast to Theorem 10, for a constraint satisfaction problem $f = \sum_{j=1}^m f_j$, $f_j \in \mathbb{B}_f$, applying Parseval's identity we have*

$$\sum_{S \subset [n]} \widehat{f}(S)^2 = \mathbf{E}[f] + 2 \sum_{i<j} \langle f_i, f_j \rangle = \widehat{f}(\emptyset) + 2 \sum_{i<j} \mathbf{E}[f_i \wedge f_j] \geq \mathbf{E}[f],$$

where $\mathbf{E}[f] := \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x)$ gives the expected value over the uniform distribution of inputs. In particular, $\sum_{S \subset [n]} \widehat{f}(S)^2 = \mathbf{E}[f]$ if and only if $\langle f_i, f_j \rangle = 0$ for all i, j. If there does exist an i, j such that $\langle f_i, f_j \rangle = 0$, then the conjunction of the clauses is unsatisfiable, i.e. $\wedge_j f_j = 0$.

B.2 Controlled Hamiltonian Evolution

In many applications we require controlled Hamiltonian evolutions. For example, in quantum phase estimation (QPE) [173], we require transformations on $(1+n)$-qubit basis states of the form

$$|0\rangle|x\rangle \to |0\rangle|x\rangle, \qquad |1\rangle|x\rangle \to e^{-iHt}|1\rangle|x\rangle,$$

for various values $t = 1, 2, 4, \ldots$; see the discussion of QPE in Chapter 3. Consider such a transformation with fixed t. Labeling the first qubit a, the overall unitary may be written as

$$\Lambda_{x_a}(e^{-iHt}) = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes e^{-iHt}. \tag{B.10}$$

Here the notation $\Lambda_{x_a}(e^{-iHt})$ indicates the unitary e^{-iHt} controlled by the classical function x_a. We obtain the Hamiltonian corresponding to this transformation by writing $\Lambda_{x_a}(e^{-iHt}) = e^{-i\widetilde{H}t}$, which gives

$$\widetilde{H} = |1\rangle\langle 1| \otimes H = x_a \otimes H = \frac{1}{2} I \otimes H - \frac{1}{2} Z_a \otimes H. \tag{B.11}$$

The control qubit is assumed precomputed here; its value may or may not depend on x. If $H = H_f$ is diagonal, then so is $\widetilde{H} = x_a H_f = H_{x_a \wedge f}$.

More generally, we can consider Hamiltonian evolution controlled by a Boolean function $g \in \mathbb{B}_k$ acting on a k-qubit register. In this case we seek to affect the unitary transformation on $(k+n)$-qubit on basis states

$$|y\rangle|x\rangle \rightarrow |y\rangle|x\rangle \qquad \text{if } g(y) = 0,$$

$$|y\rangle|x\rangle \rightarrow e^{-iHt}|y\rangle|x\rangle \qquad \text{if } g(y) = 1,$$

which gives the overall unitary

$$\Lambda_g(e^{-iHt}) = \sum_{y:g(y)=0} |y\rangle\langle y| \otimes I + \sum_{y:g(y)=1} |y\rangle\langle y| \otimes e^{-iHt} = H_{\overline{g}} \otimes I + H_g \otimes e^{-iHt},$$

corresponding to evolution under the Hamiltonian

$$\widetilde{H} = \sum_{y:g(y)=1} |y\rangle\langle y| \otimes H = H_g \otimes H. \tag{B.12}$$

These results have been summarized in Proposition 9.

As a corollary, we show that computing a Boolean function in a register is closely related to computing it as an amplitude.

Corollary 7. *For an n-bit Boolean function f, let G_f be the unitary operator on $n+1$ qubits which acts on basis states $|x\rangle|a\rangle$ as*

$$G_f|x\rangle|a\rangle = |x\rangle|a \oplus f(x)\rangle. \tag{B.13}$$

Then

$$G_f = H_f \otimes X + H_{\overline{f}} \otimes I,$$

and $G_f = I$ if and only if f is unsatisfiable.

In particular, if f is given as a CNF formula, it is #P-hard to compute the identity coefficient $\widehat{g}(\emptyset)$ of G_f, and NP-hard to decide if $\widehat{g}(\emptyset) \neq 1$.

Proof. Using Theorem 10 for H_f, we expand G_f as a sum of Pauli matrices $G_f = (1 - \widehat{f}(\emptyset))I + \widehat{f}(\emptyset)X_a + \dots$, where none of the terms to the right are proportional to I. Thus computing the I coefficient gives the number of satisfiable assignments for f. □

Observe that with an ancilla qubit a initialized to $|0\rangle$, we can simulate H_f for time t using two applications of G_f and a Z rotation,

$$e^{-iH_f t}|x\rangle|0\rangle = c\, G_f\, R_{Z_a}(-t)\, G_f|x\rangle|0\rangle, \tag{B.14}$$

where c is an unimportant global phase. This *phase kickback* is an important and well-known technique in quantum computation. Conversely, if we can simulate H_f in a controlled manner, then we can implement G_f using a single-bit quantum phase estimation, which requires two Hadamard gates and a controlled simulation of H_f. Indeed, using $e^{-i\pi H_f} = (-1)^{f(x)}$, it is easy to check

$$G_f|x\rangle|0\rangle = \mathrm{H}_a \Lambda_a(e^{-i\pi H_f})\mathrm{H}_a|x\rangle|0\rangle. \tag{B.15}$$

Thus, in this sense the operators H_f and G_f are closely related. We expand on these ideas in [111].

Finally, we remark that if we could simulate arbitrary Hamiltonians H_f efficiently, or more precisely, efficiently simulate H_f for each f in some class of n-bit Boolean functions, and if we could also somehow find and efficiently prepare an initial state $|s\rangle$ with sufficiently large projection $\|H_f|s\rangle\| = 1/\mathrm{poly}(n)$ for each such f, then we could efficiently determine if a given f is satisfiable using QPE. (Smaller state projections proportionally reduce the success probability of QPE, which can be dealt with by an inversely proportional number of algorithm repetitions [173].) Thus, as Hamiltonians representing Boolean formulas given in conjuctive normal form can be efficiently simulated using ancilla qubits (without knowing the Hamiltonian terms explicitly), we conclude that finding such an initial state must be NP-hard. Similar ideas can be applied to real functions.

Appendix C

Quantum Algorithms for Scientific Computing

In this appendix we derive theorems showing the worst-case error of the algorithms in Chapter 2.

The following corollary follows from Theorem B.1 of [56].

Corollary 8. *For $w > 1$, represented by n bits of which the first m correspond to its integer part, and for $b \geq n$, Algorithm 0 returns a value $\hat{x}_s$ approximating $\frac{1}{w}$ with error*

$$|\hat{x} - \frac{1}{w}| \leq \frac{2 + \log_2 b}{2^b}. \tag{C.1}$$

This is accomplished by performing Newton's iteration and truncating the results of each iterative step to b bits after the decimal point.

Proof. From [56], the Newton iteration of Algorithm 0 with b bits of accuracy and s iterations produces an approximation $\hat{x}$ of $\frac{1}{w}$ such that $|\hat{x} - \frac{1}{w}| \leq \left(\frac{1}{2}\right)^{2^s} + s2^{-b}$. For $s = \lceil \log_2 b \rceil$, then

$$|\hat{x} - \frac{1}{w}| \leq \frac{1}{2^b} + \lceil \log_2 b \rceil \frac{1}{2^b} \leq \frac{1}{2^b}(2 + \log_2 b).$$

□

Theorem 12. *For $w > 1$, represented by n bits of which the first m correspond to its integer part, and for $b \geq \max\{2m, 4\}$, Algorithm 1 returns a value $\hat{y}_s$ approximating $\sqrt{w}$ with error*

$$|\hat{y}_s - \sqrt{w}| \leq \left(\frac{3}{4}\right)^{b-2m} (2 + b + \log_2 b). \tag{C.2}$$

This is accomplished by performing Newton's iteration and truncating the results of each iterative step to b bits after the decimal point before passing it on the next iterative step.

Proof. The overall procedure consists of two stages of Newton's iteration, as illustrated in Fig. 2.3 above. We analyze each stage in turn.

Observe that the iteration $x_i = g_1(x_{i-1}) := -w\hat{x}_{i-1}^2 + 2\hat{x}_{i-1}$, $i = 1, 2, \dots, s_1$, $s_1 = \lceil \log_2 b \rceil$, corresponds to Newton's iteration applied to the function $f_1(x) := \frac{1}{x} - w$ for approximating $\frac{1}{w}$, with initial guess $\hat{x}_0 = 2^{-p}$ where $p \in \mathbb{N}$ and $2^p > w \geq 2^{p-1}$. It has been analyzed in detail in Theorem B.1 of [56]. Here, we briefly review some of the results. An efficient circuit for generating the initial state 2^{-p} is shown in Fig. 2.4 above, similar to that in [56].

The approximations x_i each satisfy $x_i \leq \frac{1}{w}$, i.e., we underestimate $\frac{1}{w}$. Indeed, $g_1(x) - \frac{1}{w} = \frac{1}{w}\left(2xw - w^2x^2 - 1\right) = -\frac{1}{w}(wx-1)^2 < 0$. Accounting for the truncation error, we have

$$|\hat{x}_{s_1} - \frac{1}{w}| \leq (we_0)^{2^{s_1}} \frac{1}{w} + 2^{-b}s_1 \leq \left(\frac{1}{2}\right)^{2^{s_1}} + 2^{-b}s_1 =: E. \tag{C.3}$$

This follows from the facts $w > 1$ and $we_0 \leq \frac{1}{2}$, where $e_0 = |2^{-p} - \frac{1}{w}|$, and is shown in [56]. The first term in the upper bound corresponds to the error of Newton's iteration, while the second term is the truncation error. Now we turn to the second stage. Iteration $y_j = g_2(y_{j-1}) := \frac{1}{2}(3y_{j-1} - \hat{x}_{s_1}y_{j-1}^3)$, $j = 1, 2, \dots, s_2$, $s_2 = \lceil \log_2 b \rceil$, is obtained by using Newton's method to approximate the zero of the function $f_2(y) = \frac{1}{y^2} - \mathbf{x}$, with initial guess $\hat{y}_0 = 2^{\lfloor \frac{q-1}{2} \rfloor}$ where $q \in \mathbb{N}$ and $2^{1-q} > \hat{x}_{s_1} \geq 2^{-q}$. An efficient circuit for generating the initial state $2^{\lfloor \frac{q-1}{2} \rfloor}$ is shown in Fig. 2.5. We have

$$\begin{aligned}
g_2(y) - \frac{1}{\sqrt{\mathbf{x}}} &= y - \frac{1}{\sqrt{\mathbf{x}}} + \frac{1}{2}\left(y - y^3\mathbf{x}\right) \\
&= y - \frac{1}{\sqrt{\mathbf{x}}} + \frac{\mathbf{x}y}{2}\left(\frac{1}{\mathbf{x}} - y^2\right) \\
&= \left(y - \frac{1}{\sqrt{\mathbf{x}}}\right)\left(1 - \frac{1}{2}\mathbf{x}y\left(y + \frac{1}{\sqrt{\mathbf{x}}}\right)\right) \\
&= -\frac{1}{2}\left(y - \frac{1}{\sqrt{\mathbf{x}}}\right)\left(y^2\mathbf{x} + y\sqrt{\mathbf{x}} - 2\right) \\
&= -\frac{1}{2}\left(y - \frac{1}{\sqrt{\mathbf{x}}}\right)\left(\left(y\sqrt{\mathbf{x}} - 1\right)^2 + 3y\sqrt{\mathbf{x}} - 3\right) \\
&= -\frac{1}{2}\left(y - \frac{1}{\sqrt{\mathbf{x}}}\right)\left(y - \frac{1}{\sqrt{\mathbf{x}}}\right)\sqrt{\mathbf{x}}\left(y\sqrt{\mathbf{x}} - 1 + \frac{3y\sqrt{\mathbf{x}} - 3}{y\sqrt{\mathbf{x}} - 1}\right) \\
&= -\frac{1}{2}\left(y - \frac{1}{\sqrt{\mathbf{x}}}\right)^2 \sqrt{\mathbf{x}}\left(y\sqrt{\mathbf{x}} + 2\right).
\end{aligned}$$

The last quantity is non-positive assuming $y \geq 0$. The iteration function g_2 is non-decreasing in the interval $[0, \frac{1}{\sqrt{\mathbf{x}}}]$ and $g_2(0) = 0$. Since $y_0 = 2^{-\lfloor \frac{q-1}{2} \rfloor}$, we get $y_1 \geq 0$, and inductively we see that all iterations produce positive numbers that are approximations underestimating $\frac{1}{\sqrt{\mathbf{x}}}$, i.e. $y_i \leq \frac{1}{\sqrt{\mathbf{x}}}$ for $i = 0, 1, 2, \ldots$. Then

$$e_{i+1} := |y_{i+1} - \frac{1}{\sqrt{\mathbf{x}}}| = e_i^2 \frac{1}{2} \sqrt{\mathbf{x}} |y_i \sqrt{\mathbf{x}} + 2| \leq \frac{3}{2} \sqrt{\mathbf{x}} e_i^2, \tag{C.4}$$

since $|y_i \sqrt{\mathbf{x}} + 2| \leq 3$ (y_i underestimates $\hat{x}_{s_1}^{-\frac{1}{2}}$), where $e_0 = \frac{1}{\sqrt{\mathbf{x}}} - 2^{-\lfloor \frac{q-1}{2} \rfloor} \leq 2^{-\lfloor \frac{q-1}{2} \rfloor}$.

Let $A = \frac{3}{2}\sqrt{\mathbf{x}}$. We unfold the recurrence to obtain $e_i \leq \frac{1}{A}(Ae_0)^{2^i}$, $i = 1, 2, \ldots$. We have $\sqrt{\mathbf{x}} 2^{\lfloor \frac{q-1}{2} \rfloor} \geq 2^{-q/2} 2^{\lfloor \frac{q-1}{2} \rfloor}$. For q odd, this quantity is bounded from below by $\frac{1}{\sqrt{2}}$, and for q even this is bounded by $\frac{1}{2}$. Thus $\sqrt{\mathbf{x}} e_0 = |\sqrt{\mathbf{x}} 2^{\lfloor \frac{q-1}{2} \rfloor} - 1| = 1 - \sqrt{\mathbf{x}} 2^{\lfloor \frac{q-1}{2} \rfloor} \leq \frac{1}{2}$ because we have selected the initial approximation y_0 to underestimate $\frac{1}{\sqrt{\mathbf{x}}}$. From this, we obtain $e_i \leq \frac{1}{A}\left(\frac{3}{4}\right)^{2^i}$. Using equation (C.3), we have that $\hat{x}_{s_1} \geq \frac{1}{w} - E$. Without loss of generality, $wE \leq \frac{1}{2}$. This will become apparent in a moment once we select the error parameters. Thus, $\frac{1}{A} \leq \frac{2}{3}\sqrt{2w}$. Therefore,

$$e_i \leq \frac{2}{3}\sqrt{2w}\left(\frac{3}{4}\right)^{2^i}, \quad i = 1, \ldots, s_2. \tag{C.5}$$

We now turn to the roundoff error analysis of the second stage of the algorithm. The iterations of the second stage of the algorithm would produce a sequence of approximations $y_1, \ldots, y_{s_2}$ if we did not have truncation error. Since we truncate the result of each iteration to b bits of accuracy before performing the next iteration, we have a sequence of approximations $\hat{y}_1, \ldots, \hat{y}_{s_2}$, with $\hat{y}_0 = y_0$, $\hat{y}_1 = g(\hat{y}_0) + \xi_1, \ldots, \hat{y}_i = g(\hat{y}_{i-1}) + \xi_i, \ldots$, where $|\xi_i| \leq 2^{-b}$, $i \geq 1$. Using the fact that $\sup_{x \geq 0} |g_2'(y)| \leq \frac{3}{2}$, we obtain

$$\begin{aligned}
|\hat{y}_{s_2} - y_{s_2}| &\leq |g_2(\hat{y}_{s_2-1}) - g_2(y_{s_2-1})| + |\xi_{s_2}| \\
&\leq \frac{3}{2}|\hat{y}_{s_2-1} - y_{s_2-1}| + |\xi_{s_2}| \\
&\vdots \\
&\leq \sum_{j=1}^{s_2} \left(\frac{3}{2}\right)^{s_2-j} |\xi_j| \leq 2^{-b} \frac{\left(\frac{3}{2}\right)^{s_2} - 1}{\frac{3}{2} - 1} \\
&\leq 2^{1-b}\left(\frac{3}{2}\right)^{s_2}.
\end{aligned}$$

Therefore, the total error of the second stage of the algorithm is

$$|\hat{y}_{s_2} - \frac{1}{\sqrt{\mathbf{x}}}| \leq \frac{2}{3}\sqrt{2w}\left(\frac{3}{4}\right)^{2^{s_2}} + 2^{1-b}\left(\frac{3}{2}\right)^{s_2}. \tag{C.6}$$

For $b \geq \max\{2m, 4\}$ and $s_2 = \lceil \log_2 b \rceil$, and recall that $w \leq 2^m$. Then we have

$$\begin{aligned}
|\hat{y}_{s_2} - \frac{1}{\sqrt{\mathbf{x}}}| &\leq \frac{2}{3}\sqrt{2}\, 2^{\frac{m}{2}} \left(\frac{3}{4}\right)^b + 2^{1-b}\left(\frac{3}{2}\right)^{\log_2 b+1} \\
&\leq \frac{2}{3}\sqrt{2}\,\left(\sqrt{2}\right)^m \left(\left(\frac{3}{4}\right)^2\right)^{b/2} + 2^{1-b}\, 2^{\log_2 b+1} \\
&\leq \frac{2}{3}\sqrt{2}\,\left(\frac{\sqrt{2}\,9}{16}\right)^m \left(\frac{9}{16}\right)^{b/2-m} + 2^{2-b}\, b \\
&\leq \left(\frac{3}{4}\right)^{b-2m} + 2^{2-b}\, b, \qquad \text{(C.7)}
\end{aligned}$$

Let us now consider the total error of our algorithm,

$$|\hat{y}_{s_2} - \sqrt{w}| \leq |\hat{y}_{s_2} - \frac{1}{\sqrt{\mathbf{x}}}| + |\frac{1}{\sqrt{\mathbf{x}}} - \sqrt{w}| \qquad \text{(C.8)}$$

We use equation (C.7) above to bound the first term. For the second term we have

$$|\frac{1}{\sqrt{\mathbf{x}}} - \sqrt{w}| \leq \frac{1}{2}|\frac{1}{\mathbf{x}} - w| = \frac{1}{2}\frac{w}{\mathbf{x}}|\mathbf{x} - \frac{1}{w}| \leq \frac{1}{2}\frac{w}{\mathbf{x}}E, \qquad \text{(C.9)}$$

where the first inequality follows from the mean value theorem ($|\sqrt{a} - \sqrt{b}| \leq \frac{1}{2}|a-b|$ for $a, b \geq 1$). Since $\mathbf{x} \geq \frac{1}{w} - E$,

$$\mathbf{x}^{-1} \leq \left(1 - \frac{1}{w} - E\right)^{-1} \leq 2w, \qquad \text{(C.10)}$$

for $wE \leq \frac{1}{2}$. Then (C.9) becomes

$$\begin{aligned}
|\frac{1}{\sqrt{\mathbf{x}}} - \sqrt{w}| &\leq w^2 E \leq w^2\left(\left(\frac{1}{2}\right)^{2^{s_1}} + 2^{-b}s_1\right) \\
&\leq 2^{2m}\left(\left(\frac{1}{2}\right)^{2^{s_1}} + 2^{-b}s_1\right) \\
&\leq 2^{2m}\left(2^{-b} + 2^{-b}\lceil \log_2 b\rceil\right) \\
&\leq 2^{2m-b}\left(1 + \lceil \log_2 b\rceil\right), \qquad \text{(C.11)}
\end{aligned}$$

where we set $s_1 = \lceil \log_2 b \rceil$. Combining this with equation (C.8),

$$\begin{aligned}
|\hat{y}_{s_2} - \sqrt{w}| &\leq \left(\frac{3}{4}\right)^{b-2m} + 2^{2-b}\, b + 2^{2m-b}\left(1 + \lceil \log_2 b\rceil\right) \\
&\leq \left(\frac{3}{4}\right)^{b-2m}\left(2 + b + \log_2 b\right), \qquad \text{(C.12)}
\end{aligned}$$

and the error bound in the statement of the theorem follows for $s = s_1 = s_2$.

Finally, for completeness we show that for $b \geq \max\{2m, 4\}$, $wE \leq \frac{1}{2}$. Indeed,

$$\begin{aligned} wE &\leq 2^m \left(2^{-b} + 2^{-b}\lceil \log_2 b \rceil\right) \\ &\leq 2^{-b+m}\left(2 + \log_2 b\right) \\ &\leq \left(\frac{1}{2}\right)^{b/2-m} \frac{2+\log_2 b}{2^{b/2}}. \end{aligned}$$

The first factor above is at most $\frac{1}{2}$ since $b \geq \max\{2m, 4\}$, while the second is at most 1 and this completes the proof. □

Theorem 13. *For $w > 1$, represented by n bits of which the first m correspond to its integer part, Algorithm 2 computes approximations $\hat{z}_1, \hat{z}_2, \ldots, \hat{z}_k$ such that*

$$|\hat{z}_i - w^{\frac{1}{2^i}}| \leq 2\left(\frac{3}{4}\right)^{b-2m}\left(2 + b + \log_2 b\right),\ i = 1, 2, \ldots, k,\ \text{for any k} \in \mathbb{N}. \tag{C.13}$$

This is accomplished by repeatedly calling Algorithm 1. Each z_i has $b \geq \max\{2m, 4\}$ bits after its decimal point, and by convention its integer part is m bits long.

Proof. We have

$$\begin{aligned} \hat{z}_1 &= \sqrt{w} + \xi_1 \\ \hat{z}_2 &= \sqrt{z_1} + \xi_2 \\ &\vdots \\ \hat{z}_k &= \sqrt{z_k} + \xi_k. \end{aligned}$$

Since $\hat{z}_i$ is obtained using Algorithm 1 with input $\hat{z}_{i-1}$, we use Theorem 12 to obtain $|\xi_i| \leq \left(\frac{3}{4}\right)^{b-2m}(2 + b + \log_2 b)$, $i = 1, 2, \ldots, k$. We have

$$\begin{aligned} |\hat{z}_k - w^{\frac{1}{2^k}}| &\leq |\sqrt{\hat{z}_{k-1}} - w^{\frac{1}{2^k}}| + |\xi_k| \\ &\leq \frac{1}{2}|\hat{z}_{k-1} - w^{\frac{1}{2^{k-1}}}| + |\xi_k| \\ &\vdots \\ &\leq \sum_{j=0}^{k-1} \frac{1}{2^j}|\xi_{k-j}| \\ &\leq 2\left(\frac{3}{4}\right)^{b-2m}\left(2 + b + \log_2 b\right), \end{aligned}$$

where the second inequality above follows again from $|\sqrt{a} - \sqrt{b}| \leq \frac{1}{2}|a - b|$ for $a, b \geq 1$. □

Theorem 14. *For $w > 1$, represented by n bits of which the first m bits correspond to its integer part, Algorithm 3 computes an approximation $\hat{z} := \hat{z}_p + (p-1)r$ of $\ln w$, where $2^p > w \geq 2^{p-1}$, $p \in \mathbb{N}$, and $|r - \ln 2| \leq 2^{-b}$, such that*

$$|\hat{z} - \ln w| \leq \left(\frac{3}{4}\right)^{5\ell/2} \left(m + \frac{32}{9} + 2\left(\frac{32}{9} + \frac{n}{\ln 2}\right)^3\right),$$

where $\ell \geq \lceil \log_2 8n \rceil$ is a parameter specified in the algorithm that is used to determine the number $b = \max\{5\ell, 25\}$ of bits after the decimal point in which arithmetic is be performed, and from that the error.

Proof. An overall illustration of the algorithm is given in Fig. 2.6.

Our algorithm utilizes the identity $\ln w = \ln 2 \log_2 w$, as well as other common properties of logarithms. For completeness, an example circuit for computing p is given in Fig. 2.9 above. We proceed in detail.

If the clause of the second *if* statement evaluates to true, in the case $w = 2^{p-1}$, then the algorithm sets $z_p = 0$ and returns $(p-1)r$. This quantity approximates $\ln 2^{p-1}$ with error bounded from above by $(m-1)2^{-b}$, since $p \leq m$ in the algorithm. We deal with the case w not a power of 2, for which $z_p \neq 0$. Using the same notation as the algorithm, we have

$$\begin{aligned}
|z_p + (p-1)r - \ln(w)| &\leq |z_p + \ln 2^{p-1} - \ln(w)| + |(p-1)r - \ln 2^{p-1}| \\
&= |z_p - \ln 2^{-(p-1)} w| + (m-1)2^{-b} \\
&\leq |z_p - 2^\ell \ln w_p^{\frac{1}{2^\ell}}| + (m-1)2^{-b} \\
&\leq |z_p - 2^\ell \ln \hat{t}_p| + |2^\ell \ln \hat{t}_p - 2^\ell \ln w_p \frac{1}{2^\ell}| + (m-1)2^{-b} \\
&\leq |2^\ell \hat{y}_p - 2^\ell y_p| + |2^\ell y_p - 2^\ell \ln \hat{t}_p| \qquad \text{(C.14)} \\
&\quad + |2^\ell \ln \hat{t}_p - 2^\ell \ln w_p^{\frac{1}{2^\ell}}| + (m-1)2^{-b},
\end{aligned}$$

where $w_p = 2^{1-p} w$, $z_p = 2^\ell y_p$, $\hat{z}_p = 2^\ell \hat{y}_p$ and by y_p we denote the value $(\hat{t}_p - 1) - \frac{1}{2}(\hat{t}_p - 1)^2$ before it is truncated to obtain $\hat{y}_p$. The first term is due to truncation error and is bounded from above by $2^{\ell-b}$. The last term is bounded from above by $2^\ell |\hat{t}_p - w_p^{\frac{1}{2^\ell}}|$; this is obtained using the mean value theorem for $\ln$ with argument greater than or equal to one. The second term is bounded from above by $2^\ell |2(\hat{t}_p - 1)^3|$. Indeed, recall that $\hat{t}_p - 1 = \delta \in (0,1)$ according to line 12 of the algorithm, and we have

$$\begin{aligned}
|\ln(1+\delta)| - (\delta - \frac{1}{2}\delta^2)| &= \left|\sum_{i=3}^{\infty}(-1)^{i+1}\frac{\delta^i}{i}\right| = \left|\delta^2\sum_{i=0}^{\infty}\frac{i+1}{i+3}\int_0^{\delta}(-x)^i dx\right| \\
&= \left|\delta^2\int_0^{\delta}\sum_{i=0}^{\infty}\frac{i+1}{i+3}(-x)^i dx\right| \le \delta^2\int_0^{\delta}\sum_{i=0}^{\infty}|x|^i dx \\
&\le \delta^2\int_0^{\delta}\frac{1}{1-|x|}dx \le \delta^2\int_0^{\delta}2dx = 2\delta^3,
\end{aligned}$$

assuming $\delta \le \frac{1}{2}$. We now show that in general δ is much smaller than $\frac{1}{2}$. Since we have used Algorithm 2 to compute $\hat{t}_p$ which is an approximation of $w_p^{\frac{1}{2^\ell}}$. Algorithm 2 uses Algorithm 1. Since the error bounds of Theorem 12 and 13 depend on the magnitude of w_p, the estimates of these theorems hold with $m = 1$ because $w_p \in (1, 2)$. We have

$$|\hat{t}_p - w_p^{\frac{1}{2^\ell}}| \le 2\left(\frac{3}{4}\right)^{b-2}(2 + b + \log_2 b) \le \frac{1}{8},$$

where we have set $b = \max\{5\ell, 25\}$. Thus

$$\hat{t}_p \le w_p^{\frac{1}{2^\ell}} + 2\left(\frac{3}{4}\right)^{b-2}(2 + b + \log_2 b) \le w_p^{\frac{1}{2^\ell}} + \frac{1}{8}. \qquad \text{(C.15)}$$

Now we turn to $w_p^{\frac{1}{2^\ell}}$. Let $w_p = 1 + x_p$, $x_p \in (0, 1)$. Consider the function $g(x_p) := (1 + x_p)^{\frac{1}{2^\ell}}$. We take its Taylor expansion about 0, and observing that

$$g^{(j)}(x_p) = \left(\prod_{i=0}^{j-1}\frac{1 - i2^\ell}{2^\ell}\right)(1 + x_p)^{\frac{1-j2^\ell}{2^\ell}} \le \frac{(j-1)!}{2^\ell}, \;\; j \ge 1,$$

we have

$$\begin{aligned}
w_p^{\frac{1}{2^\ell}} - 1 = (1 + x_p)^{\frac{1}{2^\ell}} - 1 &\le \frac{1}{2^\ell}\sum_{j=1}^{\infty}\frac{(j-1)!}{j!}x_p^j = \frac{1}{2^\ell}\sum_{j=1}^{\infty}\frac{1}{j}x_p^j \\
&= \frac{1}{2^\ell}\sum_{j=1}^{\infty}\int_0^{x_p}s^{j-1}ds = \frac{1}{2^\ell}\int_0^{x_p}\sum_{j=1}^{\infty}s^{j-1}ds \\
&\le \frac{1}{2^\ell}\int_0^{x_p}\frac{1}{1-s}ds = \frac{1}{2^\ell}\ln\frac{1}{1-x_p} \\
&\le \frac{1}{2^\ell}\frac{n-m+p-1}{\log_2 e} \le \frac{n-m+p-1}{2^\ell},
\end{aligned}$$

because $x_p \le 1 - 2^{-(n-m+p-1)}$, since w has a fractional part of length $n - m$. Observing that $n - m + p - 1 \le n$, and by setting $\ell \ge \lceil \log_2 8n \rceil$, we get

$$w_p^{\frac{1}{2^\ell}} - 1 \le \frac{n}{2^\ell} \le \frac{n}{8n} \le \frac{1}{8}. \qquad \text{(C.16)}$$

Using equations (C.15) and (C.16), we get

$$\hat{t}_p - 1 \leq \frac{1}{2^\ell}\frac{n}{\ln 2} + 2\left(\frac{3}{4}\right)^{b-2}(2 + b + \log_2 b) \leq \frac{1}{4}. \tag{C.17}$$

Now we turn to the error of the algorithm. We have

$$\begin{aligned} |z_p + \ln 2^{p-1} - \ln(w)| &\leq |2^\ell \hat{y}_p - 2^\ell y_p| + |2^\ell y_p - 2^\ell \ln \hat{t}_p| + |2^\ell \ln \hat{t}_p - 2^\ell \ln w_p^{\frac{1}{2^\ell}}| \\ &\leq 2^{\ell-b} + 2^\ell |2(\hat{t}_p - 1)^3| + 2^\ell |\hat{t}_p - w_p^{\frac{1}{2^\ell}}| \\ &\leq 2^{\ell-b} + 2^\ell 2\left(\frac{1}{2^\ell}\frac{n}{\ln 2} + 2\left(\frac{3}{4}\right)^{b-2}(2 + b + \log_2 b)\right)^3 \\ &+ 2^\ell\left(2\left(\frac{3}{4}\right)^{b-2}(2 + b + \log_2 b)\right) \end{aligned}$$

Since $b \geq \max\{5\ell, 25\}$, we have $\left(\frac{3}{4}\right)^{b/2} 2^\ell \leq 1$ and $\frac{2+b+\log_2 b}{\left(\frac{4}{3}\right)^{b/2}} \leq 1$. This yields

$$\begin{aligned} |z_p + \ln 2^{p-1} - \ln(w)| &\leq 2^\ell\left(2^{-b} + \frac{2}{2^{3\ell}}\left(\frac{32}{9} + \frac{n}{\ln 2}\right)^3 + \frac{32}{9}\left(\frac{3}{4}\right)^{b/2}\right) \\ &\leq 2^{-4\ell} + \frac{2}{2^{2\ell}}\left(\frac{32}{9} + \frac{n}{\ln 2}\right)^3 + \frac{32}{9}\left(\frac{3}{4}\right)^{5\ell/2} \\ &\leq \left(\frac{3}{4}\right)^{5\ell/2}\left(1 + \frac{32}{9} + 2\left(\frac{32}{9} + \frac{n}{\ln 2}\right)^3\right). \end{aligned} \tag{C.18}$$

Finally, from $b \geq 5\ell$ we have

$$(m-1)2^{-b} \leq (m-1)2^{-5\ell} \leq (m-1)\left(\frac{1}{4}\right)^{5\ell/2} \leq (m-1)\left(\frac{3}{4}\right)^{5\ell/2} \tag{C.19}$$

Combining equations (C.14), (C.18), and (C.19) then gives

$$\begin{aligned} |z_p + (p-1)r - \ln(w)| &\leq \left(\frac{3}{4}\right)^{5\ell/2}\left(1 + \frac{32}{9} + 2\left(\frac{32}{9} + \frac{n}{\ln 2}\right)^3\right) + (m-1)\left(\frac{3}{4}\right)^{5\ell/2} \\ &\leq \left(\frac{3}{4}\right)^{5\ell/2}\left(m + \frac{32}{9} + 2\left(\frac{32}{9} + \frac{n}{\ln 2}\right)^3\right), \end{aligned}$$

which is our desired bound. □

Theorem 15. *For $w > 1$, given by n bits of which the first m correspond to its integer part, and $1 \geq f \geq 0$ given by n_f bits of accuracy, Algorithm 4 computes an approximation $\hat{z}$ of w^f such that*

$$|\hat{z} - w^f| \leq \left(\frac{1}{2}\right)^{\ell-1}, \tag{C.20}$$

where $\ell \in \mathbb{N}$ is a parameter specified in the algorithm that is used to determine the number $b = \max\{n, n_f, \lceil 5(\ell + 2m + \ln n_f)\rceil, 40\}$ of bits after the decimal point in which arithmetic will be performed, and therefore it determines the error. Algorithm 4 uses Algorithm 2 which computes power of 2 roots of a given number.

Proof. First observe that the algorithm is exact for the cases $f = 1$ or $f = 0$. Therefore, without loss of generality assume $0 < f < 1$.

Consider the n_f bit number f and write its binary digits $f_i \in \{0, 1\}$ explicitly as $f = 0.f_1 f_2 ... f_{n_f} = \sum_{i=1}^{n_f} f_i/2^i$. Denote the set of non-zero digits $\mathcal{P} := \{1 \leq i \leq n_f : f_i \neq 0\}$, $p := |\mathcal{P}|$ with $1 \leq p \leq n_f$. Observe

$$w^f = w^{0.f_1 f_2 ... f_{n_f}} = w^{\sum_{i=1}^{n_f} f_i/2^i} = \prod_{i=1}^{n_f} w_i^{f_i} = \prod_{i\in\mathcal{P}} w_i,$$

where again $w_i := w^{\frac{1}{2^i}}$. Let $\hat{w}_i \simeq w^{\frac{1}{2^i}}$ denote the outputs of Algorithm 2. We have

$$\left|\hat{z} - w^f\right| \leq \left|\hat{z} - \prod_{i\in\mathcal{P}} \hat{w}_i\right| + \left|\prod_{i\in\mathcal{P}} \hat{w}_i - \prod_{i\in\mathcal{P}} w_i\right|, \tag{C.21}$$

where these terms give bounds for the repeated multiplication error, and the error from the 2^ith roots as computed by Algorithm 2, respectively.

Consider the second term. Partition $\mathcal{P}$ disjointly into two sets as $\mathcal{P}_1 := \{i \in \mathcal{P} : \hat{w}_i \geq w_i\}$ and $\mathcal{P}_2 := \{i \in \mathcal{P} : \hat{w}_i < w_i\}$. Observe that, for any i, from equation (C.13) of Theorem 13 we have $|\hat{w}_i - w_i| \leq 2\left(\frac{3}{4}\right)^{b-2m}(2 + b + \log_2 b) =: \varepsilon$. Observe $w_i, \hat{w}_i \geq 1$. First assume $\prod_{i\in\mathcal{P}} \hat{w}_i \geq \prod_{i\in\mathcal{P}} w_i$. Then

$$\begin{aligned}
\left|\prod_{i\in\mathcal{P}} \hat{w}_i - \prod_{i\in\mathcal{P}} w_i\right| &= \prod_{i\in\mathcal{P}_1} \hat{w}_i \prod_{j\in\mathcal{P}_2} \hat{w}_j - \prod_{i\in\mathcal{P}_1} w_i \prod_{j\in\mathcal{P}_2} w_j \\
&\leq \prod_{i\in\mathcal{P}_1} (w_i + \varepsilon) \prod_{j\in\mathcal{P}_2} w_j - \prod_{i\in\mathcal{P}_1} w_i \prod_{j\in\mathcal{P}_2} w_j \\
&\leq \prod_{i\in\mathcal{P}_1} w_i \prod_{j\in\mathcal{P}_2} w_j \left(\prod_{k\in\mathcal{P}_1}\left(1 + \frac{\varepsilon}{w_k}\right) - 1\right) \\
&\leq w^f\left((1+\varepsilon)^p - 1\right) \\
&\leq w^f\left(e^{p\varepsilon} - 1\right)
\end{aligned}$$

Conversely, assume $\prod_{i\in\mathcal{P}} \hat{w}_i < \prod_{i\in\mathcal{P}} w_i$. Then similarly we have

$$\begin{aligned}
\left|\prod_{i\in\mathcal{P}} \hat{w}_i - \prod_{i\in\mathcal{P}} w_i\right| &= \prod_{i\in\mathcal{P}_1} w_i \prod_{j\in\mathcal{P}_2} w_j - \prod_{i\in\mathcal{P}_1} \hat{w}_i \prod_{j\in\mathcal{P}_2} \hat{w}_j \\
&\leq \prod_{i\in\mathcal{P}_1} w_i \prod_{j\in\mathcal{P}_2} w_j - \prod_{i\in\mathcal{P}_1} w_i \prod_{j\in\mathcal{P}_2} (w_j - \varepsilon) \\
&\leq \prod_{i\in\mathcal{P}_1} w_i \prod_{j\in\mathcal{P}_2} w_j \left(1 - \prod_{k\in\mathcal{P}_2}\left(1 - \frac{\varepsilon}{w_k}\right)\right) \\
&\leq w^f \left(1 - (1-\varepsilon)^p\right) \\
&\leq w^f \left(1 - (2 - e^{p\varepsilon})\right) \\
&\leq w^f \left(e^{p\varepsilon} - 1\right),
\end{aligned}$$

where we have used the inequality $(1-\varepsilon)^p - 1 \geq 1 - e^{pe}$.[1] So conclude that $\left|\prod_{i\in\mathcal{P}} \hat{w}_i - \prod_{i\in\mathcal{P}} w_i\right| \leq w^f\left(e^{p\varepsilon} - 1\right)$ always. Furthermore, for $a \geq 0$ we have

$$e^a - 1 = a + a^2/2! + a^3/3! + \cdots = a(1 + a/2! + a^2/3! + \dots) \leq a(1 + a + a^2/2! + \dots) = ae^a$$

which yields

$$\left|\prod_{i\in\mathcal{P}} \hat{w}_i - \prod_{i\in\mathcal{P}} w_i\right| \leq w^f \; p\varepsilon \; e^{p\varepsilon} \tag{C.22}$$

Next consider the error resulting from truncation to b bits of accuracy in the products computed in step 13 of the algorithm. For each multiplication, we have $\hat{z} = z + \xi$, with error $|\xi| \leq 2^{-b}$. For notational simplicity, reindex the set $\mathcal{P}$ as $\{1, 2, \dots, p\}$ so that $\prod_{i\in\mathcal{P}} \hat{w}_i = \prod_{i=1}^{p} \hat{w}_i$. Let $z_i = \hat{w}_1 \hat{w}_2 \dots \hat{w}_i$, $i = 1, 2, \dots, p$ be the exact products, and let the approximate products be $\hat{z}_i = \hat{z}_{i-1}\hat{w}_i + \xi_i$, $i = 2, \dots, p$, $\hat{z}_1 = 1$. We have

$$\begin{aligned}
\hat{z}_1 &= \hat{w}_1 \\
\hat{z}_2 &= \hat{z}_1 \hat{w}_2 + \xi_2 \\
&\vdots \\
\hat{z}_p &= \hat{z}_{p-1}\hat{w}_p + \xi_p.
\end{aligned}$$

[1] This inequality follows trivially from term by term comparison of the binomial expansion of the left hand side with the Taylor expansion of the right hand side.

Then we have

$$\begin{aligned}
\left|\hat{z}_p - \prod_{i\in\mathcal{P}} \hat{w}_i\right| &= |\hat{z}_p - z_p| = |\hat{z}_{p-1}\hat{w}_p + \xi_p - z_{p-1}\hat{w}_p| \leq |\hat{z}_{p-1}\hat{w}_p - z_{p-1}\hat{w}_p| + |\xi_p| \\
&\leq |\hat{z}_{p-1} - z_{p-1}|\hat{w}_p + |\xi_p| \\
&\leq \left(|\hat{z}_{p-2} - z_{p-2}|\hat{w}_{p-1} + |\xi_{p-1}|\right)\hat{w}_p + |\xi_p| \\
&\leq |\hat{z}_{p-2} - z_{p-2}|\hat{w}_{p-1}\hat{w}_p + |\xi_{p-1}|\hat{w}_p + |\xi_p| \\
&\vdots \\
&\leq |\hat{z}_1 - z_1|\hat{w}_2\hat{w}_3\hat{w}_4\ldots\hat{w}_{p-1}\hat{w}_p + |\xi_2|\hat{w}_3\hat{w}_4\ldots\hat{w}_{p-1}\hat{w}_p + \cdots + |\xi_{p-1}|\hat{w}_p + |\xi_p| \\
&\leq 2^{-b}\left(\hat{w}_3\hat{w}_4\ldots\hat{w}_{p-1}\hat{w}_p + \hat{w}_4\ldots\hat{w}_{p-1}\hat{w}_p + \cdots + \hat{w}_{p-1}\hat{w}_p + \hat{w}_p + 1\right) \\
&\leq 2^{-b}(p-1)\,w^{\frac{1}{4}} \leq 2^{-b}n_f w^f \leq 2^{-b}n_f w
\end{aligned}$$

where the last line follows from observing each of the $p-1$ terms in the sum is less than $w_2 = w^{\frac{1}{4}}$.

Thus, equation (C.21) yields total error

$$\begin{aligned}
\left|\hat{z} - w^f\right| &\leq \left|\hat{z} - \prod_{i\in\mathcal{P}}\hat{w}_i\right| + \left|\prod_{i\in\mathcal{P}}\hat{w}_i - \prod_{i\in\mathcal{P}} w_i\right| \\
&\leq 2^{-b}(p-1)\,w^{\frac{1}{4}} + w^f\,p\varepsilon\,e^{p\varepsilon} \\
&\leq 2^{-b}\,n_f w + w\,n_f\varepsilon\,e^{n_f\varepsilon}.
\end{aligned}$$

Furthermore, using $\varepsilon = 2\left(\frac{3}{4}\right)^{b-2m}(2 + b + \log_2 b)$, $1 \leq w \leq 2^m$, and as we have chosen b sufficiently large such that $n_f\varepsilon \leq 1$ (to be shown later), we have

$$\begin{aligned}
\left|\hat{z} - w^f\right| &\leq 2^{-b}\,n_f w + w\,n_f\,2\left(\frac{3}{4}\right)^{b-2m}(2 + b + \log_2 b)\,e \\
&\leq n_f 2^{m-b} + n_f 2^m\left(\frac{3}{4}\right)^{b-2m} 2e\,(2 + b + \log_2 b)\,.
\end{aligned}$$

We have selected $b \geq \max\{n, n_f, \lceil 5(\ell + 2m + \log_2 n_f)\rceil, 40\}$ such that several inequalities are satisfied. From $b \geq \lceil 5(\ell + 2m + \log_2 n_f)\rceil \geq \ell + m + \log_2 n_f$, it follows that $n_f 2^{m-b} \leq 2^{-\ell}$. Furthermore, for $b \geq 40$, we have $2e\,(2 + b + \log_2 b) \leq (\frac{4}{3})^{b/2}$. Finally, $b \geq 5\,(\ell + 2m + \log_2 n_f) \geq \left(\frac{2}{\log_2 4/3}\right)(\log_2 n_f + \ell + 2m)$ implies that $n_f 2^{2m}\left(\frac{3}{4}\right)^{b/2} \leq 2^{-\ell}$. Plugging these inequalities into the previous equation yields

$$\begin{aligned}\left|\hat{z}-w^f\right| &\leq n_f 2^m \frac{1}{2^b} + n_f 2^m \left(\frac{4}{3}\right)^{2m}\left(\frac{3}{4}\right)^{b/2}\left(\left(\frac{3}{4}\right)^{b/2} 2e\left(2+b+\log_2 b\right)\right)\\ &\leq \left(\frac{1}{2}\right)^{\ell} + n_f 2^m \left(\left(\frac{4}{3}\right)^{2}\right)^{m}\left(\frac{3}{4}\right)^{b/2}\\ &\leq \left(\frac{1}{2}\right)^{\ell} + n_f 2^{2m}\left(\frac{3}{4}\right)^{b/2}\\ &\leq 2\left(\frac{1}{2}\right)^{\ell} = \left(\frac{1}{2}\right)^{\ell-1}\end{aligned}$$

as was to be shown.

Finally, for completeness, from the above inequalities and

$$b \geq 5\left(\log_2 n_f + \ell + 2m\right) \geq \frac{2}{\log_2 4/3}\left(\log_2 n_f - \log_2 e\right) + 4m,$$

we have

$$\begin{aligned} n_f \varepsilon &= n_f 2\left(\frac{3}{4}\right)^{b-2m}\left(2+b+\log_2 b\right) \leq n_f \left(\frac{3}{4}\right)^{b/2-2m}\left(\frac{1}{e}\right)\\ &\leq n_f\left(\frac{3}{4}\right)^{\log_{4/3} n_f/e}\left(\frac{1}{e}\right) \leq 1.\end{aligned}$$

□

Corollary 9. *Let $w > 1$ and $1 \geq f \geq 0$ as in Theorem 4 above. Suppose f is an approximation of a number $1 \geq F \geq 0$ accurate to n_f bits. Then Algorithm 4, with f as input, computes an approximation $\hat{z}$ of w^F such that*

$$|\hat{z} - w^F| \leq \left(\frac{1}{2}\right)^{\ell-1} + \frac{w \ln w}{2^{n_f}}. \tag{C.23}$$

Proof. Consider the error from the approximation of the exponent F by f. We have $|F-f| \leq 2^{-n_f}$. Let $g(f) := w^f$. Then $g'(f) = w^f \ln w$. By the mean value theorem, we have

$$\left|w^f - w^F\right| \leq \sup_{f\in(0,1)} g'(f)|F-f| \leq 2^{-n_f}\, w \ln w,$$

which gives

$$|\hat{z} - w^F| \leq |\hat{z} - w^f| + |w^f - w^F| \leq \left(\frac{1}{2}\right)^{\ell-1} + \frac{w \ln w}{2^{n_f}}.$$

□

Theorem 16. *For $0 \le w < 1$, represented by n bits of which the first m correspond to its integer part, and $1 \ge f \ge 0$ given by n_f bits of accuracy, Algorithm 5 computes an approximation $\hat{t}$ of w^f such that*

$$|\hat{t} - w^f| \le \frac{1}{2^{\ell-3}} \tag{C.24}$$

where $\ell \in \mathbb{N}$ is a parameter specified in the algorithm that is used to determine the number $b = \max\{n, n_f, \lceil 2\ell + 6m + 2\ln n_f \rceil, 40\}$ of bits after the decimal point in which arithmetic will be performed, and therefore also will determine the error. Algorithm 5 uses Algorithm 4, which computes w^f for the case $w \ge 1$, and also Algorithm 0 which computes the reciprocal of a number $w \ge 1$.

Proof. First observe that the algorithm is exact for the cases $f = 1$, $f = 0$, or $w = 0$. Therefore, without loss of generality assume $0 < f < 1$ and $0 < w < 1$.

Let all variables be defined as in Algorithm 5. We shall first consider the error of each variable and use this to bound the overall error of algorithm.

Firstly, the input $0 < w < 1$ is rescaled to $x := 2^k w \ge 1 > 2^{k-1} w$ exactly, by k-bit left shift,where k is a positive integer. An example circuit for computing k is given in Fig. 2.9 above. Observe that we have $w^f = x^f / 2^{kf} = x^f / (2^{\lfloor kf \rfloor} 2^{\{kf\}})$. We also have $\log_2 \frac{1}{w} \le k < \log_2 \frac{1}{w} + 1$.

The product $c = kf < k \le n - m$ is computed exactly in fixed precision arithmetic because the number of bits after the decimal point in kf is at most $n_f \le b$, where b is the number of bits in which arithmetic is performed.

Next consider $\hat{z}$ = FractionalPower(x, f, n, m, n_f, ℓ), which approximates $z = x^f$. From Theorem 15 we have $e_z := |\hat{z} - z| \le \frac{1}{2^{\ell-1}}$. Similarly, $\hat{y}$ = FractionalPower(2, $\{c\}$, n, m, n_f, ℓ) approximates $y = 2^{\{c\}}$, with $e_y := |\hat{y} - y| \le \frac{1}{2^{\ell-1}}$. Furthermore, for $\hat{s}$ = INV$(\hat{y}, n, 1, 2\ell)$ which approximates $s = 1/\hat{y}$, from Corollary 8 we have $e_s := |\hat{s} - s| \le \frac{2+\log_2 \ell}{2^{2\ell}}$, which satisfies $e_s \le \frac{1}{2^\ell}$ for $\ell \ge 2$.

Finally, observe $2^{-\lfloor c \rfloor} \hat{z}$ is computed exactly by a right shift of $\hat{z}$. This is used to compute $t = 2^{-\lfloor c \rfloor} \hat{z} \hat{s}$, which is again truncated to b decimal bits to give $\hat{t}$ with $e_t := |\hat{t} - t| \le 2^{-b}$.

Now we turn to the total error. By our variable definitions, $w^f = 2^{-\lfloor c\rfloor} z/y$. We have

$$\begin{aligned}
|\hat{t} - w^f| &\leq |\hat{t} - t| + |2^{-\lfloor c\rfloor}\hat{z}\hat{s} - 2^{-\lfloor c\rfloor}z\hat{s}| + |2^{-\lfloor c\rfloor}z\hat{s} - 2^{-\lfloor c\rfloor}zs| + \left|2^{-\lfloor c\rfloor}zs - 2^{-\lfloor c\rfloor}\frac{z}{y}\right| \\
&\leq 2^{-b} + 2^{-\lfloor c\rfloor}\hat{s}|\hat{z} - z| + 2^{-\lfloor c\rfloor}z|\hat{s} - s| + 2^{-\lfloor c\rfloor}z\left|s - \frac{1}{y}\right| \\
&\leq 2^{-b} + \frac{1}{2^{\lfloor c\rfloor}}\frac{1}{2^{\ell-1}} + w^f 2^{\{c\}}|\hat{s} - s| + \frac{x^f}{2^{\lfloor c\rfloor}}\frac{1}{2^{\{c\}}}\left|\frac{2^{\{c\}}}{\hat{y}} - 1\right|,
\end{aligned}$$

where we have used $\hat{s} \leq s = \frac{1}{\hat{y}} \leq 1$ because as remarked in the proof of Theorem 12, the algorithm computing the reciprocal underestimates it value, i.e. $\frac{1}{\hat{y}} \leq \frac{1}{y} = 2^{-\lfloor kf\rfloor} \leq 1$. Observe we have $w^f = \frac{x^f}{2^{\lfloor c\rfloor}}\frac{1}{2^{\{c\}}} \leq 1$. Moreover, $2^{\{kf\}} \leq 2$. Hence, as shown in the proof of Theorem 12, that $\hat{y} \geq 1$. This, together with the error bounds of Theorem 15 and of Corollary 8 yields

$$\begin{aligned}
|\hat{t} - w^f| &\leq 2^{-b} + \frac{1}{2^{\ell-1}} + 2|\hat{s} - s| + \frac{1}{\hat{y}}|2^{\{c\}} - \hat{y}| \\
&\leq 2^{-b} + \frac{1}{2^{\ell-1}} + \frac{2}{2^{\ell}} + \frac{1}{2^{\ell-1}} \\
&\leq 4\frac{1}{2^{\ell-1}} = \frac{1}{2^{\ell-3}}
\end{aligned}$$

□

Corollary 10. *Let* $0 \leq w < 1$ *and* $1 \geq f \geq 0$ *as in Theorem 5 above. Suppose* f *is an approximation of a number* $1 \geq F \geq 0$ *accurate to* n_f *bits. Then Algorithm 5, with* f *as input, computes an approximation* $\hat{t}$ *of* w^F *such that*

$$|\hat{t} - w^F| \leq \left(\frac{1}{2}\right)^{\ell-2} + \frac{w \ln w}{2^{n_f}}. \tag{C.25}$$

Proof. The proof is similar to that of Corollary 9. □

Appendix D

Divide and Conquer Hamiltonian Simulation

In this appendix, we review high-order splitting formulas for Hamiltonian simulation, and give the proofs for several results of Chapter 4.

D.1 Review of High-Order Splitting Formulas

Splitting formulas are a family of operator approximations based on the *Lie-Trotter product formula*

$$\lim_{n\to\infty}(e^{-iH_1t/n}e^{-iH_2t/n}\dots e^{-iH_mt/n})^n = e^{-iHt}. \tag{D.1}$$

Using this formula with finite n gives an approximation of e^{-iHt}. Without loss of generality, and to avoid dealing with absolute values, we will assume $t > 0$ here. Selecting n, often called the *Trotter number*, large enough such that the *time slice* $\Delta t := t/n$ is small $\Delta t \ll 1$, we approximate $e^{-iH\Delta t}$ by $e^{-iH_1\Delta t}e^{-iH_2\Delta t}\dots e^{-iH_m\Delta t}$ with error $O(\Delta t^2)$. This gives a second-order approximation. A third-order approximation is given by the Strang splitting formula

$$S_2(H_1,\dots,H_m,\Delta t) = e^{-iH_1\Delta t/2}\dots e^{-iH_{m-1}\Delta t/2}e^{-iH_m\Delta t}e^{-iH_{m-1}\Delta t/2}\dots e^{-iH_1\Delta t/2}, \tag{D.2}$$

with[1]

$$e^{-iH\Delta t} = S_2(\Delta t) + O(\Delta t^3), \qquad \text{as } \Delta t \to 0.$$

[1]For simplicity, when the underlying Hamiltonian decomposition is clear we will use $S_2(\Delta t)$ in place of $S_2(H_1,\dots,H_m,\Delta t)$.

Applying $S_2(\Delta t)$ over each time slice Δt yields the approximation

$$\widetilde{U} = (S_2(\Delta t))^n,$$

where $\|U - \widetilde{U}\| \to 0$ as $\Delta t \to 0$. Assume for the moment that Δt is chosen such that the number of time slices $n = t/\Delta t$ is indeed an integer. Otherwise, we would have $n = \lceil t/\Delta t \rceil$, and a single different final time slice $\Delta t' := t - \Delta t \lfloor t/\Delta t \rfloor < \Delta t$.

Suzuki [214, 215] gave high-order splitting formulas. These are recursive formulas S_{2k} of order $2k+1$, $k \in \mathbb{N}$, approximating $e^{-iH\Delta t}$ to error $O(\Delta t^{2k+1})$. They are defined by

$$S_{2k}(\Delta t) = [S_{2(k-1)}(p_k \Delta t)]^2 \; S_{2(k-1)}(q_k \Delta t) \; [S_{2(k-1)}(p_k \Delta t)]^2, \tag{D.3}$$

for $k = 2, 3, \ldots$, with $p_k = (4 - 4^{1/(2k-1)})^{-1})$ and $q_k = 1 - 4p_k$. Applying $S_{2k}(\Delta t)$ over each time slice Δt and unwinding the recurrence relation yields a product of N exponentials

$$\widetilde{U} = (S_{2k}(H_1, \ldots, H_m, t/n))^n = \prod_{\ell=1}^{N} e^{-iH_{j_\ell} t_\ell / n}, \qquad j_\ell \in \{1, \ldots, m\}, \qquad \sum_{\ell=1}^{N} t_\ell / n = mt. \tag{D.4}$$

It is important to observe that Suzuki's formulas hold asymptotically for sufficiently small Δt, and do not reveal the dependence of the error on m or the norms $\|H_j\|$, $j = 1, \ldots, m$. Application of these formulas requires explicit calculation of the prefactors in the error bounds. Typically, cost estimates for splitting methods are expressed as the product of the number of time slices and the number of exponentials required to carry out the simulation within each time slice. In particular, the estimates for the simulation error and cost in [186] depend on m, ε, k, the largest norm $\|H_1\|$, and the second largest norm $\|H_2\|$. In [186, Sec. 4] the quantity M is defined as

$$M = \left(\frac{4emt\|H_2\|}{\varepsilon} \right)^{1/2k} \frac{4em}{3} \left(\frac{5}{3} \right)^{k-1}, \tag{D.5}$$

and the time slice is given by $\Delta t = (M\|H_1\|)^{-1}$. Hence the number of intervals is $n = \lceil t/\Delta t \rceil = \lceil M\|H_1\|t \rceil$. Note that choosing M larger than necessary decreases the simulation error. Under the (weak) assumption $4emt\|H_2\| > \varepsilon$, [186, Thm. 2] shows an upper bound for the number of exponentials

$$N \leq \left((2m-1)5^{k-1} \right) \cdot \left\lceil \|H_1\| t \left(\frac{4emt\|H_2\|}{\varepsilon} \right)^{\frac{1}{2k}} \frac{4em}{3} \left(\frac{5}{3} \right)^{k-1} \right\rceil. \tag{D.6}$$

This bound is derived as the product of two terms. The first factor is the number of exponentials per time slice, which is bounded by $(2m-1)5^{k-1}$. The second factor is equal to n which bounds the number of time slices. Note that if the argument of the ceiling function is at most one, a single time interval suffices for the simulation. The cost bounds in Section 4.3 for Algorithms 1 and 2 are generalizations of (D.6).

Recall that the upper bound (D.6) does not account for any finer problem structure, such as the possibility that a number of Hamiltonians have norms significantly smaller than $\|H_2\|$. Hamiltonians with extremely small norm ($\|H_j\| = O(\varepsilon/t)$) can be ignored altogether as indicated in Proposition 1. The remaining Hamiltonians may then be partitioned into groups based on their relative norms, and each group simulated independently with our algorithms. This leads us to refined cost estimates which depend not just on m, $\|H_1\|$, and $\|H_2\|$, but on the number of Hamiltonians in each group and largest Hamiltonian norm within each group.

Furthermore, under weak conditions which guarantee the argument of the ceiling function in (D.6) is at least one (e.g. for sufficiently large m, $\|H_1\|$, t, or $1/\varepsilon$), (D.6) may be bounded to obtain

$$N(k) \quad \leq \quad \left((2m-1)5^{k-1}\right) \cdot 2\|H_1\|t\left(\frac{4emt\|H_2\|}{\varepsilon}\right)^{\frac{1}{2k}} \frac{4em}{3}\left(\frac{5}{3}\right)^{k-1} =: N(k), \quad \text{(D.7)}$$

and from this [186, Sec. 5] shows the "optimal" k (in the sense of minimizing the upper bound $N(k)$),

$$k^* := \max\left\{\text{round}\left(\sqrt{\frac{1}{2}\log_{25/3}\frac{4\text{emt}\|\text{H}_2\|}{\varepsilon}}\right), 1\right\}. \tag{D.8}$$

Setting $k = k^*$ gives the upper bound for the number of matrix exponentials

$$N \leq \frac{8e}{3}(2m-1)\; m\|H_1\|t\; e^{2\sqrt{\frac{1}{2}\ln\frac{25}{3}\ln\frac{4emt\|H_2\|}{\varepsilon}}} =: N^*. \tag{D.9}$$

We compare our results against this estimate in Section 4.3.4.

Further observe that k^* is given by an extremely slow growing function of the problem parameters. For example, for the values $m = t = \|H_2\| = \varepsilon^{-1} = 10^{10}$, (D.8) gives $k^* = 5$. Therefore, in most practical cases, one can determine the optimal value of k by inspection, with the need to carry out a formal analysis.

D.2 Divide and Conquer Hamiltonian Simulation

Discarding Small Hamiltonians

Proof of Proposition 1. Using the variation-of-constants formula [137], for any vector v we have

$$e^{-iHt}v = e^{-iAt}v - i\int_0^t e^{-iAs}Be^{-iH(t-s)}v\,ds \qquad (t \geq 0),$$

which implies

$$\|e^{-iHt} - e^{-iAt}\| \leq \|B\|t \leq \varepsilon/2.$$

Thus,

$$\|e^{-iHt} - \widetilde{U}\| \leq \|e^{-iHt} - e^{-iAt}\| + \|e^{-iAt} - \widetilde{U}\| \leq \varepsilon/2 + \varepsilon/2 \leq \varepsilon.$$

□

Recursive Lie-Trotter Formulas

We show a generalization of the Lie-Trotter formula. For simplicity and to avoid technical details we assume that H, A, and B are complex matrices.

Lemma 4 (Recursed Lie-Trotter Formula)**.** *Let $H_1, \ldots, H_m$ be Hamiltonians with $H = \sum_{j=1}^m H_j$. Consider $A = \sum_{j=1}^{m'} H_j$ and $B = \sum_{j=m'+1}^{m} H_j$. Let*

$$f(n,\alpha,\beta) := \Big((e^{-iH_1 t/\alpha n} \ldots e^{-iH_{m'} t/\alpha n})^\alpha \; (e^{-iH_{m'+1} t/\beta n} \ldots e^{-iH_m t/\beta n})^\beta \Big)^n$$

for $n, \alpha, \beta \in \mathbb{N}$. Then for fixed α, β, we have

$$\lim_{n\to\infty} f(n,\alpha,\beta) = e^{-iHt}.$$

In particular, for $\alpha = \beta = 1$ the usual Lie-Trotter formula (D.1) is reproduced.

Moreover, we may also take limits with respect to α and β, and in any order, i.e.,

$$\lim_{n,\alpha,\beta\to\infty} f(n,\alpha,\beta) = e^{-iHt}.$$

Proof. Fix α, β. Then we may expand f as

$$\begin{aligned} f(n,\alpha,\beta) \quad = \quad & \Big(\underbrace{(e^{-iH_1 t/\alpha n} \ldots e^{-iH_{m'} t/\alpha n}) \ldots (e^{-iH_1 t/\alpha n} \ldots e^{-iH_{m'} t/\alpha n})}_{\alpha} \ldots \\ & \ldots \; \underbrace{(e^{-iH_{m'+1} t/\beta n} \ldots e^{-iH_m t/\beta n}) \ldots (e^{-iH_{m'+1} t/\beta n} \ldots e^{-iH_m t/\beta n})}_{\beta} \Big)^n, \end{aligned}$$

to which we may apply the Lie-Trotter formula with respect to n to yield

$$\begin{aligned}\lim_{n\to\infty} f(n,\alpha,\beta) &= \exp\Bigg(\underbrace{(-iH_1t/\alpha - \cdots - iH_{m'}t/\alpha) + \cdots + (-iH_1t/\alpha \cdots - iH_{m'}t/\alpha)}_{\alpha} \\ &\quad + \underbrace{(-iH_{m'+1}t/\beta - \cdots - iH_mt/\beta) + \cdots + (-iH_{m'+1}t/\beta \cdots - iH_mt/\beta)}_{\beta}\Bigg) \\ &= e^{-\alpha(iH_1t/\alpha)-\cdots-\alpha(iH_{m'}t/\alpha)-\beta(iH_{m'+1}t/\beta)-\cdots-\beta(iH_mt/\beta)} = e^{-iHt}.\end{aligned}$$

This expression holds for arbitrary but fixed α and β. Now suppose we take $\alpha \to \infty$ while keeping n and β fixed. Then we have

$$\lim_{\alpha\to\infty} f(n,\alpha,\beta) = \left(e^{At/n}(e^{-iH_{m'+1}t/\beta n}\dots e^{-iH_mt/\beta n})^\beta\right)^n.$$

Taking now $n \to \infty$ yields

$$\lim_{n\to\infty}\lim_{\alpha\to\infty} f(n,\alpha,\beta) = e^{-iHt}.$$

All possible orderings of the limits with $n, \alpha, \beta \to \infty$ follow similarly. □

Error of the Strang Splitting Formula

Suzuki [213] provides error bounds for the Trotter formula, the Strang formula, and other high-order splitting formulas. We will build on the analysis of [240] to derive a useful bound for the error of the Strang splitting formula. We note that the original analysis of [240] contains a small error, which has been corrected in [242, App. A]. We also use results from [213].

We make frequent use of the inequality (e.g. [213, Lemma 1])

$$\|a^n - b^n\| \le n\|a-b\|(\max(\|a\|,\|b\|))^{n-1}, \tag{D.10}$$

for a, b elements of a Banach operator algebra and $n \in \mathbb{N}$. Also recall the identity for the commutator operator $\|[X,Y]\| \le 2\|X\|\|Y\|$, where $[X,Y] := XY - YX$.

Lemma 5 (Strang Splitting Formula Error)**.** *Let $n \in \mathbb{N}$, $t > 0$, and $\Delta t := t/n$. Let A, B be Hermitian matrices and $H = A + B$. Then*

$$\|e^{-iHt} - (S_2(A,B,\Delta t))^n\| \le \frac{1}{12}\|[[A,B],A+B]\|t\Delta t^2 \le \frac{2}{3}\|A\|\|B\|\|C\|t\Delta t^2,$$

where $\|C\| := \max\{\|A\|\,\|B\|\}$.

Proof. Let $H(x) := (1-x)A\Delta t + B\Delta t$, $0 \le x \le 1$. Then from [240, Appendix B] we have

$$\|[[A\Delta t, H(x)], H(x)]\| \le (\|[[A,B],A]\| + \|[[A,B],B]\|)\,\Delta t^3.$$

Extending the analysis of [240, Appendix B], we get

$$\begin{aligned}
\|e^{-i(A+B)\Delta t} - e^{-iA\Delta t/2}e^{-iB\Delta t}e^{-iA\Delta t/2}\| &\le \int_0^1 \left\| \int_0^1 \frac{s-s^2}{2}[[A\Delta t, H(x)], H(x)]\, ds \right\| dx \\
&= \int_0^1 \frac{s-s^2}{2}\, ds \; \int_0^1 \left\| [[A\Delta t, H(x)], H(x)] \right\| dx \\
&\le \frac{1}{12}(\|[[A,B],A]\| + \|[[A,B],B]\|)\,\Delta t^3 \\
&\le \frac{1}{12}(4\|A\|^2\|B\| + 4\|A\|\|B\|^2)\Delta t^3.
\end{aligned}$$

For $\|C\| = \max\{\|A\|\; \|B\|\}$ this yields

$$\|e^{-i(A+B)\Delta t} - e^{-iA\Delta t/2}e^{-iB\Delta t}e^{-iA\Delta t/2}\| \le \frac{2}{3}\|A\|\|B\|\|C\|\Delta t^3.$$

Finally,

$$\|(e^{-i(A+B)\Delta t})^{t/\Delta t} - S_2(A,B,\Delta t)^{t/\Delta t}\| \le (t/\Delta t)\|e^{-i(A+B)\Delta t} - e^{-iA\Delta t/2}e^{-iB\Delta t}e^{-iA\Delta t/2}\|,$$

which follows from (D.10) with unitary a and b. □

Algorithm 2

We give the details of Algorithm 2, which generalizes Algorithm 1 by first applying a splitting formula of order $2k+1$; see Figure 4.1.

We apply the results of [186], which achieves improved bounds to the simulation error and cost by rescaling the Hamiltonians to have norm at most 1. Note that such rescalings are equivalent to rescalings of the simulation time. Indeed, for Hamiltonians A, B, $H = A + B$, and $\ell > 0$ we have $U(H/\ell, t) = e^{-i(H/\ell)t} = e^{-iH(t/\ell)} = U(H, t/\ell)$ and $S_{2k}(A/\ell, B/\ell, t) = S_{2k}(A, B, t/\ell)$, where the definition of S_{2k} is given in (D.3).

Proof of Proposition 3. Recall the preliminary analysis given in Section 4.2.3. Consider the Hamiltonian H as in (4.4, 4.5), partitioned into two groups $H = A + B = (H_1 + \cdots + H_{m'}) + (H_{m'+1} + \cdots + H_m)$. We have

$$U = e^{-iHt} = e^{-i(A+B)t} = \left(e^{-i(\frac{A}{\|C\|} + \frac{B}{\|C\|})/M}\right)^{M\|C\|t} = \left(e^{-i(A+B)(M\|C\|)^{-1}}\right)^{M\|C\|t},$$

for $\|C\| := \max\{\|A\|, \|B\|\}$, and the quantity $M > 1$ is sufficiently large and will be defined shortly. Also define $\|D\| := \min\{\|A\|, \|B\|\}$. Thus the (algorithm first-step) time slice size is $(M\|C\|)^{-1}$, and the number of (first-step) intervals is $\lceil M\|C\|t\rceil$. Let

$$n_0 := \lfloor M\|C\|t\rfloor, \qquad \delta := M\|C\|t - \lfloor M\|C\|t\rfloor$$

denote the integer and fractional parts of $M\|C\|t = n_0 + \delta$, respectively.

Recall that our problem is equivalent to simulating $H/\|C\|$ for time $t\|C\|$. Let

$$\widehat{U} \;\; := \;\; \left(S_{2k}\left(\frac{A}{\|C\|}, \frac{B}{\|C\|}, \frac{1}{M}\right)\right)^{n_0} S_{2k}\left(\frac{A}{\|C\|}, \frac{B}{\|C\|}, \frac{\delta}{M}\right) \tag{D.11}$$

Unwinding the recurrence (D.3) defining S_{2k} for two Hamiltonians X and Y and $\tau \in \mathbb{R}$ yields [186, 248]

$$S_{2k}(X, Y, \tau) = \prod_{\ell=1}^{K} S_2(X, Y, z_\ell \tau),$$

where $K = 5^{k-1}$ and each z_ℓ is defined according to the recursive scheme of (D.3), $\ell = 1, \dots, K$. In particular, each z_ℓ is given as product of $k-1$ factors as $z_\ell = \prod_{r\in I_p} p_r \prod_{r\in I_q} q_r$, where the products are over the index sets I_p, I_q defined by traversing the path of the recursion tree corresponding to ℓ, and $\sum_{\ell=1}^{K} |z_\ell| = 1$; see [186, Sec. 3] for details. Recall that in Section D.1 we have defined the quantities $p_k = (4 - 4^{1/(2k-1)})^{-1})$ and $q_k = 1 - 4p_k$, for $k \in \mathbb{N}$.

Let $\widetilde{U}_{\mathcal{A}}(\tau)$ and $\widetilde{U}_{\mathcal{B}}(\tau)$ be approximations to $e^{-i(A/\|C\|)\tau}$ and $e^{-i(B/\|C\|)\tau}$, respectively, where $\mathcal{A} = A/\|C\|$ and $\mathcal{B} = B/\|C\|$. We approximate $S_2(A/\|C\|, B/\|C\|, \tau)$ by

$$\widetilde{S}_2(A/\|C\|, B/\|C\|, \tau) := \widetilde{U}_{\mathcal{A}}(\tau/2)\widetilde{U}_{\mathcal{B}}(\tau)\widetilde{U}_{\mathcal{A}}(\tau/2),$$

and this yields

$$\widetilde{S}_{2k}(A/\|C\|, B/\|C\|, \tau) := \prod_{\ell=1}^{K} \widetilde{S}_2(A/\|C\|, B/\|C\|, z_\ell\tau) = \prod_{\ell=1}^{K} \widetilde{U}_{\mathcal{A}}(z_\ell\tau/2)\widetilde{U}_{\mathcal{B}}(z_\ell\tau)\widetilde{U}_{\mathcal{A}}(z_\ell\tau/2).$$

Hence, applying the above to (D.11) we get

$$\begin{aligned}
\widetilde{U} \;\; &:= \;\; (\widetilde{S}_{2k}(A/\|C\|, B/\|C\|, 1/M))^{n_0}\widetilde{S}_{2k}(A/\|C\|, B/\|C\|, \delta/M) \\
&= \;\; \left(\prod_{\ell=1}^{K} \widetilde{S}_2(A/\|C\|, B/\|C\|, z_\ell/M)\right)^{n_0} \prod_{\ell=1}^{K} \widetilde{S}_2(A/\|C\|, B/\|C\|, z_\ell\delta/M) \\
&= \;\; \prod_{\ell'=1}^{K\lceil M\|C\|t\rceil} \widetilde{U}_{\mathcal{A}}(z_{\ell'}/2M)\widetilde{U}_{\mathcal{B}}(z_{\ell'}/M)\widetilde{U}_{\mathcal{A}}(z_{\ell'}/2M), \qquad (D.12)
\end{aligned}$$

where in the last equation we have re-indexed the product so that $z_{\ell'} = z_{((\ell' \bmod \mathrm{K})+1)}$ for $1 \leq \ell' \leq n_0 K$, and $z_{\ell'} = z_{((\ell' \bmod \mathrm{K})+1)}\delta$ for $n_0 K < \ell' \leq (n_0+1)K$. The overall term ordering and time interval sizes are easily computable from (D.3) and (D.12). Thus $\widetilde{U}$ is an ordered product of $(3K\lceil M\|C\|t\rceil)$-many applications of $\widetilde{U}_{\mathcal{A}}$ and $\widetilde{U}_{\mathcal{B}}$ (each applied for differing simulation times).

We now turn to the second step splitting formulas, i.e., the ones approximating $e^{-i(A/\|C\|)\tau}$ and $e^{-i(B/\|C\|)\tau}$ for $\tau \in \mathbb{R}$. We apply Suzuki's high-order splitting formulas, with different orders in principle.

Once more simulating $\mathcal{A} = A/\|C\|$ for time τ is equivalent to simulating A for time $\tau/\|C\|$, and this is further equivalent to simulating $A/\|H_1\|$ for time $\tau\|H_1\|/\|C\|$. Thus we define

$$\mathcal{H}_j := \begin{cases} \frac{H_j/\|C\|}{\|H_1/\|C\|\|} = \frac{H_j}{\|H_1\|} & (1 \leq j \leq m') \\ \frac{H_j/\|C\|}{\|H_{m'+1}/\|C\|\|} = \frac{H_j}{\|H_{m'+1}\|} & (m' < j \leq m). \end{cases}$$

Thus we obtain

$$\widetilde{U}_{\mathcal{A}}(z_\ell/2M) := S_{2k_A}(\mathcal{H}_1,\ldots,\mathcal{H}_{m'},1/M_A)^{\lfloor(|z_\ell|/2M)(M_A\|H_1\|/\|C\|)\rfloor} S_{2k_A}(\mathcal{H}_1,\ldots,\mathcal{H}_{m'},\delta_A/M_A), \tag{D.13}$$

$$\widetilde{U}_{\mathcal{B}}(z_\ell/M) := S_{2k_B}(\mathcal{H}_{m'+1},\ldots,\mathcal{H}_m,1/M_B)^{\lfloor M_B\|H_{m'+1}\||z_\ell|/M\|C\|\rfloor} S_{2k_B}(\mathcal{H}_{m'+1},\ldots,\mathcal{H}_m,\delta_B/M_B), \tag{D.14}$$

where $\delta_A := M_A\|H_1\||z_\ell|/2M\|C\| - \lfloor M_A\|H_1\||z_\ell|/2M\|C\|\rfloor$ and $\delta_B := M_B\|H_{m'+1}\||z_\ell|/M\|C\| - \lfloor M_B\|H_{m'+1}\||z_\ell|/M\|C\|\rfloor$, and $M_A, M_B > 1$. As before, the quantities $\lceil M_A\|H_1\||z_\ell|/2M\|C\|\rceil$ and $\lceil M_B\|H_{m'+1}\||z_\ell|/M\|C\|\rceil$ give the number of subintervals used to further subdivide intervals of length $z_\ell/2M_A$ and z_ℓ/M_B, respectively. We define M_A, M_B below. The reader may wish to recall the text after (4.22,4.23) that deals with the calculation of the number of subintervals and their lengths.

Error and Cost

Using (D.11) and (D.12), we bound the overall error by

$$\|U - \widetilde{U}\| \leq \|U - \widehat{U}\| + \|\widehat{U} - \widetilde{U}\|.$$

The first term in the right-hand side corresponds to the error of a splitting formula at the first step of the algorithm, where we pretend that exponentials $e^{-iA\tau}$, $e^{-iB\tau}$, $\tau \in \mathbb{R}$ are given to us exactly, and the second term corresponds to the error in the second step of the algorithm, i.e. the error introduced by splitting formulas approximating $e^{-iA\tau}$ and $e^{-iB\tau}$.

As explained in Section 4.2.3, to guarantee $\|U - \widehat{U}\| \leq \varepsilon/2$ we set the quantity M as in (4.17), which gives

$$M = M(k) := \left(\frac{16et\|D\|}{\varepsilon}\right)^{1/2k} \frac{8e}{3}\left(\frac{5}{3}\right)^{k-1}.$$

To apply [186, Thm. 1] for $H = A + B$ and accuracy $\varepsilon/2$, the condition of that theorem becomes

$$16et\|D\| \geq \varepsilon, \tag{D.15}$$

which implies $M \geq 1$. Hence, the number of S_{2k} comprising $\widehat{U}$ in (D.11) is at most

$$3 \cdot 5^{k-1}\lceil M\|C\|t\rceil,$$

where $\lceil M\|C\|t\rceil$ gives the number of time intervals at the first step. The interesting case is $M\|C\|t \geq 1$, for which it suffices to assume $\|C\|t \geq 1$ (otherwise, as explained in the analysis of Algorithm 1, we would be dealing with an easy problem). Then the above quantity may be further bounded by $3 \cdot 5^{k-1} 2M\|C\|t$. Let N_A and N_B be upper bounds to the number of exponentials comprising $\widetilde{U}_{\mathcal{A}}(z_\ell/2M)$ and $\widetilde{U}_{\mathcal{B}}(z_\ell/M)$, respectively, for any ℓ. Then the resulting total number of exponentials in Algorithm 2 (in $\widetilde{U}$) is

$$N \leq (2N_A + N_B)5^{k-1}2M\|C\|t. \tag{D.16}$$

In order to obtain estimates to N_A and N_B we turn to the second-step error, where we require $\|\widehat{U} - \widetilde{U}\| \leq \varepsilon/2$. We have

$$\begin{aligned} \|\widehat{U} - \widetilde{U}\| &= \| S_{2k}(A/\|C\|, B/\|C\|, 1/M)^{n_0} S_{2k}(A/\|C\|, B/\|C\|, \delta/M) - \\ &\qquad \widetilde{S}_{2k}(A/\|C\|, B/\|C\|, 1/M)^{n_0} \widetilde{S}_{2k}(A/\|C\|, B/\|C\|, \delta/M) \| \\ &\leq n_0\|S_{2k}(A, B, 1/M) - \widetilde{S}_{2k}(A, B, 1/M)\| \\ &+ \|S_{2k}(A, B, \delta/M) - \widetilde{S}_{2k}(A, B, \delta/M)\|. \end{aligned} \tag{D.17}$$

Observe the quantity $\|S_{2k}(A, B, 1/M) - \widetilde{S}_{2k}(A, B, 1/M)\|$ is equal to

$$\left\| \prod_{\ell=1}^{K} S_2(A, B, z_\ell/M) - \prod_{\ell=1}^{K} \widetilde{U}_A(z_\ell/2M)\widetilde{U}_B(z_\ell/2M)\widetilde{U}_A(z_\ell/M) \right\|$$

which is at most

$$2\sum_{\ell=1}^{K} \|e^{-iAz_\ell/2M} - \widetilde{U}_A(z_\ell/2M)\| + \sum_{\ell=1}^{K} \|e^{-iBz_\ell/M} - \widetilde{U}_B(z_\ell/M)\|.$$

The $\widetilde{U}_\mathcal{A}$ and $\widetilde{U}_\mathcal{B}$ are given by splitting formulas over time intervals of size $z_\ell/2M$, z_ℓ/M, $z_\ell\delta/2M$ and $z_\ell\delta/M$, which vary with the z_ℓ. Since $\delta < 1$ we bound the second term in (D.17) to get

$$\|\widehat{U} - \widetilde{U}\| \le (n_0+\delta)\left(2\sum_{\ell=1}^{K} \|e^{-iAz_\ell/2M} - \widetilde{U}_A(z_\ell/2M)\| + \sum_{\ell=1}^{K} \|e^{-iBz_\ell/M} - \widetilde{U}_B(z_\ell/M)\|\right). \tag{D.18}$$

Thus, sufficient conditions for $\|\widehat{U} - \widetilde{U}\| \le \varepsilon/2$ are

$$\sup_{1\le\ell\le K} \|e^{-iAz_\ell/2M} - \widetilde{U}_A(z_\ell/2M)\| \le \frac{\varepsilon}{8(n_0+\delta)K} = \frac{\varepsilon}{8M\|C\|t5^{k-1}}\ ,$$
$$\sup_{1\le\ell\le K} \|e^{-iBz_\ell/2M} - \widetilde{U}_B(z_\ell/2M)\| \le \frac{\varepsilon}{4(n_0+\delta)K} = \frac{\varepsilon}{4M\|C\|t5^{k-1}}\ .$$

We next explain how to select the subintervals for applying $\widetilde{U}_\mathcal{A}$ and $\widetilde{U}_\mathcal{B}$, keeping in mind that we eventually select the same values of M_A and M_B in all resulting time intervals due to the upper bounds (D.20, D.22) below. Note that selecting M_A or M_B to be larger than necessary can only reduce the simulation error. Thus, for convenience, we select M_A and M_B uniformly and large enough so that the resulting worst-case errors are sufficiently small.

In particular, consider $\widetilde{U}_\mathcal{A}(z_\ell/2M)$ which approximately simulates $A/\|C\|$ for time $\frac{z_\ell}{2M}$. This amount of time we further subdivide in $M_A(z_\ell/2M)$ slices. From [186], the error will be at most $\frac{\varepsilon}{8M\|C\|tK}$ if using (D.5) we set

$$\begin{aligned} M_A\left(\frac{z_\ell}{2M}\right) \quad &:= \quad \left(\frac{4em'(|z_\ell|/2M)\|H_2\|/\|C\|}{(\varepsilon/8M\|C\|tK)}\right)^{1/2k_A} \frac{4em'}{3}\left(\frac{5}{3}\right)^{k_A-1} \\ &= \quad \left(\frac{16eK|z_\ell|m't\|H_2\|}{\varepsilon}\right)^{1/2k_A} \frac{4em'}{3}\left(\frac{5}{3}\right)^{k_A-1}. \end{aligned}$$

Importantly, observe that the factors of M and $\|C\|$ have canceled. Using that [248, App. A]

$$\frac{1}{3^{k-1}} \le\ |z_\ell|\ \le \frac{4k}{3^k},$$

we have

$$\left(\frac{5}{3}\right)^{k-1} \le |z_\ell|K \le \frac{4}{5}k\left(\frac{5}{3}\right)^k. \tag{D.19}$$

Therefore

$$M_A(z_\ell/2M) \le \left(\frac{64e}{5}k\frac{m't\|H_2\|}{\varepsilon}\right)^{1/2k_A}\frac{4em'}{3}\left(\frac{5}{3}\right)^{k_A-1+k/2k_A} =: M_A, \tag{D.20}$$

for all ℓ. Hence, we will split every time interval of size $z_\ell/2M$, $\ell = 1,\dots,K$ into M_A subintervals.

To bound the cost of each $\widetilde{U}_{\mathcal{A}}(z_\ell/2M)$, we apply [186, Thm. 2]. The theorem assumes $4em'\left(\frac{|z_\ell|}{2M}\right)\frac{\|H_2\|}{\|C\|} \ge \frac{\varepsilon}{8M\|C\|tK}$, or equivalently $16em'\|H_2\|tK|z_\ell| \ge \varepsilon$, where again the M and $\|C\|$ factors have canceled. Since in the statement of the proposition we have assumed that

$$16em'\|H_2\|t \ge \varepsilon, \tag{D.21}$$

we can apply [186, Thm. 2]. Hence, the number of exponentials for each $\widetilde{U}_{\mathcal{A}}(z_\ell/2M)$, $\ell = 1,\dots,K$, is at most

$$(2m'-1)5^{k_A-1}\left\lceil M_A\frac{\|H_1\|}{\|C\|}\frac{|z_\ell|}{2M}\right\rceil \le 2m'5^{k_A-1}\left\lceil\frac{M_A}{M}\frac{\|H_1\|}{\|C\|}\frac{2k}{3^k}\right\rceil =: N_A.$$

Note that the argument of this ceiling function may be greater than or less than one, depending on the problem instance and algorithm parameters. In the latter case, the time intervals of length $z_\ell/2M$ do not require any subdivision at all.

We now consider $\widetilde{U}_{\mathcal{B}}(z_\ell/M)$ which approximately simulates $B/\|C\|$ for time z_ℓ/M, and proceed similarly. to give error at most $\frac{\varepsilon}{4M\|C\|tK}$ we select M_B from (D.5) to give

$$\begin{aligned} M_B(z_\ell/M) &= \left(\frac{4em'(|z_\ell|/M)\|H_{m'+2}\|/\|C\|}{(\varepsilon/4M\|C\|tK)}\right)^{1/2k_B}\frac{4e(m-m')}{3}\left(\frac{5}{3}\right)^{k_B-1} \\ &= \left(\frac{16eK|z_\ell|(m-m')t\|H_{m'+2}\|}{\varepsilon}\right)^{1/2k_B}\frac{4e(m-m')}{3}\left(\frac{5}{3}\right)^{k_B-1} \\ &\le \left(\frac{64e}{5}k\frac{(m-m')t\|H_{m'+2}\|}{\varepsilon}\right)^{1/2k_B}\frac{4e(m-m')}{3}\left(\frac{5}{3}\right)^{k_B-1+k/2k_B}. \end{aligned} \tag{D.22}$$

We define M_B to be the right-hand side of the final equation. Observe that the factors of M and $\|C\|$ have again canceled, and M_B is of the same form as M_A.

To apply [186, Thm. 2] to bound the cost of any $\widetilde{U}_{\mathcal{B}}(z_\ell/M)$, we require

$$4e(m-m')\left(\frac{|z_\ell|}{M}\right)\frac{\|H_{m'+2}\|}{\|C\|} \ge \frac{\varepsilon}{4M\|C\|tK},$$

or equivalently, $16e(m-m')\|H_{m'+2}\|tK|z_\ell| \ge \varepsilon$, which is valid because we have assumed that

$$16e(m-m')\|H_{m'+2}\|t \ge \varepsilon. \tag{D.23}$$

The number of exponentials for each $\widetilde{U}_{\mathcal{B}}(z_\ell/M)$ is at most

$$(2(m-m')-1)5^{k_B-1}\left\lceil M_B \frac{\|H_{m'+1}\|}{\|C\|}\frac{|z_\ell|}{M}\right\rceil \le 2m'5^{k_A-1}\left\lceil \frac{M_B}{M}\frac{\|H_{m'+1}\|}{\|C\|}\frac{4k}{3^k}\right\rceil =: N_B.$$

Thus, from (D.16) we have that the total cost (total number of exponentials) is at most

$$\begin{aligned} N &\le 2M\|C\|t5^{k-1}(2N_A+N_B) \\ &\le 2M\|C\|t\,5^{k-1}\left(4m'5^{k_A-1}\left\lceil\frac{M_A}{M}\frac{\|H_1\|}{\|C\|}\frac{2k}{3^k}\right\rceil + 2(m-m')5^{k_B-1}\left\lceil\frac{M_B}{M}\frac{\|H_{m'+1}\|}{\|C\|}\frac{4k}{3^k}\right\rceil\right). \end{aligned}$$

Letting

$$\widetilde{n} = n_0+\delta = M\|C\|t = \|C\|t\left(\frac{16et\|D\|}{\varepsilon}\right)^{1/2k}\frac{8e}{5}\left(\frac{5}{3}\right)^k,$$

which is equal to the lower bound for n as it appears in the statement of the proposition, and

$$\widetilde{n}_A = M_A\|H_1\|t\frac{2k}{3^k}, \qquad \widetilde{n}_B = M_B\|H_{m'+1}\|t\frac{4k}{3^k},$$

the cost bound becomes (cf. (4.30))

$$N \le\ 8m'5^{k+k_A-2}\widetilde{n}\left\lceil\frac{\widetilde{n}_A}{\widetilde{n}}\right\rceil + 4(m-m')5^{k+k_B-2}\,\widetilde{n}\left\lceil\frac{\widetilde{n}_B}{\widetilde{n}}\right\rceil.$$

Again applying the inequality $x\lceil y/x\rceil \le \max\{x,2y\}$ for $x,y>0$, we have

$$N \le\ 8m'5^{k+k_A-2}\max\{\widetilde{n},2\widetilde{n}_A\} + 4(m-m')5^{k+k_B-2}\max\{\widetilde{n},2\widetilde{n}_B\}.$$

Finally, we use the inequality

$$k^{1/2k'}(5/3)^{k/2k'}k/3^k \le (35/16)(3/5)^k \qquad \text{for } k,k'\in\mathbb{N} \tag{D.24}$$

to define simpler quantities n_A and n_B as

$$\begin{aligned} 2\widetilde{n}_A &= 2\|H_1\|tM_A\frac{2k}{3^k} = \|H_1\|t\left(\frac{64e}{5}k\frac{m't\|H_2\|}{\varepsilon}\right)^{1/2k_A}\frac{16em'}{3}\left(\frac{5}{3}\right)^{k_A-1+k/2k_A}\frac{k}{3^k} \\ &\le m'\|H_1\|t\left(\frac{64e}{5}\frac{m't\|H_2\|}{\varepsilon}\right)^{1/2k_A} 7e\left(\frac{5}{3}\right)^{k_A-k} =: n_A(k,k_A), \end{aligned}$$

and

$$\begin{aligned} 2\widetilde{n}_B &= 2\|H_{m'+1}\|tM_B\frac{4k}{3^k} \\ &= \|H_{m'+1}\|t\left(\frac{64e}{5}k\frac{(m-m')t\|H_{m'+2}\|}{\varepsilon}\right)^{1/2k_B}\frac{32e(m-m')}{3}\left(\frac{5}{3}\right)^{k_B-1+k/2k_B}\frac{k}{3^k} \\ &\le (m-m')\|H_{m'+1}\|t\left(\frac{64e}{5}\frac{(m-m')t\|H_{m'+2}\|}{\varepsilon}\right)^{1/2k_B} 14e\left(\frac{5}{3}\right)^{k_B-k} =: n_B(k,k_B). \end{aligned}$$

Applying these estimates, the cost bound becomes

$$N \leq\ 8m' 5^{k+k_A-2} \max\{\widetilde{n}, n_A\} + 4(m-m')5^{k+k_B-2}\max\{\widetilde{n}, n_B\}. \tag{D.25}$$

Clearly this inequality remains valid replacing $\widetilde{n}$ by any n such that $n \geq \widetilde{n}$. $\square$

Proof of Theorem 2

As the analysis is similar to that of Proposition 3, we give only the important parts. Recall the preliminary analysis given in Section 4.2.3.

Consider a Hamiltonian as in (4.4, 4.5), partitioned into μ groups $H = A_1 + \cdots + A_\mu$ as in Section 4.2.3, labeled such that $\|A_1\| \geq \|A_2\| \geq \cdots \geq \|A_\mu\|$. We approximate $U = e^{-Ht}$ with $\widetilde{U}$ given in (4.13), i.e.

$$\widetilde{U} := (\widetilde{S}_{2k}(A_1, \ldots, A_\mu, t/n))^n = \left(\prod_{\ell=1}^{N_{k,\mu}} \widetilde{U}_{A_{j_\ell}}(t_\ell/n)\right)^n, \qquad j_\ell \in \{1, \ldots, \mu\}, \quad \sum_{\ell=1}^{N_{k,\mu}} t_\ell = \mu t.$$

For the first-step error to be at most $\varepsilon/2$, we set $n = M\|A_1\|t$ with M given as in (4.17), i.e.

$$M = \left(\frac{8e\mu t\|A_2\|}{\varepsilon}\right)^{1/2k} \frac{4e\mu}{3}\left(\frac{5}{3}\right)^{k-1},$$

where in the statement of the theorem we have assumed $\mu t\|A_2\| \geq \varepsilon$. Note that we do not require $n \in \mathbb{N}$; however, as evident from (D.16) and (D.18), this assumption does not affect our analysis.

From (4.18) the second-step error is at most $\varepsilon/2$ if the error of each subroutine satisfies

$$\|e^{-iA_{j_\ell}t_\ell/n} - \widetilde{U}_{A_{j_\ell}}(t_\ell/n)\| \leq \frac{\varepsilon}{4\mu 5^{k-1}n}.$$

Thus for each $j = 1, \ldots, \mu$ we select

$$\begin{aligned} M_{A_j}\left(\frac{z_\ell}{2M}\right) &= \left(\frac{4em_j(|z_\ell|/2M)\|H_{(j,2)}\|/\|A_1\|}{(\varepsilon/4\mu M\|A_1\|tK)}\right)^{1/2k_j} \frac{4em_j}{3}\left(\frac{5}{3}\right)^{k_j-1} \\ &= \left(\frac{8e\mu K|z_\ell|m_j t\|H_{(j,2)}\|}{\varepsilon}\right)^{1/2k_j} \frac{4em_j}{3}\left(\frac{5}{3}\right)^{k_j-1} \\ &\leq \left(\frac{32e}{5}k\frac{\mu m_j t\|H_{(j,2)}\|}{\varepsilon}\right)^{1/2k_j} \frac{4em_j}{3}\left(\frac{5}{3}\right)^{k_j-1+k/2k_j} =: M_{A_j}, \end{aligned}$$

for all $\ell = 1, \ldots, K$, where in the statement of the theorem we have assumed $\mu m_j t\|H_{(j,2)}\| \geq \varepsilon$, $j = 1, \ldots, \mu$. Hence, the number of exponentials in $\widetilde{U}_{\mathcal{A}_j}(z_\ell/2M)$ is at most

$$(2m_j - 1)5^{k_j-1}\left\lceil M_{A_j}\frac{\|H_{(j,1)}\|}{\|A_1\|}\frac{|z_\ell|}{2M}\right\rceil \leq 2m_j 5^{k_j-1}\left\lceil \frac{M_{A_j}}{M}\frac{\|H_{(j,1)}\|}{\|A_1\|}\frac{2k}{3^k}\right\rceil =: N_{A_j}.$$

Recalling (4.19), from (D.16) we have that the total cost is at most

$$\begin{aligned} N &\leq 2n5^{k-1}\sum_{j=1}^{\mu} 2N_{A_j} \leq 2n5^{k-1}\sum_{j=1}^{\mu} 4m_j 5^{k_j-1}\left\lceil \frac{M_{A_j}}{M}\frac{\|H_{(j,1)}\|}{\|A_1\|}\frac{2k}{3^k}\right\rceil \\ &\leq 8\sum_{j=1}^{\mu} m_j 5^{k+k_j-2} n\left\lceil\frac{n_j}{n}\right\rceil \leq 8\sum_{j=1}^{\mu} m_j 5^{k+k_j-2}\max\{n, 2n_j\}, \end{aligned}$$

where $n_j := M_{A_j}\|H_{(j,1)}\|t\ 2k/3^k$.

We again apply (D.24) to give the simpler quantities

$$\begin{aligned} 2n_j &= 2M_{A_j}\|H_{(j,1)}\|t\frac{2k}{3^k} = \|H_{(j,1)}\|t\left(\frac{32e}{5}k\frac{\mu m_j t\|H_{(j,2)}\|}{\varepsilon}\right)^{1/2k_j}\frac{16em'}{3}\left(\frac{5}{3}\right)^{k_A-1+k/2k_j}\frac{k}{3^k} \\ &\leq m_j\|H_{(j,1)}\|t\left(\frac{32e}{5}\frac{\mu m_j t\|H_{(j,2)}\|}{\varepsilon}\right)^{1/2k_j} 7e\left(\frac{5}{3}\right)^{k_j-k} =: n_{A_j}(k,k_j), \end{aligned}$$

which gives the cost bound (4.36), i.e.,

$$N \leq 8\sum_{j=1}^{\mu} 5^{k+k_j-2} m_j \max\{n(k), n_{A_j}(k,k_j)\} =: \eta(k,k_1,\ldots,k_\mu). \tag{D.26}$$

Proof of (4.47)

Proof. Consider all problem parameters to be fixed except m, m'. Observe that (4.32) contains two maximum functions, and hence we have four cases to consider with respect to the relative magnitudes of $n(k)$, $n_A(k,k_A)$, and $n_B(k,k_B)$. Recall $n(k)$ is given in (4.44). Here we estimate the maximum function by the sum of its arguments to get

$$\begin{aligned} \eta(k,k_A,k_B) &\leq 8m'5^{k+k_A-2}\left(n_A(k,k_A)+n(k)\right) + 4(m-m')5^{k+k_B-2}\left(n_B(k,k_B)+n(k)\right) \\ &\leq 8m'5^{k+k_A-2}\, n_A(k,k_A) + 4(m-m')5^{k+k_B-2}\, n_B(k,k_B) \\ &+ (8m'5^{k+k_A-2} + 4(m-m')5^{k+k_B-2})n(k). \end{aligned}$$

Let η^* denote the minimum of (4.32) with respect to k, k_A, k_B. Let $k^{(max)}, k_A^{(max)}, k_B^{(max)}$ be defined as in Proposition 4 under the assumptions of Proposition 3. Using $\eta^* \leq \eta(k^{(max)}, k_A^{(max)}, k_B^{(max)})$, this gives

$$\begin{aligned}
N \le \eta^* &\le 8m' 5^{k^{(max)}+k_A^{(max)}-2}\, n_A(k^{(max)}, k_A^{(max)}) \\
&+ 4(m-m')5^{k^{(max)}+k_B^{(max)}-2}\, n_B(k^{(max)}, k_B^{(max)}) \\
&+ (8m' 5^{k^{(max)}+k_A^{(max)}-2} + 4(m-m')5^{k^{(max)}+k_B^{(max)}-2})\, n(k^{(max)}) \\
&= 3^{k^{(max)}} O\left(m'^2 \|H_1\| t\right) \cdot e^{2\sqrt{\frac{1}{2}\ln\frac{25}{3}\ln(\frac{64e}{5} m' t \|H_2\|/\varepsilon)}} \\
&+ 3^{k^{(max)}} O\left((m-m')^2 \|H_{m'+1}\| t\right) \cdot e^{2\sqrt{\frac{1}{2}\ln\frac{25}{3}\ln(\frac{64e}{5}(m-m') t \|H_{m'+2}\|/\varepsilon)}} \\
&+ (5^{k_A^{(max)}} + 5^{k_B^{(max)}})\, O((m-m')m' \|H_1\| t) \cdot e^{2\sqrt{\frac{1}{2}\ln\frac{25}{3}\ln(16e(m-m') t \|H_{m'+1}\|/\varepsilon)}}.
\end{aligned}$$

The quantities under the square roots are derived as in [186].

Next observe that from (4.43) we have $m'\|H_2\| \ge (m-m')\|H_{m'+1}\| \ge (m-m')\|H_{m'+2}\|$ and $(m-m')\|H_{m'+1}\| = O(m'\|H_1\|)$. Using the bound $a^{\sqrt{b}} \le \sqrt{a^{b+1}}$ for $a, b \ge 1$ we have

$$\max\{3^{k^{(max)}}, 5^{k_A^{(max)}}, 5^{k_B^{(max)}}\} \le \sqrt{5^{1+\frac{1}{2}\log_{25/3}(16em't\|H_2\|/\varepsilon)}} \le \sqrt{5}\; (16em't\|H_2\|/\varepsilon)^{0.2},$$

which gives

$$N \le \eta^* \;=\; O((m'\|H_2\|t/\varepsilon)^{0.2}) \cdot O\left(mm'\|H_1\|t\right) \cdot e^{2\sqrt{\frac{1}{2}\ln\frac{25}{3}\ln(16em't\|H_2\|/\varepsilon)}}.$$

Hence, using N^*_{prev} as defined in (D.9) we have

$$\frac{N}{N^*_{prev}} \le \frac{\eta^*}{N^*_{prev}} \;=\; O((m'\|H_2\|t/\varepsilon)^{0.2}) \cdot O\left(\frac{m'}{m}\right) \cdot \frac{e^{2\sqrt{\frac{1}{2}\ln\frac{25}{3}\ln(16em't\|H_2\|/\varepsilon)}}}{e^{2\sqrt{\frac{1}{2}\ln\frac{25}{3}\ln(4emt\|H_2\|/\varepsilon)}}}.$$

Observe that for $a < b$, the function $e^{a\sqrt{x}-b\sqrt{x}} \to 0$ as $x \to \infty$. Thus, assuming $m' = O(m^{5/6})$, we have

$$\frac{N}{N^*_{prev}} \le \frac{\eta^*}{N^*_{prev}} \xrightarrow[m\to\infty]{} 0. \tag{D.27}$$

□

Appendix E

Quantum Approximate Optimization

In this appendix we provide proofs for several results from Chapter 5.

Proof of Theorem 5. For a D-regular triangle free graph, setting partial derivatives of (5.16) to zero shows that $\tan^2\gamma = \frac{1}{D-1}$ for γ to be stationary, and thus every optimal γ^* is of the form $\pm\arctan\frac{1}{\sqrt{D-1}} + \ell\pi$ for some $\ell \in \mathbb{Z}$. In particular, $\gamma = \ell\pi + \arctan\frac{1}{\sqrt{D-1}}$ is optimal when $\sin(4\beta^*) = 1$, and $\gamma = \ell\pi - \arctan\frac{1}{\sqrt{D-1}}$ is optimal when $\sin(4\beta^*) = -1$. Taking second derivatives shows $(\gamma^*, \beta^*) = (\arctan\frac{1}{\sqrt{D-1}}, \pi/8)$ indeed gives the smallest maximum.

To classify the optimal angles for this case, observe that (5.16) may be written as a $\langle C\rangle(\gamma,\beta) = \frac{m}{2} + f(\gamma)g(\beta)$, where $f(\gamma)$ and $g(\beta)$ are odd periodic functions. Thus $\langle C\rangle$ is maximized when both f,g are maximized, or when both f,g are minimized. Indeed, the transformations $\beta \to -\beta^*$ and $\gamma \to -\gamma$ flip the signs of $g(\beta)$ and $f(\gamma)$, and hence $(-\gamma^*, -\beta^*)$ is also optimal. Moreover, as $\sin(4\beta)$ is $\frac{\pi}{2}$-periodic, the pair $(\gamma^*, \beta^* + \ell\frac{\pi}{2})$ is also optimal for any $\ell \in \mathbb{Z}$. Combining these facts with the observation that $f(\gamma)$ is π-periodic when D is even and 2π-periodic when D is odd gives the stated result.

Finally, observe that $\sum_D D\ n_D = 2m$ for any graph with m edges and n_D vertices of degree D. As $Dn_D \geq 0$, we may write (5.19) as a convex combination of (5.16) for different values of $D = d+1$, from which the third point follows from the second. $\square$

Proof of Theorem 8. We compute $\langle C\rangle$ using the Pauli Solver algorithm of Section 5.4.2, similarly to the proof of Theorem 4 for undirected MaxCut. We consider the general case first.

Recall $D_u \subset D$ is the subset of directed edges containing u, and $U_u \subset U$ are the 'undirected' edges containing u, with $d_u = |D_u|$ and $e_u = |U_u|$. For convenience, we will write $(uv) \in D$ to indicate that *one* of $|uv) \in D$ or $|vu) \in D$. From (5.30), let C_u denote the terms in C that contain vertex u, given by

$$C_u := \frac{1}{4}k_u Z_u - \frac{1}{4}\sum_{(uw)\in D_u} Z_u Z_w - \frac{1}{2}\sum_{(ut)\in U_u} Z_t Z_u. \tag{E.1}$$

Note that with this definition $C \neq \sum_u C_u$.

Let $c = \cos 2\beta$ and $s = \sin 2\beta$. For the single Z terms in (5.30), observe that

$$e^{i\beta B} Z_u e^{-i\beta B} = e^{2i\beta X_u} Z_u = (Ic + isX_u)Z_u = cZ_u + sY_u.$$

Let $c' = \cos\gamma$, $s' = \sin\gamma$, $c'' = \cos\frac{\gamma}{2}$, and $s'' = \sin\frac{\gamma}{2}$, and for each vertex u let $c'_u = \cos\frac{\gamma k_u}{2}$ and $s'_u = \sin\frac{\gamma k_u}{2}$. The Z_u term above commutes with $e^{i\gamma C}$ and thus contributes nothing to the expectation value of C (for $p = 1$). The Y_u term anti-commutes with each term in C_u, and commutes with the remaining terms in C, so we have

$$\begin{aligned}
\langle s|e^{i\gamma C} Y_u e^{-i\gamma C}|s\rangle &= \langle s|e^{2i\gamma C_u} Y_u|s\rangle \\
&= \langle s|e^{\frac{i}{2}\gamma k_u Z_u} e^{-\frac{i}{2}\gamma\sum_{(uw)\in D_u} Z_u Z_w} e^{-i\gamma\sum_{(uw)\in U_u} Z_u Z_w} Y_u|s\rangle, \\
&= \langle s|(Ic'_u + is'_u Z_u)\prod_{i=1}^{d_u}(Ic'' - is'' Z_u Z_{w_i})\prod_{j=1}^{e_u}(Ic' - is' Z_u Z_{w_j})Y_u|s\rangle.
\end{aligned}$$

Expanding the product on the right hand side of the last line again gives a sum of tensor products of Pauli operators. Recall that only terms not containing a Y or a Z factor contribute to $\langle C\rangle$, for initial state $|s\rangle = |+\rangle^{\otimes n}$. As the only vertex in common between the edges of D_u and U_u is u itself, the only term that can contribute is proportional to $Z_u * I^{\otimes d_u} * I^{\otimes e_u} * Y_u = -iX_u$, so we have

$$\langle s|e^{i\gamma C} Y_u e^{-i\gamma C}|s\rangle = \langle s|is'_u c''^{d_u} c'^{e_u}(-iX_u)|s\rangle = s'_u (c'')^{d_u}(c')^{e_u},$$

and hence

$$\langle s|e^{i\gamma C} e^{i\beta B} Z_u e^{-i\beta B} e^{-i\gamma C}|s\rangle = ss'_u (c'')^{d_u}(c')^{e_u}. \tag{E.2}$$

Turning to the ZZ terms in C in (5.30), we have

$$e^{i\beta B} Z_u Z_v e^{-i\beta B} = e^{2i\beta X_u} e^{2i\beta X_v} Z_u Z_v = c^2 Z_u Z_v + sc(Y_u Z_v + Z_u Y_v) + s^2 Y_u Y_v. \tag{E.3}$$

The first term $c^2 Z_u Z_v$ on the right commutes with $e^{i\gamma C}$ and contributes nothing to $\langle C\rangle$. We conjugate each remaining term in (E.3) separately by $e^{i\gamma C}$. We have

$$\begin{aligned}
\langle s|e^{i\gamma C} Y_u Z_v e^{-i\gamma C}|s\rangle &= \langle s|e^{2i\gamma C_u} Y_u Z_v|s\rangle \qquad \text{(E.4)}\\
&= \langle s|e^{\frac{i}{2}\gamma k_u Z_u} e^{-\frac{i}{2}\gamma \sum_{(uw)\in D_u} Z_u Z_w} e^{-i\gamma \sum_{(uw)\in U_u} Z_u Z_w} Y_u Z_v|s\rangle\\
&= \langle s|(Ic'_u + is'_u Z_u)\prod_{i=1}^{d_u}(Ic'' - is'' Z_u Z_{w_i})\prod_{j=1}^{e_u}(Ic' - is' Z_u Z_{w_j}) Y_u Z_v|s\rangle,
\end{aligned}$$

for which we have two cases depending on if the edge (uv) is directed or undirected.

First suppose $(uv) \in D$, i.e., one of the edges $|uv)$ or $|vu)$ is in the graph, and $(uv) \notin U$. Then (E.4) becomes

$$\langle s|(Ic'_u + is'_u Z_u)(Ic'' - is'' Z_u Z_v)\prod_{i=1}^{d_u-1}(Ic'' - is'' Z_u Z_{w_i})\prod_{j=1}^{e_u}(Ic' - is' Z_u Z_{w_j}) Y_u Z_v|s\rangle.$$

Expanding the product again gives a sum of Pauli terms. Similar to the undirected case, the only term that can contribute is proportional to $I * Z_u Z_v * I^{\otimes d_u - 1} * I^{\otimes e_u} * Y_u Z_v = -iX_u$, so

$$\langle s|e^{i\gamma C} Y_u Z_v e^{-i\gamma C}|s\rangle = \langle s|c'_u(-is'')c''^{d_u-1}c'^{e_u}(-iX_u)|s\rangle = -s'' c'_u (c'')^{d_u-1}(c')^{e_u}. \qquad \text{(E.5)}$$

On the other hand, suppose instead $(uv) \in U$. Then $\langle s|e^{i\gamma C} Y_u Z_v e^{-i\gamma C}|s\rangle$ is given by

$$\langle s|(Ic'_u + is'_u Z_u)(Ic' - is' Z_u Z_v)\prod_{i=1}^{d_u}(Ic'' - is'' Z_u Z_{w_i})\prod_{j=1}^{e_u-1}(Ic' - is' Z_u Z_{w_j}) Y_u Z_v|s\rangle,$$

where by the previous argument we now have

$$\langle s|e^{i\gamma C} Y_u Z_v e^{-i\gamma C}|s\rangle = \langle s|c'_u(-is'')c''^{d_u-1}c'^{e_u}(-iX_u)|s\rangle = -s' c'_u (c'')^{d_u}(c')^{e_u-1}. \qquad \text{(E.6)}$$

By symmetry, for the $Z_u Y_v$ term this gives

$$\langle s|e^{i\gamma C} Z_u Y_v e^{-i\gamma C}|s\rangle = \begin{cases} -s'' c'_v (c'')^{d_v-1}(c')^{e_v} & (uv) \in D\\ -s' c'_v (c'')^{d_v}(c')^{e_v-1} & (uv) \in U. \end{cases} \qquad \text{(E.7)}$$

The final term in (E.3) to consider is $\langle s|e^{i\gamma C} Y_u Y_v e^{-i\gamma C}|s\rangle$. We again have two cases. Suppose $(uv) \in D$. As $[YY, ZZ] = 0$, the $Z_u Z_v$ terms in $e^{\pm i\gamma C}$ will commute through $Y_u Y_v$ and cancel. Hence, define $\widetilde{C}_u = C_u + \frac{1}{4} Z_u Z_v$, which is C_u as in (E.1), but without the $Z_u Z_v$ term. Similarly, define $\widetilde{C}_v = C_v + \frac{1}{4}Z_uZ_v$, $\widetilde{D}_u = D_u\backslash\{(uv)\}$, and $\widetilde{D}_v = D_v\backslash\{(uv)\}$. Then C_u and C_v each anticommute with Y_uY_v, and we have

$$\begin{aligned}
\langle s|e^{i\gamma C}Y_uY_ve^{-i\gamma C}|s\rangle &= \langle s|e^{2i\gamma\widetilde{C}_u}e^{2i\gamma\widetilde{C}_v}Y_uY_v|s\rangle \\
&= \langle s|e^{\frac{i}{2}\gamma k_uZ_u}e^{-\frac{i}{2}\gamma\sum_{uw)\in\widetilde{D}_u}Z_uZ_w}e^{-i\gamma\sum_{(uw)\in U_u}Z_uZ_w} \\
&\quad\cdot e^{\frac{i}{2}\gamma k_vZ_v}e^{-\frac{i}{2}\gamma\sum_{(vw)\in\widetilde{D}_v}Z_vZ_w}e^{-i\gamma\sum_{(vt)\in U_v}Z_vZ_t}Y_uY_v|s\rangle, \\
&= \langle s|(Ic'_u+is'_uZ_u)\prod_{i=1}^{d_u-1}(Ic''-is''Z_uZ_{w_i})\prod_{j=1}^{e_u}(Ic'-is'Z_uZ_{w_j}) \\
&\quad\cdot(Ic'_v+is'_vZ_v)\prod_{i=1}^{d_v-1}(Ic''-is''Z_vZ_{w_i})\prod_{j=1}^{e_v}(Ic'-is'Z_vZ_{w_j})Y_uY_v|s\rangle \\
&= \langle s|\left(Ic'_uc'_v+ic'_vs'_uZ_u+ic'_us'_vZ_v-s'_us'_vZ_uZ_v\right) \qquad \text{(E.8)} \\
&\quad\cdot\prod_{(ab)\in D_{uv}}(Ic''-is''Z_aZ_b)\prod_{(ab)\in U_{uv}}(Ic'-is'Z_aZ_b)Y_uY_v|s\rangle,
\end{aligned}$$

where in the last line we have defined $D_{uv} = \widetilde{D}_u\cup\widetilde{D}_v$ and $U_{uv} = U_u\cup U_v$, with $d_{uv} := |D_{uv}| = d_u+d_v-2$ and $e_{uv} := |E_{uv}| = e_u+e_v$.

Consider each term in the first parenthesis $(c'_uc'_vI+ic'_vs'_uZ_u+ic'_us'_vZ_v-s'_us'_vZ_uZ_v)$ in the last line above. It is easy to see that the single Z_u and Z_v terms cannot combine with the remaining terms in the product to produce terms composed of X and I factors, so they contribute nothing to the expectation value and can be ignored. For the I and Z_uZ_v terms in the parenthesis, similar to the argument used in the proof of Theorem 4, these terms can combine and contribute in many ways, depending on the number of triangles containing (uv) in the graph. The I term contribution depends on the number of ways to pick an odd number of triangles, as was the case in deriving (5.15), and the Z_uZ_v term depends on the number of ways to pick an even number of triangles. Unfortunately, our present situation is much more complicated, as we now have 4 different types of triangles (from whether each of $(uw),(vw)$ are directed or undirected), and must consider all possible combinations of different triangle types. This leads to an analysis and result significantly more complicated than that of (5.15). Instead, we consider two special cases leading to simpler results, triangle-free and oriented graphs.

First, suppose the graph is triangle-free. Then only the Z_uZ_v term from the first parenthesis in (E.8) can contribute, and we have

$$\langle s|e^{i\gamma C}Y_uY_ve^{-i\gamma C}|s\rangle = -s'_us'_uZ_uZ_v(c'')^{d_{uv}}(c')^{e_{uv}}Y_uY_v = s'_us'_v(c'')^{d_u+d_v-2}(c')^{e_u+e_v}.$$

For the other case $(uv) \in U$, repeating the above argument gives

$$\langle s|e^{i\gamma C}Y_uY_ve^{-i\gamma C}|s\rangle = s'_us'_v(c'')^{d_u+d_v}(c')^{e_u+e_v-2}$$

Combining this with the above results gives the second part of the Theorem.

Finally, we consider oriented graphs. In this case, every edge is in D and U is empty. Thus, the previous results for the edges in D apply. Let $f = f_{uv}$ be the number of triangles in the graph containing edge (uv). Recall that we define a triangle to be any three edges (uv), (uw), (vw), independently of the direction of each edge. For this case, the derivation of all quantities up to the last one remain valid by setting $U = \emptyset$. For the last quantity $\langle s|e^{i\gamma C}Y_uY_ve^{-i\gamma C}|s\rangle$, we have

$$\langle s|\left(c'_uc'_vI + ic'_vs'_uZ_u + ic'_us'_vZ_v - s'_us'_vZ_uZ_v\right)\prod_{i=1}^{d_u-1}(Ic''-is''Z_uZ_{w_i})\prod_{i=1}^{d_v-1}(Ic''-is''Z_vZ_{w_i})Y_uY_v|s\rangle.$$

By the previous argument, only the I and Z_uZ_v terms in the first parenthesis can possibly contribute. By inspection, the I term contribution follows identically from the derivation of (5.14) to give (cf. (5.15))

$$\langle s|\prod_{i=1}^{d_u-1}(Ic''-is''Z_uZ_{w_i})\prod_{i=1}^{d_v-1}(Ic''-is''Z_vZ_{w_i})Y_uY_v|s\rangle = \frac{1}{2}(c'')^{d_u+d_v-2-2f}(1-(c')^f).$$

By similar arguments (now summing even numbers of triangles rather than odd), and using the formula for the sum over even indices $\sum_{i=0,2,4,\dots}^{f} a^{f-i}b^i = \frac{1}{2}((a+b)^f + (a-b)^f)$, we have

$$\langle s|Z_uZ_v\prod_{i=1}^{d_u-1}(Ic''-is''Z_uZ_{w_i})\prod_{i=1}^{d_v-1}(Ic''-is''Z_vZ_{w_i})Y_uY_v|s\rangle = -\frac{1}{2}(c'')^{d_u+d_v-2-2f}(1+(c')^f).$$

Hence,

$$\langle s|(e^{i\gamma C}Y_uY_ve^{-i\gamma C}|s\rangle = \frac{1}{2}(c'')^{d_u+d_v-2-2f}\left(c'_uc'_v(1-c'^f) + s'_us'_v(1+c'^f)\right).$$

Thus, putting all these quantities together and applying some basic trigonometric identities gives the first statement of the theorem. □