ABSTRACT

Title of Dissertation: Quantum Computing for Optimization and Machine Learning

Shouvanik Chakrabarti
Doctor of Philosophy, 2022

Dissertation Directed by: Assistant Professor Xiaodi Wu
Department of Computer Science

Quantum Computing leverages the quantum properties of subatomic matter to enable computations faster than those possible on a regular computer. Quantum Computers have become increasingly practical in recent years, with some small-scale machines becoming available for public use. The rising importance of machine learning has highlighted a large class of computing and optimization problems that process massive amounts of data and incur correspondingly large computational costs. This raises the natural question of how quantum computers may be leveraged to solve these problems more efficiently. This dissertation presents some encouraging results on the design of quantum algorithms for machine learning and optimization.

We first focus on tasks with provably more efficient quantum algorithms. We show a quantum speedup for convex optimization by extending quantum gradient estimation algorithms to efficiently compute subgradients of non-differentiable functions. We also develop a quantum framework for simulated annealing algorithms which is used to show a quantum speedup in estimating the volumes of convex bodies. Finally, we demonstrate a quantum algorithm for solving matrix games, which can be applied to a variety of learning problems such as linear classification, minimum enclosing ball, and $\ell - 2$ margin SVMs.

We then shift our focus to variational quantum algorithms, which describe a family of heuristic algorithms that use parameterized quantum circuits as function models that can be fit for various learning and optimization tasks. We seek to analyze the properties of these algorithms including their efficient formulation and training, expressivity, and the convergence of the associated optimization problems. We formulate a model of quantum Wasserstein GANs in order to facilitate the robust and scalable generative learning of quantum states. We also investigate the expressivity of so called *Quantum Neural Networks* compared to classical ReLU networks and investigate both theoretical and empirical separations. Finally, we leverage the theory of overparameterization in variational systems to give sufficient conditions on the convergence of *Variational Quantum Eigensolvers*. We use these conditions to design principles to study and evaluate the design of these systems.

QUANTUM COMPUTING FOR OPTIMIZATION AND MACHINE LEARNING

by

Shouvanik Chakrabarti

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2022

Acknowledgments

This dissertation has been made possible due to the help and encouragement of countless people, who have made my graduate studies an enjoyable and rewarding experience that I will cherish forever.

First and foremost, I would like to thank my advisor, Xiaodi Wu, for his invaluable support and guidance throughout the process. I started my PhD studies with very little prior exposure to formal research projects and have been greatly infuenced by his research style, his approach to technical problems, and to formulating ambitious yet actionable research questions. Throughout my tenure in graduate school, Xiaodi always took the time to meet me at least once every week to guide me and keep me on track towards my research goals: beyond technical discussions, he has also given me extremely helpful advice on writing, structuring, and communicating my work that will continue to be of assistance to me in the years to come. I have also greatly enjoyed our many conversations about perspectives and directions in academia as well as life in general.

I thank Andrew M. Childs for being a great source of support throughout and a collaborator on two of the papers presented in this dissertation; as well as offering two great courses on Quantum Computation and Quantum Algorithms that helped me get a solid pedagogical footing in the field. Our interactions have taught me a lot about academic maturity and critical thinking.

I thank Soheil Feizi for introducing me to heuristic machine learning algorithms; and teaching me a lot about effectively combining theoretical and empirical research. I am grateful to him also for many discussions introducing me to interesting problems on the cutting edge of machine learning research.

I thank the members of my dissertation committee, Xiaodi Wu, Alexander Barg, Andrew M. Childs, Soheil Feizi, and Furong Huang for taking the time to read my dissertation and provide valuable feedback during the writing and defense.

The work contained in this dissertation is the product of collaborations with Andrew M. Childs, Soheil Feizi, Yiming Huang, Shih-han Hung, Chunhao Wang, Xiaodi Wu, and Xuchen You. I have also had the good fortune of collaborating with Rajiv Krishnakumar, Guglielmo Mazzola, Nikitas Stamatapoulos, Stefan Woerner, William J. Zeng, and Shaopeng Zhu, on projects not covered here. I thank you all for your contributions and consider myself very lucky to have benefitted from your expertise. I especially thank Tongyang Li, Chunhao Wang, and Xuchen You with whom I have collaborated extensively on multiple projects.

I thank Tongyang Li for being not only a long time collaborator but effectively an additional mentor who has helped me at every stage of graduate life, I am very grateful for his assistance and example. I am also thankful to Xuchen You and Shaopeng Zhu who have been wonderful friends and whose constant company and support cannot be overstated. I thank all my other fellow students at QuICS as well as the CS department, especially Andrew Fichman, Nathaniel Grammel, Sankha Narayan Guria, Yiming Huang, Marina Knittel, Jiaqi Leng, Jingling Li, Yuxiang Peng, Debjani Saha, Katherine Scola, and Jessica Thompson; for their help, as well as many stimulating conversations, lunches, and board game nights that have made graduate school much more enjoyable. A special thanks also to my roommates Siddharth Nag and Joyneel Misra for greatly simplifying life outside of school.

I thank the staff at QuICS and the CS department: especially Tom Hurst and Andrea Svejda for their generous help in streamlining every departmental formality.

During my dissertation research I have been lucky to participate in internships with the Quantum Research Group at Goldman Sachs in the summer of 2020, and the FLARE Research Team at JPMorgan Chase in the summer of 2021. I thank all my collaborators and mentors at both locations for helping me gather new perspectives and learn state-of-the-art results in new areas. I especially thank Rajiv Krishnakumar, Nikitas Stamatapoulos, and William J. Zeng at Goldman Sachs; and Pierre Minssen, Romina Yalovetzky, Shaohan Hu, and Marco Pistoia at JPMorgan Chase for their support throughout. I also thank my fellow interns during this period, especially Arthur Rattew with whom I enjoyed many illuminating discussions.

Finally, I thank my mother, father, and sister for their emotional support and for believing in me even more than myself; and my grandfather Prof. Bijan Kumar Gupta for being my earliest role model and the scientist I always strive to emulate.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Motivations

Quantum Computers were conceptualized in the early 1980s [4], and were initially seen as being necessary to solve the problem of simulating general quantum mechanical processes [5], which seemed infeasible using only classical computers. The work of Deutsch and Josza [6], and Simon [7] made it apparent that speedups from quantum computation could apply to general computing problems that are unmotivated by quantum physics. This culminated in the seminal work of Shor [8], that showed quantum algorithms for the prime factorization and discrete logarithmic problems that are exponentially faster than the best known classical alternatives. Around the same time, Grover [9] developed an algorithm for unstructured search that was quadratically faster than the classical alternative. Exponential quantum speedups naturally draw the most attention, however these have only been found for problems with some algebraic structure, or those with quantum mechanical motivations or properties. On the other hand, polynomial quantum speedups seem viable for a much larger class of computing problems.

The success of machine learning in the 21st century, has highlighted a host of interesting computational problems. These problems are often of great practical significance, and are applied to massive datasets consisting of very high-dimensional data. The amount of data being processed makes it so that even polynomial quantum speedups could have significant ramifications in the practical solution of these problems. Recent years have seen many efforts in the development of quantum algorithms, including algorithms with provable gaurantees (see for eg. [10, 11]), and heuristic approaches (see for eg. [12, 13]). This raises the natural question of whether similar algorithms can be obtained for machine learning tasks, or the optimization problems that power them. The search for a *quantum advantage* in machine learning has been the subject of countless recent works (see for eg. [14, 15, 16, 17]) This dissertation will present several new works in this direction. The focus of these works will be two-fold:

1. To establish quantum algorithms for important machine learning and optimization tasks that require computational costs that are *provably lower* than their best known classical counterparts, and to investigate limitations on the degree of quantum advantage possible.

2. To explore the properties of heuristic quantum algorithms; in an attempt to obtain principled approaches to improving their robustness, scalability, trainability, and usefulness. To this end we theoretically investigate simplified models of these algorithms and verify and extend our hypotheses through empirical study.

1.2 Contributions

Algorithms with provable guarantees. Quantum algorithms with provable guarantees have been traditionally the most studied route to quantum advantage; since their computational requirements can be bounded analytically one can be certain that an advantage is obtained over the best classical counterparts. Such guarantees are especially desirable in the case of quantum algorithms, since the hardware challenges in constructing large scale quantum computers can limit the scope of empirical analysis. Such algorithms have been proposed in the domain of machine learning for tasks including semi-definite programming [17, 18, 19], principle component analysis [15], and singular value decomposition [16], and learning with quantum examples [14].

Interest in provable quantum advantage for machine learning grew after the seminal work of Harrow, Hassidim and Lloyd [20] that demonstrated a quantum algorithm for solving linear systems of equations. This algorithm has been further refined several times to improve the scaling on the error margin and condition number of the equation system [10, 21]. The ubiquity of linear systems in machine learning led to the anticipation of quantum linear system algorithms being applied in many areas of machine learning. The algorithm in [20] suffers however from two issues that limit its usefulness; it requires the data to be encoded in a quantum data structure, and while it requires exponentially fewer resources than a classical linear equation solver, it provides a much weaker solution wherein instead of a full description of the solution we obtain instead a quantum system allowing us to sample based on its coefficients. This algorithm has however been employed in several settings; most notably in an algorithm with claimed exponential advantage for recommendation systems [22], however certain preprocessing assumptions in [22] allow a classical algorithm [23] to also obtain performance exponentially faster than the previous classical state of the art. In order to avoid the subtleties arising from such a comparison, we focus on algorithms that return a complete classical description of the output; at the cost of smaller (polynomial) speedups.

This dissertation presents three works on quantum algorithms with provable guarantees for optimization and machine learning tasks:

1. Chapter 2 describes a quantum algorithm for convex optimization (based on [24] published in Quantum in 2020 and presented at QIP 2019. Convex optimization represents the largest

class of efficiently solvable classical optimization problems and is the central computational step in many machine learning applications. The general problem asks for an estimate of the minimum of a convex function f over a convex set $\mathcal{C}$; using queries to oracles that evaluate f and indicate membership in $\mathcal{C}$. Our algorithm is found to require quadratically fewer queries to problem oracles than the best known classical counterpart (in terms of the input dimension n); corresponding lower bounds are also established that rule out an exponential quantum speedup.

2. Chapter 3 describes a quantum algorithm for estimating the volumes of convex bodies (based on [25]) that was presented at QIP 2020. This is a task of historic importance in convex geometry and has intimate connections with sampling and optimization routines that are commonplace in machine learning. The problem is to estimate the volume of a convex body given access only to a membership oracle. We obtain quantum speedups in both arithmetic and oracle complexity, and show corresponding lower bounds.

3. Chapter 4 describes an optimal quantum algorithm for linear classification that is quadratically faster than the classical state of the art. The approach is generalized to obtain new optimal classical and quantum algorithms for a range of matrix games and is applied to other problems such as the caratheodery, SVMs and zero-sum games. This chapter is based on [26] and [27] published in the proceedings of ICML 2019 and AAAI 2020 respectively.

Variational Quantum Algorithms. Quantum Variational Methods (QVMs) (see for eg. [12, 28, 29] have become a leading candidate for quantum applications on Near-Term Intermediate Scale Quantum Computers. These algorithms use *classically parameterized quantum circuits* as function models that can be trained to satisfy various properties. The classical parameters are optimized on a classical computer, thereby eliminating the need to have precise control operations and arithmetic executed on a quantum machine. The quantum machine is then simply used to evaluate the parameterized function models, and these models can be chosen to be amenable to implementation on near-term hardware. It is commonly conjectured that QVMs will help resolve quantum physics related computational problems in the near future. They are also likely helpful for solving general information/computational tasks, especially when the nature of these tasks exhibits certain structures that can be exploited by quantum mechanics. The success of deep learning in classical computer science has led to a paradigm shift where case by case algorithm designs are often replaced by fitting extremely flexible function models.

A lot of study has already been devoted to the design, analysis, and small-scale implementation of QVMs (e.g., see the survey [30]). A prominent example is the *variational quantum eigensolver* (VQE) [28] which is a QVM that finds the ground state/energy of physically-interesting Hamiltonian systems and finds promising applications in quantum chemistry. Another one is the *quantum approximate optimization algorithm* (QAOA) [29] which proposes a near-term feasible variational circuits that mimic the behavior of quantum adiabatic algorithms to solve optimization problems. One can also further leverage variational quantum circuits for classification [12] (under the name of Quantum Neural Networks (QNNs)), generative models [13], and several other learning tasks.

In contrast to deep learning, where empirical research has played a major role in investigating the training methods, empirical research in quantum variational methods is limited by the available quantum hardware as well as the exponential complexity of simulating them by classical means. As a result, current empirical findings do not necessarily generalize to intermediate-size variational quantum circuits, which are predicted to be available in the near future. A more principled approach is thus required to understand the properties of these systems including their robustness, expressivity, and trainability. This dissertation presents three works that leverage ideas from the theoretical study of deep learning to better understand these aspects of QVMs.

1. Chapter 5 presents a proposal of a quantum Wasserstein GAN that is used to facilitate more robust and scalable learning of unknown target quantum states. This chapter is based on [31] published in the proceedings of NeurIPS 2019.

2. Chapter 6 (based on work currently under review) makes a compares the expressive power of QNNs and feed-forward ReLU networks. Separations are discovered in both directions; indicating that the choice of classical vs quantum neural networks is dependent on the particular problem and disproving a commonly held notion that QNNs are a strictly more powerful alternative.

3. Chapter 7 (based on work currently under review) studies the convergence of the non-convex optimization problems involved in VQEs. We provide the first rigorous proof of convergence for *over-parameterized* VQEs and obtain sufficient conditions on the number of required parameters. These conditions are then used as guiding principles to study and evaluate the design of parameterized ansatz that are commonly used in VQEs.

Acknowledgement. Chapter 2 is based on joint work with Andrew Childs, Tongyang Li, and Xiaodi Wu. Chapter 3 is based on joint work with Andrew Childs, Tongyang Li, Shih-Han Hung, Chunhao Wang, and Xiaodi Wu. Chapter 4 is based on joint work with Tongyang Li, Chunhao Wang, and Xiaodi Wu. Chapter 5 is based on joint work with Yiming Huang, Tongyang Li,

Soheil Feizi, and Xiaodi Wu. Chapter 6 is bsaed on joint work with Xiaodi Wu. Chapter 7 is based on joint work with Xuchen You and Xiaodi Wu.

1.3 Preliminaries on Quantum Information

Quantum systems are represented by quantum states, which are L_2-normalized vectors in a complex vector space (or Hilbert Space). The Hilbert space has an orthonormal basis, where each basis vector corresponds to a distinct classical outcome. For example, a quantum bit (or qubit) is represented by a state in the vector space spanned by $|0\rangle$ (the 0 basis vector) and $|1\rangle$ (the 1 basis vector). In general therefore, a qubit can be in the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $\alpha, \beta \in \mathbb{C}$ such that $|\alpha^2| + |\beta^2| = 1$. A state that is a linear superposition of the basis states is said to be in *superposition*. The coefficients α, β are referred to as the amplitudes of the quantum state. The Hilbert space corresponding to a system made up of disjoint subsystems is the tensor product of the underlying Hilbert spaces.

Quantum computations are normally carried out on an array of one or more quantum bits. The associated Hilbert space is spanned by a basis consisting of vectors corresponding to each possible bitstring (this is termed the computational basis). Computational basis vectors are often associated with the integer represented by the bitstring. A system of n qubits thus has an associated Hilbert space of dimension 2^n. State vectors can also be viewed as normalized column vectors, in which case the computational basis consists of the fundamental basis vectors $\mathbf{e_i}$. The innner product of two vectors $|\psi\rangle, |\phi\rangle$ is denoted by $\langle\psi|\phi\rangle$ and defined as a bilinear, complex conjugate map that maintains the orthonormality of the computational basis. In column vector form, $\langle\psi|\phi\rangle$ is given by the product of the conjugate transpose of $|\psi\rangle$ with $|\phi\rangle$.

The dynamics of quantum states are given by linear maps that preserve their normalization. Such mappings U are called unitary maps and satisfy the condition that $U^\dagger U = UU^\dagger = I$ where $\dagger$ denotes the adjoint operation. In column vector form, each mapping M is denoted by a matrix with entries $M_{ij} = \langle i|M|j\rangle$ where $|i\rangle, |j\rangle$ are the i^{th} and j^{th} compuational basis vectors respectively. In matrix form, the adjoint of a mapping is given by its conjugate transpose. An important (non-unitary) linear map, is the inner product between two vectors $(|\phi\rangle, |\psi\rangle)$ denoted by $|\phi\rangle\langle\psi|$ that maps any vector $|\nu\rangle$ to $\langle\psi|\nu\rangle|\phi\rangle$. The inner product of a vector with itself is the projection map into that vector. In general, any map P such that $P^2 = P$ is a projection operator onto some subspace of the Hilbert space.

An observation (or measurement) of a quantum state associates a real value (or outcome) to every vector of some basis of the Hilbert space. For a basis $|b_j\rangle$ and outcomes m_j the measurement can be encoded in the self-adjoint (Hermitian) matrix $M = \sum_j m_j |b_j\rangle\langle b_j|$. Corresp-ondingly, any self-adjoint operator represents a measurement, where its eigenvectors form the basis of measurement and the eigenvalues represent the associated outcomes. On applying measurement $M = \sum_j m_j |b_j\rangle\langle b_j|$ to a quantum state $|\psi\rangle$ the result m_j is obtained with probability $|\langle b_j|m_j\rangle|^2$.

A vector in a composite Hilbert space is said to be entangled if it is not the tensor product of two vectors from the Hilbert Spaces of the subsystems. The quantum state $|\beta_{00}\rangle = \frac{1}{\sqrt{2}}(|00\rangle+|11\rangle)$ over 2 qubits, is entangled as it cannot be written as the tensor product of two single qubit states. In contrast, the state $\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ is unentangled.

Any unitary matrix U is invertible. This implies that any quantum operation must be reversible, unlike classical computing where common operations such as AND are irreversible. There is a standard recipe to make any classical operation into a quantum operation as follows:

given a classical operation $x \to f(x)$ on bitstrings, we define the unitary U_f that operates as $|x\rangle|y\rangle \to |x\rangle|y \oplus f(x)\rangle$ when $|x\rangle, |y\rangle$ are compuational basis states. This unitary map can be used to evaluate the function $f(x)$ by setting y to the zero bitstring.

Quantum Gates: Quantum Circuits are built out of universal gate sets consisting of 1 or 2 qubit operations known as quantum gates. It is known that these gate sets can all approximate each other with overhead that is polylogarithmic in the approximation error.

Complexity of a Quantum Algorithm: The analogue of runtime of a classical algorithm is the number of gates of a universal gate set required to implement the required quantum circuits. This is referred to as *gate complexity* but is often used interchangeably with runtime while comparing the performance of algorithms.

Ensembles of quantum states: Consider an ensemble of quantum states $\{(p_i, |\psi_i\rangle)\}$. This ensemble can be encoded in a Hermitian positive definite matrix with trace 1 called the *density matrix* as $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$. Conversely, any Hermitian p.s.d matrix with unit trace is a valid density matrix as its eigenbasis forms a valid ensemble. The expectation value when a measurement operator M is applied to a density matrix ρ is $\mathrm{Tr}(\rho M)$. Density matrices that represent one quantum state are referred to as *pure* states and have a rank of 1. Ensembles of more than one state are referred to as *mixed* states, and the corresponding matrix has a rank greater than 1.

For a Hilbert space $\mathcal{X}$ the sets of Hermitian, Unitary and density matrices are denoted by $\mathcal{H}(\mathcal{X})$, $U(\mathcal{X})$, and $\mathcal{D}(\mathcal{X})$ respectively.

Chapter 2: Quantum Algorithms and Lower Bounds for Convex Optimization

In this section we study the problem of convex optimization on quantum computers. The presented discussion is based on results initially obtained in [32].

2.1 Introduction

Convex optimization has been a central topic in the study of mathematical optimization, theoretical computer science, and operations research over the last several decades. On the one hand, it has been used to develop numerous algorithmic techniques for problems in combinatorial optimization, machine learning, signal processing, and other areas. On the other hand, it is a major class of optimization problems that admit efficient classical algorithms [33, 34]. Approaches to convex optimization include the ellipsoid method [34], interior-point methods [35, 36], cutting-plane methods [37, 38], and random walks [39, 40].

The fastest known classical algorithm for general convex optimization solves an n-dimensional instance using $\tilde{O}(n^2)$ queries to oracles for the convex body and the objective function, and runs in time $\tilde{O}(n^3)$ [41].[1] The novel step of [41] is a construction of a separation oracle by a subgradient calculation with $O(n)$ objective function calls and $O(n)$ extra time. It then relies on a reduction from optimization to separation that makes $\tilde{O}(n)$ separation oracle calls and runs

[1]The notation $\tilde{O}$ suppresses poly-logarithmic factors in n, R, r, ϵ, i.e., $\tilde{O}(f(n)) = f(n)\log^{O(1)}(nR/r\epsilon)$.

in time $\tilde{O}(n^3)$ [42]. Although it is unclear whether the query complexity of $\tilde{O}(n^2)$ is optimal for all possible classical algorithms, it is the best possible result using the above framework. This is because it takes $\tilde{\Omega}(n)$ queries to compute the (sub)gradient (see Section 2.5.1) and it also requires $\Omega(n)$ queries to produce an optimization oracle from a separation oracle (see [43] and [44, Section 10.2.2]).

It is natural to ask whether quantum computers can solve convex optimization problems faster. Recently, there has been significant progress on quantum algorithms for solving a special class of convex optimization problems called semidefinite programs (SDPs). SDPs generalize the better-known linear programs (LPs) by allowing positive semidefinite matrices as variables. For an SDP with n-dimensional, s-sparse input matrices and m constraints, the best known classical algorithm [42] finds a solution in time $\tilde{O}(m(m^2 + n^\omega + mns)\operatorname{poly}\log(1/\epsilon))$, where ω is the exponent of matrix multiplication and ϵ is the accuracy of the solution. Brandão and Svore gave the first quantum algorithm for SDPs with worst-case running time $\tilde{O}(\sqrt{mn}s^2(Rr/\varepsilon)^{32})$, where R and r upper bound the norms of the optimal primal and dual solutions, respectively [17]. Compared to the aforementioned classical SDP solver [42], this gives a polynomial speedup in m and n. Van Apeldoorn et al. [45] further improved the running time of a quantum SDP solver to $\tilde{O}(\sqrt{mn}s^2(Rr/\epsilon)^8)$, which was subsequently improved to $\tilde{O}\big((\sqrt{m} + \sqrt{n}(Rr/\epsilon))s(Rr/\epsilon)^4\big)$ [46, 47]. The latter result is tight in the dependence of m and n since there is a quantum lower bound of $\Omega(\sqrt{m} + \sqrt{n})$ for constant R, r, s, ϵ [17].

However, semidefinite programming is a structured form of convex optimization that does not capture the problem in general. In particular, SDPs are specified by positive semidefinite matrices, and their solution is related to well-understood tasks in quantum computation such as solving linear systems (e.g., [10, 20]) and Gibbs sampling (e.g., [46, 47]). General convex optimization need not include such structural information, instead only offering the promise that the objective function and constraints are convex. Currently, little is known about whether quantum computers could provide speedups for general convex optimization. Our goal is to shed light on this question.

2.1.1 Convex optimization

We consider the following general minimization problem:

$$\min_{x \in K} f(x), \text{ where } K \subseteq \mathbb{R}^n \text{ is a convex set and } f\colon K \to \mathbb{R} \text{ is a convex function.} \tag{2.1}$$

We assume we are given upper and lower bounds on the function values, namely $m \leq \min_{x \in K} f(x) \leq M$, and inner and outer bounds on the convex set K, namely

$$B_2(0, r) \subseteq K \subseteq B_2(0, R), \tag{2.2}$$

where $B_2(x, l)$ is the ball of radius l in L_2 norm centered at $x \in \mathbb{R}^n$. We ask for a solution $\tilde{x} \in K$ with precision ϵ, in the sense that

$$f(\tilde{x}) \leq \min_{x \in K} f(x) + \epsilon. \tag{2.3}$$

We consider the very general setting where the convex body K and convex function f are only specified by oracles. In particular, we have:

- A *membership oracle* O_K for K, which determines whether a given $x \in \mathbb{R}^n$ belongs to K;

- An *evaluation oracle* O_f for f, which outputs $f(x)$ for a given $x \in K$.

Convex optimization has been well-studied in the model of membership and evaluation oracles since this provides a reasonable level of abstraction of K and f, and it helps illuminate the algorithmic relationship between the optimization problem and the relatively simpler task of determining membership [34, 41, 42]. The efficiency of convex optimization is then measured by the number of queries to the oracles (i.e., the *query complexity*) and the total number of other elementary gates (i.e., the *gate complexity*).

It is well known that a general bounded convex optimization problem is equivalent to one with a linear objective function over a different bounded convex set. In particular, if promised that $\min_{x \in K} f(x) \leq M$, (2.1) is equivalent to the problem

$$\min_{x' \in \mathbb{R},\, x \in K} x' \quad \text{such that} \quad f(x) \leq x' \leq M. \tag{2.4}$$

Observe that a membership query to the new convex set

$$K' := \{(x', x) \in \mathbb{R} \times K \mid f(x) \leq x' \leq M\} \tag{2.5}$$

can be implemented with one query to the membership oracle for K and one query to the evaluation oracle for f. Thus the ability to optimize a linear function

$$\min_{x \in K} c^T x \tag{2.6}$$

for any $c \in \mathbb{R}^n$ and convex set $K \subseteq \mathbb{R}^n$ is essentially equivalent to solving a general convex optimization problem. A procedure to solve such a problem for any specified c is known as an *optimization oracle*. Thus convex optimization reduces to implementing optimization oracles over general convex sets (Lemma 2.2.1). The related concept of a *separation oracle* takes as input a point $p \notin K$ and outputs a hyperplane separating p from K.

In the quantum setting, we model oracles by unitary operators instead of classical procedures. In particular, in the quantum model of membership and evaluation oracles, we are promised to have unitaries O_K and O_f such that

- For any $x \in \mathbb{R}^n$, $O_K|x,0\rangle = |x, \delta[x \in K]\rangle$, where $\delta[P]$ is 1 if P is true and 0 if P is false;
- For any $x \in \mathbb{R}^n$, $O_f|x,0\rangle = |x, f(x)\rangle$.

In other words, we allow coherent superpositions of queries to both oracles. If the classical oracles can be implemented by explicit circuits, then the corresponding quantum oracles can be implemented by quantum circuits of about the same size, so the quantum query model provides a useful framework for understanding the quantum complexity of convex optimization.

2.1.2 Contributions

Our first main result is a quantum algorithm for optimizing a convex function over a convex body. Specifically, we show the following:

Theorem 2.1.1. *There is a quantum algorithm for minimizing a convex function f over a convex set $K \subseteq \mathbb{R}^n$ using $\tilde{O}(n)$ queries to an evaluation oracle for f and $\tilde{O}(n)$ queries to a membership oracle for K. The gate complexity of this algorithm is $\tilde{O}(n^3)$.*

Recall that the state-of-the-art classical algorithm [41] for general convex optimization with evaluation and membership oracles uses $\tilde{O}(n^2)$ queries to each. Thus our algorithm provides a quadratic improvement over the best known classical result. While the query complexity of [41] is not known to be tight, it is the best possible result that can be achieved using subgradient computation to implement a separation oracle, as discussed above.

The proof of Theorem 2.1.1 follows the aforementioned classical strategy of constructing a separating hyperplane for any given point outside the convex body [41]. We find this hyperplane using a fast quantum algorithm for gradient estimation using $\tilde{O}(1)$ evaluation queries[2], as first proposed by Jordan [48] and later refined by [49] with more rigorous analysis. However, finding a suitable hyperplane in general requires calculating approximate subgradients of convex functions that may not be differentiable, whereas the algorithms in [48] and [49] both require bounded second derivatives or more stringent conditions. To address this issue, we introduce classical randomness into the algorithm to produce a suitable approximate subgradient with $\tilde{O}(1)$ evaluation queries, and show how to use such an approximate subgradient in the separation framework to produce a faster quantum algorithm. Moreover, our subgradient algorithm only relies on an *approximate* membership oracle with precision $1/\operatorname{poly}(n)$ (see Definition 2.2.5).

Our new quantum algorithm for subgradient computation is the source of the quantum speedup of the entire algorithm and establishes a separation in query complexity for the subgradient computation between quantum ($\tilde{O}(1)$) and classical ($\tilde{\Omega}(n)$, see Section 2.5.1) algorithms. This subroutine could also be of independent interest, in particular in the study of quantum algorithms based on gradient descent and its variants (e.g., [50, 51]).

On the other hand, we also aim to establish corresponding quantum lower bounds to understand the potential for quantum speedups for convex optimization. To this end, we prove:

Theorem 2.1.2. *There exists a convex body $K \subseteq \mathbb{R}^n$, a convex function f on K, and a precision*

[2]Here $\tilde{O}(1)$ has the same definition as footnote 1, i.e., $\tilde{O}(1) = \log^{O(1)}(nR/r\epsilon)$.

$\epsilon > 0$, *such that a quantum algorithm needs at least* $\Omega(\sqrt{n})$ *queries to a membership oracle for* K *and* $\Omega(\sqrt{n}/\log n)$ *queries to an evaluation oracle for* f *to output a point* $\tilde{x}$ *satisfying*

$$f(\tilde{x}) \leq \min_{x \in K} f(x) + \epsilon \tag{2.7}$$

with high success probability (say, at least 0.8).

We establish the query lower bound on the membership oracle by reductions from search with wildcards [52]. The lower bound on evaluation queries uses a similar reduction, but this only works for an evaluation oracle with low precision. To prove a lower bound on precise evaluation queries, we propose a discretization technique that relates the difficulty of the continuous problem to a corresponding discrete one. This approach might be of independent interest since optimization problems naturally have continuous inputs and outputs, whereas most previous work on quantum lower bounds focuses on discrete inputs. Using this technique, we can simulate one perfectly precise query by one low-precision query at discretized points, thereby establishing the evaluation lower bound as claimed in Theorem 2.1.2. As a side point, this evaluation lower bound holds even for an unconstrained convex optimization problem on $\mathbb{R}^n$, which might be of independent interest since this setting has also been well-studied classically [33, 43, 44, 53].

We summarize our main results in Table 2.1.

	Classical bounds	Quantum bounds (this paper)
Membership queries	$\tilde{O}(n^2)$ [41], $\Omega(n)$ [54]	$\tilde{O}(n)$, $\Omega(\sqrt{n})$
Evaluation queries	$\tilde{O}(n^2)$ [41], $\Omega(n)$ [54]	$\tilde{O}(n)$, $\tilde{\Omega}(\sqrt{n})$
Time complexity	$\tilde{O}(n^3)$ [41]	$\tilde{O}(n^3)$

Table 2.1: Summary of classical and quantum complexities of convex optimization.

2.1.3 Overview of techniques

2.1.3.1 Upper bound

To prove our upper bound result in Theorem 2.1.1, we use the well-known reduction from general convex optimization to the case of a linear objective function, which simplifies the problem to implementing an optimization oracle using membership oracle queries (Lemma 2.2.1). For the reduction from optimization to membership, we follow the best known classical result in [41] which implements an optimization oracle using $\tilde{O}(n^2)$ membership queries and $\tilde{O}(n^3)$ arithmetic operations. In [41], the authors first show a reduction from separation oracles to membership oracles that uses $\tilde{O}(n)$ queries and then use a result from [42] to implement an optimization oracle using $\tilde{O}(n)$ queries to a separation oracle, giving an overall query complexity of $\tilde{O}(n^2)$.

The reduction from separation to membership involves the calculation of a *height function* defined by the authors (see Eq. (2.50)), whose evaluation oracle can be implemented in terms of the membership oracle of the original set. A separating hyperplane is determined by computing a subgradient, which already takes $\tilde{O}(n)$ queries. In fact, it is not hard to see that any classical algorithm requires $\tilde{\Omega}(n)$ classical queries (see Section 2.5.1), so this part of the algorithm cannot be improved classically. The possibility of using the quantum Fourier transform to compute the gradient of a function using $\tilde{O}(1)$ evaluation queries ([48, 49]) suggests the possibility of replacing the subgradient procedure with a faster quantum algorithm. However, the techniques described in [48, 49] require the function whose gradient is to be computed to have bounded second (or even higher) derivatives, and the height function is only guaranteed to be Lipschitz continuous (Definition 2.2.9) and in general is not even differentiable.

To compute subgradients of general (non-differentiable) convex functions, we introduce classical randomness (taking inspiration from [41]) and construct a quantum subgradient algorithm that uses $\tilde{O}(1)$ queries. Our proof of correctness (Section 2.2.2) has three main steps:

1. We analyze the average error incurred when computing the gradient using the quantum Fourier transform. Specifically, we show that this approach succeeds if the function has bounded second derivatives in the vicinity of the point where the gradient is to be calculated (see Algorithm 1, Algorithm 2, and Lemma 2.2.3). Some of our calculations are inspired by [49].

2. We use the technique of *mollifier functions* (a common tool in functional analysis [55], suggested to us by [54] in the context of [41]) to show that it is sufficient to treat infinitely differentiable functions (the mollified functions) with bounded first derivatives (but possibly large second derivatives). In particular, it is sufficient to output an approximate gradient of the mollified function at a point near the original point where the subgradient is to be calculated (see Lemma 2.2.4).

3. We prove that convex functions with bounded first derivatives have second derivatives that lie below a certain threshold with high probability for a random point in the vicinity of the original point (Lemma 2.2.5). Furthermore, we show that a bound on the second derivatives can be chosen so that the smooth gradient calculation techniques work on a sufficiently large fraction of the neighborhood of the original point, ensuring that the final subgradient error is small (see Algorithm 3 and Theorem 2.2.2).

The new quantum subgradient algorithm is then used to construct a separation oracle as in

[41] (and a similar calculation is carried out in Theorem 2.2.3). Finally the reduction from [42] is used to construct the optimization oracle using $\tilde{O}(n)$ separation queries. From Lemma 2.2.1, this shows that the general convex optimization problem can be solved using $\tilde{O}(n)$ membership and evaluation queries and $\tilde{O}(n^3)$ gates.

2.1.3.2 Lower bound

We prove our quantum lower bounds on membership and evaluation queries separately before showing how to combine them into a single optimization problem. Both lower bounds work over n-dimensional hypercubes.

In particular, we prove both lower bounds by reductions from search with wildcards [52]. In this problem, we are given an n-bit binary string s and the task is to determine all bits of s using wildcard queries that check the correctness of any subset of the bits of s: more formally, the input in the wildcard model is a pair (T, y) where $T \subseteq [n]$ and $y \in \{0,1\}^{|T|}$, and the query returns 1 if $s_{|T} = y$ (here the notation $s_{|T}$ represents the subset of the bits of s restricted to T). Reference [52] shows that the quantum query complexity of search with wildcards is $\Omega(\sqrt{n})$.

For our lower bound on membership queries, we consider a simple objective function, the sum of all coordinates $\sum_{i=1}^{n} x_i$. In other words, we take $c = \mathbf{1}^n$ in (2.6). However, the position of the hypercube is unknown, and to solve the optimization problem (formally stated in Definition 2.3.1), one must use the membership oracle to locate it.

Specifically, the hypercube takes the form $\bigtimes_{i=1}^{n}[s_i - 2, s_i + 1]$ (where $\bigtimes$ is the Cartesian product) for some offset binary string $s \in \{0,1\}^n$. In Section 2.3.1, we prove:

- Any query $x \in \mathbb{R}^n$ to the membership oracle of this problem can be simulated by one query to the search-with-wildcards oracle for s. To achieve this, we divide the n coordinates of x into four sets: $T_{x,0}$ for those in $[-2,-1)$, $T_{x,1}$ for those in $(1,2]$, $T_{x,\text{mid}}$ for those in $[-1,1]$, and $T_{x,\text{out}}$ for the rest. Notice that $T_{x,\text{mid}}$ corresponds to the coordinates that are always in the hypercube and $T_{x,\text{out}}$ corresponds to the coordinates that are always out of the hypercube; $T_{x,0}$ (resp., $T_{x,1}$) includes the coordinates for which $s_i = 0$ (resp., $s_i = 1$) impacts the membership in the hypercube. We prove in Section 2.3.1 that a wildcard query with $T = T_{x,0} \cup T_{x,1}$ can simulate a membership query to x.

- The solution of the sum-of-coordinates optimization problem explicitly gives s, i.e., it solves search with wildcards. This is because this solution must be close to the point $(s_1 - 2, \ldots, s_n - 2)$, and applying integer rounding would recover s.

These two points establish the reduction of search with wildcards to the optimization problem, and hence establishes the $\Omega(\sqrt{n})$ membership quantum lower bound in Theorem 2.1.2 (see Theorem 2.3.2).

For our lower bound on evaluation queries, we assume that membership is trivial by fixing the hypercube at $\mathcal{C} = [0,1]^n$. We then consider optimizing the max-norm function

$$f(x) = \max_{i \in [n]} |x_i - c_i| \tag{2.8}$$

for some unknown $c \in \{0,1\}^n$. Notice that learning c is equivalent to solving the optimization problem; in particular, outputting an $\tilde{x} \in \mathcal{C}$ satisfying (2.3) with $\epsilon = 1/3$ would determine the string c. This follows because for all $i \in [n]$, we have $|\tilde{x}_i - c_i| \leq \max_{i \in [n]} |\tilde{x}_i - c_i| \leq 1/3$, and c_i must be the integer rounding of $\tilde{x}_i$, i.e., $c_i = 0$ if $\tilde{x}_i \in [0, 1/2)$ and $c_i = 1$ if $\tilde{x}_i \in [1/2, 1]$. On the other hand, if we know c, then we know the optimum $x = c$.

We prove an $\Omega(\sqrt{n}/\log n)$ lower bound on evaluation queries for learning c. Our proof, which appears in Section 2.3.2, is composed of three steps:

1) We first prove a weaker lower bound with respect to the precision of the evaluation oracle. Specifically, if $f(x)$ is specified with b bits of precision, then using binary search, a query to $f(x)$ can be simulated by b queries to an oracle that inputs $(f(x), t)$ for some $t \in \mathbb{R}$ and returns 1 if $f(x) \leq t$ and returns 0 otherwise. We further without loss of generality assume $x \in [0,1]^n$. If $x \notin [0,1]^n$, we assign a penalty of the L_1 distance between x and its projection $\pi(x)$ onto $[0,1]^n$; by doing so, $f(\pi(x))$ and x fully characterizes $f(x)$ (see (2.81)). Therefore, $f(x) \in [0,1]$, and $f(x)$ having b bits of precision is equivalent to having precision 2^{-b}.

   Similar to the interval dividing strategy in the proof of the membership lower bound, we prove that one query to such an oracle can be simulated by one query to the search-with-wildcards oracle for s. Furthermore, the solution of the max-norm optimization problem explicitly gives s, i.e., it solves the search-with-wildcards problem. This establishes the reduction to search with wildcards, and hence establishes an $\Omega(\sqrt{n}/b)$ lower bound on the number of quantum queries to the evaluation oracle f with precision 2^{-b} (see Lemma 2.3.1).

2) Next, we introduce a technique we call *discretization*, which effectively simulates queries over an (uncountably) infinite set by queries over a discrete set. This technique might be of independent interest since proving lower bounds on functions with an infinite domain can be challenging.

   We observe that the problem of optimizing (2.8) has the following property: if we are given two strings $x, x' \in [0,1]^n$ such that $x_1, \ldots, x_n, 1-x_1, \ldots, 1-x_n$ and $x'_1, \ldots, x'_n, 1-x'_1, \ldots, 1-x'_n$ have the same ordering (for instance, $x = (0.1, 0.2, 0.7)$ and $x' = (0.1, 0.3, 0.6)$ both have the ordering $x_1 \le x_2 \le 1-x_3 \le x_3 \le 1-x_2 \le 1-x_1$), then

$$\arg\max_{i\in[n]} |x_i - c_i| = \arg\max_{i\in[n]} |x'_i - c_i|. \tag{2.9}$$

Furthermore, if $x'_1, \ldots, x'_n, 1-x'_1, \ldots, 1-x'_n$ are $2n$ different numbers, then knowing the value of $f(x')$ implies the value of the $\arg\max$ in (2.9) (denoted i^*) and the corresponding c_{i^*}, and we can subsequently recover $f(x)$ given x since $f(x) = |x_{i^*} - c_{i^*}|$. In other words, $f(x)$ can be computed given x and $f(x')$.

Therefore, it suffices to consider all possible ways of ordering $2n$ numbers, rendering the problem discrete. Without loss of generality, we focus on x' satisfying $\{x'_1, \ldots, x'_n, 1-x'_1, \ldots, 1-x'_n\} = \{\frac{1}{2n+1}, \ldots, \frac{2n}{2n+1}\}$, and we denote the set of all such x' by D_n (see also (2.97)). In Lemma 2.3.4, we prove that one classical (resp., quantum) evaluation query from $[0,1]^n$ can be simulated by one classical evaluation query (resp., two quantum evaluation queries) from D_n using Algorithm 5. To illustrate this, we give a concrete example with $n = 3$ in Section 2.3.2.2.

3) Finally, we use discretization to show that one perfectly precise query to f can be simulated by one query to f with precision $\frac{1}{5n}$; in other words, b in step 1) is at most $\lceil \log_2 5n \rceil = O(\log n)$ (see Lemma 2.3.3). This is because by discretization, the input domain can be limited to the discrete set D_n. Notice that for any $x \in D_n$, $f(x)$ is an integer multiple of $\frac{1}{2n+1}$; even if $f(x)$ can only be computed with precision $\frac{1}{5n}$, we can round it to the closest

integer multiple of $\frac{1}{2n+1}$ which is exactly $f(x)$, since the distance $\frac{2n+1}{5n} < \frac{1}{2}$. As a result, we can precisely compute $f(x)$ for all $x \in D_n$, and thus by discretization we can precisely compute $f(x)$ for all $x \in [0,1]^n$.

In all, the three steps above establish an $\Omega(\sqrt{n}/\log n)$ quantum lower bound on evaluation queries to solve the problem in Eq. (2.8) (see Theorem 2.3.2). In particular, this lower bound is proved for an unconstrained convex optimization problem on $\mathbb{R}^n$, which might be of independent interest.

As a side result, we prove that our quantum lower bound is optimal for the problem in (2.8) (up to poly-logarithmic factors in n), as we can prove a matching $\tilde{O}(\sqrt{n})$ upper bound (Theorem 2.5.1). Therefore, a better quantum lower bound on the number of evaluation queries for convex optimization would require studying an essentially different problem.

Having established lower bounds on both membership and evaluation queries, we combine them to give Theorem 2.1.2. This is achieved by considering an optimization problem of dimension $2n$; the first n coordinates compose the sum-of-coordinates function in Section 2.3.1, and the last n coordinates compose the max-norm function in Section 2.3.2. We then concatenate both parts and prove Theorem 2.1.2 via reductions to the membership and evaluation lower bounds, respectively (see Section 2.3.3).

In addition, all lower bounds described above can be adapted to a convex body that is contained in the unit hypercube and that contains the discrete set D_n to facilitate discretization; we present a "smoothed" hypercube (see Section 2.3.4) as a specific example.

Organization. Our quantum upper bounds are given in Section 2.2 and lower bounds are given in Section 2.3. Technical details that are not essential to the main discussion, included auxiliary lemmas (Section 2.5.1) and proof details for upper bounds (Section 2.5.2) and lower bounds (Section 2.5.3) are provided in an appendix section at the end of the chapter for completeness.

2.2 Upper bound

In this section, we prove:

Theorem 2.2.1. *An optimization oracle for a convex set $K \subseteq \mathbb{R}^n$ can be implemented using $\tilde{O}(n)$ quantum queries to a membership oracle for K, with gate complexity $\tilde{O}(n^3)$.*

The following lemma shows the equivalence of optimization oracles to a general convex optimization problem.

Lemma 2.2.1. *Suppose a reduction from an optimization oracle to a membership oracle for convex sets requires $O(g(n))$ queries to the membership oracle. Then the problem of optimizing a convex function over a convex set can be solved using $O(g(n))$ queries to both the membership oracle and the evaluation oracle.*

Proof. The problem $\min_{x \in K} f(x)$ reduces to the problem $\min_{(x',x) \in K'} x'$ where K' is defined as in (2.4). K' is the intersection of convex sets and is therefore itself convex. A membership oracle for K' can be implemented using 1 query each to the membership oracle for K and the evaluation oracle for f. Since $O(g(n))$ queries to the membership oracle for K' are sufficient to optimize any linear function, the result follows. □

Theorem 2.1.1 directly follows from Theorem 2.2.1 and Lemma 2.2.1.

Overview. This part of the paper is organized following the plan outlined in Section 2.1.3.1. Precise definitions of oracles and other relevant terminology appear in Section 2.2.1. Section 2.2.2

develops a fast quantum subgradient procedure that can be used in the classical reduction from optimization to membership. This is done in two parts:

1. Section 2.2.2.1 presents an algorithm based on the quantum Fourier transform that calculates the gradient of a function with bounded second derivatives (i.e., a β-smooth function) with bounded expected one-norm error.

2. Section 2.2.2.2 uses mollification to restrict the analysis to infinitely differentiable functions without loss of generality, and then uses classical randomness to eliminate the need for bounded second derivatives.

In Section 2.2.3 we show that the new quantum subgradient algorithm fits into the classical reduction from [41]. Finally, we describe the reduction from optimization to membership in Section 2.2.4.

2.2.1 Oracle definitions

In this section, we provide precise definitions for the oracles for convex sets and functions that we use in our algorithm and its analysis. We also provide precise definitions of Lipschitz continuity and β-smoothness, which we will require in the rest of the section.

Definition 2.2.1 (Ball in L_p norm)**.** *The ball of radius $r > 0$ in L_p norm $\|\cdot\|_p$ centered at $x \in \mathbb{R}^n$ is $B_p(x, r) := \{y \in \mathbb{R}_n \mid \|x - y\|_p \leq r\}$.*

Definition 2.2.2 (Interior of a convex set)**.** *For any $\delta > 0$, the δ-interior of a convex set K is defined as $B_2(K, -\delta) := \{x \mid B_2(x, \delta) \subseteq K\}$.*

Definition 2.2.3 (Neighborhood of a convex set). *For any $\delta > 0$, the δ-neighborhood of a convex set K is defined as $B_2(K, \delta) := \{x \mid \exists y \in K \text{ s.t. } \|x - y\|_2 \leq \delta\}$.*

Definition 2.2.4 (Evaluation oracle). *When queried with $x \in \mathbb{R}^n$ and $\delta > 0$, output α such that $|\alpha - f(x)| \leq \delta$. We use $\mathrm{EVAL}_\delta(f)$ to denote the time complexity. The classical procedure or quantum unitary representing the oracle is denoted by O_f.*

Definition 2.2.5 (Membership oracle). *When queried with $x \in \mathbb{R}^n$ and $\delta > 0$, output an assertion that $x \in B_2(K, \delta)$ or $x \notin B_2(K, -\delta)$. The time complexity is denoted by $\mathrm{MEM}_\delta(K)$. The classical procedure or quantum unitary representing the membership oracle is denoted by O_K.*

Definition 2.2.6 (Separation oracle). *When queried with $x \in \mathbb{R}^n$ and $\delta > 0$, with probability $1 - \delta$, either*

- *assert $x \in B_2(K, \delta)$ or*
- *output a unit vector $\hat{c}$ such that $\hat{c}^T x \leq \hat{c}^T y + \delta$ for all $y \in B_2(K, -\delta)$.*

The time complexity is denoted by $\mathrm{SEP}_\delta(K)$.

Definition 2.2.7 (Optimization oracle). *When queried with a unit vector c, find $y \in \mathbb{R}^n$ such that $c^T x \leq c^T y + \delta$ for all $x \in B_2(K, -\delta)$ or asserts that $B_2(K, \delta)$ is empty. The time complexity of the oracle is denoted by $\mathrm{OPT}_\delta(K)$.*

Definition 2.2.8 (Subgradient). *A subgradient of a convex function $f\colon \mathbb{R}^n \to \mathbb{R}$ at x, is a vector g such that*

$$f(y) \geq f(x) + \langle g, y - x \rangle \tag{2.10}$$

for all $y \in \mathbb{R}^n$. For a differentiable convex function, the gradient is the only subgradient. The set of subgradients of f at x is called the subdifferential at x and denoted by $\partial f(x)$.

Definition 2.2.9 (L-Lipschitz continuity). *A function $f\colon \mathbb{R}^n \to \mathbb{R}$ is said to be L-Lipschitz continuous (or simply L-Lipschitz) in a set S if for all $x \in S$, $\|g\|_\infty \leq L$ for any $g \in \partial f(x)$. An immediate consequence of this is that for any $x, y \in S$,*

$$|f(y) - f(x)| \leq L\|y - x\|_\infty. \tag{2.11}$$

Definition 2.2.10 (β-smoothness). *A function $f\colon \mathbb{R}^n \to \mathbb{R}$ is said to be β-smooth in a set S if for all $x \in S$, the magnitudes of the second derivatives of f in all directions are bounded by β. This also means that the largest magnitude of an eigenvalue of the Hessian $\nabla^2 f(x)$ is at most β. Consequently, for any $x, y \in S$, we have*

$$f(y) \leq f(x) + \langle \nabla f(x), y - x\rangle + \frac{\beta}{2}\|y - x\|_\infty^2. \tag{2.12}$$

2.2.2 Evaluation to subgradient

In this section we present a procedure that, given an evaluation oracle for an L-Lipschitz continuous function $f\colon \mathbb{R}^n \to \mathbb{R}$ with evaluation error at most $\epsilon > 0$, a point $x \in \mathbb{R}^n$, and an "approximation scale" factor $r_1 > 0$, computes an approximate subgradient $\tilde{g}$ of f at x. Specifically, $\tilde{g}$ satisfies

$$f(q) \geq f(x) + \langle \tilde{g}, q - x\rangle - \zeta\|q - x\|_\infty - 4nr_1L \tag{2.13}$$

for all $q \in \mathbb{R}^n$, where $\mathbb{E}\zeta \leq \xi(r_1, \epsilon)$ and ξ must monotonically increase with ϵ as ϵ^α for some $\alpha > 0$. Here ζ is the error in the subgradient that is bounded in expectation by the function ξ.

2.2.2.1 Smooth functions

We first describe how to approximate the gradient of a smooth function. Algorithm 1 and Algorithm 2 use techniques from [48] and [49] to evaluate the gradient of a function with bounded second derivatives in the neighborhood of the evaluation point. To analyze their behavior, we begin with the following lemma showing that Algorithm 1 provides a good estimate of the gradient with bounded failure probability.

```
Algorithm 1: GradientEstimate($f, \epsilon, L, \beta, x_0$)
    Data: Function $f$, evaluation error $\epsilon$, Lipschitz constant $L$, smoothness parameter $\beta$,
          and point $x_0$.
    Define
      • $l = 2\sqrt{\epsilon/n\beta}$ to be the size of the grid used,
      • $b \in \mathbb{N}$ such that $\frac{24\pi\sqrt{n\epsilon\beta}}{L} \leq \frac{1}{2^b} = \frac{1}{N} \leq \frac{48\pi\sqrt{n\epsilon\beta}}{L}$,
      • $b_0 \in \mathbb{N}$ such that $\frac{N\epsilon}{2Ll} \leq \frac{1}{2^{b_0}} = \frac{1}{N_0} \leq \frac{N\epsilon}{Ll}$,
      • $F(x) = \frac{N}{2Ll}[f(x_0 + \frac{l}{N}(x - N/2)) - f(x_0)]$, and,
      • $\gamma : \{0, 1, \ldots, N-1\} \to G := \{-N/2, -N/2+1, \ldots, N/2-1\}$ such that
        $\gamma(x) = x - N/2$.
    Let $O_F$ denote a unitary operation acting as $O_F|x\rangle = e^{2\pi i \tilde{F}(x)}|x\rangle$, where
    $|\tilde{F}(x) - F(x)| \leq \frac{1}{N_0}$, with $x$ represented using $b$ bits and $\tilde{F}(x)$ represented using $b_0$
    bits.
  1 Start with $n$ $b$-bit registers set to 0 and Hadamard transform each to obtain
        $$\frac{1}{\sqrt{N^n}} \sum_{x_1,\ldots,x_d \in \{0,1,\ldots,N-1\}} |x_1, \ldots, x_d\rangle; \qquad (2.14)$$
  2 Perform the operation $O_F$ and the map $|x\rangle \mapsto |\gamma(x)\rangle$ to obtain
        $$\frac{1}{N^{n/2}} \sum_{g \in G^n} e^{2\pi i \tilde{F}(g)}|g\rangle; \qquad (2.15)$$
  3 Apply the inverse QFT over $G$ to each of the registers;
  4 Measure the final state to get $k_1, k_2, \ldots, k_d$ and report $\tilde{g} = \frac{2L}{N}(k_1, k_2, \ldots, k_d)$ as the
     result.
```

Lemma 2.2.2. *Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be an L-Lipschitz function that is specified by an evaluation* *oracle with error at most ϵ. Let f be β-smooth in $B_\infty(x, 2\sqrt{\epsilon/\beta})$, and let $\tilde{g}$ be the output of* $\texttt{GradientEstimate}(f, \epsilon, L, \beta, x_0)$ *(from Algorithm 1). Then*

$$\Pr\left[|\tilde{g}_i - \nabla f(x)_i| > 1500\sqrt{n\epsilon\beta}\right] < \frac{1}{3}, \quad \forall i \in [n]. \tag{2.16}$$

The proof of Lemma 2.2.2 is deferred to Lemma 2.5.5 in the final section.

Next we analyze Algorithm 2, which uses several calls to Algorithm 1 to provide an estimate of the gradient that is close in expected L_1 distance to the true value.

```
Algorithm 2: SmoothQuantumGradient($f, \epsilon, L, \beta, x$)
    Data: Function $f$, evaluation error $\epsilon$, Lipschitz constant $L$, smoothness parameter $\beta$,
          and point $x$.
  1 Set $T$ such that $2e^{-T^2/24} \le 1500\sqrt{n\epsilon\beta}/L$;
  2 for $t = 1, 2, \ldots, T$ do
  3     $e^{(t)} \leftarrow$ GradientEstimate($f, \epsilon, L, \beta, x$);
  4 for $i = 1, 2, \ldots, n$ do
  5     If more than $T/2$ of $e_i^{(t)}$ lie in an interval of size $3000\sqrt{n\epsilon\beta}$, set $\tilde{g}_i$ to be the median
          of the points in that interval;
  6     Otherwise, set $\tilde{g}_i = 0$;
  7 Output $\tilde{g}$.
```

Lemma 2.2.3. *Let f be a convex, L-Lipshcitz continuous function that is specified by an evaluation oracle with error at most ϵ. Suppose f is β-smooth in $B_\infty(x, 2\sqrt{\epsilon/\beta})$. Let*

$$\tilde{g} = \texttt{SmoothQuantumGradient}(f, \epsilon, L, \beta, x) \tag{2.17}$$

(from Algorithm 2). Then for any $i \in [n]$, we have $|\tilde{g}_i| \le L$ and $\mathbb{E}|\tilde{g}_i - \nabla f(x)_i| \le 3000\sqrt{n\epsilon\beta}$;

hence

$$\mathbb{E}\|\tilde{g} - \nabla f(x)\|_1 \le 3000 n^{3/2}\sqrt{\epsilon\beta}. \tag{2.18}$$

If L, $1/\beta$, and $1/\epsilon$ are $\mathrm{poly}(n)$*, the* `SmoothQuantumGradient` *algorithm uses $\tilde{O}(1)$ queries to the evaluation oracle and $\tilde{O}(n)$ gates.*

Proof. For each dimension $i \in [n]$ and each iteration $t \in [T]$, consider the random variable

$$X_i^t = \begin{cases} 1 & \text{if } |e_i^{(t)} - \nabla f(x)_i| > 1500\sqrt{n\epsilon\beta} \\ 0 & \text{otherwise.} \end{cases} \tag{2.19}$$

From the conditions on function f, Lemma 2.2.2 applies to `GradientEstimate`$(f, \epsilon, L, \beta, x)$, and thus $\Pr(X_i^t = 1) < 1/3$. Thus, by the Chernoff bound, $\Pr\left[|\tilde{g}_i - \nabla f(x)_i| \le 1500\sqrt{n\epsilon\beta}\right] > 1 - 2e^{-T^2/24} \ge 1 - 1500\sqrt{n\epsilon\beta}/L$. In the remaining cases, $|\tilde{g}_i - \nabla f(x)_i| \le L$ (see Line 4 of Algorithm 1). Thus $\mathbb{E}|\tilde{g}_i - \nabla f(x)_i| \le 3000\sqrt{n\epsilon\beta}$ for all $i \in [n]$, and (2.18) follows.

The algorithm makes $T = \mathrm{poly}(\log(1/n\epsilon\beta))$ calls to a procedure that makes one query to the evaluation oracle. Thus the query complexity is $\tilde{O}(1)$. To evaluate the gate complexity, observe that we iterate over n dimensions, using $\mathrm{poly}(b) = \mathrm{poly}(\log(1/n\epsilon\beta))$ gates for the quantum Fourier transform over each. This process is repeated $T = \mathrm{poly}(\log(1/n\epsilon\beta))$ times. Thus the entire algorithm uses $\tilde{O}(n)$ gates. □

2.2.2.2 Extension to non-smooth functions

Now consider a general L-Lipschitz continuous convex function f. We show that any such function is close to a smooth function, and we consider the relationship between the subgradients of the original function and the gradient of its smooth approximation.

For any $\delta > 0$, let $m_\delta\colon \mathbb{R}^n \to \mathbb{R}$ be the *mollifier function of width* δ, defined as

$$m_\delta(x) := \begin{cases} \frac{1}{I_n} \exp\left(-\frac{1}{1-\|x/\delta\|_2^2}\right) & x \in B_2(0,\delta) \\ 0 & \text{otherwise,} \end{cases} \tag{2.20}$$

where I_n is chosen such that $\int_{B_2(0,\delta)} m_\delta(x)\,\mathrm{d}^n x = 1$. The mollification of f, denoted $F_\delta := f * m_\delta$, is obtained by convolving it with the mollifier function, i.e.,

$$F_\delta(x) = (f * m_\delta)(x) = \int_{\mathbb{R}^n} f(x-y) m_\delta(y)\,\mathrm{d}^n x. \tag{2.21}$$

The mollification of f has several key properties, as follows:

Proposition 2.2.1. *Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be an L-Lipschitz convex function with mollification F_δ. Then*

(i) *F_δ is infinitely differentiable,*

(ii) *F_δ is convex,*

(iii) *F_δ is L-Lipschitz continuous, and*

(iv) *$|F_\delta(x) - f(x)| \le L\delta$.*

These properties of the mollifier function are well known in functional analysis [55]. For completeness a proof is provided in Lemma 2.5.2.

Lemma 2.2.4. *Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be an infinitely differentiable L-Lipschitz continuous convex function with mollification F_δ. Then any $\tilde{g}$ satisfying $\|\tilde{g} - \nabla F_\delta(y)\|_1 = \zeta$ for some $y \in B_\infty(x, r_1)$ satisfies*

$$f(q) \geq f(x) + \langle \tilde{g}, q - x\rangle - \zeta\|q - x\|_\infty - 4nr_1L - 2L\delta. \tag{2.22}$$

Here ζ is the error in the subgradient and δ is the parameter used in the mollifier function.

Proof. For all $q \in \mathbb{R}^n$, convexity of F_δ implies

$$\begin{aligned}
F_\delta(q) &\geq F_\delta(y) + \langle \nabla F_\delta(y), q - y\rangle && (2.23)\\
&= F_\delta(x) + \langle \nabla F_\delta(y), q - x\rangle + \langle \nabla F_\delta(y), x - y\rangle + (F_\delta(y) - F_\delta(x)) && (2.24)\\
&\geq F_\delta(x) + \langle \nabla F_\delta(y), q - x\rangle - 4nr_1L && (2.25)\\
&\geq F_\delta(x) + \langle \tilde{g}, q - x\rangle - \zeta\|q - x\|_\infty - 4nr_1L, && (2.26)
\end{aligned}$$

so (2.22) follows from Proposition 2.2.1(iv). □

Now consider δ such that $L\delta \ll \epsilon$. Then the evaluation oracle with error ϵ for f is also an evaluation oracle for F_δ with error $\epsilon + L\delta \approx \epsilon$. Thus the given evaluation oracle is also the evaluation oracle for an infinitely differentiable convex function with the same Lipschitz constant, with almost equal error, allowing us to analyze infinitely differentiable functions without loss of generality (as long as we make no claim about the second derivatives). This idea is made precise in Theorem 2.2.2. (Note that the mollification of f is never computed or estimated by our algorithm. It is only a tool for analysis.)

Unfortunately, Lemma 2.2.3 cannot be directly used to calculate subgradients for F_δ as $\delta \to 0$. Note that for the given evaluation oracle for f to also be an $\sim \epsilon$-evaluation oracle for F_δ, we must have $\delta \leq \epsilon$. Furthermore, there exist convex functions (such as $f(x) = |x|$) where if $|f(x) - g(x)| \leq \delta$ and $g(x)$ is β-smooth, then $\beta\delta \geq c$ for some constant c (see Lemma 2.5.3 in the final section). Thus using the `SmoothQuantumGradient` algorithm at $x = 0$ will give us a one-norm error of $3000n^{3/2}\sqrt{\epsilon\beta} \geq 3000n^{3/2}\sqrt{c}$, which is independent of ϵ.

To avoid this problem, we take inspiration from [41] and introduce classical randomness into the gradient evaluation. In particular, the following lemma shows that for a Lipschitz continuous function, if we sample at random from the neighborhood of any given point, the probability of having large second derivatives is small. Let $y \in_{\mathrm{R}} Y$ indicate that y is sampled uniformly at random from the set Y. Also, let $\lambda(x)$ be the largest eigenvalue of the Hessian matrix $\nabla^2 f(x)$ at x. Since the Hessian is positive semidefinite, we have $\lambda(x) \leq \Delta f(x) := \mathrm{Tr}(\nabla^2 f(x))$. Thus the second derivatives of a function are bounded by $\Delta f(x)$.

Lemma 2.2.5. *Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be a twice differentiable L-Lipschitz function. Then*

$$\mathbb{E}_{y\in_{\mathrm{R}} B_\infty(x,r_1)} \Delta f(y) \leq \frac{nL}{r_1}. \tag{2.27}$$

Proof. We have

$$\begin{aligned}
\mathbb{E}_{y\in_{\mathrm{R}} B_\infty(x,r_1)} \Delta f(y) &= \frac{1}{(2r_1)^n} \int\limits_{B_\infty(x,r_1)} \Delta f(y)\, \mathrm{d}^n y && (2.28)\\
&= \frac{1}{(2r_1)^n} \int\limits_{\partial B_\infty(x,r_1)} \langle \nabla f(y), \eta(y) \rangle\, \mathrm{d}^{n-1} y && (2.29)\\
&\leq \frac{1}{(2r_1)^n} (2n)(2r_1)^{n-1} L = \frac{nL}{r_1} && (2.30)
\end{aligned}$$

where (2.29) comes from the divergence theorem (the integral of the divergence of a vector field over a set is equal to the integral of the vector field over the surface of the set) and $\eta(y)$ is the area element on the surface $\partial B_\infty(x, r_1)$ defined as

$$\eta(y)_i = \begin{cases} 1 & \text{if } y_i - x_i \geq r_1 \\ 0 & \text{otherwise.} \end{cases} \tag{2.31}$$

$\square$

Using Markov's inequality with Lemma 2.2.5, we have

$$\Pr_{y \sim B_\infty(x,r_1)} \left[\Delta f(y) > \frac{pnL}{r_1} \right] \leq \frac{1}{p} \tag{2.32}$$

for $p > 1$. We use this fact to argue that at most points $y \in B_\infty(x, r_1)$, we can use the `SmoothQuantumGradient` procedure (with a second derivative bound $\beta_0 = pnL/r_1$) and obtain good estimates to the gradient (with error that monotonically decreases with ϵ).

From Lemma 2.2.3, we see that for `SmoothQuantumGradient` to be successful at a point y, the second derivative bound $\beta_0 = pnL/r_1$ must hold not only at y, but at every point $z \in B_\infty(y, l)$, where $l := 2\sqrt{\epsilon/\beta_0}$. Thus we wish to upper bound the probability that a point y lies in the l-neighborhood of the set of points with second derivatives greater than β_0. Specifically, we have the following.

Lemma 2.2.6. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be an L-Lipschitz convex function with $L \geq 1$. Suppose $n > 1$ and $\epsilon \in (0, 1)$. Then for any $r_1 > 0$,*

$$\Pr_{y \in_{\mathrm{R}} B_\infty(x,r_1)} \left[\exists\, z \in B_\infty(y, l), \Delta f(z) \geq \frac{16nL}{\epsilon^{1/3}} \right] \leq \frac{\epsilon^{1/3}}{8r_1} \tag{2.33}$$

where $l = \epsilon^{2/3}/2\sqrt{nL}$ (i.e., $l = 2\sqrt{\epsilon/\beta_0}$ with $\beta_0 = pnL/r_1$ and $p = 16r_1/\epsilon^{1/3}$).

Proof. We denote the measure of a set S by $\mathcal{M}(S)$. Consider $y \in_{\mathrm{R}} B_\infty(x, r_1)$. Then the probability that $y \in S \subseteq B_\infty(x, r_1)$ is $\mathcal{M}(S)/(2r_1)^n$. Let $B_\infty(S, l) := \{y \mid \exists\, z \in B_\infty(y, l), z \in S\}$. From the union bound,

$$\mathcal{M}(B_\infty(S,l)) \leq \mathcal{M}(S) + \mathcal{M}(S)\mathcal{M}(B_\infty(0,l)) = \big(1 + (2l)^n\big)\mathcal{M}(S). \tag{2.34}$$

Therefore, we have

$$\begin{aligned}
\Pr_{y \in_{\mathrm{R}} B_\infty(x,r_1)}[\exists\, z \in B_\infty(y,l), z \in S] &= \frac{\mathcal{M}(B_\infty(S,l))}{(2r_1)^n} && (2.35)\\
&\leq \big(1 + (2l)^n\big)\frac{\mathcal{M}(S)}{(2r_1)^n} && (2.36)\\
&= \big(1 + (2l)^n\big) \Pr_{y \in_{\mathrm{R}} B_\infty(x,r_1)}[y \in S]. && (2.37)
\end{aligned}$$

Considering $S = \{z \mid \Delta f(x) \geq pnL/r_1\}$ for any $p > 1$, combining (2.37) and (2.32) gives

$$\begin{aligned}
\Pr_{y \in_{\mathrm{R}} B_\infty(x,r_1)}\left[\exists\, z \in B_\infty(y,l), \Delta f(z) \geq \frac{pnL}{r_1}\right] &\leq \frac{1}{p} + \frac{(2l)^n}{p} && (2.38)\\
&= \frac{1}{p} + \frac{4^n}{p}\left(\frac{\epsilon}{\beta_0}\right)^{n/2} && (2.39)\\
&= \frac{1}{p} + \frac{1}{p}\left(\frac{16\epsilon r_1}{pnL}\right)^{n/2}. && (2.40)
\end{aligned}$$

Using the assumptions that $n > 1$, $\epsilon < 1$, and $L \geq 1$, we have

$$\Pr_{y \in_{\mathrm{R}} B_\infty(x,r_1)}\left[\exists\, z \in B_\infty(y,l), \Delta f(z) \geq \frac{pnL}{r_1}\right] \leq \frac{1}{p} + \frac{1}{p}\left(\frac{16r_1}{p}\right)^{n/2}. \tag{2.41}$$

Finally, with $p = 16r_1/\epsilon^{1/3}$, we have

$$\Pr_{y \in_{\mathrm{R}} B_\infty(x,r_1)} \left[\exists\, z \in B_\infty(y,l), \Delta f(z) \geq \frac{16nL}{\epsilon^{1/3}} \right] \leq \frac{\epsilon^{1/3}}{16r_1} + \frac{\epsilon^{1/3}\epsilon^{n/6}}{16r_1} \leq \frac{\epsilon^{1/3}}{8r_1} \tag{2.42}$$

as claimed. □

Thus we have shown that if we choose a point at random in the r_1-neighborhood of the given point x, every point in its l-neighborhood has small second derivatives with high probability. Note the assumption that $L \geq 1$ is without loss of generality since otherwise we could simply run the algorithm with $L = 1$.

```
Algorithm 3: QuantumSubgradient($f, \epsilon, L, x, r_1$)
    Data: Function $f$, evaluation error $\epsilon$, Lipschitz constant $L$, point $x \in \mathbb{R}^n$, length
          $r_1 > 0$.
  1 Sample $y \in_{\mathrm{R}} B_\infty(x, r_1)$;
  2 Output $\tilde{g} =$ SmoothQuantumGradient($f, \epsilon, L, 16nL/\epsilon^{1/3}, y$).
```

Now we are ready to show that Algorithm 3 produces a good approximate subgradient.

Theorem 2.2.2. *Let f be a convex, L-Lipschitz function that is specified by an evaluation oracle with error $\epsilon < \min\{1, 8192r_1^3\}$. Let $\tilde{g} =$ `QuantumSubgradient`(f, ϵ, L, x, r_1) (Algorithm 3). Then for all $q \in \mathbb{R}^n$,*

$$f(q) \geq f(x) + \langle \tilde{g}, q - x \rangle - \zeta \|q - x\|_\infty - 4nr_1L, \tag{2.43}$$

where $\mathbb{E}\zeta \leq L\epsilon^{1/3}\big(15000n^2 + \frac{n}{4r_1}\big)$.

Proof. Consider F_δ such that $L\delta \ll \epsilon$. From Proposition 2.2.1, F_δ is infinitely differentiable, convex, and L-Lipschitz. The given evaluation oracle for f is also an evaluation oracle for F_δ

with error $\epsilon_1 = \epsilon + L\delta \leq \epsilon$.

We have $\epsilon < 8192r_1^3$ and $\epsilon_1 < 4096r_1^3$, so $p = 16r_1/\epsilon_1^{1/3} > 1$. Thus by Lemma 2.2.6, an invocation of the algorithm $g = \texttt{SmoothQuantumGradient}(F_\delta, \epsilon_1, L, 16r_1/\epsilon_1^{1/3}, y)$ behaves correctly with probability at least $1 - \epsilon_1^{1/3}/8r_1$. Thus for each $i \in [n]$, we have:

1. With probability at least $1 - \epsilon_1^{1/3}/8r_1$,

$$\mathbb{E}|g_i - \nabla F_\delta(y)_i| \leq 3000\sqrt{\frac{16r_1 n^2 L\epsilon_1}{r_1\epsilon_1^{1/3}}} < 15000\epsilon_1^{1/3} n L^{1/2} \leq 15000\epsilon_1^{1/3} Ln. \tag{2.44}$$

2. With probability at most $\epsilon_1^{1/3}/8r_1$, the algorithm fails. From Lipschitz continuity, $|\nabla F_\delta(x)_i| \leq L$, and from Lemma 2.2.3, $|g_i| \leq L$. Therefore,

$$\mathbb{E}|g_i - \nabla F_\delta(y)_i| \leq 2L. \tag{2.45}$$

Finally, we have

$$\mathbb{E}_{y\in_{\mathrm{R}} B_\infty(x,r_1)}|g_i - \nabla F_\delta(y)_i| \leq 15000L\epsilon_1^{1/3} n + \frac{L\epsilon_1^{1/3}}{4r_1}, \tag{2.46}$$

so

$$\mathbb{E}_{y\in_{\mathrm{R}} B_\infty(x,r_1)}\|g - \nabla F_\delta(y)\|_1 \leq L\epsilon_1^{1/3}\left(15000n^2 + \frac{n}{4r_1}\right). \tag{2.47}$$

Thus from Lemma 2.2.4,

$$f(q) \geq f(x) + \langle g, q - x\rangle - \zeta\|q - x\|_\infty - 4nr_1L - 2L\delta \tag{2.48}$$

for all $q \in \mathbb{R}^n$ where $\mathbb{E}\zeta \leq L\epsilon_1^{1/3}\big(15000n^2 + \frac{n}{4r_1}\big)$. Now let $\delta \to 0$. Then $F_\delta \to f$, $\epsilon_1 \to \epsilon$, and

$g \to \tilde{g}$. Finally,

$$f(q) \geq f(x) + \langle \tilde{g}, q - x \rangle - \zeta \|q - x\|_\infty - 4nr_1 L \tag{2.49}$$

for all $q \in \mathbb{R}^n$, where $\mathbb{E}\zeta \leq L\epsilon^{1/3}\big(15000n^2 + \frac{n}{4r_1}\big)$. $\square$

2.2.3 Membership to separation

Algorithm 4: `SeparatingHalfspace`(K, p, ρ, δ)

```
Data: Convex set $K$ such that $B_2(0,r) \subset K \subset B_2(0,R)$, $\kappa = R/r$, $\delta$-precision
      membership oracle for $K$, point $p$.
1 if the membership oracle asserts that $p \in B_2(K,\delta)$ then
2     Output: $p \in B_2(K,\delta)$.
3 else if $p \notin B_2(0,R)$ then
4     Output: the halfspace $\{x \in \mathbb{R}^n \mid 0 > \langle x - p, p\rangle\}$.
5 else
6     Define $h_p(x)$ as in (2.50). The evaluation oracle for $h_p(x)$ for any $x \in B(0,r/2)$
        can be implemented to precision $\epsilon = 7\kappa\delta$ using $\log(1/\epsilon)$ queries to the
        membership oracle for $K$;
7     Compute $\tilde{g}$ = QuantumSubgradient$(h_p, \epsilon, L, 0, \frac{R^{1/2}\epsilon^{1/3}}{4\kappa^{1/2}})$;
8     Output: the halfspace
        $\{x \in \mathbb{R}^n \mid \big(100000R + 12R^{1/2} + 1\big)\, n^2\epsilon^{1/6}\kappa^{3/2}/\rho \geq \langle \tilde{g}, x - p\rangle\}$.
```

In this section we show how the approximate subgradient procedure Algorithm 3 fits into the reduction from separation to membership presented in [41]. We use the *height function* $h_p\colon \mathbb{R}^n \to \mathbb{R}$ defined in [41] for any vector $p \in \mathbb{R}^n$, as

$$h_p(x) = -\max\{t \in \mathbb{R} \mid x + t\hat{p} \in K\}. \tag{2.50}$$

, where $\hat{p}$ is the unit vector in the direction of p. The height function has the following properties:

Proposition 2.2.2 (Lemmas 11 and 12 of [41])**.** *Let $K \subset \mathbb{R}^n$ be a convex set with $B_2(0, r) \subseteq K \subseteq B_2(0, R)$ for some $R > r > 0$. Then for any $p \in \mathbb{R}^n$, the height function (2.50) satisfies*

(i) *$h_p(x)$ is convex,*

(ii) *$h_p(x) \le 0$ for all $x \in K$, and*

(iii) *for all $\delta > 0$, $h_p(x)$ is $\frac{R+\delta}{r-\delta}$-Lipschitz continuous for $x \in B_2(0, \delta)$.*

Now we are ready to analyze Algorithm 4.

Theorem 2.2.3. *Let $K \subset \mathbb{R}^n$ be a convex set such that $B_2(0, r) \subseteq K \subseteq B_2(0, R)$ for some $R > r > 0$. Let $\rho \in (0, 1)$ and $\delta \in (0, \min\{r/7\kappa, 1/7\kappa\})$. Then with probability at least $1 - \rho$,* `SeparatingHalfspace`(K, p, ρ, δ) *outputs a halfspace that contains K and not p.*

Proof. Since $\delta \le \min\{r/7\kappa, 1/7\kappa\}$, $\epsilon \le \min\{1, r\}$.

If $p \in B_2(K, \delta)$ the algorithm is trivially correct.

If $p \notin B_2(0, R)$, the algorithm outputs a halfspace that contains $B_2(0, R)$ (and therefore contains K), and not p.

Finally, suppose $p \notin B_2(K, -\delta)$ and $p \in B_2(0, R)$. Since $\epsilon \ge \delta$, $p \notin B_2(K, -\epsilon)$. The height function $h_p(x)$ is 3κ-Lipschitz for all $x \in B_2(0, r/2)$, where $\kappa := R/r$. Since $\epsilon < \min\{1, r\}$, we have $\epsilon < r$, so Theorem 2.2.2 implies

$$h_p(x) \ge h_p(0) + \langle \tilde{g}, x \rangle - \zeta \|x\|_\infty - 12nr_1\kappa \tag{2.51}$$

for any $x \in K$, where $\mathbb{E}\zeta \le 3\kappa\epsilon^{1/3}(15000n^2 + \frac{n}{4r_1})$.

Notice that $-p/\kappa \in K$ and $h_p(-p/\kappa) = h_p(0) - \frac{1}{\kappa}\|p\|_2$. From (2.51),

$$h_p(0) - \frac{1}{\kappa}\|p\|_2 \geq h_p(0) + \langle \tilde{g}, -p/\kappa \rangle - \frac{1}{\kappa}\zeta\|p\|_\infty - 12nr_1\kappa \tag{2.52}$$

so

$$\langle \tilde{g}, p \rangle \geq \|p\|_2 - \zeta\|p\|_\infty - 12nr_1\kappa^2. \tag{2.53}$$

As claimed in Line 6 of Algorithm 4, $h_p(x)$ can be evaluated with any precision ϵ such that $7\kappa\delta \leq \epsilon$ using $O(\log(1/\epsilon))$ queries to a membership oracle with error δ; the proof is deferred to Lemma 2.5.6.

Since the membership oracle returns a negative response $p \notin B_2(K, -\delta)$, and the error ϵ in $h_p(x)$ must be $\geq \delta$, $p \notin B_2(K, -\epsilon)$. We are also given that $B_2(0, r) \subseteq K$. Thus we have $\left(1 - \frac{\epsilon}{r}\right) K \subseteq B_2(K, -\epsilon)$. Thus,

$$h_p(0) \geq -\left(1 - \frac{\epsilon}{r}\right)\|p\|_2 \geq -\|p\|_2 + \epsilon\kappa. \tag{2.54}$$

From (2.51), (2.52), and (2.54), we have

$$\begin{aligned} h_p(x) &\geq \langle \tilde{g}, x - p - \zeta\|x\|_\infty - \zeta\|p\|_\infty - 12nr_1\kappa - 12nr_1\kappa^2 - \epsilon\kappa & (2.55) \\ &\geq \langle \tilde{g}, x - p \rangle - 2\zeta R - 24nr_1\kappa^2 - \epsilon\kappa, & (2.56) \end{aligned}$$

so $\langle \tilde{g}, x-p\rangle \leq \tilde{\zeta}$ for all $x \in K$, where

$$\mathbb{E}\tilde{\zeta} \leq 6R\kappa\epsilon^{1/3}\left(15000n^2 + \frac{n}{4r_1}\right) + 24nr_1\kappa^2 + \epsilon\kappa \tag{2.57}$$

$$\leq 90000Rn^2\epsilon^{1/3}\kappa + 12nR^{1/2}\epsilon^{1/6}\kappa^{3/2} + \epsilon\kappa \tag{2.58}$$

$$< (100000R + 12R^{1/2} + 1)n^2\epsilon^{1/6}\kappa^{3/2}. \tag{2.59}$$

Thus the result follows from Markov's inequality. □

Theorem 2.2.4. *Let $K \subset \mathbb{R}^n$ be a convex set with $B_2(0,r) \subseteq K \subseteq B_2(0,R)$ and $\kappa = R/r$ for some $R > r > 0$, and let $\eta > 0$ be fixed. Further suppose that $R, r, \kappa \in \text{poly}(n)$. Then a separating oracle for K with error η can be implemented using $\tilde{O}(1)$ queries to a membership oracle for K and $\tilde{O}(n)$ gates.*

Proof. Clearly, the unit vector in the direction $\tilde{g}$ (from Algorithm 4) determines a separating hyperplane given a point $p \notin B_2(K, -\epsilon)$.

From (2.52), we have

$$\langle \tilde{g}, p\rangle \geq \|p\|_2 - 3\kappa\epsilon^{1/3}\left(15000n^2 + \frac{n}{4r_1}\right)\|p\|_\infty. \tag{2.60}$$

Letting $3\epsilon^{1/3}\left(15000n^2 + \frac{n}{4r_1}\right) < \frac{1}{2\kappa^2}$, we have

$$\|\tilde{g}\|_2 R \geq r - \frac{R}{2\kappa} \;\Rightarrow\; \|g\| \geq \frac{1}{2\kappa}. \tag{2.61}$$

Thus, we have a separating oracle with error margin $\left(200000R + 24R^{1/2} + 2\right)n^2\epsilon^{1/6}\kappa^{5/2}\rho^{-1}$ and failure probability ρ. Setting $\rho = \left((200000R + 24R^{1/2} + 2)n^2\epsilon^{1/6}\kappa^{5/2}\right)^{1/2}$, we have a composite error of $(200000R + 24R^{1/2} + 2)n^2\epsilon^{1/6}\kappa^{5/2}$. To have error at most η, we take $\epsilon \leq \eta^6/\big((200000R + 24R^{1/2} + 1)^6 n^{12}\kappa^{15}\big)$.

We finally obtain

$$\delta = \frac{\epsilon}{7\kappa} \leq \frac{1}{7\kappa}\min\left\{\frac{\eta^6}{\left(200000R + 24R^{1/2} + 1\right)^6 n^{12}\kappa^{15}}, \frac{1}{216\kappa^6\left(15000n^2 + \frac{n}{4r_1}\right)^3}, r, 1\right\}. \tag{2.62}$$

Consequently, we have $\mathrm{SEP}_\eta = \tilde{O}(1)\,\mathrm{MEM}_\delta$, where

$$\delta = \frac{1}{7\kappa}\min\left\{\frac{\eta^6}{\left(200000R + 24R^{1/2} + 1\right)^6 n^{12}\kappa^{15}}, \frac{1}{216\kappa^6\left(15000n^2 + \frac{n}{8r_1}\right)^3}, r, 1\right\}. \tag{2.63}$$

Therefore, $1/\epsilon$ and $1/\delta$ are both $O(\mathrm{poly}(n))$. Implementing the evaluation oracle takes $\mathrm{poly}(\log(1/\epsilon))$ membership queries and a further $\tilde{O}(1)$ queries are used for the sub-gradient.

The evaluation requires $\tilde{O}(1/\epsilon)$ gates and `SmoothQuantumGradient` uses $n\,\mathrm{poly}(\log(1/\epsilon))$ gates. Thus a total of $\mathrm{poly}(\log(1/\eta))$ queries and $n\,\mathrm{poly}(\log(1/\eta))$ gates are used. $\square$

2.2.4 Separation to optimization

It is known that an optimization oracle for a convex set can be implemented in $\tilde{O}(n)$ queries to a separation oracle. Specifically, Theorem 15 of [41] states:

Theorem 2.2.5 (Separation to Optimization)**.** *Let K be a convex set satisfying $B_2(0, r) \subset K \subset B_2(0, R)$ and let $\kappa = 1/r$. For any $0 < \epsilon < 1$, with probability $1 - \epsilon$, we can compute $x \in B_2(K, \epsilon)$ such that $c^T x \leq \min_{x\in K} c^T x + \epsilon\|c\|_2$, using $O(n\log(n\kappa/\epsilon))$ queries to $\mathrm{SEP}_\eta(K)$,* *where* $\eta = \text{poly}(\epsilon/n\kappa)$*, and* $\tilde{O}(n^3)$ *arithmetic operations.*

From Theorem 2.2.5 and Theorem 2.2.4, we have the following result

Theorem 2.2.6 (Membership to Optimization). *Let* K *be a convex set satisfying* $B_2(0,r) \subset K \subset B_2(0,R)$ *and let* $\kappa = 1/r$*. For any* $0 < \epsilon < 1$*, with probability* $1-\epsilon$*, we can compute* $x \in B_2(K,\epsilon)$ *such that* $c^T x \leq \min_{x\in K} c^T x + \epsilon$*, using* $\tilde{O}(n)$ *queries to a membership oracle for* K *with error* δ*, where* $\delta = O(\text{poly}(\epsilon))$*, and* $\tilde{O}(n^3)$ *gates.*

Proof. Using Theorem 2.2.4 with $\eta = \text{poly}(\epsilon/n\kappa)$, each query to the separation oracle requires $\tilde{O}(1)$ queries to a membership oracle with error $\delta = O(\text{poly}(\epsilon))$. We make $\tilde{O}(n)$ separation queries and perform a further $\tilde{O}(n^3)$ arithmetic operations, so the result follows. □

Theorem 2.2.1 follows directly from Theorem 2.2.6.

2.3 Lower bound

In this section, we prove our quantum lower bound on convex optimization (Theorem 2.1.2). We prove separate lower bounds on membership queries (Section 2.3.1) and evaluation queries (Section 2.3.2). We then combine these lower bounds into a single optimization problem in Section 2.3.3, establishing Theorem 2.1.2.

2.3.1 Membership queries

In this subsection, we establish a membership query lower bound using a reduction from the following search-with-wildcards problem:

Theorem 2.3.1 ([52, Theorem 1])**.** *For any $s \in \{0,1\}^n$, let O_s be a wildcard oracle satisfying*

$$O_s|T\rangle|y\rangle|0\rangle = |T\rangle|y\rangle|Q_s(T,y)\rangle \tag{2.64}$$

for all $T \subseteq [n]$ and $y \in \{0,1\}^{|T|}$, where $Q_s(T,y) = \delta[s_{|T} = y]$. Then the bounded-error quantum query complexity of determining s is $O(\sqrt{n}\log n)$ and $\Omega(\sqrt{n})$.

We use Theorem 2.3.1 to give an $\Omega(\sqrt{n})$ lower bound on membership queries for convex optimization. Specifically, we consider the following *sum-of-coordinates optimization problem*:

Definition 2.3.1. *Let*

$$\mathcal{C}_s := \bigtimes_{i=1}^{n} [s_i - 2, s_i + 1], \qquad s_i \in \{0,1\} \ \forall\, i \in [n], \tag{2.65}$$

where $\bigtimes$ is the Cartesian product on different coordinates. In the sum-of-coordinates optimization *problem, the goal is to minimize*

$$f(x) = \sum_{i\in[n]} x_i \quad \text{s.t. } x \in \mathcal{C}_s. \tag{2.66}$$

Intuitively, Definition 2.3.1 concerns an optimization problem on a hypercube where the function is simply the sum of the coordinates, but the position of the hypercube is unknown. Note that the function f in (2.66) is convex and 1-Lipschitz continuous.

We prove the hardness of solving sum-of-coordinates optimization using its membership oracle:

Theorem 2.3.2. *Given an instance of the sum-of-coordinates optimization problem with members-*

hip oracle $O_{\mathcal{C}_s}$, it takes $\Omega(\sqrt{n})$ quantum queries to $O_{\mathcal{C}_s}$ to output an $\tilde{x} \in \mathcal{C}_s$ such that

$$f(\tilde{x}) \leq \min_{x \in \mathcal{C}_s} f(x) + \frac{1}{3}, \tag{2.67}$$

with success probability at least 0.9.

Proof. Assume that we are given an arbitrary string $s \in \{0,1\}^n$ together with the membership oracle $O_{\mathcal{C}_s}$ for the sum-of-coordinates optimization problem.

We prove that a quantum query to $O_{\mathcal{C}_s}$ can be simulated by a quantum query to the oracle O_s in (2.64) for search with wildcards. Consider an arbitrary point $x \in \mathbb{R}^n$ in the sum-of-coordinates problem. We partition $[n]$ into four sets:

$$T_{x,0} := \big\{i \in [n] \mid x_i \in [-2,-1)\big\} \tag{2.68}$$

$$T_{x,1} := \big\{i \in [n] \mid x_i \in (1,2]\big\} \tag{2.69}$$

$$T_{x,\text{mid}} := \big\{i \in [n] \mid x_i \in [-1,1]\big\} \tag{2.70}$$

$$T_{x,\text{out}} := \big\{i \in [n] \mid |x_i| > 2\big\}, \tag{2.71}$$

and denote $T_x := T_{x,0} \cup T_{x,1}$ and $y^{(x)} \in \{0,1\}^{|T_x|}$ such that

$$y_i^{(x)} = \begin{cases} 0 & \text{if } i \in T_{x,0} \\ 1 & \text{if } i \in T_{x,1}. \end{cases} \tag{2.72}$$

We prove that $O_{\mathcal{C}_s}(x) = Q_s(T_x, y^{(x)})$ if $T_{x,\text{out}} = \varnothing$, and $O_{\mathcal{C}_s}(x) = 0$ otherwise. On the one hand, if $O_{\mathcal{C}_s}(x) = 1$, we have $x \in \mathcal{C}_s$. Because for all $i \in [n]$, $x_i \in [s_i - 2, s_i + 1] \subset [-2, 2]$ for both $s_i = 0$ and $s_i = 1$, we must have $T_{x,\text{out}} = \varnothing$. Now consider any $i \in T_x$. If $i \in T_{x,0}$, then $x_i \in [-2, -1)$. Because $x_i \in [0-2, 0+1]$ and $x_i \notin [1-2, 1+1]$, we must have $s_i = 0$ since $x_i \in [s_i - 2, s_i + 1]$. Similarly, if $i \in T_{x,1}$, then we must have $s_i = 1$. As a result of (2.72), for all $i \in T_x$ we have $s_i = y_i^{(x)}$; in other words, $s_{|T_x} = y^{(x)}$ and $Q_s(T_x, y^{(x)}) = 1 = O_{\mathcal{C}_s}(x)$.

On the other hand, if $O_{\mathcal{C}_s}(x) = 0$, there exists an $i_0 \in [n]$ such that $x_{i_0} \notin [s_{i_0} - 2, s_{i_0} + 1]$. We must have $i_0 \notin T_{x,\text{mid}}$ because $[-1, 1] \subset [s_{i_0} - 2, s_{i_0} + 1]$ regardless of whether $s_{i_0} = 0$ or $s_{i_0} = 1$. Next, if $i_0 \in T_{x,\text{out}}$, then $T_{x,\text{out}} \neq \varnothing$ and we correctly obtain $O_{\mathcal{C}_s}(x) = 0$. The remaining cases are $i_0 \in T_{x,0}$ and $i_0 \in T_{x,1}$. If $i_0 \in T_{x,0}$, because $x_{i_0} \in [-2, -1) \subset [0-2, 0+1]$ and $x_{i_0} \notin [s_{i_0} - 2, s_{i_0} + 1]$, we must have $s_{i_0} = 1$, and thus $s_{|T_x} \neq y^{(x)}$ because $y_{i_0}^{(x)} = 0$ by (2.72). If $i_0 \in T_{x,1}$, we similarly have $s_{i_0} = 0$, $y_{i_0}^{(x)} = 1$, and thus $s_{|T_x} \neq y^{(x)}$. In both cases, $s_{|T_x} \neq y^{(x)}$, so $Q_s(T_x, y^{(x)}) = 0 = O_{\mathcal{C}_s}(x)$.

Therefore, we have established that $O_{\mathcal{C}_s}(x) = Q_s(T_x, y^{(x)})$ if $T_{x,\text{out}} = \varnothing$, and $O_{\mathcal{C}_s}(x) = 0$ otherwise. In other words, a quantum query to $O_{\mathcal{C}_s}$ can be simulated by a quantum query to O_s.

We next prove that a solution $\tilde{x}$ of the sum-of-coordinates problem satisfying (2.67) solves the search-with-wildcards problem in Theorem 2.3.1. Because $\min_{x \in \mathcal{C}_s} f(x) = \sum_{i=1}^n (s_i - 2)$, we have

$$f(\tilde{x}) = \sum_{i=1}^n \tilde{x}_i \leq \frac{1}{3} + \sum_{i=1}^n (s_i - 2). \tag{2.73}$$

On the one hand, for all $j \in [n]$ we have $\tilde{x}_j \geq s_j - 2$ since $\tilde{x} \in \mathcal{C}_s$; on the other hand, by (2.73) we have

$$\frac{1}{3}+\sum_{i=1}^{n}(s_i-2)\geq\sum_{i=1}^{n}\tilde{x}_i\geq\tilde{x}_j+\sum_{i\in[n],\ i\neq j}(s_i-2),\tag{2.74}$$

which implies $\tilde{x}_j \leq s_j - 2 + \frac{1}{3}$. In all,

$$\tilde{x}_i \in [s_i-2, s_i-2+\tfrac{1}{3}] \quad \forall\, i\in[n].\tag{2.75}$$

Define a rounding function $\mathrm{sgn}_{-3/2}\colon \mathbb{R}\to\{0,1\}$ as

$$\mathrm{sgn}_{-3/2}(z)=\begin{cases}0 & \text{if } z<-3/2\\ 1 & \text{otherwise.}\end{cases}\tag{2.76}$$

We prove that $\mathrm{sgn}_{-3/2}(\tilde{x}) = s$ (here $\mathrm{sgn}_{-3/2}$ is applied on all n coordinates, respectively). For all $i\in[n]$, if $s_i = 0$, then $\tilde{x}_i \in [-2, -\frac{5}{3}] \subset (-\infty, -\frac{3}{2})$ by (2.75), which implies $\mathrm{sgn}_{-3/2}(\tilde{x}_i) = 0$ by (2.76). Similarly, if $s_i = 1$, then $\tilde{x}_i \in [-1, -\frac{2}{3}] \subset (-\frac{3}{2}, +\infty)$ by (2.75), which implies $\mathrm{sgn}_{-3/2}(\tilde{x}_i) = 1$ by (2.76).

In all, if we can solve the sum-of-coordinates optimization problem with an $\tilde{x}$ satisfying (2.67), we can solve the search-with-wildcards problem. By Theorem 2.3.2, the search-with-wildcards problem has quantum query complexity $\Omega(\sqrt{n})$; since a query to the membership oracle $O_{\mathcal{C}_s}$ can be simulated by a query to the wildcard oracle O_s, we have established an $\Omega(\sqrt{n})$ quantum lower bound on the number of membership queries needed to solve the sum-of-coordinates optimization problem. $\square$

2.3.2 Evaluation queries

In this subsection, we establish an evaluation query lower bound by considering the following *max-norm optimization problem*:

Definition 2.3.2. *In the* max-norm optimization problem*, the goal is to minimize a function* $f_c\colon \mathbb{R}^n \to \mathbb{R}$ *satisfying*

$$f_c(x) = \max_{i\in[n]} |\pi(x_i) - c_i| + \Big(\sum_{i=1}^{n} |\pi(x_i) - x_i|\Big) \tag{2.77}$$

for some $c \in \{0,1\}^n$*, where* $\pi\colon \mathbb{R} \to [0,1]$ *is defined as*

$$\pi(x) = \begin{cases} 0 & \textit{if } x < 0 \\ x & \textit{if } 0 \le x \le 1 \\ 1 & \textit{if } x > 1. \end{cases} \tag{2.78}$$

Observe that for all $x \in [0,1]^n$, we have $f_c(x) = \max_{i\in[n]} |x_i - c_i|$. Intuitively, Definition 2.3.2 concerns an optimization problem under the max-norm (i.e., L_∞ norm) distance from c for all x in the unit hypercube $[0,1]^n$; for all x not in the unit hypercube, the optimizing function pays a penalty of the L_1 distance between x and its projection $\pi(x)$ onto the unit hypercube. The function f_c is 2-Lipschitz continuous with a unique minimum at $x = c$; we prove in Lemma 2.5.7 that f_c is convex.

We prove the hardness of solving max-norm optimization using its evaluation oracle:

Theorem 2.3.3. *Given an instance of the max-norm optimization problem with an evaluation* *oracle O_{f_c}, it takes $\Omega(\sqrt{n}/\log n)$ quantum queries to O_{f_c} to output an $\tilde{x} \in [0,1]^n$ such that*

$$f_c(\tilde{x}) \leq \min_{x \in [0,1]^n} f_c(x) + \frac{1}{3}, \tag{2.79}$$

with success probability at least 0.9.

The proof of Theorem 2.3.3 has two steps. First, we prove a weaker lower bound with respect to the precision of the evaluation oracle:

Lemma 2.3.1. *Suppose we are given an instance of the max-norm optimization problem with an evaluation oracle O_{f_c} that has precision $0 < \delta < 0.05$, i.e., f_c is provided with $\lceil \log_2(1/\delta) \rceil$ bits of precision. Then it takes $\Omega(\sqrt{n}/\log(1/\delta))$ quantum queries to O_{f_c} to output an $\tilde{x} \in [0,1]^n$ such that*

$$f_c(\tilde{x}) \leq \min_{x \in [0,1]^n} f_c(x) + \frac{1}{3}, \tag{2.80}$$

with success probability at least 0.9.

The second step simulates a perfectly precise query to f_c by a rough query:

Lemma 2.3.2. *One classical (resp., quantum) query to O_{f_c} with perfect precision can be simulated by one classical query (resp., two quantum queries) to O_{f_c} with precision $1/5n$.*

Theorem 2.3.3 simply follows from the two propositions above: by Lemma 2.3.2, we can assume that the evaluation oracle O_{f_c} has precision $1/5n$, so Lemma 2.3.1 implies that it takes $\Omega(\sqrt{n}/\log 5n) = \Omega(\sqrt{n}/\log n)$ quantum queries to O_{f_c} to output an $\tilde{x} \in [0,1]^n$ satisfying (2.79) with success probability 0.9.

The proofs of Lemma 2.3.1 and Lemma 2.3.2 are given in Section 2.3.2.1 and Section 2.3.2.2, respectively.

2.3.2.1 $\tilde{\Omega}(\sqrt{n})$ quantum lower bound on a low-precision evaluation oracle

Similar to the proof of Theorem 2.3.2, we also use Theorem 2.3.1 (the quantum lower bound on search with wildcards) to give a quantum lower bound on the number of evaluation queries required to solve the max-norm optimization problem.

Proof of 2.3.1. Assume that we are given an arbitrary string $c \in \{0,1\}^n$ together with the evaluation oracle O_{f_c} for the max-norm optimization problem. To show the lower bound, we reduce the search-with-wildcards problem to the max-norm optimization problem.

We first establish that an evaluation query to O_f can be simulated using wildcard queries on c. Notice that if we query an arbitrary $x \in \mathbb{R}^n$, by (2.77) we have

$$f_c(x) = \max_{i\in[n]} |\pi(x_i) - c_i| + \Big(\sum_{i=1}^{n} |\pi(x_i) - x_i|\Big) = f_c(\pi(x)) + \Big(\sum_{i=1}^{n} |\pi(x_i) - x_i|\Big) \qquad (2.81)$$

where $\pi(x) := (\pi(x_1), \ldots, \pi(x_n))$. In particular, the difference of $f_c(x)$ and $f_c(\pi(x))$ is an explicit function of x that is independent of c. Thus the query $O_{f_c}(x)$ can be simulated using one query to $O_{f_c}(\pi(x))$ where $\pi(x) \in [0,1]^n$. It follows that we can restrict ourselves without loss of generality to implementing evaluation queries for $x \in [0,1]^n$.

Now we consider a decision version of oracle queries to f_c, denoted $O_{f_{c,\text{dec}}}$, where

$f_{c,\text{dec}} \colon [0,1]^n \times [0,1] \to \{0,1\}$ such that

$$f_{c,\text{dec}}(x,t) = \delta[f_c(x) \leq t]. \tag{2.82}$$

(We restrict to $t \in [0,1]$ because $f_c(x) \in [0,1]$ always holds for $x \in [0,1]^n$.) Using binary search, a query to O_{f_c} with precision δ can be simulated by at most $\lceil \log_2(1/\delta) \rceil = O(\log 1/\delta)$ queries to the oracle $O_{f_{c,\text{dec}}}$.

Next, we prove that a query to $O_{f_{c,\text{dec}}}$ can be simulated by a query to the search-with-wildcards oracle O_c in (2.64). Consider an arbitrary query $(x,t) \in [0,1]^n \times [0,1]$ to $O_{f_{c,\text{dec}}}$. For convenience, we denote $J_{0,t} := [0,t]$, $J_{1,t} := [1-t,1]$, and

$$I_{0,t} := J_{0,t} - (J_{0,t} \cap J_{1,t}) \tag{2.83}$$

$$I_{1,t} := J_{1,t} - (J_{0,t} \cap J_{1,t}) \tag{2.84}$$

$$I_{\text{mid},t} := J_{0,t} \cap J_{1,t} \tag{2.85}$$

$$I_{\text{out},t} := [0,1] - (J_{0,t} \cup J_{1,t}). \tag{2.86}$$

We partition $[n]$ into four sets:

$$T_{x,0,t} := \{i \in [n] \mid x_i \in I_{0,t}\} \tag{2.87}$$

$$T_{x,1,t} := \{i \in [n] \mid x_i \in I_{1,t}\} \tag{2.88}$$

$$T_{x,\text{mid},t} := \{i \in [n] \mid x_i \in I_{\text{mid},t}\} \tag{2.89}$$

$$T_{x,\text{out},t} := \{i \in [n] \mid x_i \in I_{\text{out},t}\}. \tag{2.90}$$

The strategy here is similar to the proof of Theorem 2.3.2: $T_{x,\text{mid},t}$ corresponds to the coordinates such that $|x_i - c_i| \leq t$ regardless of whether $c_i = 0$ or 1 (and hence c_i does not influence whether or not $\max_{i\in[n]} |x_i - c_i| \leq t$); $T_{x,\text{out},t}$ corresponds to the coordinates such that $|x_i - c_i| > t$ regardless of whether $c_i = 0$ or 1 (so $\max_{i\in[n]} |x_i - c_i| > t$ provided $T_{x,\text{out},t}$ is nonempty); and $T_{x,0,t}$ (resp., $T_{x,1,t}$) corresponds to the coordinates such that $|x_i - c_i| \leq t$ only when $c_i = 0$ (resp., $c_i = 1$).

Denote $T_{x,t} := T_{x,0,t} \cup T_{x,1,t}$ and let $y^{(x,t)} \in \{0,1\}^{|T_{x,t}|}$ such that

$$y_i^{(x,t)} = \begin{cases} 0 & \text{if } i \in T_{x,0,t} \\ 1 & \text{if } i \in T_{x,1,t}. \end{cases} \tag{2.91}$$

We will prove that $O_{f_{c,\text{dec}}}(x) = Q_c(T_{x,t}, y^{(x,t)})$ if $T_{x,\text{out},t} = \varnothing$, and $O_{f_{c,\text{dec}}}(x) = 0$ otherwise.

On the one hand, if $O_{f_{c,\text{dec}}}(x) = 1$, we have $f_c(x) \leq t$. In other words, for all $i \in [n]$ we have $|x_i - c_i| \leq t$, which implies

$$x_i \in J_{c_i,t} \quad \forall\, i \in [n]. \tag{2.92}$$

Since $J_{c_i,t} \subseteq J_{0,t} \cup J_{1,t}$, we have $x_i \in J_{0,t} \cup J_{1,t}$ for all $i \in [n]$, and thus $T_{x,\text{out},t} = \varnothing$ by (2.86) and (2.90). Now consider any $i \in T_{x,t}$. If $i \in T_{x,0,t}$, then $x_i \in I_{0,t}$ by (2.87). By (2.83) we have $x_i \in J_{0,t}$ and $x_i \notin J_{1,t}$, and thus $c_i = 0$ by (2.92). Similarly, if $i \in T_{x,1,t}$, then we must have $c_i = 1$. As a result of (2.91), for all $i \in T_{x,t}$ we have $c_i = y_i^{(x,t)}$; in other words, $c_{|T_{x,t}} = y^{(x,t)}$ and $Q_c(T_{x,t}, y^{(x,t)}) = 1 = O_{f_{c,\text{dec}}}(x)$.

On the other hand, if $O_{f_{c,\text{dec}}}(x) = 0$, there exists an $i_0 \in [n]$ such that

$$x_{i_0} \notin J_{c_{i_0},t}. \tag{2.93}$$

Therefore, we must have $i_0 \notin T_{x,\text{mid},t}$ since (2.85) implies $I_{\text{mid},t} = J_{0,t} \cap J_{1,t} \subseteq J_{c_{i_0},t}$. Next, if $i_0 \in T_{x,\text{out},t}$, then $T_{x,\text{out},t} \neq \varnothing$ and we correctly obtain $O_{f_{c,\text{dec}}}(x) = 0$. The remaining cases are $i_0 \in T_{x,0,t}$ and $i_0 \in T_{x,1,t}$.

If $i_0 \in T_{x,0,t}$, then $y_{i_0}^{(x,t)} = 0$ by (2.91). By (2.87) we have $x_{i_0} \in I_{0,t}$, and by (2.83) we have $x_{i_0,t} \in J_{0,t}$ and $x_{i_0} \notin J_{1,t}$; therefore, we must have $c_{i_0} = 1$ by (2.93). As a result, $c_{|T_{x,t}} \neq y^{(x,t)}$ at i_0. If $i_0 \in T_{x,1,t}$, we similarly have $c_{i_0} = 0$, $y_{i_0}^{(x,t)} = 1$, and thus $c_{|T_{x,t}} \neq y^{(x,t)}$ at i_0. In either case, $c_{|T_{x,t}} \neq y^{(x,t)}$, and $Q_c(T_{x,t}, y^{(x,t)}) = 0 = O_{f_{c,\text{dec}}}(x)$.

We have established that $O_{f_{c,\text{dec}}}(x) = Q_c(T_{x,t}, y^{(x,t)})$ if $T_{x,\text{out},t} = \varnothing$, and $O_{f_{c,\text{dec}}}(x) = 0$ otherwise. In other words, a quantum query to $O_{f_{c,\text{dec}}}$ can be simulated by a quantum query to the search-with-wildcards oracle O_c. Together with the fact that a query to O_{f_c} with precision δ can be simulated by $O(\log 1/\delta)$ queries to $O_{f_{c,\text{dec}}}$, it can also be simulated by $O(\log 1/\delta)$ queries to O_c.

We next prove that a solution $\tilde{x}$ of the max-norm optimization problem satisfying (2.80) solves the search-with-wildcards problem in Theorem 2.3.1. Because $\min_{x\in[0,1]^n} f_c(x) = 0$, considering the precision of at most $\delta < 0.05$ we have

$$f_c(\tilde{x}) \leq \tfrac{1}{3} + \delta \leq 0.4. \tag{2.94}$$

In other words,

$$\tilde{x}_i \in [c_i - 0.4, c_i + 0.4] \quad \forall\, i \in [n]. \tag{2.95}$$

Similar to (2.76), we define a rounding function $\operatorname{sgn}_{1/2} \colon \mathbb{R} \to \{0,1\}$ as

$$\operatorname{sgn}_{1/2}(z) = \begin{cases} 0 & \text{if } z < 1/2 \\ 1 & \text{otherwise.} \end{cases} \tag{2.96}$$

We prove that $\operatorname{sgn}_{1/2}(\tilde{x}) = c$ (here $\operatorname{sgn}_{1/2}$ is applied coordinate-wise). For all $i \in [n]$, if $c_i = 0$, then $\tilde{x}_i \in [0, 0.4] \subset (-\infty, 1/2)$ by (2.95), which implies $\operatorname{sgn}_{1/2}(\tilde{x}_i) = 0$ by (2.96). Similarly, if $c_i = 1$, then $\tilde{x}_i \in [0.6, 1] \subset (1/2, +\infty)$ by (2.95), which implies $\operatorname{sgn}_{1/2}(\tilde{x}_i) = 1$ by (2.96).

We have shown that if we can solve the max-norm optimization problem with an $\tilde{x}$ satisfying (2.80), we can solve the search-with-wildcards problem. By Theorem 2.3.2, the search-with-wildcards problem has quantum query complexity $\Omega(\sqrt{n})$; since a query to the evaluation oracle O_{f_c} can be simulated by $O(\log 1/\delta)$ queries to the wildcard oracle O_c, we have established an $\Omega(\sqrt{n}/\log(1/\delta))$ quantum lower bound on the number of evaluation queries needed to solve the max-norm optimization problem. □

2.3.2.2 Discretization: simulating perfectly precise queries by low-precision queries

In this subsection we prove Lemma 2.3.2, which we rephrase more formally as follows. For our convenience, *the function f_c in (2.77) is abbreviated as f throughout this subsection.*

Lemma 2.3.3. *Assume that $\hat{f}\colon [0,1]^n \to [0,1]$ satisfies $|\hat{f}(x) - f(x)| \le \frac{1}{5n}\ \forall\, x \in [0,1]^n$. Then one classical (resp., quantum) query to O_f can be simulated by one classical query (resp., two quantum queries) to $O_{\hat{f}}$.*

To achieve this, we present an approach that we call *discretization*. Instead of considering queries on all of $[0,1]^n$, we only consider a discrete subset $D_n \subseteq [0,1]^n$ defined as

$$D_n := \bigl\{\chi(a,\pi) \mid a \in \{0,1\}^n \text{ and } \pi \in S_n\bigr\}, \tag{2.97}$$

where S_n is the symmetric group on $[n]$ and $\chi\colon \{0,1\}^n \times S_n \to [0,1]^n$ satisfies

$$\chi(a,\pi)_i = (1-a_i)\tfrac{\pi(i)}{2n+1} + a_i(1 - \tfrac{\pi(i)}{2n+1}) \quad \forall\, i \in [n]. \tag{2.98}$$

Observe that D_n is a subset of $[0,1]^n$.

Since $|S_n| = n!$ and there are 2^n choices for $a \in \{0,1\}^n$, we have $|D_n| = 2^n n!$. For example, when $n = 2$, we have

$$D_2 = \bigl\{\bigl(\tfrac{1}{5},\tfrac{2}{5}\bigr),\bigl(\tfrac{1}{5},\tfrac{3}{5}\bigr),\bigl(\tfrac{4}{5},\tfrac{2}{5}\bigr),\bigl(\tfrac{4}{5},\tfrac{3}{5}\bigr),\bigl(\tfrac{2}{5},\tfrac{1}{5}\bigr),\bigl(\tfrac{2}{5},\tfrac{4}{5}\bigr),\bigl(\tfrac{3}{5},\tfrac{1}{5}\bigr),\bigl(\tfrac{3}{5},\tfrac{4}{5}\bigr)\bigr\} \tag{2.99}$$

with $|D_2| = 2^2 \cdot 2! = 8$.

We denote the restriction of the oracle O_f to D_n by $O_{f|D_n}$, i.e.,

$$O_{f|D_n}|x\rangle|0\rangle = |x\rangle|f(x)\rangle \qquad \forall\, x \in D_n. \tag{2.100}$$

In fact, this restricted oracle entirely captures the behavior of the unrestricted function.

Lemma 2.3.4 (Discretization)**.** *A classical (resp., quantum) query to O_f can be simulated using one classical query (resp., two quantum queries) to $O_{f|D_n}$.*

Algorithm 5: Simulate one query to O_f using one query to $O_{f|D_n}$.

```
Input: $x \in [0,1]^n$;
Output: $f(x) \in [0,1]$;
1 Compute $b \in \{0,1\}^n$ and $\sigma \in S_n$ such that the $2n$ numbers
    $x_1, x_2, \ldots, x_n, 1-x_1, \ldots, 1-x_n$ are arranged in decreasing order as
    $$b_{\sigma(1)}x_{\sigma(1)} + (1-b_{\sigma(1)})(1-x_{\sigma(1)}) \geq \cdots \geq b_{\sigma(n)}x_{\sigma(n)} + (1-b_{\sigma(n)})(1-x_{\sigma(n)}) \geq (1-b_{\sigma(n)})x_{\sigma(n)} + b_{\sigma(n)}(1-x_{\sigma(n)}) \geq \cdots \geq (1-b_{\sigma(1)})x_{\sigma(1)} + b_{\sigma(1)}(1-x_{\sigma(1)}); \qquad (2.101)$$
2 Compute $x^* \in D_n$ such that $\chi(b, \sigma^{-1}) = x^*$ (where $\chi$ is defined in (2.98));
3 Query $f(x^*)$ and let $k^* = (2n+1)(1-f(x^*))$;
4 Return
    $$f(x) = \begin{cases} (1-b_{\sigma(n)})x_{\sigma(n)} + b_{\sigma(n)}(1-x_{\sigma(n)}) & \text{if } k^* = n+1 \\ b_{\sigma(k^*)}x_{\sigma(k^*)} + (1-b_{\sigma(k^*)})(1-x_{\sigma(k^*)}) & \text{otherwise.} \end{cases} \qquad (2.102)$$
```

We prove this proposition by giving an algorithm (Algorithm 5) that performs the simulation. The main idea is to compute $f(x)$ only using x and $f(x^*)$ for some $x^* \in D_n$. We observe that max-norm optimization has the following property: if two strings $x \in [0,1]^n$ and $x^* \in D_n$ satisfy that $x_1, \ldots, x_n, 1-x_1, \ldots, 1-x_n$ and $x_1^*, \ldots, x_n^*, 1-x_1^*, \ldots, 1-x_n^*$ have the same ordering, then

$$\arg\max_{i \in [n]} |x_i - c_i| = \arg\max_{i \in [n]} |x_i^* - c_i|. \qquad (2.103)$$

Furthermore, $x^* \in D_n$ promises that $\{x_1^*, \ldots, x_n^*, 1-x_1^*, \ldots, 1-x_n^*\} = \{\frac{1}{2n+1}, \ldots, \frac{2n}{2n+1}\}$ are $2n$ different numbers, and hence knowing the value of $f(x^*)$ implies the value of the $\arg\max$ above (denoted i^*) and the corresponding c_{i^*}; we can subsequently recover $f(x)$ given x since

$f(x) = |x_{i^*} - c_{i^*}|$. In other words, $f(x)$ can be computed given x and $f(x^*)$. Moreover, $f(x^*)$ is an integer multiple of $\frac{1}{2n+1}$; even if $f(x^*)$ can only be computed with precision $\frac{1}{5n}$, we can round it to the closest integer multiple of $\frac{1}{2n+1}$ which is exactly $f(x^*)$, since the distance $\frac{2n+1}{5n} < \frac{1}{2}$. As a result, we can precisely compute $f(x^*)$ for all $x \in D_n$, and thus we can precisely compute $f(x)$.

We illustrate Algorithm 5 by a simple example. For convenience, we define an order function $\mathrm{Ord}\colon [0,1]^n \to \{0,1\}^n \times S_n$ by $\mathrm{Ord}(x) = (b, \sigma)$ for all $x \in [0,1]^n$, where b and σ satisfy Eq. (2.101).

An example with $n = 3$. Consider the case where the ordering in (2.101) is

$$1 - x_3 \geq x_1 \geq x_2 \geq 1 - x_2 \geq 1 - x_1 \geq x_3. \tag{2.104}$$

Then Algorithm 5 proceeds as follows:

- Line 1: With the ordering (2.104), we have $\sigma(1) = 3$, $\sigma(2) = 1$, $\sigma(3) = 2$; $b_3 = 0$, $b_1 = 1$, $b_2 = 1$.

- Line 2: The point $x^* \in D_3$ that we query given $\mathrm{Ord}(x)$ satisfies $1 - x_3^* = 6/7$, $x_1^* = 5/7$, $x_2^* = 4/7$, $1 - x_2^* = 3/7$, $1 - x_1^* = 2/7$, and $x_3^* = 1/7$; in other words, $x^* = (5/7, 4/7, 1/7)$.

- Line 3: Now we query $f(x^*)$. Since $f(x^*)$ is a multiple of $1/7$ and $f(x^*) \in [1/7, 6/7]$, there are only 6 possibilities: $f(x^*) = 6/7$, $f(x^*) = 5/7$, $f(x^*) = 4/7$, $f(x^*) = 3/7$, $f(x^*) = 2/7$, or $f(x^*) = 1/7$.

After running Line 1, Line 2, and Line 3, we have a point x^* from the discrete set D_3 such that $\mathrm{Ord}(x) = \mathrm{Ord}(x^*)$. Since they have the same ordering and $|x_i - c_i|$ is either x_i or

$1 - x_i$ for all $i \in [3]$, the function value $f(x^*)$ should essentially reflect the value of $f(x)$; this is made precise in Line 4.

- Line 4: Depending on the value of $f(x^*)$, we have six cases:
  - $f(x^*) = 6/7$: In this case, we must have $c_3 = 1$, so that $|x_3 - c_3| = |1/7 - 1| = 6/7$ ($|x_1 - c_1|$ can only give 5/7 or 2/7, and $|x_2 - c_2|$ can only give 4/7 or 3/7). Because $1 - x_3$ is the largest in (2.104), we must have $f(x) = 1 - x_3$.
  - $f(x^*) = 5/7$: In this case, we must have $c_1 = 0$, so that $|x_1 - c_1| = |5/7 - 0| = 5/7$. Furthermore, we must have $c_3 = 1$ (otherwise if $c_3 = 0$, $f(x) \geq |x_3 - c_3| = 6/7$). As a result of (2.104), we must have $f(x) = x_1$ since $x_1 \geq x_3$ and $x_1 \geq \max\{x_2, 1-x_2\}$.
  - $f(x^*) = 4/7$: In this case, we must have $c_2 = 0$, so that $|x_2 - c_2| = |4/7 - 0| = 4/7$. Furthermore, we must have $c_3 = 1$ (otherwise if $c_3 = 0$, $f(x) \geq |x_3 - c_3| = 6/7$) and $c_1 = 1$ (otherwise if $c_1 = 0$, $f(x) \geq |x_1 - c_1| = 5/7$). As a result of (2.104), we must have $f(x) = x_2$ since $x_2 \geq 1 - x_1 \geq 1 - x_3$.
  - $f(x^*) = 3/7$: In this case, we must have $c_2 = 1$, so that $|x_2 - c_2| = |4/7 - 1| = 3/7$. Furthermore, we must have $c_3 = 1$ (otherwise if $c_3 = 0$, $f(x) \geq |x_3 - c_3| = 6/7$) and $c_1 = 1$ (otherwise if $c_1 = 0$, $f(x) \geq |x_1 - c_1| = 5/7$). As a result of (2.104), we must have $f(x) = 1 - x_2$ since since $1 - x_2 \geq 1 - x_1 \geq 1 - x_3$.
  - $f(x^*) = 2/7$ or $f(x^*) = 1/7$: This two cases are impossible because $f(x^*) \geq |x_2 - c_2| = |4/7 - c_2| \geq 3/7$, no matter $c_2 = 0$ or $c_2 = 1$.

While Algorithm 5 is a classical algorithm for querying O_f using a query to $O_{f|D_n}$, it is straightforward to perform this computation in superposition using standard techniques to obtain a quantum query to O_f. However, note that this requires two queries to a quantum oracle for $O_{f|D_n}$ since we must uncompute $f(x^*)$ after computing $f(x)$.

Having the discretization technique at hand, Lemma 2.3.3 is straightforward.

Proof of Lemma 2.3.3. Recall that $|\hat{f}(x) - f(x)| \leq \frac{1}{5n}\ \forall x \in [0,1]^n$. We run Algorithm 5 to compute $f(x)$ for the queried value of x, except that in Line 3 we take $k^* = \lceil (2n{+}1)(1{-}\hat{f}(x^*)) \rfloor$ (here $\lceil a \rfloor$ is the closest integer to a). Because $|\hat{f}(x^*) - f(x^*)| \leq \frac{1}{5n}$, we have

$$\big|(2n+1)(1-\hat{f}(x^*)) - (2n+1)(1-f(x^*))\big| = (2n+1)|\hat{f}(x^*) - f(x^*)| \leq \tfrac{2n+1}{5n} < \tfrac{1}{2}; \tag{2.105}$$

as a result, $k^* = (2n{+}1)(1{-}f(x^*))$ because the latter is an integer (see Lemma 2.5.9). Therefore, due to the correctness of Algorithm 5 established in Section 2.5.3, and noticing that the evaluation oracle is only called at Line 3 (with the replacement described above), we successfully simulate one query to O_f by one query to $O_{\hat{f}}$ (actually, to $O_{\hat{f}|D_n}$). $\square$

The full analysis of Algorithm 5 is deferred to Section 2.5.3. In particular,

- In Section 2.5.3 we prove that the discretized vector x^* obtained in Line 2 is a good approximation of x in the sense that $\mathrm{Ord}(x^*) = \mathrm{Ord}(x)$;
- In Section 2.5.3 we prove that the value k^* obtained in Line 3 satisfies $k^* \in \{1, \ldots, n+1\}$;
- In Section 2.5.3 we finally prove that the output returned in Line 4 is correct.

2.3.3 Proof of full lower bound

We now prove Theorem 2.1.2 using Theorem 2.3.2 and Theorem 2.3.3. Recall that our lower bounds on membership and evaluation queries are both proved on the n-dimensional hypercube. It remains to combine the two lower bounds to establish them simultaneously.

Theorem 2.3.4. *Let $\mathcal{C}_s := \bigtimes_{i=1}^{n}[s_i - 2, s_i + 1]$ for some $s \in \{0,1\}^n$. Consider a function $f\colon \mathcal{C}_s \times [0,1]^n \to \mathbb{R}$ such that $f(x) = f_{\mathrm{M}}(x) + f_{\mathrm{E},c}(x)$, where for any $x = (x_1, x_2, \ldots, x_{2n}) \in \mathcal{C}_s \times [0,1]^n$,*

$$f_{\mathrm{M}}(x) = \sum_{i=1}^{n} x_i, \qquad f_{\mathrm{E},c}(x) = \max_{i\in\{n+1,\ldots,2n\}} |x_i - c_{i-n}| \tag{2.106}$$

for some $c \in \{0,1\}^n$. Then outputting an $\tilde{x} \in \mathcal{C}_s \times [0,1]^n$ satisfying

$$f(\tilde{x}) \leq \min_{x\in\mathcal{C}_s\times[0,1]^n} f(x) + \tfrac{1}{3} \tag{2.107}$$

with success probability at least 0.8 requires $\Omega(\sqrt{n})$ quantum queries to $O_{\mathcal{C}_s\times[0,1]^n}$ and $\Omega(\sqrt{n}/\log n)$ quantum queries to O_f.

Notice that the dimension of the optimization problem above is $2n$ instead of n; however, the constant overhead of 2 does not influence the asymptotic lower bounds.

Proof of Theorem 2.3.4. First, we prove that

$$\min_{x\in\mathcal{C}_s\times[0,1]^n} f(x) = S \qquad \text{and} \qquad \arg\min_{x\in\mathcal{C}_s\times[0,1]^n} f(x) = (s - 2_n, c), \tag{2.108}$$

where 2_n is the n-dimensional all-twos vector and $S := \sum_{i=1}^{n}(s_i - 2)$. On the one hand,

$$f_{\mathrm{M}}(x) \geq S \qquad \forall\, x \in \mathcal{C}_s \times [0,1]^n, \tag{2.109}$$

with equality if and only if $(x_1, \ldots, x_n) = s - 2_n$. On the other hand,

$$f_{\mathrm{E},c}(x) \geq 0 \qquad \forall\, x \in \mathcal{C}_s \times [0,1]^n, \tag{2.110}$$

with equality if and only if $(x_{n+1}, \ldots, x_{2n}) = c$. Thus $f(x) = f_{\mathrm{M}}(x) + f_{\mathrm{E},c}(x) \geq S$ for all $x \in \mathcal{C}_s \times [0,1]^n$, with equality if and only if $x = (x_1, \ldots, x_n, x_{n+1}, \ldots, x_{2n}) = (s - 2_n, c)$.

If we can solve this optimization problem with an output $\tilde{x}$ satisfying (2.107), then

$$f_{\mathrm{M}}(\tilde{x}) + f_{\mathrm{E},c}(\tilde{x}) = f(\tilde{x}) \leq S + \tfrac{1}{3}. \tag{2.111}$$

Eqs. (2.109), (2.110), and (2.111) imply

$$f_{\mathrm{M}}(\tilde{x}) \leq S + \tfrac{1}{3} = \min_{x \in \mathcal{C}_s \times [0,1]^n} f_{\mathrm{M}}(x) + \tfrac{1}{3}; \tag{2.112}$$

$$f_{\mathrm{E},c}(\tilde{x}) \leq \tfrac{1}{3} = \min_{x \in \mathcal{C}_s \times [0,1]^n} f_{\mathrm{E},c}(x) + \tfrac{1}{3}. \tag{2.113}$$

On the one hand, Eq. (2.112) says that $\tilde{x}$ also minimizes f_{M} with approximation error $\epsilon = \frac{1}{3}$. By Theorem 2.3.2, this requires $\Omega(\sqrt{n})$ queries to the membership oracle $O_{\mathcal{C}_s}$. Also notice that one query to $O_{\mathcal{C}_s \times [0,1]^n}$ can be trivially simulated one query to $O_{\mathcal{C}_s}$; therefore, minimizing f with approximation error $\epsilon = \frac{1}{3}$ with success probability 0.9 requires $\Omega(\sqrt{n})$ quantum queries to $O_{\mathcal{C}_s \times [0,1]^n}$.

On the other hand, Eq. (2.113) says that $\tilde{x}$ minimizes $f_{\mathrm{E},c}$ with approximation error $\epsilon = \frac{1}{3}$. By Theorem 2.3.3, it takes $\Omega(\sqrt{n}/\log n)$ queries to $O_{f_{\mathrm{E},c}}$ to output $\tilde{x}$. Also notice that

$$f(x) = f_{\mathrm{M}}(x) + f_{\mathrm{E},c}(x) = \sum_{i=1}^{n} x_i + f_{\mathrm{E},c}(x); \tag{2.114}$$

therefore, one query to O_f can be simulated by one query to $O_{f_{\mathrm{E},c}}$. Therefore, approximately minimizing f with success probability 0.9 requires $\Omega(\sqrt{n}/\log n)$ quantum queries to O_f.

In addition, f_{M} is independent of the coordinates $x_{n+1}, \ldots, x_{2n}$ and only depends on the coordinates $x_1, \ldots, x_n$, whereas $f_{\mathrm{E},c}$ is independent of the coordinates $x_1, \ldots, x_n$ and only depends on the coordinates $x_{n+1}, \ldots, x_{2n}$. As a result, the oracle $O_{\mathcal{C}_s \times [0,1]^n}$ reveals no information about c, and O_f reveals no information about s. Since solving the optimization problem reveals both s and c, the lower bounds on query complexity must hold simultaneously.

Overall, to output an $\tilde{x} \in \mathcal{C}_s \times [0,1]^n$ satisfying (2.107) with success probability at least $0.9 \cdot 0.9 > 0.8$, we need $\Omega(\sqrt{n})$ quantum queries to $O_{\mathcal{C}_s \times [0,1]^n}$ and $\Omega(\sqrt{n}/\log n)$ quantum queries to O_f, as claimed. □

2.3.4 Smoothed hypercube

As a side point, our quantum lower bound in Theorem 2.3.4 also holds for a smooth convex body. Given an n-dimensional hypercube $\mathcal{C}_{x,l} := \bigtimes_{i=1}^{n} [x_i - l, x_i]$, we define a smoothed version as

$$\mathcal{SC}_{x,l} := B_2\bigg(\bigtimes_{i=1}^{n} \Big[x_i - \frac{2n}{2n+1} l, x_i - \frac{1}{2n+1} l\Big], \frac{1}{2n+1} l\bigg) \tag{2.115}$$

using Definition 2.2.3. For instance, a smoothed 3-dimensional cube is shown in Figure 2.1.

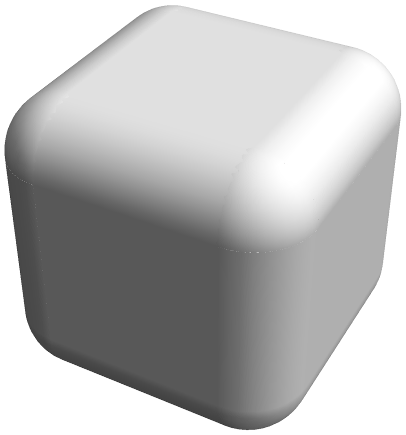


Figure 2.1: Smoothed hypercube of dimension 3.

The smoothed hypercube satisfies

$$\mathcal{C}_{x-\frac{1}{2n+1}l_n,\frac{2n-1}{2n+1}l} \subseteq \mathcal{SC}_{x,l} \subseteq \mathcal{C}_{x,l} \tag{2.116}$$

where l_n is l times the n-dimensional all-ones vector; in other words, it is contained in the original (non-smoothed) hypercube, and it contains the hypercube with the same center but edge length $\frac{2n-1}{2n+1}l$. For instance, $\bigtimes_{i=1}^{n}[\frac{1}{2n+1},\frac{2n}{2n+1}] \subseteq \mathcal{SC}_{1_n,1} \subseteq \bigtimes_{i=1}^{n}[0,1]$; by Eq. (2.97), $D_n \subseteq \mathcal{SC}_{1_n,1}$. It can be verified that the proof of Theorem 2.3.2 still holds if the hypercube $\bigtimes_{i=1}^{n}[s_i-2,s_i+1] = \mathcal{C}_{s+1_n,3}$ is replaced by $\mathcal{SC}_{s+1_n,3}$, and the proof of Theorem 2.3.3 still holds if the unit hypercube $[0,1]^n$ is replaced by $\mathcal{SC}_{1_n,1}$; consequently Theorem 2.3.4 also holds. More generally, the proofs remain valid as long as the smoothed hypercube is contained in $[0,1]^n$ and contains D_n (for discretization).

2.4 Conclusions

This chapter has presented quantum algorithms for convex optimization that are quadratically faster than the best known classical algorithms [41] in terms of the membership and evaluation oracles required. The corresponding lower bounds show that despite these speedups, no exponential quantum speedups are possible for this problem.

Related independent work. In independent simultaneous work, van Apeldoorn, Gilyén, Gribling, and de Wolf [56] establish a similar upper bound, showing that $\tilde{O}(n)$ quantum queries to a membership oracle suffice to optimize a linear function over a convex body (i.e., to implement an optimization oracle). Their proof follows a similar strategy to ours, using a quantum algorithm for evaluating gradients in $\tilde{O}(1)$ queries to implement a separation oracle. Those authors also establish quantum lower bounds on the query complexity of convex optimization, showing in particular that $\Omega(\sqrt{n})$ quantum queries to a separation oracle are needed to implement an optimization oracle, implying an $\Omega(\sqrt{n})$ quantum lower bound on the number of membership queries required to optimize a convex function. While Ref. [56] does not explicitly focus on evaluation queries, those authors have pointed out to us that an $\Omega(\sqrt{n})$ lower bound on evaluation queries can be obtained from their lower bound on membership queries, using a careful application of techniques inspired by [41] (although our approach gives a bound with a better Lipschitz parameter).

Open questions. This work leaves several natural open questions for future investigation. In particular:

- Can we close the gap for both membership and evaluation queries? Our upper bounds on both oracles in Theorem 2.1.1 uses $\tilde{O}(n)$ queries, whereas the lower bounds of Theorem 2.1.2 are only $\tilde{\Omega}(\sqrt{n})$.

- Can we improve the time complexity of our quantum algorithm? The time complexity $\tilde{O}(n^3)$ of our current quantum algorithm matches that of the classical state-of-the-art algorithm [41] since our second step, the reduction from optimization to separation, is entirely classical. Is it possible to improve this reduction quantumly?

- What is the quantum complexity of convex optimization with a first-order oracle (i.e., with direct access to the gradient of the objective function)? This model has been widely considered in the classical literature (see for example Ref. [53]).

2.5 Deferred Technical Details

2.5.1 Auxiliary lemmas

Classical gradient computation. Here we prove that the classical query complexity of gradient computation is linear in the dimension.

Lemma 2.5.1. *Let f be an L-Lipschitz convex function that is specified by an evaluation oracle with precision $\delta = 1/\operatorname{poly}(n)$. Any (deterministic or randomized) classical algorithm to calculate a subgradient of f with L_∞-norm error $\epsilon = 1/\operatorname{poly}(n)$ must make $\tilde{\Omega}(n)$ queries to the evaluation oracle.*

Proof. Consider the linear function $f(x) = c^T x$ where each $c_i \in [0,1]$. Since each c_i must be determined to precision ϵ, the problem hides $n \log(1/\epsilon)$ bits of information. Furthermore, since the evaluation oracle has precision δ, each query reveals only $\log(1/\delta)$ bits of information. Thus any classical algorithm must make at least $\frac{n\log(1/\epsilon)}{\log(1/\delta)} = n/\log(n)$ evaluation queries. $\square$

Mollified functions. The following lemma establishes properties of mollified functions:

Lemma 2.5.2 (Mollifier properties). *Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be an L-Lipschitz convex function with mollification $F_\delta = f * m_\delta$, where m_δ is defined in (2.20). Then*

(i) *F_δ is infinitely differentiable,*

(ii) *F_δ is convex,*

(iii) *F_δ is L-Lipschitz continuous, and*

(iv) *$|F_\delta(x) - f(x)| \leq L\delta$.*

Proof.

(i) Convolution satisfies $\frac{\mathrm{d}(p*q)}{\mathrm{d}x} = p * \frac{\mathrm{d}q}{\mathrm{d}x}$, so because m_δ is infinitely differentiable, F_δ is infinitely differentiable.

(ii) We have $F_\delta(x) = \int_{\mathbb{R}^n} f(x-z)m_\delta(z)\,\mathrm{d}z = \int_{\mathbb{R}^n} f(z)m_\delta(x-z)\,\mathrm{d}z$. Thus

$$F_\delta(\lambda x + (1-\lambda)y) = \int\limits_{\mathbb{R}^n} f(\lambda x + (1-\lambda)y - z)m_\delta(z)\,\mathrm{d}z \tag{2.117}$$

$$\geq \int\limits_{\mathbb{R}^n} [\lambda f(x-z) + (1-\lambda)f(y-z)]m_\delta(z)\,\mathrm{d}z \tag{2.118}$$

$$= \lambda F_\delta(x) + (1-\lambda)F_\delta(y), \tag{2.119}$$

where the inequality holds by convexity of f and the fact that $m_\delta \geq 0$. Thus F_δ is convex.

(iii) We have

$$\|F_\delta(x) - F_\delta(y)\| = \|\int\limits_{\mathbb{R}^n} [f(x-z) - f(y-z)] m_\delta(z)\,\mathrm{d}z\| \tag{2.120}$$

$$\leq \int\limits_{\mathbb{R}^n} \|f(x-z) - f(y-z)\| m_\delta(z)\,\mathrm{d}z \tag{2.121}$$

$$\leq L\|x-y\| \int\limits_{\mathbb{R}^n} m_\delta(z)\,\mathrm{d}z \tag{2.122}$$

$$= L\|x-y\|. \tag{2.123}$$

Thus from Definition 2.2.9, F_δ is L-Lipschitz.

(iv) We have

$$|F_\delta(x) - f(x)| = \left| \int\limits_{\mathbb{R}^n} f(x-z) g(z)\,\mathrm{d}z - \int\limits_{\mathbb{R}^n} f(x) g(z)\,\mathrm{d}z \right| \tag{2.124}$$

$$\leq \int\limits_{\mathbb{R}^n} |f(x-z) - f(z)|\, g(z)\,\mathrm{d}z \tag{2.125}$$

$$\leq L \int\limits_{\mathbb{R}^n} |z|\, g(z)\,\mathrm{d}z \tag{2.126}$$

$$= L \int\limits_{B_2(0,\delta)} \frac{|z|}{I_n} \exp\left(-\frac{1}{1 - \|z/\delta\|^2}\right)\,\mathrm{d}z \tag{2.127}$$

$$= L\delta \int\limits_{B_2(0,1)} \frac{|u|}{I_n} \exp\left(-\frac{1}{1 - \|u\|^2}\right)\,\mathrm{d}u \tag{2.128}$$

$$\leq L\delta \int\limits_{B_2(0,1)} \frac{1}{I_n} \exp\left(-\frac{1}{1 - \|u\|^2}\right)\,\mathrm{d}u \tag{2.129}$$

$$= L\delta \tag{2.130}$$

as claimed.

□

The following lemma shows strong convexity of mollified functions, ruling out the possibility of directly applying Lemma 2.5.5 to calculate subgradients.

Lemma 2.5.3. *There exists a* 1*-Lipschitz convex function* f *such that for any* β*-smooth function* g *with* $|f(x) - g(x)| \leq \delta$ *for all* x*,* $\beta\delta \geq c$ *where* c *is a constant.*

Proof. Let $f(x) = |x|$. Consider $x \geq 0$. By the smoothness of g,

$$g(x) \leq g(0) + \nabla g(0)^T x + \frac{\beta}{2}x^2, \tag{2.131}$$

$$g(-x) \leq g(0) - \nabla g(0)^T x + \frac{\beta}{2}x^2. \tag{2.132}$$

As a result, we have $g(x) + g(-x) \leq 2g(0) + \beta x^2$ for all $x > 0$. Since $|f(x) - g(x)| \leq \delta$,

$$f(x) + f(-x) \leq 2f(0) + \beta x^2 + 4\delta \quad \Rightarrow \quad \beta x^2 - 2x + 4\delta \geq 0 \tag{2.133}$$

for all $x > 0$.

Since $4\delta > 0$, the discriminant must be non-positive. Therefore, $16 - 16\beta\delta \leq 0$, so $\beta\delta \geq 1$. □

2.5.2 Proof details for upper bound

We give the complete proof of Lemma 2.2.2 in this section.

Given a quantum oracle that computes the function N_0F in the form

$$U_F|x\rangle|y\rangle = |x\rangle|y \oplus (N_0F(x) \mod N)\rangle, \tag{2.134}$$

it is well known that querying U_F with

$$|y_0\rangle = \frac{1}{\sqrt{N_0}} \sum_{i\in\{0,1,\dots,N-1\}} e^{\frac{2\pi ix}{N_0}}|i\rangle \tag{2.135}$$

allows us to implement the phase oracle O_F in one query. This is a common technique used in quantum algorithms known as *phase kickback*.

First, we prove the following lemma:

Lemma 2.5.4. *Let* $G := \{-N/2, -N/2+1, \dots, N/2-1\}$ *and define* $\gamma\colon \{0,1,\dots,N-1\} \to G$ *by* $\gamma(x) = x - N/2$ *for all* $x \in \{0,1,\dots,N-1\}$*. Consider the inverse quantum Fourier transforms*

$$\mathrm{QFT}_N^{-1}|x\rangle := \frac{1}{\sqrt{N}} \sum_{y\in[0,N-1]} e^{-\frac{2\pi ixy}{N}}|y\rangle, \quad \forall\, x \in [0, N-1]; \tag{2.136}$$

$$\mathrm{QFT}_G^{-1}|\gamma(x)\rangle := \frac{1}{\sqrt{N}} \sum_{\gamma(y)\in G} e^{-\frac{2\pi i\gamma(x)\gamma(y)}{N}}|\gamma(y)\rangle, \quad \forall\, \gamma(x) \in G \tag{2.137}$$

over $[0, N-1] := \{0,1,\dots,N-1\}$ *and G, respectively. Then we have* $\mathrm{QFT}_G^{-1} = U\,\mathrm{QFT}_N^{-1}\,U$*, where* U *is a tensor product of* $b = \log_2 N$ *single-qubit unitaries.*

Proof. For any $x \in [0, N-1]$, we have

$$\mathrm{QFT}_G^{-1}|x\rangle = \frac{1}{\sqrt{N}} \sum_{y \in [0,N-1]} e^{-\frac{2\pi i \gamma(x)\gamma(y)}{N}} |y\rangle \tag{2.138}$$

which is equivalent to

$$\frac{1}{\sqrt{N}} \sum_{y \in [0,N-1]} e^{-\frac{2\pi i x y}{N}} e^{\pi i (x+y)} |y\rangle \tag{2.139}$$

up to a global phase. Setting $U|x\rangle = e^{\pi i x}|x\rangle$ for all $x \in \{0, 1, \ldots, N-1\}$, we have the result. □

The above shows that we can implement QFT_G^{-1} on a single b-bit register using $O(b)$ gates. Thus there is no significant overhead in gate complexity that results from using QFT_G instead of the usual QFT.

Now we prove Lemma 2.2.2, which is rewritten below:

Lemma 2.5.5. *Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be an L-Lipschitz function that is specified by an evaluation oracle with error at most ϵ. Let f be β-smooth in $B_\infty(x, 2\sqrt{\epsilon/\beta})$, and let $\tilde{g}$ be the output of* `GradientEstimate`$(f, \epsilon, L, \beta, x_0)$ *(from Algorithm 1). Let $g = \nabla f(x_0)$. Then*

$$\Pr\left[|\tilde{g}_i - g_i| > 1500\sqrt{n\epsilon\beta}\right] < \frac{1}{3}, \quad \forall\, i \in [n]. \tag{2.140}$$

Proof. To analyze the `GradientEstimate` algorithm, let the actual state obtained before applying the inverse QFT over G be

$$|\psi\rangle = \frac{1}{N^{n/2}} \sum_{x \in G^d} e^{2\pi i \tilde{F}(x)} |x\rangle, \tag{2.141}$$

where $|\tilde{F}(x) - \frac{N}{2Ll}[f(x_0 + \frac{lx}{N}) - f(x_0)]| \leq \frac{1}{N_0}$. Also consider the idealized state

$$|\phi\rangle = \frac{1}{N^{n/2}} \sum_{x \in G^d} e^{\frac{2\pi i g \cdot x}{2L}} |x\rangle. \tag{2.142}$$

From Lemma 2.5.4 we can efficiently apply the inverse QFT over G; from the analysis of phase estimation (see [57]), we know that

$$\forall\, i \in [n] \quad \Pr\left[\left|\frac{Ng_i}{2L} - k_i\right| > w\right] < \frac{1}{2(w-1)}, \tag{2.143}$$

so in particular,

$$\forall\, i \in [n] \quad \Pr\left[\left|\frac{Ng_i}{2L} - k_i\right| > 4\right] < \frac{1}{6}. \tag{2.144}$$

Now, let $g = \nabla f(x_0)$. The difference in the probabilities of any measurement on $|\psi\rangle$ and $|\phi\rangle$ is bounded by the trace distance between the two density matrices, which is

$$\| |\psi\rangle\langle\psi| - |\phi\rangle\langle\phi| \|_1 = 2\sqrt{1 - |\langle\psi|\phi\rangle|^2} \leq 2\| |\psi\rangle - |\phi\rangle \|. \tag{2.145}$$

Since f is β-smooth in $B_\infty(x, 2\sqrt{\frac{\epsilon}{\beta}})$,

$$\tilde{F}(x) \leq \frac{N}{2Ll}\left[f\left(x_0 + \frac{lx}{N}\right) - f(x_0)\right] + \frac{1}{N_0} \tag{2.146}$$

$$\leq \frac{N}{2Ll}\left(\frac{l}{N}\nabla f(x_0) \cdot x + \frac{\beta l^2 x^2}{2N^2}\right) + \frac{1}{N_0} \tag{2.147}$$

$$\leq \frac{1}{2L}\nabla f(x_0) \cdot x + \frac{N\beta ln}{4L} + \frac{N\epsilon}{Ll}. \tag{2.148}$$

Then we have

$$\||\psi\rangle - |\phi\rangle\|^2 = \frac{1}{N^d}\sum_{x \in G_d} |e^{2\pi i \tilde{F}(x)} - e^{\frac{2\pi i g.x}{2L}}|^2 \tag{2.149}$$

$$= \frac{1}{N^d}\sum_{x \in G_d} 4\pi^2 \left(\tilde{F}(x) - \frac{g \cdot x}{2L}\right)^2 \tag{2.150}$$

$$\leq \frac{1}{N^d}\sum_{x \in G_d} \frac{4\pi^2 N^2}{L^2}\left(\frac{\beta ln}{4} + \frac{\epsilon}{l}\right)^2 \tag{2.151}$$

$$= \frac{4\pi^2 N^2 \beta \epsilon n}{L^2}. \tag{2.152}$$

In Algorithm 1, N is chosen such that $N \leq \frac{L}{24\pi\sqrt{n\epsilon\beta}}$. Plugging this into (2.149),

$$\||\psi\rangle - |\phi\rangle\|^2 \leq \frac{1}{144}. \tag{2.153}$$

Thus the trace distance is at most $\frac{1}{6}$. Therefore, $\Pr\left[\left|k_i - \frac{Ng_i}{2L}\right| > 4\right] < \frac{1}{3}$. Thus we have

$$\Pr\left[|\tilde{g}_i - g_i| > \frac{8L}{N}\right] < \frac{1}{3}, \quad \forall i \in [n]. \tag{2.154}$$

From Algorithm 1, $\frac{1}{N} \leq \frac{48\pi\sqrt{n\epsilon\beta}}{L}$, so $\frac{8L}{N} < 384\pi\sqrt{n\epsilon\beta} < 1500\sqrt{n\epsilon\beta}$, and the result follows. □

Finally, we prove that the height function h_p can be evaluated with precision ϵ using $O(\log(1/\epsilon))$ queries to a membership oracle:

Lemma 2.5.6. *The function $h_p(x)$ can be evaluated for any $x \in B_\infty(0, r/2)$ with any precision $\epsilon \geq 7\kappa\delta$ using $O(\log(1/\epsilon))$ queries to a membership oracle with error δ.*

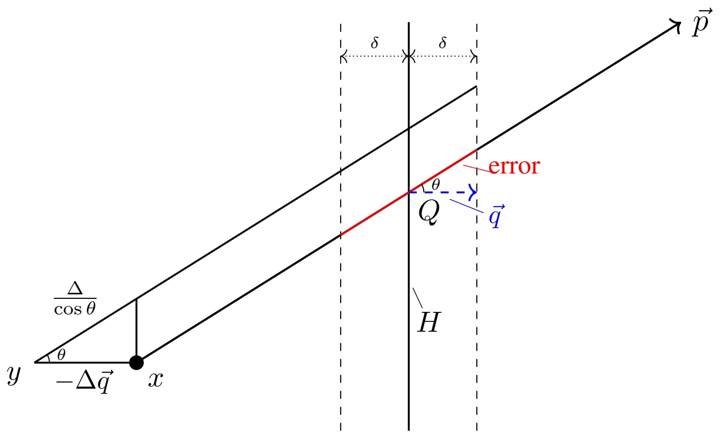


Figure 2.2: Relating the error to 2δ in $n = 2$ dimensions.

Proof. We denote the intersection of the ray $x + t\vec{p}$ and the boundary of K by Q, and let H be an $(n-1)$-dimensional hyperplane that is tangent to K at Q. Because K is convex, it lies on only one side of H; we let $\vec{q}$ denote the unit vector at Q that is perpendicular to H and points out of K. Let $\theta := \arccos\langle\vec{p}, \vec{q}\rangle$.

Using binary search with $\log(1/\delta)$ queries, we can find a point P on the ray $x + t\vec{p}$ such that $P \notin B(K, -\delta)$ and $P \in B(K, \delta)$. The total error for t is then at most $\frac{2\delta}{\cos\theta}$. Now consider $y = x - \Delta\vec{q}$ for some small $\Delta > 0$. Then $h_p(y) - h_p(x) = \frac{\Delta}{\cos\theta} + o(\frac{\Delta}{\cos\theta})$ (see Figure 2.2 for an illustration with $n = 2$).

By Proposition 2.2.2, $h_p(x)$ is 3κ-Lipschitz for any $x \in B(0, r/2)$; therefore, $h_p(y) -$

$h_p(x) \leq 3\kappa\|y - x\| = 3\kappa\Delta$, and hence

$$\frac{\Delta}{\cos\theta} + o\Big(\frac{\Delta}{\cos\theta}\Big) \leq 3\kappa\Delta \quad \Rightarrow \quad \frac{1}{\cos\theta} \leq 3.5\kappa \tag{2.155}$$

for a small enough $\Delta > 0$. Thus the error in $h_p(x)$ is at most $\frac{2\delta}{\cos\theta} \leq 7\kappa\delta$, and the result follows. □

2.5.3 Proof details for lower bound

In this section, we give proof details for our claims in Section 2.3.2.

Convexity of max-norm optimization. In this subsection, we prove:

Lemma 2.5.7. *The function*

$$f_c(x) = \max_{i\in[n]} |\pi(x_i) - c_i| + \Big(\sum_{i=1}^{n} |\pi(x_i) - x_i|\Big) \tag{2.156}$$

is convex on $\mathbb{R}^n$, where $c \in \{0,1\}^n$ and $\pi \colon \mathbb{R} \to [0,1]$ is defined as

$$\pi(x) = \begin{cases} 0 & \textit{if } x < 0 \\ x & \textit{if } 0 \leq x \leq 1 \\ 1 & \textit{if } x > 1. \end{cases} \tag{2.157}$$

Proof. For convenience, we define $g_i \colon \mathbb{R}^n \to \mathbb{R}$ for $i \in [n]$ as

$$g_i(x) := |\pi(x_i) - x_i| = \begin{cases} -x_i & \text{if } x_i < 0 \\ 0 & \text{if } 0 \le x_i \le 1 \\ x_i - 1 & \text{if } x_i > 1 \end{cases} \tag{2.158}$$

where the second equality follows from (2.157). From (2.158), it is clear that $g_i(x) = \max\{-x_i, 0, x_i - 1\}$. Since the pointwise maximum of convex functions is convex, $g_i(x)$ is convex for all $i \in [n]$.

Moreover, for all $i \in [n]$ we define $h_{c,i} \colon \mathbb{R}^n \to \mathbb{R}$ as $h_{c,i}(x) := |\pi(x_i) - c_i| + |\pi(x_i) - x_i|$. We claim that $h_{c,i}(x) = |x_i - c_i|$, and thus $h_{c,i}$ is convex. If $c_i = 0$, then $|\pi(x_i) - c_i| + |\pi(x_i) - x_i| = \pi(x_i) + |\pi(x_i) - x_i|$; as a result,

$$x_i < 0 \quad \Rightarrow \quad \pi(x_i) + |\pi(x_i) - x_i| = 0 + |0 - x_i| = -x_i; \tag{2.159}$$

$$0 \le x_i \le 1 \quad \Rightarrow \quad \pi(x_i) + |\pi(x_i) - x_i| = x_i + |x_i - x_i| = x_i; \tag{2.160}$$

$$x_i > 1 \quad \Rightarrow \quad \pi(x_i) + |\pi(x_i) - x_i| = 1 + |1 - x_i| = x_i. \tag{2.161}$$

Therefore, $\forall\, i \in [n], h_{c,i}(x) = |x_i - c_i|$. The proof is similar when $c_i = 1$.

Now we have

$$f_c(x) = \max_{i\in[n]} \left(|\pi(x_i) - c_i| + \sum_{j=1}^{n} |\pi(x_j) - x_j| \right) \tag{2.162}$$

$$= \max_{i\in[n]} \left(\left(|\pi(x_i) - c_i| + |\pi(x_i) - x_i| \right) + \sum_{j\neq i} g_j(x) \right) \tag{2.163}$$

$$= \max_{i\in[n]} \left(h_{c,i}(x) + \sum_{j\neq i} g_j(x) \right). \tag{2.164}$$

Because $h_{c,i}$ and g_j are both convex functions on $\mathbb{R}^n$ for all $i, j \in [n]$, the function $h_{c,i}(x) + \sum_{j\neq i} g_j(x)$ is convex on $\mathbb{R}^n$. Thus f_c is the pointwise maximum of n convex functions and is therefore itself convex. □

Proof of Lemma 2.3.4.

Correctness of Line 1 and Line 2. In this subsection, we prove:

Lemma 2.5.8. *Let b and σ be the values computed in Line 1 of Algorithm 5, and let $x^* = \chi(b, \sigma^{-1})$. Then* $\mathrm{Ord}(x^*) = \mathrm{Ord}(x)$.

Proof. First, observe that $b \in \{0,1\}^n$ and $\sigma \in S_n$ because

- For all $i \in [n]$, both x_i and $1 - x_i$ can be written as $b_i x_i + (1 - b_i)(1 - x_i)$ for some $b_i \in \{0, 1\}$;
- $\mathrm{Ord}(x)$ is *palindrome*, i.e., if x_{i_1} is the largest in $\{x_1, \dots, x_n, 1 - x_1, \dots, 1 - x_n\}$ then $1 - x_{i_1}$ is the smallest in $\{x_1, \dots, x_n, 1 - x_1, \dots, 1 - x_n\}$; if $1 - x_{i_2}$ is the second largest in $\{x_1, \dots, x_n, 1 - x_1, \dots, 1 - x_n\}$ then x_{i_2} is the second smallest in $\{x_1, \dots, x_n, 1 - x_1, \dots, 1 - x_n\}$; etc.

Recall that in (2.101), the decreasing order of $\{x_1, \ldots, x_n, 1 - x_1, \ldots, 1 - x_n\}$ is

$$b_{\sigma(1)}x_{\sigma(1)} + (1 - b_{\sigma(1)})(1 - x_{\sigma(1)}) \geq \cdots \geq b_{\sigma(n)}x_{\sigma(n)} + (1 - b_{\sigma(n)})(1 - x_{\sigma(n)})$$
$$\geq (1 - b_{\sigma(n)})x_{\sigma(n)} + b_{\sigma(n)}(1 - x_{\sigma(n)}) \geq \cdots \geq (1 - b_{\sigma(1)})x_{\sigma(1)} + b_{\sigma(1)}(1 - x_{\sigma(1)}). \quad (2.165)$$

On the other hand, by the definition of D_n, we have

$$\{x_1^*, \ldots, x_n^*, 1 - x_1^*, \ldots, 1 - x_n^*\} = \left\{\frac{1}{2n+1}, \frac{2}{2n+1}, \ldots, \frac{2n}{2n+1}\right\}. \quad (2.166)$$

Combining (2.165) and (2.166), it suffices to prove that for any $i \in [n]$,

$$b_{\sigma(i)}x_{\sigma(i)}^* + (1 - b_{\sigma(i)})(1 - x_{\sigma(i)}^*) = 1 - \frac{i}{2n+1}; \quad (2.167)$$
$$(1 - b_{\sigma(i)})x_{\sigma(i)}^* + b_{\sigma(i)}(1 - x_{\sigma(i)}^*) = \frac{i}{2n+1}. \quad (2.168)$$

We only prove (2.167); the proof of (2.168) follows symmetrically.

By (2.98), we have $x_j^* = (1 - b_j)\frac{\sigma^{-1}(j)}{2n+1} + b_j(1 - \frac{\sigma^{-1}(j)}{2n+1})$ for all $j \in [n]$; taking $j = \sigma(i)$, we have $x_{\sigma(i)}^* = (1 - b_{\sigma(i)})\frac{i}{2n+1} + b_{\sigma(i)}(1 - \frac{i}{2n+1})$. Moreover, since $b_{\sigma(i)} \in \{0, 1\}$ implies that

$b_{\sigma(i)}(1-b_{\sigma(i)})=0$ and $b_{\sigma(i)}^2+(1-b_{\sigma(i)})^2=1$, we have

$$\begin{aligned} b_{\sigma(i)}x^*_{\sigma(i)}+(1-b_{\sigma(i)})(1-x^*_{\sigma(i)}) &= b_{\sigma(i)}\left[(1-b_{\sigma(i)})\tfrac{i}{2n+1}+b_{\sigma(i)}\left(1-\tfrac{i}{2n+1}\right)\right] \\ &\quad +(1-b_{\sigma(i)})\left[b_{\sigma(i)}\tfrac{i}{2n+1}+(1-b_{\sigma(i)})\left(1-\tfrac{i}{2n+1}\right)\right] \qquad (2.169) \\ &= 2b_{\sigma(i)}(1-b_{\sigma(i)})\tfrac{i}{2n+1}+\left(b_{\sigma(i)}^2+(1-b_{\sigma(i)})^2\right)\left(1-\tfrac{i}{2n+1}\right) \qquad (2.170) \\ &= 1-\tfrac{i}{2n+1}, \qquad (2.171) \end{aligned}$$

which is exactly (2.167). □

Correctness of Line 3. In this subsection, we prove:

Lemma 2.5.9. *There is some $k^*\in\{1,\dots,n+1\}$ such that $f(x^*)=1-\frac{k^*}{2n+1}$.*

Proof. Because $|x^*_i-c_i|$ is an integer multiple of $\frac{1}{2n+1}$ for all $i\in[n]$, $f(x^*)$ must also be an integer multiple of $\frac{1}{2n+1}$. As a result, $k^*=(2n+1)(1-f(x^*))\in\mathbb{Z}$.

It remains to prove that $1\le k^*\le n+1$. By the definition of D_n in (2.97), we have

$$x^*_i=(1-b_i)\frac{\sigma^{-1}(i)}{2n+1}+b_i\left(1-\frac{\sigma^{-1}(i)}{2n+1}\right) \qquad \forall\, i\in[n]; \tag{2.172}$$

since $b_i=0$ or 1, we have $x^*_i\in\{\frac{\sigma^{-1}(i)}{2n+1},1-\frac{\sigma^{-1}(i)}{2n+1}\}$. Because we also have $c_i\in\{0,1\}$,

$$|x^*_i-c_i|\le 1-\frac{\sigma^{-1}(i)}{2n+1}\le\frac{2n}{2n+1}. \tag{2.173}$$

As a result,

$$f(x^*) = \max_{i\in[n]} |x_i^* - c_i| \leq \frac{2n}{2n+1} \;\Rightarrow\; k^* \geq 1. \tag{2.174}$$

It remains to prove $k^* \leq n+1$. By (2.172), we have

$$x^*_{\sigma(n)} \in \left\{\frac{n}{2n+1}, \frac{n+1}{2n+1}\right\}; \tag{2.175}$$

because $c_{\sigma(n)} \in \{0,1\}$, we have

$$|x^*_{\sigma(n)} - c_{\sigma(n)}| \geq \frac{n}{2n+1}. \tag{2.176}$$

Therefore, we have

$$f(x^*) = \max_{i\in[n]} |x_i^* - c_i| \geq |x^*_{\sigma(n)} - c_{\sigma(n)}| \geq \frac{n}{2n+1}, \tag{2.177}$$

which implies $k^* \leq n+1$. □

Correctness of Line 4. In this subsection, we prove:

Lemma 2.5.10. *The output of $f(x)$ in Line 4 is correct.*

Proof. A key observation we use in the proof, following directly from (2.172), is that

$$|x^*_{\sigma(i)} - c_{\sigma(i)}| = \begin{cases} \frac{i}{2n+1} & \text{if } c_{\sigma(i)} = b_{\sigma(i)}; \\ 1 - \frac{i}{2n+1} & \text{if } c_{\sigma(i)} = 1 - b_{\sigma(i)}. \end{cases} \tag{2.178}$$

First, assume that $k^* \in \{1, \ldots, n\}$ (i.e., the "otherwise" case in (2.102) happens). By (2.178),

$$x^*_{\sigma(k^*)} \in \left\{\frac{k^*}{2n+1}, 1 - \frac{k^*}{2n+1}\right\}; \quad x^*_{\sigma(i)} \notin \left\{\frac{k^*}{2n+1}, 1 - \frac{k^*}{2n+1}\right\} \quad \forall\, i \neq k^*, \tag{2.179}$$

which implies that for all $i \neq k^*$, $|x^*_{\sigma(i)} - c_{\sigma(i)}| \neq 1 - \frac{k^*}{2n+1}$ since $c_{\sigma(i)} \in \{0, 1\}$. As a result, we must have

$$|x^*_{\sigma(k^*)} - c_{\sigma(k^*)}| = 1 - \frac{k^*}{2n+1}. \tag{2.180}$$

Together with (2.178), this implies

$$c_{\sigma(k^*)} = 1 - b_{\sigma(k^*)}. \tag{2.181}$$

For any $i < k^*$, if $c_{\sigma(i)} = 1 - b_{\sigma(i)}$, then (2.178) implies that

$$f(x^*) \geq |x^*_{\sigma(i)} - c_{\sigma(i)}| = 1 - \frac{i}{2n+1} > 1 - \frac{k^*}{2n+1}, \tag{2.182}$$

which contradicts with the assumption that $f(x^*) = 1 - \frac{k^*}{2n+1}$. Therefore, we must have

$$c_{\sigma(i)} = b_{\sigma(i)} \quad \forall\, i \in \{1, \ldots, k^* - 1\}. \tag{2.183}$$

Recall that the decreasing order of $\{x_1, \ldots, x_n, 1-x_1, \ldots, 1-x_n\}$ is

$$\begin{aligned} & b_{\sigma(1)}x_{\sigma(1)} + (1-b_{\sigma(1)})(1-x_{\sigma(1)}) \geq \cdots \geq b_{\sigma(n)}x_{\sigma(n)} + (1-b_{\sigma(n)})(1-x_{\sigma(n)}) \\ & \quad \geq (1-b_{\sigma(n)})x_{\sigma(n)} + b_{\sigma(n)}(1-x_{\sigma(n)}) \geq \cdots \geq (1-b_{\sigma(1)})x_{\sigma(1)} + b_{\sigma(1)}(1-x_{\sigma(1)}). \end{aligned} \tag{2.184}$$

Based on (2.181), (2.183), and (2.184), we next prove

$$|x_{\sigma(k^*)} - c_{\sigma(k^*)}| \geq |x_{\sigma(i)} - c_{\sigma(i)}| \quad \forall\, i \in [n]. \tag{2.185}$$

If (2.185) holds, it implies

$$f(x) = \max_{i \in [n]} |x_i - c_i| = |x_{\sigma(k^*)} - c_{\sigma(k^*)}|. \tag{2.186}$$

If $b_{\sigma(k^*)} = 0$, then (2.181) implies $c_{\sigma(k^*)} = 1$, (2.186) implies $f(x) = 1 - x_{\sigma(k^*)}$, and the output in Line 4 satisfies

$$b_{\sigma(k^*)}x_{\sigma(k^*)} + (1-b_{\sigma(k^*)})(1-x_{\sigma(k^*)}) = 1 - x_{\sigma(k^*)} = f(x); \tag{2.187}$$

If $b_{\sigma(k^*)} = 1$, then (2.181) implies $c_{\sigma(k^*)} = 0$, (2.186) implies $f(x) = x_{\sigma(k^*)}$, and the output in Line 4 satisfies

$$b_{\sigma(k^*)}x_{\sigma(k^*)} + (1-b_{\sigma(k^*)})(1-x_{\sigma(k^*)}) = x_{\sigma(k^*)} = f(x). \tag{2.188}$$

The correctness of Line 4 follows.

It remains to prove (2.185). We divide its proof into two parts:

- Suppose $i < k^*$. By (2.184), we have

$$b_{\sigma(k^*)}x_{\sigma(k^*)} + (1 - b_{\sigma(k^*)})(1 - x_{\sigma(k^*)}) \geq (1 - b_{\sigma(i)})x_{\sigma(i)} + b_{\sigma(i)}(1 - x_{\sigma(i)}). \tag{2.189}$$

  - If $b_{\sigma(k^*)} = 0$ and $b_{\sigma(i)} = 0$, we have $c_{\sigma(k^*)} = 1$ and $c_{\sigma(i)} = 0$ by (2.181) and (2.183), respectively; (2.189) reduces to $1 - x_{\sigma(k^*)} \geq x_{\sigma(i)}$;
  - If $b_{\sigma(k^*)} = 0$ and $b_{\sigma(i)} = 1$, we have $c_{\sigma(k^*)} = 1$ and $c_{\sigma(i)} = 1$ by (2.181) and (2.183), respectively; (2.189) reduces to $1 - x_{\sigma(k^*)} \geq 1 - x_{\sigma(i)}$;
  - If $b_{\sigma(k^*)} = 1$ and $b_{\sigma(i)} = 0$, we have $c_{\sigma(k^*)} = 0$ and $c_{\sigma(i)} = 0$ by (2.181) and (2.183), respectively; (2.189) reduces to $x_{\sigma(k^*)} \geq x_{\sigma(i)}$;
  - If $b_{\sigma(k^*)} = 1$ and $b_{\sigma(i)} = 1$, we have $c_{\sigma(k^*)} = 0$ and $c_{\sigma(i)} = 1$ by (2.181) and (2.183), respectively; (2.189) reduces to $x_{\sigma(k^*)} \geq 1 - x_{\sigma(i)}$.

  In each case, the resulting expression is exactly (2.185). Overall, we see that (2.185) is always true when $i < k^*$.

- Suppose $i > k^*$. By (2.184), we have

$$b_{\sigma(k^*)}x_{\sigma(k^*)} + (1 - b_{\sigma(k^*)})(1 - x_{\sigma(k^*)}) \geq b_{\sigma(i)}x_{\sigma(i)} + (1 - b_{\sigma(i)})(1 - x_{\sigma(i)}); \tag{2.190}$$

$$b_{\sigma(k^*)}x_{\sigma(k^*)} + (1 - b_{\sigma(k^*)})(1 - x_{\sigma(k^*)}) \geq (1 - b_{\sigma(i)})x_{\sigma(i)} + b_{\sigma(i)}(1 - x_{\sigma(i)}). \tag{2.191}$$

  - If $b_{\sigma(k^*)} = 0$, we have $c_{\sigma(k^*)} = 1$ by (2.181); (2.190) and (2.191) give $1 - x_{\sigma(k^*)} \geq \max\{x_{\sigma(i)}, 1 - x_{\sigma(i)}\}$;

- If $b_{\sigma(k^*)} = 1$, we have $c_{\sigma(k^*)} = 0$ by (2.181); (2.190) and (2.191) give $x_{\sigma(k^*)} \geq \max\{x_{\sigma(i)}, 1 - x_{\sigma(i)}\}$.

Both cases imply (2.185), so we see this also holds for $i > k^*$.

The same proof works when $k^* = n + 1$. In this case, there is no $i \in [n]$ such that $i > k^*$; on the other hand, when $i < k^*$, we replace (2.189) by

$$(1 - b_{\sigma(n)})x_{\sigma(n)} + b_{\sigma(n)}(1 - x_{\sigma(n)}) \geq (1 - b_{\sigma(i)})x_{\sigma(i)} + b_{\sigma(i)}(1 - x_{\sigma(i)}), \tag{2.192}$$

and the argument proceeds unchanged. □

Optimality of Theorem 2.3.3. In this section, we prove that the lower bound in Theorem 2.3.3 is optimal (up to poly-logarithmic factors in n) for the max-norm optimization problem:

Theorem 2.5.1. *Let $f_c \colon [0,1]^n \to [0,1]$ be an objective function for the max-norm optimization problem (Definition 2.3.2). Then there exists a quantum algorithm that outputs an $\tilde{x} \in [0,1]^n$ satisfying (2.79) with $\epsilon = 1/3$ using $O(\sqrt{n}\log n)$ quantum queries to O_f, with success probability at least* 0.9.

In other words, the quantum query complexity of the max-norm optimization problem is $\tilde{\Theta}(\sqrt{n})$.

We prove Theorem 2.5.1 also using search with wildcards (Theorem 2.3.1).

Proof. It suffices to prove that one query to the wildcard query model O_c in (2.64) can be simulated by one query to O_{f_c}, where the c in (2.77) is the string c in the wildcard query model.

Assume that we query (T, y) using the wildcard query model. Then we query $O_{f_c}(x^{(T,y)})$

where for all $i \in [n]$,

$$x_i^{(T,y)} = \begin{cases} \frac{1}{2} & \text{if } i \notin T; \\ 0 & \text{if } i \in T \text{ and } y_i = 0; \\ 1 & \text{if } i \in T \text{ and } y_i = 1. \end{cases} \tag{2.193}$$

If $c_{|T} = y$, then

- if $|T| = n$ (i.e., $T = [n]$), then

$$f_c(x) = \max_{i\in[n]} |x_i^{(T,y)} - c_i| = 0 \tag{2.194}$$

  because for any $i \in [n]$, $x_i^{(T,y)} = y_i = c_i$;

- if $|T| \leq n-1$, then

$$f_c(x) = \max_{i\in[n]} |x_i^{(T,y)} - c_i| + g_i = \frac{1}{2}, \tag{2.195}$$

  because for all $i \in T$ we have $x_i^{(T,y)} = y_i = c_i$ and hence $|x_i^{(T,y)} - c_i| = 0$, and for all $i \notin T$ we have $|x_i^{(T,y)} - c_i| = |\frac{1}{2} - c_i| = \frac{1}{2}$.

Therefore, if $c_{|T} = y$, then we must have $f_c(x^{(T,y)}) \in \{0, \frac{1}{2}\}$.

On the other hand, if $c_{|T} \neq y$, then there exists an $i_0 \in T$ such that $c_{i_0} \neq y_{i_0}$. This implies $x_{i_0}^{(T,y)} = 1 - c_{i_0}$; as a result, $f_c(x^{(T,y)}) = 1$ because on the one hand $f_c(x^{(T,y)}) \geq |1 - c_{i_0} - c_{i_0}| = 1$, and on the other hand $f_c(x^{(T,y)}) \leq 1$ as $|x_i^{(T,y)} - c_i| \leq 1$ for all $i \in [n]$.

Notice that the sets $\{0, \frac{1}{2}\}$ and $\{1\}$ do not intersect. Therefore, after we query $O_{f_c}(x^{(T,y)})$

and obtain the output, we can tell $Q_s(T, y) = 1$ in (2.64) if $O_{f_c}(x^{(T,y)}) \in \left\{0, \frac{1}{2}\right\}$, and output $Q_s(T, y) = 0$ if $O_{f_c}(x^{(T,y)}) = 1$. In all, this gives a simulation of one query to the wildcard query model O_c by one query to O_{f_c}.

As a result of Theorem 2.3.1, there is a quantum algorithm that outputs the c in (2.77) using $O(\sqrt{n} \log n)$ quantum queries to O_f. If we take $\tilde{x} = c$, then $f_c(\tilde{x}) = \max_i |c_i - c_i| = 0$, which is actually the optimal solution with $\epsilon = 0$ in (2.79). This establishes Theorem 2.5.1. $\square$

Chapter 3: Quantum Algorithms for Estimating Volumes of Convex Bodies

In this chapter we present a quantum algorithm with a polynomial speedup for estimating the volumes of convex bodies. The results presented were first established in [58].

3.1 Introduction

Estimating the volume of a convex body is a central challenge in theoretical computer science. Volume estimation is a basic problem in convex geometry and can be viewed as a continuous version of counting. Furthermore, algorithms for a generalization of volume estimation, namely log-concave sampling—can be directly used to perform convex optimization, and hence can be widely applied to problems in statistics, machine learning, operations research, etc. See the survey [59] for a more comprehensive introduction.

Volume estimation is a notoriously difficult problem. References [60, 61] proved that any *deterministic algorithm* that approximates the volume of an n-dimensional convex body within a factor of $n^{o(n)}$ necessarily makes exponentially many queries to a membership oracle for the convex body. Furthermore, Refs. [62, 63, 64] showed that estimating the volume exactly (deterministically) is #P-hard, even for explicitly described polytopes.

Surprisingly, volumes of convex bodies can be approximated efficiently by *randomized algorithms*. Dyer, Frieze, and Kannan [65] gave the first polynomial-time randomized algorithm for estimating the volume of a convex body in $\mathbb{R}^n$. They present an iterative algorithm that constructs a sequence of convex bodies. The volume of the convex body of interest can be written as the telescoping product of the ratios of the volumes of consecutive convex bodies, and these ratios are estimated by Markov chain Monte Carlo (MCMC) methods via random walks inside these convex bodies. The algorithm in [65] has complexity[1] $\tilde{O}(n^{23})$ with multiplicative error $\epsilon = \Theta(1)$. Subsequent work [66, 67, 68, 69, 70, 71, 72] improved the design of the iterative framework and the choice of the random walks. The state-of-the-art algorithm for estimating the volume of a general convex body [73] uses $\tilde{O}(n^4)$ queries to the oracle for the convex body and $\tilde{O}(n^6)$ additional arithmetic operations.

It is natural to ask whether quantum computers can solve volume estimation even faster than classical randomized algorithms. Although there are frameworks with potential quantum speedup for simulated annealing algorithms in general, with volume estimation as a possible application [74], we are not aware of any previous quantum speedup for volume estimation. There are several reasons to develop such a result. First, quantum algorithms for volume estimation can be seen as performing a continuous version of quantum counting [75, 76], a key algorithmic technique with wide applications in quantum computing. Second, quantum algorithms for volume estimation can exploit quantum MCMC methods (e.g., [11, 77, 78]), and a successful quantum volume estimation algorithm may illuminate the application of quantum MCMC methods in other scenarios. Third, there has been recent progress on quantum algorithms for convex optimization [32, 79], so it is natural to study the closely related task of estimating volumes of convex bodies.

[1]Throughout the paper, $\tilde{O}$ omits factors in $\mathrm{poly}(\log R/r, \log 1/\epsilon, \log n)$ where R and r are defined in (3.2).

Formulation Given a convex set $\mathrm{K} \subset \mathbb{R}^n$, we consider the problem of estimating its volume

$$\mathrm{Vol}(\mathrm{K}) := \int_{x \in \mathrm{K}} \mathrm{d}x. \tag{3.1}$$

To get a basic sense about the location of K, we assume that it contains the origin. Furthermore, we assume that we are given inner and outer bounds on K, namely

$$\mathrm{B}_2^n(0, r) \subseteq \mathrm{K} \subseteq \mathrm{B}_2^n(0, R), \tag{3.2}$$

where $\mathrm{B}_2^n(x, l)$ is the ball of radius l in ℓ_2-norm centered at $x \in \mathbb{R}^n$. Denote $D := R/r$.

We consider the very general setting where the convex body K is only specified by an oracle. In particular, we have a *membership oracle*[2] for K that determines whether a given $x \in \mathbb{R}^n$ belongs to K. The efficiency of volume estimation is then measured by the number of queries to the membership oracle (i.e., the *query complexity*) and the total number of other arithmetic operations.

In the quantum setting, the membership oracle is a unitary operator O_{K}. Specifically, we have

$$O_{\mathrm{K}}|x, 0\rangle = |x, \delta[x \in \mathrm{K}]\rangle \qquad \forall x \in \mathbb{R}^n, \tag{3.3}$$

where $\delta[P]$ is 1 if P is true and 0 if P is false.[3] In other words, we allow coherent superpositions

[2]The membership oracle is commonly used in convex optimization research (see for example [34]). This model is not only general but also of practical interest. For instance, when K is a bounded convex polytope, the membership oracle can be efficiently implemented by checking if all its linear constraints are satisfied; see also [80].

[3]Here x can be approximated just as in the classical algorithms, such as with fixed-point numbers. Our algorithmic approach is robust under discretization (see Section 3.5), and our quantum lower bound holds even when x is stored with arbitrary precision (Section 3.6). We mostly assume for convenience that O_{K} operates on

of queries to the membership oracle. If the classical membership oracle can be implemented by an explicit classical circuit, then the corresponding quantum membership oracle can be implemented by a quantum circuit of about the same size. Therefore, the quantum query model provides a useful framework for understanding the quantum complexity of volume estimation.

3.1.1 Contributions

Our main result is a quantum algorithm for estimating volumes of convex bodies:

Theorem 3.1.1 (Main Theorem)**.** *Let $\mathrm{K} \subset \mathbb{R}^n$ be a convex set with $\mathrm{B}_2^n(0,r) \subseteq \mathrm{K} \subseteq \mathrm{B}_2^n(0,R)$. Assume $0 < \epsilon < 1/2$. Then there is a quantum algorithm that returns a value $\widetilde{\mathrm{Vol}(\mathrm{K})}$ satisfying*

$$\frac{1}{1+\epsilon}\mathrm{Vol}(\mathrm{K}) \leq \widetilde{\mathrm{Vol}(\mathrm{K})} \leq (1+\epsilon)\mathrm{Vol}(\mathrm{K}) \tag{3.4}$$

with probability at least $2/3$ using $\tilde{O}(n^3 + n^{2.5}/\epsilon)$ quantum queries to the membership oracle O_{K} (defined in (4.8)) and $\tilde{O}(n^5 + n^{4.5}/\epsilon)$ additional arithmetic operations.[4]

To the best of our knowledge, this is the first quantum algorithm that achieves quantum speedup for this fundamental problem, compared to the classical state-of-the-art algorithm [73, 82] that uses $\tilde{O}(n^4 + n^3/\epsilon^2)$ classical queries and $\tilde{O}(n^6 + n^5/\epsilon^2)$ additional arithmetic operations.[5] Furthermore, our quantum algorithm not only achieves a quantum speedup in query complexity, but also in the number of arithmetic operations for executing the algorithm. This differs from

$x \in \mathbb{R}^n$, since this neither presents a serious obstacle nor conveys significant power.

[4]Arithmetic operations (e.g., addition, subtraction, multiplication, and division) can be in principle implemented by a universal set of quantum gates using the Solovay-Kitaev Theorem [81] up to a small overhead. In our quantum algorithm, the number of arithmetic operations is dominated by n-dimensional matrix-vector products computed in superposition for rounding the convex body (see Section 3.4.4).

[5]This is achieved by applying [73] to preprocess the convex body to be well-rounded (i.e. $R/r = O(\sqrt{n})$) using $\tilde{O}(n^4)$ queries and then applying [82] using $\tilde{O}(n^3/\epsilon^2)$ queries to estimate the volume of the (well-rounded) convex body. The number of additional arithmetic operations has an overhead of $O(n^2)$ due to the affine transformation.

previous quantum algorithms for convex optimization [32, 79] where only the query complexity is improved, but the gate complexity is the same as that of the classical state-of-the-art algorithm [42, 83].

On the other hand, we prove in Section 3.6.1 that volume estimation with $\epsilon = \Theta(1)$ requires $\Omega(\sqrt{n})$ quantum queries to the membership oracle, ruling out the possibility of achieving superpolynomial quantum speedup for volume estimation. Classically, the best-known lower bound on the query complexity of volume estimation is $\tilde{\Omega}(n^2)$ due to Rademacher and Vempala [84], but there are technical difficulties to lift it to a quantum lower bound (see Section 3.1.2.3). For the dependence on $1/\epsilon$, we establish a quantum query lower bound of $\Omega(1/\epsilon)$, and the same argument shows a classical query lower bound of $\Omega(1/\epsilon^2)$ (see Section 3.6.2). As a result, our quantum algorithm in Theorem 4.1.1 achieves a provable quadratic quantum speedup in $1/\epsilon$ and is optimal in $1/\epsilon$ up to poly-logarithmic factors.

Technically, we refine a framework for achieving quantum speedups of simulated annealing algorithms, which might be of independent interest. Our framework applies to MCMC algorithms with cooling schedules that ensure each ratio in a telescoping product has bounded variance, an approach known as *Chebyshev cooling*. Furthermore, we propose several novel techniques when implementing this framework, including a theory of continuous-space quantum walks with rigorous bounds on discretization error, a quantum algorithm for nondestructive mean estimation, and a quantum algorithm with interlaced rounding and volume estimation of convex bodies (as described further in Section 4.3.1 below). In principle, our techniques apply not only to the integral of the identity function (as in Theorem 4.1.1), but could also be applied to any log-concave function defined on a convex body, following the approach in [40].

We summarize our main results in Table 3.1.

	Classical bounds	Quantum bounds (this paper)
Query complexity	$\tilde{O}(n^4 + n^3/\epsilon^2)$ [73, 82], $\tilde{\Omega}(n^2)$ [84]	$\tilde{O}(n^3 + n^{2.5}/\epsilon)$, $\Omega(\sqrt{n} + 1/\epsilon)$
Total complexity	$\tilde{O}\big((n^2 + C_{\text{MEM}}) \cdot (n^4 + n^3/\epsilon^2)\big)$ [73, 82]	$\tilde{O}\big((n^2 + C_{\text{MEM}}) \cdot (n^3 + n^{2.5}/\epsilon)\big)$

Table 3.1: Summary of complexities of volume estimation, where n is the dimension of the convex body, ϵ is the multiplicative precision of volume estimation, and C_{MEM} is the cost of applying the membership oracle once. Total complexity refers to the cost of the of queries plus the number of additional arithmetic operations.

3.1.2 Techniques

We now summarize the key technical aspects of our work.

3.1.2.1 Classical volume estimation framework

Volume estimation by simulated annealing The volume of a convex body K can be estimated using simulated annealing. Consider the value

$$Z(a) := \int_{\mathrm{K}} e^{-a\|x\|_2}\,\mathrm{d}x, \tag{3.5}$$

where $\|x\|_2 := \sqrt{x_1^2 + \cdots + x_n^2}$ is the ℓ_2-norm of x. On the one hand, $Z(0) = \mathrm{Vol}(\mathrm{K})$; on the other hand, because $e^{-\|x\|_2}$ decays exponentially fast with $\|x\|_2$, taking a large enough a ensures that the vast majority of $Z(a)$ concentrates near 0, so it can be well approximated by integrating on a small ball centered at 0. Therefore, a natural strategy is to consider a sequence $a_0 > a_1 > \cdots > a_m$ with a_0 sufficiently large and a_m close to 0. We consider a simulated annealing algorithm that iteratively changes a_i to a_{i+1} and estimates $\mathrm{Vol}(\mathrm{K})$ by the telescoping product

$$\mathrm{Vol}(\mathrm{K}) \approx Z(a_m) = Z(a_0) \prod_{i=0}^{m-1} \frac{Z(a_{i+1})}{Z(a_i)}. \tag{3.6}$$

In the i^{th} step, a random walk is used to sample the distribution over K with density proportional to $e^{-a_i\|x\|_2}$. Denote one such sample by X_i, and let $V_i := e^{(a_i - a_{i+1})\|X_i\|_2}$. Then we have

$$\mathbb{E}[V_i] = \int_{\mathrm{K}} e^{(a_i - a_{i+1})\|x\|_2} \frac{e^{-a_i\|x\|_2}}{Z(a_i)}\,\mathrm{d}x = \int_{\mathrm{K}} \frac{e^{-a_{i+1}\|x\|_2}}{Z(a_i)}\,\mathrm{d}x = \frac{Z(a_{i+1})}{Z(a_i)}. \tag{3.7}$$

Therefore, each ratio $\frac{Z(a_{i+1})}{Z(a_i)}$ can be estimated by taking i.i.d. samples X_i, computing the correspo-

nding V_is, and taking their average.

We can analyze this volume estimation algorithm by considering its behavior at three levels:

1) High level: The algorithm follows the simulated annealing framework described above, where the volume is estimated by a telescoping product as in (3.6).

2) Middle level: The number of i.i.d. samples used to estimate $\mathbb{E}[V_i]$ (a ratio in the telescoping product given by (3.7)) is small. Intuitively, the annealing schedule should be slow enough that V_i has small variance.

3) Low level: The random walk converges fast so that we can take each i.i.d. sample of V_i efficiently.

Classical volume estimation algorithm Our approach follows the classical volume estimation algorithm in [73] (see also Section 3.4.1). At the high level, it is a simulated annealing algorithm that estimates the volume of an alternative convex body K' produced by the *pencil construction*, which intersects a cylinder $[0, 2R/r] \times \mathrm{K}$ and a cone $\mathrm{C} := \{x \in \mathbb{R}^{n+1} : x_0 \geq 0, \|x\|_2 \leq x_0\}$. This construction shares the same intuition as above, but replaces the integral (3.5) by $Z(a) = \int_{\mathrm{K}'} e^{-ax_0}\,\mathrm{d}x$ because it is easier to calculate while can be directly used to estimate $\mathrm{Vol}(\mathrm{K})$ when $a \approx 0$ by a standard Monte Carlo approach (see Lemma 3.4.1).

Without loss of generality, assume that $r = 1$. Lovász and Vempala [73] proved that if we take the sequence $a_0 > \cdots > a_m$ where $a_0 = 2n$, $a_{i+1} = (1 - \frac{1}{\sqrt{n}})a_i$, and $m = \tilde{O}(\sqrt{n})$, then $Z(a_0) \approx \int_{\mathrm{C}} e^{-a_0 x_0}\,\mathrm{d}x$ and

$$\mathrm{Var}[V_i^2] = O(1) \cdot \mathbb{E}[V_i]^2,\ \forall\, i \in [m], \tag{3.8}$$

i.e., the variance of V_i is bounded by a constant multiple of the square of its expectation. Such a simulated annealing schedule is known as *Chebyshev cooling* (see also Section 3.4.3.3). This establishes the middle-level requirement of the simulated annealing framework. Furthermore, [73] proves that the product of the average of $\tilde{O}(\sqrt{n}/\epsilon^2)$ i.i.d. samples of V_i for all $i \in [m]$ gives an estimate of $\mathrm{Vol}(\mathrm{K}')$ within multiplicative error ϵ with high success probability.

At the low level, Ref. [73] uses a *hit-and-run walk* to sample X_i. In this walk, starting from a point p, we uniformly sample a line ℓ through p and move to a random point along the chord $\ell \cap \mathrm{K}$ with density proportional to e^{-ax_0} (see Section 3.2.4 for details). Reference [72] analyzes the convergence of the hit-and-run walk, proving that it converges to the distribution over K with density proportional to e^{-ax_0} within $\tilde{O}(n^3)$ steps, assuming that K is *well-rounded* (i.e., $R/r = O(\sqrt{n})$).

Finally, Ref. [73] constructs an affine transformation that transforms a general K to be well-rounded with $\tilde{O}(n^4)$ classical queries to its membership oracle, hence removing the constraint of the previous steps that K be well-rounded. Because the affine transformation is an n-dimensional matrix-vector product, this introduces an overhead of $O(n^2)$ in the number of arithmetic operations.

Overall, the algorithm has $\tilde{O}(\sqrt{n})$ iterations, where each iteration takes $\tilde{O}(\sqrt{n}/\epsilon^2)$ i.i.d. samples, and each sample takes $\tilde{O}(n^3)$ steps of the hit-and-run walk. In total, the query complexity is

$$\tilde{O}(\sqrt{n}) \cdot \tilde{O}(\sqrt{n}/\epsilon^2) \cdot \tilde{O}(n^3) = \tilde{O}(n^4/\epsilon^2). \tag{3.9}$$

The number of additional arithmetic operations is $\tilde{O}(n^4/\epsilon^2) \cdot O(n^2) = \tilde{O}(n^6/\epsilon^2)$ due to the affine transformation for rounding the convex body.

3.1.2.2 Quantum algorithm for volume estimation

It is natural to consider a quantum algorithm for volume estimation following the classical framework in Section 3.1.2.1. A naive attempt might be to develop a quantum walk that achieves a generic quadratic speedup of the mixing time. However, this is unfortunately difficult to achieve in general. Quantum walks are unitary processes that do not converge to stationary distributions in the classical sense. As a result, alternative and indirect quantum analogues of mixing properties of Markov chains have been proposed and studied (see Section 3.1.3.2 for more detail). None of these methods provide a direct replacement for classical mixing, and we cannot directly apply them in our context.

Instead, we adapt one of the frameworks proposed in [78]. To give a quantum speedup for volume estimation by this method, we address the following additional technical challenges:

- **Quantum walks in continuous space:** Quantum walks have mainly been studied in discrete spaces [85, 86], and we need to understand how to define a quantum counterpart of the hit-and-run walk.

- **Quantum mean estimation:** Quantum counting [76] is a general tool for estimating a probability $p \in [0, 1]$ with quadratic quantum speedup compared to classical sampling. However, estimating the mean of an unbounded random variable with a quantum version of Chebyshev concentration requires more advanced tools.

- **Rounding:** Classically, rounding a general convex body takes $\tilde{O}(n^4)$ queries [73], more expensive than volume estimation of a well-rounded body using $\tilde{O}(n^3/\epsilon^2)$ queries [82]. To achieve an overall quantum speedup, we also need to give a fast quantum algorithm for rounding convex bodies.

- **Error analysis of the quantum hit-and-run walk:** We must bound the error incurred when implementing the quantum walk on a digital quantum computer with finite precision. Existing classical error analyses (e.g., [87]) do not automatically cover the quantum case.

We develop several novel techniques to resolve these issues:

Theory of continuous-space quantum walks (Section 3.3) Our first technical contribution is to develop a quantum implementation of the low-level framework, i.e., to replace the classical hit-and-run walk by a *quantum hit-and-run walk*. However, although quantum walks in discrete spaces have been well studied (see for example [85, 86]), we are not aware of comparable results that can be used to analyze spectral properties and mixing times of quantum walks in continuous space. Here we describe a framework for continuous-space quantum walks that can be instantiated to give a quantum version of the hit-and-run walk. In particular, we formally define such walks and analyze their spectral properties, generalizing Szegedy's theory [85] to continuous spaces (Section 3.3.1). We also show a direct correspondence between the stationary distribution of a classical walk and a certain eigenvector of the corresponding quantum walk (Section 3.3.2).

Quantum volume estimation algorithm via simulated annealing (Section 3.4.2) Having described a quantum hit-and-run walk, the next step is to understand the high-level simulated annealing framework. As mentioned above, it is nontrivial to directly prepare stationary states of quantum walks. In this paper, we follow a quantum MCMC framework proposed by [78] that can prepare stationary states of quantum walks by simulated annealing (see Section 3.2.2). In this framework, we have a sequence of slowly-varying Markov chains, and the stationary state of the initial Markov chain can be efficiently prepared. In each iteration, we apply fixed-point amplitude amplification of the quantum walk operator [1] due to Grover to transform the current stationary state to the next one; compared to classical slowly-varying Markov chains, the convergence rate of such quantum procedure is *quadratically better in spectral gap.*

Our **main technical contribution** is to show how to adapt the Chebyshev cooling schedule in [73] to the quantum MCMC framework in [78] using our quantum hit-and-run walk. The conductance lower bound together with the classical $\tilde{O}(n^3)$ mixing time imply that we can perform one step of fixed-point amplitude amplification using $\tilde{O}(n^{1.5})$ queries to O_{K}. Furthermore, the inner product between consecutive stationary states is a constant. These two facts ensure that the stationary state in each iteration can be prepared with $\tilde{O}(n^{1.5})$ queries to the membership oracle O_{K}. The total number of iterations is still $\tilde{O}(\sqrt{n})$, as in the classical case.

Quantum algorithm for nondestructive mean estimation (Section 3.4.3.3) In the next step, we consider how to estimate each ratio in the telescoping product at the middle level. The basic tool is quantum counting [76], which estimates a probability $p \in [0,1]$ with error ϵ and high success probability using $O(1/\epsilon)$ quantum queries, a quadratic speedup compared to the classical complexity $O(1/\epsilon^2)$. However, in our case we need to estimate the expectation of a random variable with bounded variance. We use the "quantum Chebyshev inequality" developed in [88] which truncates the random variable with reasonable upper and lower bounds and then reduces to quantum counting; see Section 3.2.3.[6] Compared to the classical counterpart, it achieves quadratic speedup in the dependences on both variance and multiplicative error.

There is an additional technical difficulty in quantum simulated annealing: classically, it is implicitly assumed that in the $(i+1)^{\text{st}}$ iteration we have samples to the stationary distribution in the i^{th} iteration. Applying existing quantum mean estimation techniques to the quantum stationary state in the i^{th} iteration would ruin that state and make it hard to use in the subsequent $(i+1)^{\text{st}}$ iteration. To resolve this issue, we estimate the mean *nondestructively* in the quantum Chebyshev inequality while keeping its quadratic speedup in the error dependence using a nondestructive amplitude estimation technique developed in [89]. Nondestructive mean estimation relies on the following observation: applying amplitude estimation on a state $|\psi\rangle$ results with high probability in the measurement collapsing to one of two states $|\psi_+\rangle, |\psi_-\rangle$ with constant overlap with ψ. The algorithm repeatedly projects these states onto $|\psi\rangle$: if the projection is successful then the state is restored, otherwise amplitude estimation can be performed again to obtain $|\psi_+\rangle, |\psi_-\rangle$ and the projection can be repeated. Due to the constant overlap, $\mathrm{poly}(\log(\delta^{-1}))$ repititions suffice to

[6]A related technique is the quantum Monte Carlo method of Montanaro [11]. Here we use [88] for two reasons: first, it has the advantage of handling multiplicative instead of additive errors, which is appropriate for estimating the telescoping ratios. Second, its quantum algorithm is based on amplitude estimation and hence can readily be made nondestructive, as discussed below.

ensure that at least one of the projections succeeds with probability δ. It remains to implement the required projection efficiently: we show how this can be accomplished using quantum walk operators corresponding to the Markov Chains in the MCMC framework; see Section 3.4.3.4.

In our quantum volume estimation algorithm, we apply the quantum Chebyshev inequality under the same compute-uncompute procedure. This gives a quadratic speedup in ϵ^{-1} when estimating the $\mathbb{E}[V_i]$ in (3.7), so that $\tilde{O}(\sqrt{n}/\epsilon)$ copies of the stationary state suffice[7] (see Lemma 3.4.3).

Quantum algorithm for volume estimation with interlaced rounding (Section 3.4.4) The stationary states of the quantum hit-and-run walk can be prepared with $\tilde{O}(n^{1.5})$ queries to O_{K} only when the corresponding density functions are *well-rounded*, i.e., every level set with probability μ contains a ball of radius μr and the variance of the density is bounded by R^2, where $R/r = O(\sqrt{n})$.[8] It remains to show how to ensure that the convex body is well-rounded.

Classically, Ref. [73] gave a rounding algorithm that transforms a convex body to ensure that all the densities sampled in the volume estimation algorithm are well-rounded. This algorithm uses $\tilde{O}(n^4)$ queries, via $\tilde{O}(n)$ iterations of simulated annealing. A quantization of this algorithm along the same lines as detailed above gives an algorithm with $\tilde{O}(n^{3.5})$ quantum queries.

To improve over that approach, we instead follow a classical framework for directly rounding logconcave densities [40]. The rounding is interlaced with the volume estimation algorithm, so that in each iteration of the simulated annealing framework, we use some of the samples to calculate an affine transformation that makes the next stationary state well-rounded. This ensures that the quantum hit-and-run walk continues to take only $\tilde{O}(n^{1.5})$ queries for each sample. Our

[7]It is possible to use fewer copies of the stationary state. See Footnote 14.

[8]When the density function is uniform in K, this definition of well-roundedness reduces to that in Footnote 5. The definition of level sets is the same as in [73].

algorithm maintains $\tilde{O}(n)$ extra quantum states for rounding, and the quantum hit-and-run walk is used to transform them from one stationary distribution to the next. In each iteration, we use a nondestructive measurement to sample the required affine transformation. With $\tilde{O}(\sqrt{n})$ iterations this results in an additional $\tilde{O}(\sqrt{n}) \cdot \tilde{O}(n) \cdot \tilde{O}(n^{1.5}) = \tilde{O}(n^3)$ cost for rounding.

We also show that this framework can be used as a preprocessing step that puts the convex body itself in well-rounded position (i.e., $\mathrm{B}_2(0,r) \subseteq \mathrm{K} \subseteq \mathrm{B}_2(0,R)$ with $R/r = O(\sqrt{n})$) using $\tilde{O}(n^3)$ quantum queries. Putting a convex body in well-rounded position implies that several random walks used in simulated annealing algorithms (including the hit-and-run walk) mix fast without the need for further rounding. Therefore, as an alternative, we could preprocess the convex body to be well-rounded and then apply the simulated annealing algorithm to obtain a volume estimation algorithm that uses $\tilde{O}(n^3 + n^{2.5}/\epsilon)$ quantum queries.

Error analysis of discretized hit-and-run walks (Section 3.5) Although we defined quantum hit-and-run walks abstractly in Section 3.3, implementing a continuous-space quantum walk on a digital quantum computers will lead to discretization error, and the error analysis of classical walks in a discrete space approximating $\mathbb{R}^n$ (such as [87]) does not automatically apply to the quantum counterpart. To ensure that discretization errors do not affect a realistic implementation of our algorithm, in Section 3.5 we propose a *discretized hit-and-run walk* and provide rigorous bounds on the discretization error.

Summary Our quantum volume estimation algorithm can be summarized as follows.

1) High level: The quantum algorithm follows a simulated annealing framework using a quantum MCMC method [78], where the volume is estimated by a telescoping product (as in (3.6)); the

number of iterations is $\tilde{O}(\sqrt{n})$.

2) Middle level: We estimate the $\mathbb{E}[V_i]$ in (3.7), a ratio in the telescoping product, using the nondestructive version of the quantum Chebyshev inequality [88]. This takes $\tilde{O}(\sqrt{n}/\epsilon)$ implementations of the quantum hit-and-run walk operators.

3) Low level: If the convex body K is well-rounded (i.e., $R/r = O(\sqrt{n})$), each quantum hit-and-run walk operator can be implemented using $\tilde{O}(n^{1.5})$ queries to the membership oracle O_{K} in (4.8).

Finally, we give a quantum algorithm that interlaces rounding and volume estimation of the convex body, using an additional $\tilde{O}(n^{2.5})$ quantum queries to O_{K} in each iteration. Because the affine transformation is an n-dimensional matrix-vector product, it introduces an overhead of $O(n^2)$ in the number of arithmetic operations (just as in the classical rounding algorithm).

Overall, our quantum volume estimation algorithm has $\tilde{O}(\sqrt{n})$ iterations. Each iteration implements $\tilde{O}(\sqrt{n}/\epsilon)$ quantum hit-and-run walks, and each quantum hit-and-run walk uses $\tilde{O}(n^{1.5})$ queries; there is also a cost of $\tilde{O}(n^{2.5})$ for rounding. Thus the quantum query complexity is

$$\tilde{O}(\sqrt{n}) \cdot \big(\tilde{O}(\sqrt{n}/\epsilon) \cdot \tilde{O}(n^{1.5}) + \tilde{O}(n^{2.5})\big) = \tilde{O}(n^3 + n^{2.5}/\epsilon). \tag{3.10}$$

The number of additional arithmetic operations is $\tilde{O}(n^3 + n^{2.5}/\epsilon) \cdot O(n^2) = \tilde{O}(n^5 + n^{4.5}/\epsilon)$ due to the affine transformations for interlaced rounding of the convex body.

Figure 3.1 summarizes our techniques. The volume estimation and interlaced rounding algorithms are given as Algorithm 8 and Algorithm 9, respectively, in Section 3.4.

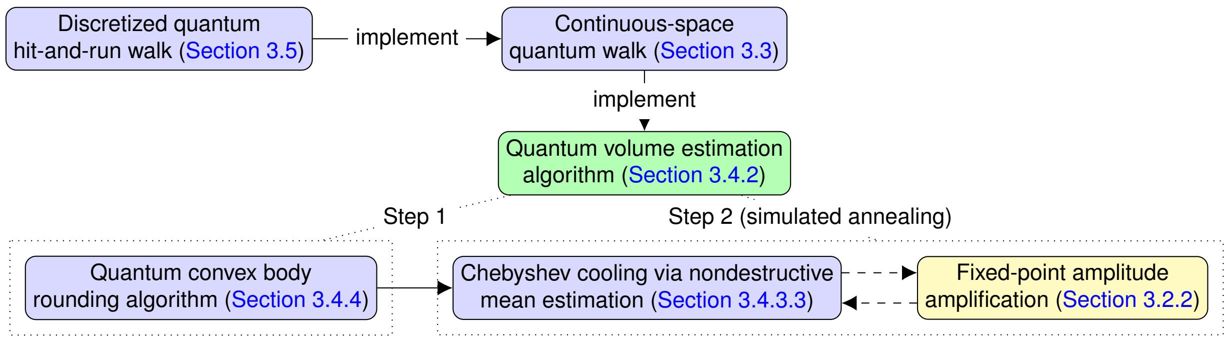

Discretized quantum hit-and-run walk (Section 3.5)
implement
Continuous-space quantum walk (Section 3.3)
implement
Quantum volume estimation algorithm (Section 3.4.2)
Step 1
Step 2 (simulated annealing)
Quantum convex body rounding algorithm (Section 3.4.4)
Chebyshev cooling via nondestructive mean estimation (Section 3.4.3.3)
Fixed-point amplitude amplification (Section 3.2.2)

Figure 3.1: The structure of our quantum volume estimation algorithm. The four purple frames represent the four novel techniques that we propose, the yellow frame represents the known technique from [1], and the green frame at the center represents our quantum algorithm.

3.1.2.3 Quantum lower bounds (Section 3.6)

The classical state-of-the-art query lower bound for volume estimation is a $\tilde{\Omega}(n^2)$ bound for n-dimensional parallelopipeds [84]. The argument uses Yao's principle [90] to reduce the problem of estimating the volume of parallelopipeds to a corresponding average-case lower bound for deterministic algorithms. However, in the quantum setting, it is unclear how to apply a similar argument since such a reduction to the deterministic case does not work in general.

Nevertheless, we prove that volume estimation requires $\Omega(\sqrt{n})$ quantum queries to the membership oracle, ruling out the possibility of exponential quantum speedup (see Theorem 3.6.1). We establish this by a reduction to search: for a hyper-rectangle $\mathrm{K} = \bigtimes_{i=1}^{n}[0, 2^{s_i}]$ specified by a binary string $s = (s_1, \ldots, s_n) \in \{0,1\}^n$ with $|s| = 0$ or 1, we prove that a membership query to K can be simulated by a query to s. Thus, since $\mathrm{Vol}(\mathrm{K}) = 2$ if and only if $|s| = 1$, the $\Omega(\sqrt{n})$ quantum lower bound on search [91] applies to volume estimation.

In addition, we prove that volume estimation requires $\Omega(1/\epsilon)$ quantum queries (see Theorem 3.6.2), which means that our quantum algorithm is optimal in $1/\epsilon$ up to poly-logarithmic factors. The idea is to construct a convex body whose volume estimation reduces to the Hamming distance problem with known tight quantum query complexity [92]. To be more specific, we consider the n-dimensional unit hypercube and attach "hyperpyramids" to its faces, such that its central axis passes through the center of the hypercube. We show that adding or deleting any hyperpyramid of volume $1/2n$ does not influence the convexity of the convex body, and calculating the volume of the body reveals the Hamming weight of a binary string that encodes the presence or absence of the hyperpyramids.

3.1.3 Related work

While our paper gives the first quantum algorithm for volume estimation, classical volume estimation algorithms have been well-studied, as we review in Section 3.1.3.1. Our quantum algorithm builds upon quantum analogs of Markov chain Monte Carlo methods that we review in Section 3.1.3.2.

3.1.3.1 Classical volume estimation algorithms

There is a rich literature on classical algorithms for estimating volumes of convex bodies (e.g., see the surveys [59, 93]). The general approach is to consider a sequence of random walks inside the convex body K whose stationary distributions converge quickly to the uniform distribution on K. Applying simulated annealing to this sequence of walks (as in Section 4.3.1), the volume of K can be approximated by a telescoping product.

The first polynomial-time algorithm for volume estimation was given by [65]. It uses a *grid walk* in which the convex body K is approximated by a grid mesh K_{grid} of spacing δ (i.e., K_{grid} contains the points in K whose coordinates are integer multiples of δ). The walk proceeds as follows:

1. Pick a grid point y uniformly at random from the neighbors of the current point x.

2. If $y \in \mathrm{K}_{\text{grid}}$, go to y; else stay at x.

Dyer, Frieze, and Vempala [65] proved that for a properly chosen δ, the grid walk converges to the uniform distribution on K_{grid} in $\tilde{O}(n^{23})$ steps, and that $\delta^n |\mathrm{K}_{\text{grid}}|$ is a good approximation of $\mathrm{Vol}(\mathrm{K})$ (in the sense of (3.4)). Subsequently, more refined analysis of the grid walk improved its cost to $\tilde{O}(n^8)$ [66, 67, 68]. However, this is still inefficient in practice.

Intuitively, the grid walk converges slowly because each step only moves locally in K. Subsequent work improved the complexity by considering other types of random walk. These improvements mainly use two types of walk: the *hit-and-run walk* and the *ball walk*. In this paper, we use the hit-and-run walk (see also Section 3.2.4), which behaves as follows:

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

2. Move to a uniformly random point along the chord $\ell \cap \mathrm{K}$.

Smith [94] proved that the stationary distribution of the hit-and-run walk is the uniform distribution on K. Regarding the convergence of the hit-and-run walk, [71] showed that it mixes in $\tilde{O}(n^3)$ steps from a warm start after appropriate preprocessing, and [72] subsequently proved that the hit-and-run walk mixes rapidly from any interior starting point (see also Theorem 3.2.4). Under the simulated annealing framework, the hit-and-run walk gives the state-of-the-art volume estimation algorithm with query complexity $\tilde{O}(n^4)$ [40, 73]. Our quantum volume estimation algorithm can be viewed as a quantization of this classical hit-and-run algorithm.

Given a radius parameter δ, the ball walk is defined as follows:

1. Pick a uniformly random point y from the ball of radius δ centered at the current point x.

2. If $y \in \mathrm{K}$, go to y; else stay at x.

Lovász and Simonovits [69] proved that the ball walk mixes in $\tilde{O}(n^6)$ steps. Kannan et al. [70] subsequently improved the mixing time to $\tilde{O}(n^3)$ starting from a warm start, giving a total query complexity of $\tilde{O}(n^5)$ for the volume estimation problem.

The analysis of the ball walk relies on a central conjecture in convex geometry, the Kannan-Lovász-Simonovits (KLS) conjecture (see [93]). The KLS conjecture states that the Cheeger constant of any log-concave density is achieved to within a universal, dimension-independent constant factor by a hyperplane-induced subset, where the Cheeger constant is the minimum ratio between the measure of the boundary of a subset to the measure of the subset or its complement, whichever is smaller. Although this quantity is conjectured to be a constant, the best known upper bound is only $O(n^{1/4})$ [95]. However, in the special case when the convex body is well-rounded (i.e., $R/r = O(\sqrt{n})$), a recent breakthrough by Cousins and Vempala [82, 96] proved the KLS conjecture for Gaussian distributions. In other words, they established a volume estimation algorithm with query complexity $\tilde{O}(n^3)$ in the well-rounded case.

Table 3.2 summarizes classical algorithms for volume estimation.

Method	State-of-the-art query complexity	Restriction on the convex body
Grid walk	$\tilde{O}(n^8)$ [68]	General ($R/r = \mathrm{poly}(n)$)
Hit-and-run walk	$\tilde{O}(n^4)$ [40, 73]	General ($R/r = \mathrm{poly}(n)$)
Ball walk	$\tilde{O}(n^3)$ [82, 96]	Well-rounded ($R/r = O(\sqrt{n})$)

Table 3.2: Summary of classical methods for estimating the volume of a convex body $\mathrm{K} \subset \mathbb{R}^n$ when $\epsilon = \Theta(1)$, where R, r are the radii of the balls centered at the origin that contain and are contained by the convex body, respectively.

3.1.3.2 Quantum Markov chain Monte Carlo methods

The performance of Markov chain Monte Carlo (MCMC) methods is determined by the rate of convergence to their stationary distributions (i.e., the mixing time). Suppose we have a reversible, ergodic Markov chain with unique stationary distribution π. Let π_k denote the distribution obtained by applying the Markov chain for k steps from some arbitrary initial state. It is well-known (see for example [97]) that $O(\frac{1}{\Delta}\log(1/(\epsilon \min_x \pi(x))))$ steps suffice to ensure $\|\pi_k - \pi\|_1 \leq \epsilon$, where Δ is the spectral gap of the Markov chain.

Many authors have studied quantum analogs of Markov chains (in both continuous [98] and discrete [85, 99, 100] time) and their mixing properties. While a quantum walk is a unitary process and hence does not converge to a stationary distribution, one can define notions of quantum mixing time by choosing the number of steps at random or by adding decoherence [77, 99, 100, 101, 102, 103, 104], and compare them to the classical mixing time. Note that distribution sampled by such a process may or may not be the same as the stationary distribution π of the corresponding classical Markov process, depending on the structure of the process and the notion of mixing. It is also natural to ask how efficiently we can prepare a quantum state close to $|\pi\rangle := \sum_x \sqrt{\pi_x}|x\rangle$, which can be viewed as a "quantum sample" from π. However, it is unclear how to do this efficiently in general, even in cases where a corresponding classical Markov process mixes quickly; in particular, a generic quantum algorithm for this task could be used to solve graph isomorphism [105, Section 8.4].

It is also possible to achieve quantum speedup of MCMC methods by not demanding speedup of the mixing time of each separate Markov chain, but only for the procedure as a whole. In particular, MCMC methods are often implemented by simulated annealing algorithms where the final output is a telescoping product of values at different temperatures. From this perspective, Somma et al. [106, 107, 108] used quantum walks to accelerate classical simulated annealing processes by exploiting the quantum Zeno effect, using measurements implemented by phase estimation of the quantum walk operators of these Markov chains. References [109, 110] also introduced how to implement Metropolis sampling on quantum computers.

Our quantum volume estimation algorithm is most closely related to work of Wocjan and Abeyesinghe [78], which achieves complexity $\tilde{O}(1/\sqrt{\Delta})$ for preparing the final stationary distribution of a sequence of slowly varying Markov chains, where Δ is the minimum of their spectral gaps. Their quantum algorithm transits between the stationary states of consecutive Markov chains by fixed-point amplitude amplification [1], which is implemented by amplitude estimation with $\tilde{O}(1/\sqrt{\Delta})$ implementations of the quantum walk operators of these Markov chains (see Section 3.2.2 for more details).

Our simulated annealing procedure preserves the slowly-varying property, so we adopt the framework of [78] in our algorithm for volume estimation (see Section 3.4.3.2). We develop several novel techniques (described in Section 4.3.1) that allow us to implement the steps of this framework efficiently. Note that the slowly-varying property also facilitates other frameworks that give efficient adiabatic [105] or circuit-based [111] quantum algorithms for generating quantum samples of the stationary state.

Previous work has mainly applied these quantum simulated annealing algorithms to estimating partition functions of discrete systems. Given an inverse temperature $\beta > 0$ and a classical

Hamiltonian $H\colon \Omega \to \mathbb{R}$ where Ω is a finite space, the goal is to estimate the partition function

$$Z(\beta) := \sum_{x\in\Omega} e^{-\beta H(x)} \tag{3.11}$$

within multiplicative error $\epsilon > 0$. Wocjan et al. [74] gave a quantum algorithm that achieves quadratic quantum speedup with respect to both mixing time and accuracy.

The classical algorithm that [74] quantizes uses $\tilde{O}(\log|\Omega|)$ annealing steps to ensure that each ratio $Z(\beta_{i+1})/Z(\beta_i)$ is bounded. In fact, it is possible to relax this requirement and use a cooling schedule with only $\tilde{O}(\sqrt{\log|\Omega|})$ steps such that the variance of each ratio is bounded, so its mean can be well-approximated by Chebyshev's inequality; this is exactly the Chebyshev cooling technique [112] introduced in Section 4.3.1 (see also Section 3.4.3.3). Montanaro [11] improves upon [74] using Chebyshev cooling; more recently, Harrow and Wei [89] further quadratically improved the spectral gap dependence of the estimation of the partition function.

Organization We review necessary background in Section 4.2. We describe the theory of continuous-space quantum walks in Section 3.3. In Section 3.4, we first review the classical state-of-the-art volume estimation algorithm in Section 3.4.1, and then give our quantum algorithm for estimating volumes of well-rounded convex bodies in Section 3.4.2. The proofs of our quantum algorithms are given in Section 3.4.3, and the quantum algorithm for rounding convex bodies is given in Section 3.4.4. The details of our discretized hit-and-run walk are given in Section 3.5, and we conclude with our quantum lower bound on volume estimation in Section 3.6.

3.2 Preliminaries

We summarize necessary tools used in this paper as follows.

3.2.1 Classical and quantum walks

A Markov chain over a finite state space Ω is a sequence of random variables $X_0, X_1, \dots$ such that for each $i \in \mathbb{N}$, the probability of transition to the next state $y \in \Omega$,

$$\Pr[X_{i+1} = y \mid X_i = x, X_{i-1} = x_{i-1}, \dots, X_0 = x_0] = \Pr[X_{i+1} = y \mid X_i = x] =: p_{x \to y}$$

only depends on the present state $x \in \Omega$. The Markov chain can be represented by the transition probabilities $p_{x\to y}$ satisfying $\sum_y p_{x\to y} = 1$. For each $i \in \mathbb{N}$, we denote by π_i the distribution over Ω with density $\pi_i(x) = \Pr[X_i = x]$. A *stationary distribution* π satisfies $\sum_{x\in\Omega} p_{x\to y}\pi(x) = \pi(y)$. A Markov chain is *reversible* if it has a stationary distribution π such that $\pi(x)p_{x\to y} = \pi(y)p_{y\to x}$ for all $x, y \in \Omega$. The *conductance* of a reversible Markov chain is defined as

$$\Phi := \inf_{\mathcal{S}\subseteq\Omega} \frac{\sum_{x\in\mathcal{S}}\sum_{y\in\Omega/\mathcal{S}} \pi(x)p_{x\to y}}{\min\{\sum_{x\in\mathcal{S}}\pi(x), \sum_{x\in\Omega/\mathcal{S}}\pi(x)\}}. \tag{3.12}$$

The theory of discrete-time quantum walks has also been well developed. Given a classical reversible Markov chain on Ω with transition probability p, we define a unitary operator U_p on $\mathbb{C}^{|\Omega|} \otimes \mathbb{C}^{|\Omega|}$ such that

$$U_p|x\rangle|0\rangle = |x\rangle|p_x\rangle, \text{ where } |p_x\rangle := \sum_{y\in\Omega}\sqrt{p_{x\to y}}|y\rangle. \tag{3.13}$$

The quantum walk is then defined as [85]

$$W_p := S\big(2U_p(I_\Omega \otimes |0\rangle\langle 0|)U_p^\dagger - I_\Omega \otimes I_\Omega\big), \tag{3.14}$$

where I_Ω is the identity map on $\mathbb{C}^{|\Omega|}$ and $S := \sum_{x,y\in\Omega} |x,y\rangle\langle y,x| = S^\dagger$ is the swap gate on $\mathbb{C}^{|\Omega|} \otimes \mathbb{C}^{|\Omega|}$.

To understand the quantum walk, it is essential to analyze the spectrum of W_p. First, observing that $W_p = S(2\Pi - I)$ where $\Pi = U_p(I_\Omega \otimes |0\rangle\langle 0|)U_p^\dagger = \sum_{x\in\Omega} |x\rangle\langle x| \otimes |p_x\rangle\langle p_x|$ projects onto the span of the states $|x\rangle \otimes |p_x\rangle$, we consider the eigenvector $|\lambda\rangle$ of $\Pi S \Pi$ with eigenvalue λ. We have $\Pi S\Pi = \sum_{x\in\Omega} D_{xy}|x\rangle\langle y| \otimes |p_x\rangle\langle p_y|$ where $D_{xy} := \sqrt{p_{x\to y}p_{y\to x}}$. Since $W_p|\lambda\rangle = S|\lambda\rangle$ and $W_pS|\lambda\rangle = 2\lambda S|\lambda\rangle - |\lambda\rangle$, the subspace span$\{|\lambda\rangle, S|\lambda\rangle\}$ is invariant under W_p. The eigenvalues of W_p within this subspace are $\lambda \pm i\sqrt{1-\lambda^2} = e^{\pm i \arccos\lambda}$. For more details, see [85].

The phase gap $\arccos\lambda \geq \sqrt{2(1-\lambda)} \geq \sqrt{2\delta}$, where δ is the spectral gap of D. Therefore, applying phase estimation using $O(1/\sqrt{\delta})$ calls to W_p suffices to distinguish the state corresponding to the stationary distribution of the classical Markov chain from the other eigenvectors.

3.2.2 Quantum speedup of MCMC sampling via simulated annealing

Consider a Markov chain with spectral gap Δ and stationary distribution π. Classically, it takes $\Theta(\frac{1}{\Delta}\log(1/\epsilon\pi_{\min}))$ steps to sample from a distribution $\tilde{\pi}$ such that $\|\tilde{\pi} - \pi\| \leq \epsilon$, where $\pi_{\min} := \min_i \pi_i$. Quantumly, [78] proved the following result about a sequence of slowly varying Markov chains:

Theorem 3.2.1 ([78, Theorem 2]). *Let $p_1, \ldots, p_r$ be the transition probabilities of r Markov* *chains with stationary distributions* $\pi_1, \dots, \pi_r$*, spectral gaps* $\delta_1, \dots, \delta_r$*, and quantum walk operators* $W_1, \dots, W_r$*, respectively; let* $\Delta := \min\{\delta_1, \dots, \delta_r\}$*. Assume that* $|\langle \pi_i | \pi_{i+1} \rangle|^2 \geq p$ *for some* $0 < p < 1$ *and all* $i \in [r-1]$*, and assume that we can efficiently prepare the state* $|\pi_1\rangle$ *(where each* $|\pi_i\rangle$ *is a quantum sample defined as in Section 3.1.3.2). Then, for any* $0 < \epsilon < 1$*, there is a quantum algorithm that produces a quantum state* $|\tilde{\pi}_r\rangle$ *such that* $\| |\tilde{\pi}_r\rangle - |\pi_r\rangle \| \leq \epsilon$*, using* $\tilde{O}(r/(p\sqrt{\Delta}))$ *steps of the quantum walk operators* $W_1, \dots, W_r$*, where the* $\tilde{O}$ *omits poly-logarithmic terms in* r*,* $1/\epsilon$*, and* $1/p\sqrt{\Delta}$*.*[9]

Their quantum algorithm produces the states $|\pi_1\rangle, \dots, |\pi_r\rangle$ sequentially, and can do so rapidly if consecutive states have significant overlap and the walks mix rapidly. Intuitively, this is achieved by amplitude amplification. However, to avoid overshooting, the paper uses a variant of standard amplitude amplification, known as $\pi/3$-*amplitude amplification* [1], that we now review.

Given two states $|\psi\rangle$ and $|\phi\rangle$, we let $\Pi_\psi := |\psi\rangle\langle\psi|$, $\Pi_\psi^\perp := I - \Pi_\psi$, $\Pi_\phi := |\phi\rangle\langle\phi|$, and $\Pi_\phi^\perp := I - \Pi_\phi$. Define the unitaries

$$R_\psi := \omega \Pi_\psi + \Pi_\psi^\perp, \quad R_\phi := \omega \Pi_\phi + \Pi_\phi^\perp \qquad \text{where} \quad \omega = e^{i\frac{\pi}{3}}. \tag{3.15}$$

Given $|\langle \psi | \phi \rangle|^2 \geq p$, it can be shown that $|\langle \phi | R_\psi R_\phi | \psi \rangle|^2 \geq 1 - (1-p)^3$. Recursively, one can establish the following:

Lemma 3.2.1 ([78, Lemma 1])**.** *Let* $|\psi\rangle$ *and* $|\phi\rangle$ *be two quantum states with* $|\langle \psi | \phi \rangle|^2 \geq p$ *for some* $0 < p \leq 1$*. Define the unitaries* R_ψ, R_ϕ *as in (3.15) and the unitaries* U_m *recursively as*

[9] Note that this is quadratically worse in $1/p$ than the Grover's algorithm [113] with complexity $O(1/\sqrt{p})$. This is because we use a simple fixed-point quantum search algorithm [1] that does not require knowing p in advance. Notice that there exist fixed-point quantum search algorithms that preserve the $O(1/\sqrt{p})$ speedup (e.g., [114], [115, Chapter 6]), but in our quantum algorithm, the simpler algorithm suffices as $p = \Theta(1)$ (see Lemma 3.4.2).

follows:

$$U_0 = I, \qquad U_{m+1} = U_m\, R_\psi\, U_m^\dagger\, R_\phi\, U_m. \tag{3.16}$$

Then we have

$$|\langle\phi|U_m|\psi\rangle|^2 \geq 1 - (1-p)^{3^m}, \tag{3.17}$$

and the unitaries in $\{R_\psi, R_\psi^\dagger, R_\phi, R_\phi^\dagger\}$ are used at most 3^m times in U_m.

Taking $m = \lceil \log_3(\ln(1/\epsilon)/p) \rceil$, the inner product between $|\phi\rangle$ and $U_m|\psi\rangle$ in (3.17) is at least $1-\epsilon$, and we use $3^m = O(\log(1/\epsilon)/p)$ unitaries from the set $\{R_\psi, R_\psi^\dagger, R_\phi, R_\phi^\dagger\}$.

To establish Theorem 3.2.1 by Lemma 3.2.1, it remains to construct the unitaries $R_i := \omega|\pi_i\rangle\langle\pi_i| + (I - |\pi_i\rangle\langle\pi_i|)$. In [78], this is achieved by phase estimation of the quantum walk operator W_i with precision $\sqrt{\Delta}/2$. Recall that if a classical Markov chain has spectral gap δ, then the corresponding quantum walk operator has phase gap of at least $2\sqrt{\delta}$ (see Section 3.2.1). Therefore, phase estimation with precision $\sqrt{\Delta}/2$ suffices to distinguish between $|\pi_i\rangle$ and other eigenvectors of W_i. As a result, we can take

$$R_i = \mathsf{PhaseEst}(W_i)^\dagger \big(I \otimes \big(\omega|0\rangle\langle 0| + (I - |0\rangle\langle 0|)\big)\big)\mathsf{PhaseEst}(W_i). \tag{3.18}$$

3.2.3 Quantum Chebyshev inequality

Assume we are given a unitary U such that

$$U|0\rangle|0\rangle = \sqrt{p}|0\rangle|\phi\rangle + |0^{\perp}\rangle, \tag{3.19}$$

where $|\phi\rangle$ is a normalized pure state and $(\langle 0| \otimes I)|0^{\perp}\rangle = 0$. If we measure the output state, we get 0 in the first register with probability p; by the Chernoff bound, it takes $\Theta(1/\epsilon^2)$ samples to estimate p within ϵ with high success probability. However, there is a more efficient quantum algorithm, called *amplitude estimation* [76], that estimates the value of p using only $O(1/\epsilon)$ calls to U:

Theorem 3.2.2 ([76, Theorem 12]). *Given U satisfying (3.19), the amplitude estimation algorithm in Figure 3.2 outputs an angle $\tilde{\theta}_p \in [-\pi, \pi]$ such that $\tilde{p} := \sin^2(\tilde{\theta}_p)$ satisfies*

$$|\tilde{p} - p| \leq \frac{2\pi\sqrt{p(1-p)}}{M} + \frac{\pi^2}{M^2} \tag{3.20}$$

with success probability at least $8/\pi^2$, using M calls to U and $U^{\dagger}$.

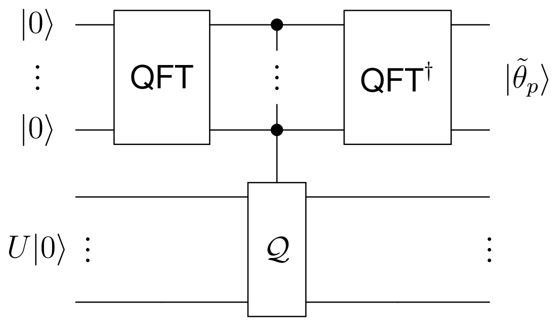


Figure 3.2: The quantum circuit for amplitude estimation.

Here QFT denotes the quantum Fourier transform over $\mathbb{Z}_M$ and $\mathcal{Q} := -US_0U^\dagger S_1$ where S_0 and S_1 are reflections about $|0\rangle$ and the target state, respectively, following the pattern of Grover search. The controlled-$\mathcal{Q}$ gate denotes the operation $\sum_{j=0}^{M-1} |j\rangle\langle j| \otimes \mathcal{Q}^j$. In fact, it was shown in the proof of [76, Theorem 12] that the state after applying the circuit in Figure 3.2 is

$$\frac{e^{i\theta_p}}{\sqrt{2}}|\tilde{\theta}_p\rangle|\Psi_+\rangle - \frac{e^{-i\theta_p}}{\sqrt{2}}|-\tilde{\theta}_p\rangle|\Psi_-\rangle \tag{3.21}$$

where $\theta_p \in [0,\pi]$ such that $p = \sin^2(\theta_p)$, and $\tilde{\theta}_p \in [0,\pi]$ is a random variable such that $\tilde{p} = \sin^2(\tilde{\theta}_p)$, and $|\Psi_\pm\rangle$ are two eigenvectors of $\mathcal{Q}$. Measuring the first register either gives $\tilde{\theta}_p$ or $-\tilde{\theta}_p$ with probability $1/2$, but since $\sin^2(\tilde{\theta}_p) = \sin^2(-\tilde{\theta}_p) = \tilde{p}$, this does not influence the success of Theorem 3.2.2.

In (3.20), if we take $M = \left\lceil 2\pi\left(\frac{2\sqrt{p}}{\epsilon} + \frac{1}{\sqrt{\epsilon}}\right)\right\rceil = O(1/\epsilon)$, we get

$$|\tilde{p} - p| \leq \frac{2\pi\sqrt{p(1-p)}}{2\pi}\epsilon + \frac{\pi^2}{4\pi^2}\epsilon^2 \leq \frac{\epsilon}{2} + \frac{\epsilon}{4} \leq \epsilon. \tag{3.22}$$

Furthermore, the success probability $8/\pi^2$ can be boosted to $1-\nu$ by executing the algorithm $\Theta(\log 1/\nu)$ times and taking the median of the estimates.

Amplitude estimation can be generalized from estimating a single probability $p \in [0,1]$ to estimating the expectation of a random variable. Assume that U is a unitary acting on $\mathbb{C}^S \otimes \mathbb{C}^{|\Omega|}$ such that

$$U|0\rangle|0\rangle = \sum_{x\in\Omega} \sqrt{p_x}|\psi_x\rangle|x\rangle \tag{3.23}$$

where $S \in \mathbb{N}$ and $\{|\psi_x\rangle : x \in \Omega\}$ are unit vectors in $\mathbb{C}^S$. Let

$$\mu_U := \sum_{x\in\Omega} p_x x, \qquad \sigma_U^2 := \sum_{x\in\Omega} p_x (x-\mu_U)^2 \tag{3.24}$$

denote the expectation and variance of the random variable, respectively. Several quantum algorithms have given speedups for estimating μ_U. Specifically, Ref. [11] showed how to estimate μ_U within additive error ϵ by $\tilde{O}(\sigma_U/\epsilon)$ calls to U and $U^\dagger$. Given an upper bound H and a lower bound $L > 0$ on the random variable, Ref. [116] showed how to estimate μ_U with multiplicative error ϵ using $\tilde{O}(\sigma_U/\epsilon\mu_U \cdot H/L)$ calls to U and $U^\dagger$. More recently, Ref. [88] mutually generalized these results and proposed a significantly better quantum algorithm:

Theorem 3.2.3 ([88, Theorem 3.5])**.** *There is a quantum algorithm that, given a quantum sampler U as in (3.23), an integer Δ_U, a value $H > 0$, and two reals $\epsilon, \delta \in (0,1)$, outputs an estimate $\tilde{\mu}_U$. If $\Delta_U \geq \sqrt{\sigma_U^2 + \mu_U^2}/\mu_U$ and $H > \mu_U$, then $|\tilde{\mu}_U - \mu_U| \leq \epsilon\mu_U$ with probability at least $1-\delta$, and the algorithm uses $\tilde{O}(\Delta_U/\epsilon \cdot \log^3(H/\mu_U)\log(1/\delta))$ calls to U and $U^\dagger$.*

The quantum algorithm works as follows. First, assume $\Omega \subseteq [L, H]$ for given real numbers $L, H \geq 0$, there is a basic estimation algorithm (denoted BasicEst) that estimates $H^{-1}\mu_U$ up to ϵ-multiplicative error:

However, usually the bounds L and H are not explicitly given. In this case, Ref. [88] considered the truncated mean $\mu_{<b}$ defined by replacing the outcomes larger than b with 0. The paper then runs Algorithm 6 (BasicEst) to estimate $\mu_{<b}/b$. A crucial observation is that $\sqrt{b/\mu_{<b}}$ is smaller than Δ_U for large values of b, and it becomes larger than Δ_U when $b \approx \mu_U\Delta_U^2$. As a result, by repeatedly running BasicEst with Δ_U quantum samples, and applying $O(\log(H/L))$ steps of a binary search on the values of b, the first non-zero value is obtained when $b/\Delta_U^2 \approx \mu_U$.

Algorithm 6: BasicEst: the basic estimation algorithm.

```
Input: A quantum sampler $U$ acting on $\mathbb{C}^S \otimes \mathbb{C}^{|\Omega|}$, interval $[L, H]$, precision
       parameter $\epsilon \in (0,1)$, failure parameter $\delta \in (0,1)$.
Output: $\epsilon$-multiplicative approximation of $H^{-1}\mu_U$.
1 Use controlled rotation to implement a unitary $R_{L,H}$ acting on $\mathbb{C}^{|\Omega|} \otimes \mathbb{C}^2$ such that for
    all $x \in \Omega$, $R_{L,H}|x\rangle|0\rangle = \begin{cases} |x\rangle(\sqrt{1-\frac{x}{H}}|0\rangle + \sqrt{\frac{x}{H}}|1\rangle) & \text{if } L \leq x < H \\ |x\rangle|0\rangle & \text{otherwise} \end{cases}$;
2 Let $V = (I_S \otimes R_{L,H})(U \otimes I_2)$ and $\Pi = I_S \otimes I_\Omega \otimes |1\rangle\langle 1|$;
3 for $i = 1, \ldots, \Theta(\log(1/\delta))$ do
4     Compute $\tilde{p}_i$ by Theorem 3.2.2 with $U \leftarrow V$, $S_1 \leftarrow 2\Pi - I$, and
        $M \leftarrow \Theta(1/(\epsilon\sqrt{H^{-1}\mu_U}))$;
5 Return $\tilde{p} = \mathrm{median}\{\tilde{p}_1, \ldots, \tilde{p}_{\Theta(\log(1/\delta))}\}$.
```

In [88], more precise truncation means are used to improve the precision of the result to $\tilde{O}(1/\epsilon)$ and remove the dependence on L.

Note that the quantum algorithm for Theorem 3.2.3 only relies on BasicEst. This is crucial when we estimate the mean of our simulated annealing algorithm in different iterations *nondestructively* (see Section 3.4.2 for more details).

3.2.4 Hit-and-run walk

As introduced in Section 3.1.3.1, there are various random walks that mix fast in a convex body K, such as the grid walk [65] and the ball walk [69, 82]. In this paper, we mainly use the *hit-and-run walk* [71, 72, 94]. It is defined as follows:

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

2. Move to a uniform random point along the chord $\ell \cap \mathrm{K}$.

For any two points $p, q \in \mathrm{K}$, we let $\ell(p, q)$ denote the length of the chord in K through p and q. Then the transition probability of the hit-and-run walk is determined by the following lemma:

Lemma 3.2.2 ([71, Lemma 3]). *If the current point of the hit-and-run walk is u, then the density function of the distribution of the next point $x \in \mathrm{K}$ is*

$$p_u(x) = \frac{2}{nv_n} \cdot \frac{1}{\ell(u,x)|x-u|^{n-1}}, \tag{3.25}$$

where $v_n := \pi^{\frac{n}{2}}/\Gamma(1+\frac{n}{2})$ is the volume of the n-dimensional unit ball. In other words, the probability that the next point is in a (measurable) set $\mathrm{A} \subseteq \mathrm{K}$ is

$$P_u(\mathrm{A}) = \int_{\mathrm{A}} \frac{2}{nv_n} \cdot \frac{1}{\ell(u,x)|x-u|^{n-1}} \, \mathrm{d}x. \tag{3.26}$$

In general, we can also define a hit-and-run walk with a given density. Let f be a density function in $\mathbb{R}^n$. For any points $u, v \in \mathbb{R}^n$, we let

$$\mu_f(u,v) := \int_0^1 f((1-t)u + tv) \, \mathrm{d}t. \tag{3.27}$$

For any line ℓ, let ℓ^+ and ℓ^- be the endpoints of the chord $\ell \cap \mathrm{K}$ (with $+$ and $-$ assigned arbitrarily). The density f specifies the following hit-and-run walk:

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

2. Move to a random point x along the chord $\ell \cap \mathrm{K}$ with density $\frac{f(x)}{\mu_f(\ell^-,\ell^+)}$.

Let π_{K} denote the uniform distribution over K. Smith [94] proved that the stationary distribution of the hit-and-run walk with uniform density is π_{K}. Furthermore, Lovász and Vempala [72] proved that the hit-and-run walk mixes rapidly from any initial distribution:

Theorem 3.2.4 ([72, Theorem 1.1]). *Let* K *be a convex body that satisfies (3.2):* $B_2(0,r) \subseteq \mathrm{K} \subseteq B_2(0,R)$*. Let* σ *be a starting distribution and let* $\sigma^{(m)}$ *be the distribution of the current point after* m *steps of the hit-and-run walk in* K*. Let* $\epsilon > 0$*, and suppose that the density function* $\mathrm{d}\sigma/\mathrm{d}\pi_{\mathrm{K}}$ *is upper bounded by* M *except on a set* S *with* $\sigma(\mathrm{S}) \leq \epsilon/2$*. Then for any*

$$m > 10^{10}\frac{n^2R^2}{r^2}\ln\frac{M}{\epsilon}, \tag{3.28}$$

the total variation distance between $\sigma^{(m)}$ *and* π_{K} *is less than* ϵ*.*

Theorem 3.2.4 can also be generalized to exponential distributions on K:

Theorem 3.2.5 ([72, Theorem 1.3]). *Let* $\mathrm{K} \subset \mathbb{R}^n$ *be a convex body and let* f *be a density supported on* K *that is proportional to* e^{-a^Tx} *for some vector* $a \in \mathbb{R}^n$*. Assume that the level set of* f *of probability* $1/8$ *contains a ball of radius* r*, and* $\mathbb{E}_f(|x - z_f|^2) \leq R^2$*, where* z_f *is the centroid of* f*. Let* σ *be a starting distribution and let* σ^m *be the distribution for the current point after* m *steps of the hit-and-run walk applied to* f*. Let* $\epsilon > 0$*, and suppose that the density function* $\frac{\mathrm{d}\sigma}{\mathrm{d}\pi_f}$ *is upper bounded by* M *except on a set* S *with* $\sigma(S) \leq \frac{\epsilon}{2}$*. Then for*

$$m > 10^{30}\frac{n^2R^2}{r^2}\ln^5\frac{MnR}{r\epsilon},$$

the total variation distance between σ^m *and* π_f *is less than* ϵ*.*

Roughly speaking, the proofs of Theorem 3.2.4 and Theorem 3.2.5 have two steps. First, for any random walk on a continuous domain Ω with transition probability p and stationary distribution π, we define its conductance (which generalizes the discrete case in Eq. (3.12)) as

$$\Phi := \inf_{\mathcal{S}\subseteq\Omega} \frac{\int_{\mathcal{S}}\int_{\Omega/\mathcal{S}} \mathrm{d}x\,\mathrm{d}y\,\pi_x p_{x\to y}}{\min\{\int_{\mathcal{S}}\mathrm{d}x\,\pi_x, \int_{\Omega/\mathcal{S}}\mathrm{d}x\,\pi_x\}}. \tag{3.29}$$

It is well-known that the mixing time of this random walk is proportional to $1/\Phi^2$ up to logarithmic factors. This is captured by the following proposition:

Proposition 3.2.1 ([69, Corollary 1.5]). *Let $M := \sup_{\mathcal{S}\subseteq\Omega}\frac{\sigma(\mathcal{S})}{\pi(\mathcal{S})}$ where σ is the initial distribution. Then for every $\mathcal{S}\subseteq\Omega$,*

$$\left|\sigma^{(k)}(\mathcal{S}) - \pi(\mathcal{S})\right| \le \sqrt{M}\left(1-\frac{1}{2}\Phi^2\right)^k. \tag{3.30}$$

Furthermore, the conductance in Proposition 3.2.1 can be relaxed to that of sets with a fixed small probability p:

Proposition 3.2.2 ([69, Corollary 1.6]). [10] *Let $M := \sup_{\mathcal{S}\subseteq\Omega}\frac{\sigma(\mathcal{S})}{\pi(\mathcal{S})}$. If the conductance for all connected, measurable set $\mathrm{A}\subseteq\Omega$ such that $\pi(\mathrm{A}) = p \le 1/2$ is at least Φ_p, then for all $\mathrm{S}\subseteq\Omega$, we have*

$$\left|\sigma^{(k)}(\mathrm{S}) - \pi(\mathrm{S})\right| \le 2Mp + 2M\left(1-\frac{1}{2}\Phi_p^2\right)^k. \tag{3.31}$$

Second, Ref. [72] proved a lower bound on the conductance of the hit-and-run walk with exponential density:

[10] Note that in the original statement ([69, Corollary 1.6]), the conditions are given in a slightly different formulation, but it is not hard to obtain the conditions in the original formulation from the conditions of this proposition.

Proposition 3.2.3 ([72, Theorem 6.9])**.** *Let f be a density in $\mathbb{R}^n$ proportional to $e^{-a^T x}$ whose support is a convex body* K *of diameter* d*. Assume that* $\mathrm{B}_2(0, r) \subseteq \mathrm{K}$*. Then for any subset* S *with* $\pi_f(\mathrm{S}) = p \leq 1/2$*, the conductance of the hit-and-run walk satisfies*

$$\phi(S) \geq \frac{r}{10^{13} nd \ln(nd/rp)}. \tag{3.32}$$

Proposition 3.2.1 and Proposition 3.2.3 imply Theorem 3.2.4 and Theorem 3.2.5; complete proofs are given in [72].

For the conductance of the hit-and-run walk with a uniform distribution, Ref. [72] established a stronger lower bound that is independent of p:

Proposition 3.2.4 ([72, Theorem 4.2])**.** *Assume that* K *has diameter* d *and contains a unit ball. Then the conductance of the hit-and-run in* K *with uniform distribution is at least* $\frac{1}{2^{24} nd}$.

3.3 Theory of continuous-space quantum walks

In this section, we develop the theory of continuous-space, discrete-time quantum walks.

Specifically, we generalize the discrete-time quantum walk of Szegedy [85] to continuous space. Let $n \in \mathbb{N}$ and suppose Ω is a continuous[11] subset of $\mathbb{R}^n$. A probability transition density p on Ω is a continuous function $p\colon \Omega \times \Omega \to [0, +\infty)$ such that

$$\int_\Omega \mathrm{d}y\, p(x, y) = 1 \qquad \forall\, x \in \Omega. \tag{3.33}$$

We also write $p_{x \to y} := p(x, y)$ for the transition density from x to y. Together, Ω and p specify a

[11]We say that Ω is continuous if for any $x, y \in \Omega$ there is a continuous function $f_{x,y}\colon [0, 1] \to \Omega$ such that $f_{x,y}(0) = x$ and $f_{x,y}(1) = y$.

continuous-space Markov chain that we denote (Ω, p) throughout the paper.

For background on the mathematical foundations of quantum mechanics over continuous state spaces, see [117, Chapter 1]. In this section, we treat quantum states as square integrable functions $f\colon \Omega \to \mathbb{R}$ in $L^2(\Omega)$ if $\int_\Omega \mathrm{d}x\, |f(x)|^2 < \infty$. The inner product $\langle \cdot, \cdot \rangle$ on $L^2(\Omega)$ is defined by

$$\langle f, g \rangle := \int_\Omega \mathrm{d}x\, f(x) g(x) \qquad \forall\, f, g \in L^2(\Omega). \tag{3.34}$$

Note that by the Cauchy-Schwarz inequality, we have

$$|\langle f, g \rangle|^2 \leq \left(\int_\Omega \mathrm{d}x\, |f(x)|^2 \right) \left(\int_\Omega \mathrm{d}x\, |g(x)|^2 \right) < \infty; \tag{3.35}$$

the norm of an $f \in L^2(\Omega)$ is subsequently defined as $\|f\| := \sqrt{\langle f, f \rangle}$. The pure states in Ω correspond to functions in the set

$$\mathrm{St}(\Omega) := \left\{ f\colon \Omega \to \mathbb{R} \;\middle|\; \int_\Omega \mathrm{d}x\, |f(x)|^2 = 1 \right\}. \tag{3.36}$$

The computational basis elements $|y\rangle$ for all $y \in \Omega$ correspond to Dirac delta functions $\delta(x - y)$ centered at y, where $\delta(x) = 0$ for all $x \neq 0$, and $\int_{\mathbb{R}^n} \delta(x)\, \mathrm{d}x = 1$. Delta functions are not members of the Hilbert space $L^2(\Omega)$, however we interpret them in the following sense: for any $y \in \mathrm{int}_\epsilon(\Omega)$ we consider a normalized Gaussian with width ϵ, given by $\delta_{y,\epsilon}(x) \propto \frac{1}{(2\pi\epsilon^2)^{n/2}} e^{-\|x-y\|^2/2\epsilon^2}$ for $x \in \Omega$ and 0 for $x \notin \Omega$. It is clear that $\delta_{y,\epsilon} \in L_2(\Omega)$ and its behavior approaches that of the delta function. In the remainder of the section statements such as $A = B$ are to be interpreted as $\lim_{\epsilon \to 0} |A_\epsilon - B_\epsilon| = 0$ where A_ϵ, B_ϵ are obtained from A, B by replacing every occurence of a computational basis vector $|y\rangle$ by $\delta_{y,\epsilon}$.[12] Integrals over $L_2(\Omega)$ are computed pointwise. It can be verified that

$$\int_\Omega \mathrm{d}x\, |x\rangle\langle x| = I \tag{3.37}$$

and

$$\langle x|x'\rangle = \delta(x - x'), \quad \forall x, x' \in \Omega. \tag{3.38}$$

3.3.1 Continuous-space quantum walk

Given a transition density function p, the quantum walk is characterized by the following states:

$$|\phi_x\rangle := |x\rangle \otimes \int_\Omega \mathrm{d}y\, \sqrt{p_{x\to y}}|y\rangle \qquad \forall x \in \mathbb{R}^n. \tag{3.39}$$

Since $p_{x\to y}$ is a normalized probability density function, $|\phi_x\rangle \in L_2(\Omega)$. Now, denote

$$U := \int_\Omega \mathrm{d}x\, |\phi_x\rangle(\langle x| \otimes \langle 0|),\ \Pi := \int_\Omega \mathrm{d}x\, |\phi_x\rangle\langle\phi_x|,\ S := \int_\Omega\int_\Omega \mathrm{d}x\,\mathrm{d}y\, |x,y\rangle\langle y,x|. \tag{3.40}$$

[12] As in most treatments of continous quantum mechanics, we shall not be fully rigorous with respect to operations such as interchanging orders of limits. We have two reasons to believe that pathological cases do not occur: (1) The states that occur during the algorithm are mostly well-behaved and correspond to probability distributions that are obtained during classical geometric random walks. (2) We later show Section 3.5.1 that our algorithm can also be executed discretely, and we work in the continuous setting for ease of analysis.

Notice that Π is the projection onto $\operatorname{span}\{|\phi_x\rangle\}_{x\in\mathbb{R}^n}$ because

$$\Pi^2 = \int_\Omega \int_\Omega \mathrm{d}x\,\mathrm{d}x'\,|\phi_x\rangle\langle\phi_x|\phi_{x'}\rangle\langle\phi_{x'}| = \int_\Omega \int_\Omega \mathrm{d}x\,\mathrm{d}x'\,\delta(x-x')|\phi_x\rangle\langle\phi_{x'}| = \Pi, \tag{3.41}$$

and S is the swap operator for the two registers. A single step of the quantum walk is defined as the unitary operator

$$W := S(2\Pi - I). \tag{3.42}$$

The first main result of this subsection is the following theorem:

Theorem 3.3.1. *Let*

$$D := \int_\Omega \int_\Omega \mathrm{d}x\,\mathrm{d}y\,\sqrt{p_{x\to y}p_{y\to x}}|x\rangle\langle y| \tag{3.43}$$

denote the discriminant operator *of p. Let Λ be the set of eigenvalues of D, so that $D = \int_\Lambda \mathrm{d}\lambda\,\lambda|\lambda\rangle\langle\lambda|$. Then the eigenvalues of the quantum walk operator W in (3.42) are ± 1 and $\lambda \pm i\sqrt{1-\lambda^2}$ for all $\lambda \in \Lambda$.*

To prove Theorem 3.3.1, we first prove the following lemma:

Lemma 3.3.1. *For any $\lambda \in \Lambda$, we have $|\lambda| \le 1$.*

Proof. Since λ is an eigenvalue of D, we have $D|\lambda\rangle = \lambda|\lambda\rangle$. As a result, we have

$$
\begin{aligned}
|\lambda|\delta(0) &= |\lambda|\langle\lambda|\lambda\rangle && (3.44)\\
&= |\langle\lambda|D|\lambda\rangle| && (3.45)\\
&= \Big|\int_\Omega\int_\Omega \mathrm{d}x\,\mathrm{d}y\,\sqrt{p_{y\to x}p_{x\to y}}\langle\lambda|x\rangle\langle y|\lambda\rangle\Big| && (3.46)\\
\text{(by Cauchy-Schwarz)}\quad &\leq \sqrt{\Big(\int_\Omega\int_\Omega \mathrm{d}x\,\mathrm{d}y\,p_{y\to x}|\langle y|\lambda\rangle|^2\Big)\Big(\int_\Omega\int_\Omega \mathrm{d}x\,\mathrm{d}y\,p_{x\to y}|\langle\lambda|x\rangle|^2\Big)}\\
&= \sqrt{\Big(\int_\Omega \mathrm{d}y\,|\langle y|\lambda\rangle|^2\Big)\Big(\int_\Omega \mathrm{d}x\,|\langle\lambda|x\rangle|^2\Big)} \quad \Big(\text{by} \int_\Omega \mathrm{d}y\,p_{x\to y} = 1\Big) && (3.47)\\
&= \int_\Omega \mathrm{d}x\,\langle\lambda|x\rangle\langle x|\lambda\rangle && (3.48)\\
&= \langle\lambda|\Big(\int_\Omega \mathrm{d}x\,|x\rangle\langle x|\Big)|\lambda\rangle \quad \text{(by (3.37))} && (3.49)\\
&= \delta(0). && (3.50)
\end{aligned}
$$

Hence the result follows. □

Proof of Theorem 3.3.1. Define an isometry

$$T := \int_\Omega \mathrm{d}x\,|\phi_x\rangle\langle x| = \int_\Omega\int_\Omega \mathrm{d}x\,\mathrm{d}y\,\sqrt{p_{x\to y}}|x,y\rangle\langle x|. \tag{3.51}$$

Then

$$TT^\dagger = \int_\Omega\int_\Omega \mathrm{d}x\,\mathrm{d}y\,|\phi_x\rangle\langle x|y\rangle\langle\phi_y| = \int_\Omega \mathrm{d}x\,|\phi_x\rangle\langle\phi_x| = \Pi, \tag{3.52}$$

and

$$T^\dagger T = \int_\Omega \int_\Omega \mathrm{d}x\,\mathrm{d}y\,|x\rangle\langle\phi_x|\phi_y\rangle\langle y| \tag{3.53}$$

$$= \int_\Omega \int_\Omega \int_\Omega \int_\Omega \mathrm{d}x\,\mathrm{d}y\,\mathrm{d}a\,\mathrm{d}b\,\langle x|y\rangle\langle a|b\rangle\sqrt{p_{x\to a}p_{y\to b}}|x\rangle\langle y| \tag{3.54}$$

$$= \int_\Omega \int_\Omega \mathrm{d}x\,\mathrm{d}a\,p_{x\to a}|x\rangle\langle x| \tag{3.55}$$

$$= \int_\Omega \mathrm{d}x\,|x\rangle\langle x| \tag{3.56}$$

$$= I. \tag{3.57}$$

Furthermore,

$$T^\dagger S T = \int_\Omega \int_\Omega \mathrm{d}x\,\mathrm{d}y\,|x\rangle\langle\phi_x|S|\phi_y\rangle\langle y| \tag{3.58}$$

$$= \int_\Omega \int_\Omega \int_\Omega \int_\Omega \mathrm{d}x\,\mathrm{d}y\,da\,\mathrm{d}b\,\langle x,a|S|y,b\rangle\sqrt{p_{x\to a}p_{y\to b}}|x\rangle\langle y| \tag{3.59}$$

$$= \int_\Omega \int_\Omega \mathrm{d}x\,\mathrm{d}a\,\sqrt{p_{x\to a}p_{a\to x}}|x\rangle\langle a| \tag{3.60}$$

$$= D. \tag{3.61}$$

As a result, for any $\lambda \in \Lambda$ we have

$$WT|\lambda\rangle = S(2\Pi - I)T|\lambda\rangle = (2STT^\dagger T - ST)|\lambda\rangle = (2ST - ST)|\lambda\rangle = ST|\lambda\rangle. \tag{3.62}$$

Similarly, we have

$$\begin{aligned} WST|\lambda\rangle &= S(2\Pi - I)ST|\lambda\rangle \\ &= (2STT^\dagger ST - S^2T)|\lambda\rangle = (2STD - T)|\lambda\rangle = (2\lambda S - I)T|\lambda\rangle. \end{aligned} \tag{3.63}$$

By Lemma 3.3.1, $|\lambda| \leq 1$. As a result, we have

$$W\big(I - (\lambda + i\sqrt{1-\lambda^2})S\big)T|\lambda\rangle = WT|\lambda\rangle - (\lambda + i\sqrt{1-\lambda^2})WST|\lambda\rangle \tag{3.64}$$

$$= ST|\lambda\rangle - (\lambda + i\sqrt{1-\lambda^2})(2\lambda S - I)T|\lambda\rangle \tag{3.65}$$

$$= \big(S - (\lambda + i\sqrt{1-\lambda^2})(2\lambda S - I)\big)T|\lambda\rangle \tag{3.66}$$

$$= (\lambda + i\sqrt{1-\lambda^2})\big(I - (\lambda + i\sqrt{1-\lambda^2})S\big)T|\lambda\rangle; \tag{3.67}$$

in other words, $\lambda + i\sqrt{1-\lambda^2}$ is an eigenvalue of W with eigenvector $\big(I - (\lambda + i\sqrt{1-\lambda^2})S\big)T|\lambda\rangle$.

Similarly, we have

$$W\big(I - (\lambda - i\sqrt{1-\lambda^2})S\big)T|\lambda\rangle = (\lambda - i\sqrt{1-\lambda^2})\big(I - (\lambda - i\sqrt{1-\lambda^2})S\big)T|\lambda\rangle, \tag{3.68}$$

i.e., $\lambda - i\sqrt{1-\lambda^2}$ is an eigenvalue of W with eigenvector $\big(I - (\lambda - i\sqrt{1-\lambda^2})S\big)T|\lambda\rangle$.

Finally, note that for any vector $|u\rangle$ in the orthogonal complement of the space $\mathrm{span}_{\lambda\in\Lambda}\{T|\lambda\rangle, ST|\lambda\rangle\}$, W simply acts as $-S$ since

$$\Pi = TT^\dagger = \int_\Lambda \mathrm{d}\lambda\, T|\lambda\rangle\langle\lambda|T^\dagger, \tag{3.69}$$

which projects onto $\mathrm{span}_{\lambda\in\Lambda}\{T|\lambda\rangle\}$. In this orthogonal complement subspace, the eigenvalues are ± 1 because $S^2 = I$. $\square$

3.3.2 Stationary distribution

Classically, the density $\pi = (\pi_x)_{x\in\Omega}$ corresponding to the stationary distribution of a Markov chain (Ω, p) satisfies

$$\int_\Omega \mathrm{d}x\, \pi_x = 1; \qquad \int_\Omega \mathrm{d}y\, p_{y\to x}\pi_y = \pi_x \qquad \forall\, x \in \Omega. \tag{3.70}$$

In other words, we can naturally define a transition operator as

$$P := \int_\Omega \int_\Omega \mathrm{d}x\, \mathrm{d}y\, p_{y\to x}|x\rangle\langle y|, \tag{3.71}$$

and the stationary density π satisfies $P\pi = \pi$. The Markov chain (Ω, p) is *reversible* if there exists a classical density $\sigma = (\sigma_x)_{x\in\Omega}$ such that

$$p_{y\to x}\sigma_y = p_{x\to y}\sigma_x \qquad \forall\, x, y \in \Omega. \tag{3.72}$$

(This is called the *detailed balance condition*.) Notice that for all $x \in \Omega$,

$$\int_\Omega \mathrm{d}y\, p_{y\to x}\sigma_y = \int_\Omega \mathrm{d}y\, p_{x\to y}\sigma_x = \sigma_x \int_\Omega \mathrm{d}y\, p_{x\to y} = \sigma_x; \tag{3.73}$$

therefore, we must have $P\sigma = \sigma$, i.e., σ is a stationary density of P. In this paper, we focus on Markov chains (Ω, p) that are reversible and have a unique stationary distribution (i.e., $\sigma = \pi$).

Such assumptions are natural for Markov chains in practice, including the Metropolis-Hastings algorithm, simple random walks on graphs, etc.

We point out that if π is the classical stationary density of a reversible Markov chain (Ω, p), then

$$|\pi_W\rangle := \int_\Omega \mathrm{d}x \, \sqrt{\pi_x} |\phi_x\rangle \tag{3.74}$$

is the unique eigenvalue-1 eigenstate of the quantum walk operator W restricted to the subspace $\mathrm{span}_{\lambda \in \Lambda}\{T|\lambda\rangle, ST|\lambda\rangle\}$. First, a simple calculation shows that

$$\begin{aligned}
W|\pi_W\rangle &= S(2\Pi - I)|\pi_W\rangle && (3.75)\\
&= S|\pi_W\rangle && (3.76)\\
&= \left(\int_\Omega \int_\Omega \mathrm{d}x \, \mathrm{d}y \, |x, y\rangle\langle y, x|\right)\left(\int_\Omega \int_\Omega \mathrm{d}x \, \mathrm{d}y \, \sqrt{\pi_y p_{y\to x}} |y, x\rangle\right) && (3.77)\\
&= \int_\Omega \int_\Omega \mathrm{d}x \, \mathrm{d}y \, \sqrt{\pi_y p_{y\to x}} |x, y\rangle && (3.78)\\
&= \int_\Omega \int_\Omega \mathrm{d}x \, \mathrm{d}y \, \sqrt{\pi_x p_{x\to y}} |x\rangle|y\rangle && (3.79)\\
&= \int_\Omega \mathrm{d}x \, \sqrt{\pi_x} |x\rangle \left(\int_\Omega \mathrm{d}y \, \sqrt{p_{x\to y}} |y\rangle\right) && (3.80)\\
&= \int_\Omega \mathrm{d}x \, \sqrt{\pi_x} |\phi_x\rangle && (3.81)\\
&= |\pi_W\rangle, && (3.82)
\end{aligned}$$

where (3.76) follows from $|\pi_W\rangle \in \mathrm{span}_{x\in\Omega}\{|\phi_x\rangle\}$, (3.77) follows from the definition of S in (3.40), (3.79) follows from (3.72), and (3.79) follows from the definition of $|\phi_x\rangle$ in (3.39). Thus $|\pi_W\rangle$ is an eigenvector of W with eigenvalue 1. On the other hand, since (Ω, p) is reversible, P

is similar to D: if we denote $D_\pi := \int_\Omega \mathrm{d}x\, \sqrt{\pi_x}|x\rangle\langle x|$, then

$$\begin{aligned} D_\pi D D_\pi^{-1} &= \Big(\int_\Omega \mathrm{d}x \sqrt{\pi_x}|x\rangle\langle x|\Big)\Big(\int_\Omega\int_\Omega \mathrm{d}x\,\mathrm{d}y \sqrt{p_{x\to y}p_{y\to x}}|x\rangle\langle y|\Big)\Big(\int_\Omega \mathrm{d}y \sqrt{\pi_y^{-1}}|y\rangle\langle y|\Big) \\ &= \int_\Omega\int_\Omega \mathrm{d}x\,\mathrm{d}y\, \sqrt{\pi_x\pi_y^{-1}p_{x\to y}p_{y\to x}}|x\rangle\langle y| && (3.83) \\ &= \int_\Omega\int_\Omega \mathrm{d}x\,\mathrm{d}y\, p_{y\to x}|x\rangle\langle y| \quad \text{(by (3.72))} && (3.84) \\ &= P. && (3.85) \end{aligned}$$

As a result, D and P have the same set of eigenvalues. Furthermore, Lemma 3.3.1 implies that all eigenvalues of P have norm at most 1, and the proof of Theorem 3.3.1 shows that $|\pi_W\rangle$ is the unique eigenvector with this eigenvalue within $\mathrm{span}_{\lambda\in\Lambda}\{T|\lambda\rangle, ST|\lambda\rangle\}$.

The state

$$|\pi\rangle := \int_\Omega \mathrm{d}x\, \sqrt{\pi_x}|x\rangle \tag{3.86}$$

represents a quantum sample from the density π; in particular, measuring $|\pi\rangle$ in the computational basis gives a classical sample from π. Furthermore, the unitary operator in (3.40) satisfies

$$U^\dagger|\pi_W\rangle = \Big(\int_\Omega \mathrm{d}x\, |x\rangle|0\rangle\langle\phi_x|\Big)\Big(\int_\Omega \mathrm{d}x\, \sqrt{\pi_x}|\phi_x\rangle\Big) = \int_\Omega \mathrm{d}x\, \sqrt{\pi_x}|x\rangle|0\rangle = |\pi\rangle|0\rangle, \tag{3.87}$$

so we have $U|\pi\rangle|0\rangle = |\pi_W\rangle$.

3.4 Quantum speedup for volume estimation

In this section, we present and analyze our quantum algorithm for volume estimation. First, we review the classical state-of-the-art volume estimation algorithm in Section 3.4.1. We then describe our quantum algorithm for estimating the volume of well-rounded convex bodies (i.e., $R/r = O(\sqrt{n})$) with query complexity $\tilde{O}(n^{2.5}/\epsilon)$ in Section 3.4.2, with detailed proofs given in Section 3.4.3. Finally, we remove the well-rounded condition by giving a quantum algorithm with interlaced rounding and volume estimation with additional cost $\tilde{O}(n^{2.5})$ in each iteration in Section 3.4.4.

3.4.1 Review of classical algorithms for volume estimation

The state-of-the-art classical algorithm for volume estimation uses $\tilde{O}(n^4 + n^3/\epsilon^2)$ classical queries, where $\tilde{O}(n^4)$ queries are used to construct the affine transformation that makes convex body well-rounded [73] and $\tilde{O}(n^3/\epsilon^2)$ queries are used to estimate the volume of the well-rounded convex body (after the affine transformation) [82].

We now review the algorithm of [73] for estimating volumes of well-rounded convex bodies. This algorithm estimates the volume of a convex body obtained by the following *pencil construction*. Define the cone

$$\mathrm{C} := \Big\{x \in \mathbb{R}^{n+1} : x_0 \geq 0, \sum_{i=1}^{n} x_i^2 \leq x_0^2\Big\}. \tag{3.88}$$

Let K' be the intersection of the cone C and a cylinder $[0, 2D] \times \mathrm{K}$, i.e.,

$$\mathrm{K}' := ([0, 2D] \times \mathrm{K}) \cap \mathrm{C} \tag{3.89}$$

(recall $D = R/r$). Without loss of generality we renormalize to $r = 1$, so that $\mathrm{B}_2(0, 1) \subseteq \mathrm{K} \subseteq \mathrm{B}_2(0, D)$. Since $D\mathrm{Vol}(\mathrm{K}) \leq \mathrm{Vol}(\mathrm{K}') \leq 2D\mathrm{Vol}(\mathrm{K})$, with the knowledge of $\mathrm{Vol}(\mathrm{K}')$ we can estimate $\mathrm{Vol}(\mathrm{K})$ with multiplicative error ϵ by generating $O(1/\epsilon^2)$ sample points from the uniform distribution on $[0, 2D] \times \mathrm{K}$ and then counting how many of them fall into K'. Such an approximation succeeds with high probability by a Chernoff-type argument (see Section 3.4.3.1 for a formal proof).

Lovász and Vempala [73] considers simulated annealing under the pencil construction. For any $a > 0$, define

$$Z(a) := \int_{\mathrm{K}'} e^{-ax_0}\,\mathrm{d}x. \tag{3.90}$$

It can be shown that for any $a \leq \epsilon/D$,

$$(1 - \epsilon)\mathrm{Vol}(\mathrm{K}') \leq Z(a) \leq \mathrm{Vol}(\mathrm{K}'). \tag{3.91}$$

On the other hand, it is shown in [73, Section 2.3] that for any $a \geq 2n$ and $\epsilon > (3/4)^n$,

$$(1 - \epsilon) \int_{\mathrm{C}} e^{-ax_0}\,\mathrm{d}x \leq Z(a) \leq \int_{\mathrm{C}} e^{-ax_0}\,\mathrm{d}x. \tag{3.92}$$

This suggests using a simulated annealing procedure for estimating $\mathrm{Vol}(\mathrm{K}')$. Specifically, if we select a sequence $a_0 > a_1 > \cdots > a_m$ for which $a_0 = 2n$ and $a_m \leq \epsilon/D$, then we can estimate $\mathrm{Vol}(\mathrm{K}')$ by

$$Z(a_m) = Z(a_0)\prod_{i=0}^{m-1}\frac{Z(a_{i+1})}{Z(a_i)}. \tag{3.93}$$

(Note that this procedure uses an increasing sequence of temperatures $1/a_i$, unlike standard simulated annealing in which temperature is decreased.)

Let π_i be the probability distribution over K' with density proportional to $e^{-a_i x_0}$, i.e., $\mathrm{d}\pi_i(x) = \frac{e^{-a_i x_0}}{Z(a_i)}\mathrm{d}x$. Let X_i be a random sample from π_i, and let $(X_i)_0$ be its first coordinate; define $V_i := e^{(a_i - a_{i+1})(X_i)_0}$. We have

$$\mathbb{E}_{\pi_i}[V_i] = \int_{\mathrm{K}'} e^{(a_i-a_{i+1})x_0}\,\mathrm{d}\pi_i(x) = \int_{\mathrm{K}'} e^{(a_i-a_{i+1})x_0}\frac{e^{-a_i x_0}}{Z(a_i)}\,\mathrm{d}x = \frac{Z(a_{i+1})}{Z(a_i)}. \tag{3.94}$$

Furthermore, if the simulated annealing schedule satisfies $a_{i+1} \geq (1-\frac{1}{\sqrt{n}})a_i$, then V_i satisfies (see [73, Lemma 4.1])

$$\frac{\mathbb{E}_{\pi_i}[V_i^2]}{\mathbb{E}_{\pi_i}[V_i]^2} \leq \left(\frac{a_{i+1}^2}{a_i(2a_{i+1}-a_i)}\right)^{n+1} < 8 \qquad \forall\, i \in [m], \tag{3.95}$$

i.e., the variance of V_i is bounded by a constant multiple of the square of its expectation. Thus, this simulated annealing procedure constitutes *Chebyshev cooling* (see also Section 3.4.3.3), ensuring its correctness (see Proposition 3.4.1).

Algorithm 7 presents this approach in detail. Note that sampling from π_0 in Line 2 is straightforward: select a random positive real number x_0 from the distribution with density e^{-2nx} and choose a uniformly random point $(v_1, \ldots, v_n)$ from the unit ball. If $X = (x_0, x_0v_1, \ldots, x_0v_n) \notin$

K', try again; else return X. Equation (3.92) ensures that we succeed with probability at least $1-\epsilon$ for each sample.

Algorithm 7: Volume estimation of well-rounded K with $\tilde{O}(n^4/\epsilon^2)$ classical queries [73].

```
**Input:** Membership oracle $O_{\mathrm{K}}$ of $\mathrm{K}$; $R$ such that $\mathrm{B}_2(0,1) \subseteq \mathrm{K} \subseteq \mathrm{B}_2(0,R)$;
        $R = O(\sqrt{n})$, i.e., $\mathrm{K}$ is well-rounded.
**Output:** $\epsilon$-multiplicative approximation of $\mathrm{Vol}(\mathrm{K})$.
1 Set $m = 2\lceil \sqrt{n}\ln(n/\epsilon)\rceil$, $k = \frac{512}{\epsilon^2}\sqrt{n}\ln(n/\epsilon)$, $\delta = \epsilon^2 n^{-10}$, and $a_i = 2n(1-\frac{1}{\sqrt{n}})^i$ for
   $i \in [m]$;
2 Take $k$ samples $X_0^{(1)}, \ldots, X_0^{(k)}$ from $\pi_0$;
3 **for** $i \in [m]$ **do**
4     Take $k$ samples from $\pi_i$ with error parameter $\delta$ and starting points $X_{i-1}^{(1)}, \ldots, X_{i-1}^{(k)}$,
        giving points $X_i^{(1)}, \ldots, X_i^{(k)}$;
5     Compute $V_i = \frac{1}{k}\sum_{j=1}^{k} e^{(a_i - a_{i+1})(X_i^{(j)})_0}$;
6 Return $n! v_n (2n)^{-(n+1)} V_1 \cdots V_m$ as the estimate of the volume of $\mathrm{K}'$, where
   $v_n := \pi^{\frac{n}{2}}/\Gamma(1+\frac{n}{2})$ is the volume of the $n$-dimensional unit ball;
```

3.4.2 Quantum algorithm for volume estimation

As introduced in Section 4.3.1, our quantum algorithm has four main improvements that contribute to the quantum speedup of Algorithm 7:

1. We replace the classical hit-and-run walk in Section 3.2.4 by a quantum hit-and-run walk, defined using the framework of Section 3.3. Classically, the hit-and-run walk mixes in $\tilde{O}(n^3)$ steps in a well-rounded convex body given a warm start (see Theorem 3.2.4). Quantumly, we can use the quantum hit-and-run walk operator to prepare its stationary state given a warm start state using only $\tilde{O}(n^{1.5})$ queries to the membership oracle for the well-rounded convex body.

2. We replace the simulated annealing framework in Algorithm 7 by the quantum MCMC frame-

work described in Section 3.2.2. Classically, we sample from π_i in the i^{th} iteration by running the classical hit-and-run walk starting from the samples taken in the $(i-1)^{\text{st}}$ iteration. Quantumly, we prepare the quantum sample $|\pi_i\rangle$ in the i^{th} iteration by applying $\pi/3$-amplitude amplification to a quantum sample produced in the $(i-1)^{\text{st}}$ iteration, where the unitaries in the $\pi/3$-amplitude amplification are implemented by phase estimation of the quantum hit-and-run walk operators as in (3.18).

3. We use the quantum Chebyshev inequality (see Section 3.2.3) to give a quadratic quantum speedup in ϵ^{-1} when taking the average $e^{(a_i-a_{i+1})(\bar{X}_i)_0}$ in Line 5 of Algorithm 7. However, we must be cautious because the resulting points $X_i^{(1)},\ldots,X_i^{(k)}$ in Line 4 follow the distribution π_i, which varies in different iterations of simulated annealing. Instead, our quantum algorithm must be *nondestructive*: it must retain a copy of $|\pi_i\rangle$ after estimating the average $e^{(a_i-a_{i+1})(\bar{X}_i)_0}$, so that we can map this state to $|\pi_{i+1}\rangle$ by $\pi/3$-amplitude amplification for the next iteration. This is achieved in Section 3.4.3.3.

4. In Section 3.4.4, we show how the densities can be transformed to be well-rounded by an affine transformation at each stage of the algorithm. This is to ensure that the hit-and-run walk mixes fast assuming the densities π_i to be sampled from are well-rounded (see Theorem 3.2.5). The high-level idea is to sample points from density π_i and compute an affine transformation S_{i+1} that rounds π_i and the next density π_{i+1} (see Lemma 3.4.11). To sample these points, we use $\pi/3$-amplitude amplification to map the states corresponding to the uniform distributions for one stage to those for the next. The affine transformation can be computed coherently using nondestructive mean estimation, with $\tilde{O}(n^{2.5})$ quantum queries in each iteration.

Algorithm 8 is our quantum volume estimation algorithm that satisfies our main theorem:

Theorem 3.1.1 (Main Theorem)**.** *Let* $\mathrm{K} \subset \mathbb{R}^n$ *be a convex set with* $\mathrm{B}_2^n(0,r) \subseteq \mathrm{K} \subseteq \mathrm{B}_2^n(0,R)$*. Assume* $0 < \epsilon < 1/2$*. Then there is a quantum algorithm that returns a value* $\widetilde{\mathrm{Vol}(\mathrm{K})}$ *satisfying*

$$\frac{1}{1+\epsilon}\mathrm{Vol}(\mathrm{K}) \leq \widetilde{\mathrm{Vol}(\mathrm{K})} \leq (1+\epsilon)\mathrm{Vol}(\mathrm{K}) \tag{3.4}$$

with probability at least $2/3$ *using* $\tilde{O}(n^3 + n^{2.5}/\epsilon)$ *quantum queries to the membership oracle* O_{K} *(defined in (4.8)) and* $\tilde{O}(n^5 + n^{4.5}/\epsilon)$ *additional arithmetic operations.*[13]

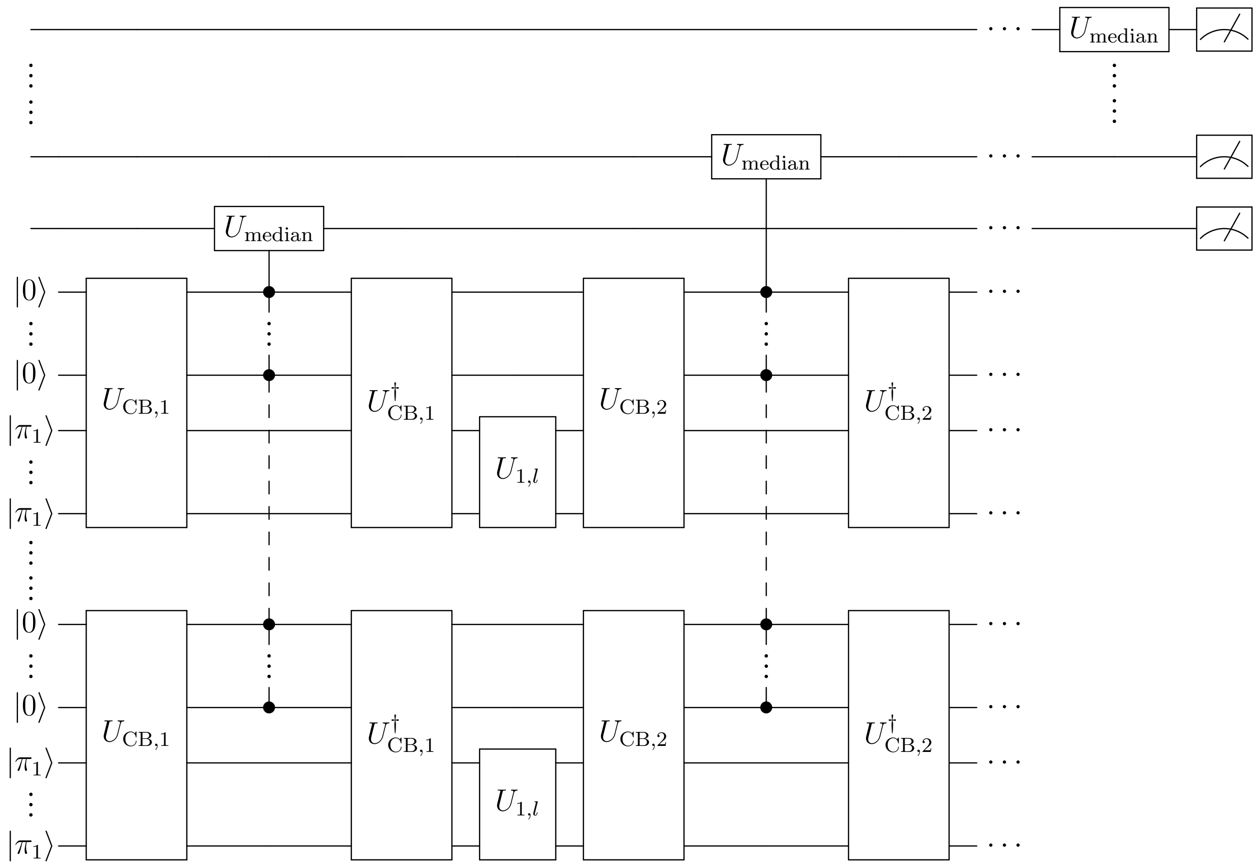


Figure 3.3: The quantum circuit for Algorithm 8 (assuming well-roundedness). Here $U_{\text{CB},i}$ is the circuit of the quantum Chebyshev inequality (Theorem 3.2.3) in the i^{th} iteration and $U_{i,l}$ is $\pi/3$-amplitude amplification from $|\pi_i\rangle$ to $|\pi_{i+1}\rangle$.

[13]Arithmetic operations (e.g., addition, subtraction, multiplication, and division) can be in principle implemented by a universal set of quantum gates using the Solovay-Kitaev Theorem [81] up to a small overhead. In our quantum algorithm, the number of arithmetic operations is dominated by n-dimensional matrix-vector products computed in superposition for rounding the convex body (see Section 3.4.4).

Algorithm 8: Volume estimation of convex K with $\tilde{O}(n^3+n^{2.5}/\epsilon)$ quantum queries.

```
Input: Membership oracle $O_{\mathrm{K}}$ for $\mathrm{K}$; $R = O(\sqrt{n})$ such that
        $\mathrm{B}_2(0,1) \subseteq \mathrm{K} \subseteq \mathrm{B}_2(0,R)$.
Output: $\epsilon$-multiplicative approximation of $\mathrm{Vol}(\mathrm{K})$.
1 Set $m = \Theta(\sqrt{n}\log(n/\epsilon))$ to be the number of iterations of simulated annealing and
    $a_i = 2n(1-\frac{1}{\sqrt{n}})^i$ for $i \in [m]$. Let $\pi_i$ be the probability distribution over $\mathrm{K}'$ with
    density proportional to $e^{-a_i x_0}$;
  Set error parameters $\delta, \epsilon' = \Theta(\epsilon/m^2)$, $\epsilon_1 = \epsilon/2m$; let $k = \tilde{\Theta}(\sqrt{n}/\epsilon)$ be the number of
    copies of stationary states for applying the quantum Chebyshev inequality; let
    $l = \tilde{\Theta}(n)$ be the number of copies of stationary states needed to obtain the affine
    transformation $S_i$;
  Prepare $k+l$ (approximate) copies of $|\pi_0\rangle$, denoted $|\tilde{\pi}_0^{(1)}\rangle, \ldots, |\tilde{\pi}_0^{(k+l)}\rangle$ (Lemma 3.4.4);
2 for $i \in [m]$ do
3     Use the quantum Chebyshev inequality on the $k$ copies of the state $|\tilde{\pi}_{i-1}\rangle$ with
        parameters $\epsilon_1, \delta$ to estimate the expectation $\mathbb{E}_{\pi_i}[V_i]$ (in Eq. (3.94)) as $\tilde{V}_i$
        (Lemma 3.4.9 and Figure 3.4). The post-measurement states are denoted
        $|\hat{\pi}_{i-1}^{(1)}\rangle, \ldots, |\hat{\pi}_{i-1}^{(k)}\rangle$;
4     Use the $l$ copies of the state $|\pi_{i-1}\rangle$ to nondestructively obtain the affine
        transformation $S_i$ that rounds $\pi_{i-1}$ and $\pi_i$ (Section 3.4.4). The post-measurement
        states are denoted $|\hat{\pi}_{i-1}^{(k+1)}\rangle, \ldots, |\hat{\pi}_{i-1}^{(k+l)}\rangle$;
5     Apply $\pi/3$-amplitude amplification with error $\epsilon'$ (Section 3.2.2 and Lemma 3.4.8)
        and affine transformation $S_i$ to map $|S_i\hat{\pi}_{i-1}^{(1)}\rangle, \ldots, |S_i\hat{\pi}_{i-1}^{(k+l)}\rangle$ to
        $|S_i\tilde{\pi}_i^{(1)}\rangle, \ldots, |S_i\tilde{\pi}_i^{(k+l)}\rangle$, using the quantum hit-and-run walk;
6     Invert $S_i$ to get $k+l$ (approximate) copies of the stationary distribution $|\pi_i\rangle$ for
        use in the next iteration;
7 Compute an estimate $\widetilde{\mathrm{Vol}(\mathrm{K}')} = n! v_n (2n)^{-(n+1)} \tilde{V}_1 \cdots \tilde{V}_m$ of the volume of $\mathrm{K}'$, where
    $v_n$ is the volume of the $n$-dimensional unit ball;
8 Use $\widetilde{\mathrm{Vol}(\mathrm{K}')}$ to estimate the volume of $\mathrm{K}$ as $\widetilde{\mathrm{Vol}(\mathrm{K})}$ (Section 3.4.3.1).
```

More generally, our framework paves the way of combining several different ingredients in quantum computing, and it could be used to provide quantum speedup for classical simulated annealing algorithms based on Chebyshev cooling, i.e., those with the property that the random variable in each iteration has bounded ratio between its variance and the square of its expectation. This might be of independent interest.

The proof of Theorem 4.1.1 is organized as follows. We first assume that in each iteration,

S_{i+1} puts π_{i+1} in isotropic position, i.e., the densities are promised to be well-rounded. The rest of this subsection presents an overview of the proof of Theorem 4.1.1 (including a quantum circuit in Figure 3.3), and proofs details are given in Section 3.4.3. In Section 3.4.4, we show how the well-roundedness be maintained at an additional cost of $\tilde{O}(n^{2.5})$ quantum queries in each iteration.

Following the discussion in Section 4.3.1, our proof can be described at three levels:

High level (the simulated annealing framework) In Section 3.4.3.1, we show how to estimate $\mathrm{Vol}(\mathrm{K})$ given an estimate of the volume of the pencil construction, $\mathrm{Vol}(\mathrm{K}')$:

Lemma 3.4.1. *If we have access to $\widetilde{\mathrm{Vol}(\mathrm{K}')}$ such that*

$$\frac{1}{1+\epsilon/2}\mathrm{Vol}(\mathrm{K}') \leq \widetilde{\mathrm{Vol}(\mathrm{K}')} \leq (1+\epsilon/2)\mathrm{Vol}(\mathrm{K}') \tag{3.96}$$

with probability at least 0.7, then we can return a value $\widetilde{\mathrm{Vol}(\mathrm{K})}$ such that

$$\frac{1}{1+\epsilon}\mathrm{Vol}(\mathrm{K}) \leq \widetilde{\mathrm{Vol}(\mathrm{K})} \leq (1+\epsilon)\mathrm{Vol}(\mathrm{K}) \tag{3.97}$$

holds with probability at least $2/3$, using $\tilde{O}(n^{2.5}/\epsilon)$ quantum queries to the membership oracle O_{K}.

In Section 3.4.3.2, we prove that the inner product between stationary states of consecutive simulated annealing steps is at least a constant:

Lemma 3.4.2. *Let $|\pi_i\rangle$ be the stationary distribution state of the quantum walk W_i for $i \in [m]$ defined in (3.86). For $n \geq 2$, we have $\langle \pi_i | \pi_{i+1} \rangle > 1/3$ for $i \in [m-1]$.*

This allows $\pi/3$-amplitude amplification to transform the stationary state of one Markov chain to that of the next. The total number of iterations of $\pi/3$-amplitude amplification is thus $\tilde{O}(\sqrt{n})$, which equals to the number of iterations of the classical volume estimation algorithm by [73].

Middle level (each telescoping ratio) In Section 3.4.3.3, we describe how we apply the quantum Chebyshev inequality (Theorem 3.2.3) to the Chebyshev cooling schedule.

Lemma 3.4.3. *Given $\tilde{O}(\log(1/\delta)/\epsilon)$ copies of the quantum states $|\pi_{i-1}\rangle$, there exists a quantum algorithm that outputs an estimate of $\mathbb{E}_{\pi_i}[V_i]$ (in Eq. (3.94)) with relative error less than ϵ with probability at least $1-O(\delta)$ using $\tilde{O}(\mathcal{C}\log(1/\delta)/\epsilon)$ oracle calls, where $\mathcal{C}$ oracle calls are required to implement a sampler for $|\pi_i\rangle$. Moreover, this quantum algorithm is nondestructive, i.e., the initial copies of quantum states $|\pi_{i-1}\rangle$ are restored after the computation with probability at least $1-O(\delta)$.*

Because the relative error in each iteration for estimating the volume via Chebyshev cooling is $\Theta(\epsilon/m) = \tilde{\Theta}(\epsilon/\sqrt{n})$, Lemma 3.4.3 implies that $O(\log(1/\delta)/(\epsilon/\sqrt{n})) = \tilde{O}(\sqrt{n}/\epsilon)$ copies of the stationary state suffice for the simulated annealing framework.[14]

Low level (the quantum hit-and-run walk) In Section 3.4.3.4, we give a careful analysis of the errors coming from the quantum Chebyshev inequality as well as the $\pi/3$-amplitude amplification:

[14]Notice that in the quantum Chebyshev inequality by Hamoudi and Magniez, they did not distinguish the number of copies of the initial state from the number of quantum samples [88, Theorem 5.3]. In fact, their proof uses only $O(\log(1/\delta))$ copies of the initial state $|\pi_{i-1}\rangle$ in Lemma 3.4.3, which reduces the total number of copies of the stationary states in the simulated annealing framework to $O(\log(1/\delta))$. Nevertheless, this does not change the total query and time complexities of our quantum algorithms because the total number of calls to the quantum sampler remains the same.

Lemma 3.4.4. *For $\epsilon_1 < 1$, given $\tilde{O}(\log(1/\delta)/\epsilon_1)$ copies of a state $|\tilde{\pi}_{i-1}\rangle$ such that $\||\tilde{\pi}_{i-1}\rangle - |\pi_{i-1}\rangle\| \leq \epsilon_1$, there exists a quantum procedure (using $\pi/3$-amplitude amplification and the quantum Chebyshev inequality) that outputs a $\tilde{V}_i$ such that $|\tilde{V}_i - \mathbb{E}_{\pi_i}[V_i]| \leq \epsilon_1 \mathbb{E}_{\pi_i}[V_i]$ (where $\mathbb{E}_{\pi_i}[V_i]$ is defined in Eq. (3.94)) with success probability $1 - \delta^4$ using $\tilde{O}(n^{3/2}\log(1/\delta)/\epsilon_1 + n^{3/2}\log(1/\epsilon'))$ calls to the membership oracle and returns $\tilde{O}(\log(1/\delta)/\epsilon_1)$ copies of final states $|\tilde{\pi}_i\rangle$ such that $\||\tilde{\pi}_i\rangle - |\pi_i\rangle\| = O(\epsilon_1 + \delta + \epsilon')$.*

Having the four lemmas above from all the three levels, we establish Theorem 4.1.1 as follows.

Proof of Theorem 4.1.1. We prove the correctness and analyze the cost separately.

Correctness By Lemma 3.4.1, it suffices to compute the volume of the pencil construction K' to relative error $\epsilon/2$ with probability at least 0.7 in order to compute the volume of the well-rounded convex body K. This is computed as a telescoping sum of $m = O(\sqrt{n}\log n/\epsilon)$ products of the form $Z(a_{i+1})/Z(a_i)$. The random variable V_i is an unbiased estimator for $Z(a_{i+1})/Z(a_i)$, i.e., $\mathbb{E}_{\pi_i}[V_i] = Z(a_{i+1})/Z(a_i)$. Consider applying Lemma 3.4.4 m times with sufficiently small $\delta, \epsilon' \leq \epsilon/12m^2$ and $\epsilon_1 = \epsilon/3m$. This will promise that $\epsilon_1 + \delta + \epsilon' \leq \epsilon/2m$. At each iteration i we have a state $|\tilde{\pi}_{i-1}\rangle$ such that $\||\tilde{\pi}_{i-1}\rangle - |\pi_{i-1}\rangle\| \leq O(\epsilon/4m)$. Thus each telescoping sum can be computed with a relative error of $\epsilon/2m$, resulting in a relative error of less than $\epsilon/2$ for the final volume. The probability of success for each iteration is at least $1 - \delta^4 = 1 - \Theta(\epsilon^4/4m^8)$. Thus the probability of success for the whole algorithm is at least $1 - \Theta(\epsilon^4/4m^7) = 1 - \tilde{O}(\epsilon^{11}/n^{3.5})$, which is greater than 0.7 for a large enough n.

Cost Ignoring the cost of obtaining the affine transformation to round the logconcave densities to be sampled (assuming that all the relevant densities are well rounded), we have from Lemma 3.4.4, the number of calls to the membership oracle in each iteration of Algorithm 8 is $\tilde{O}(n^{3/2}\log(1/\delta)/\epsilon_1 + n^{3/2}\log(1/\epsilon')) = \tilde{O}(n^2/\epsilon)$. The total number of oracle calls is thus $\tilde{O}(n^{2.5}/\epsilon)$. The argument for correctness above applies for well-rounded logconcave densities. This is ensured by maintaining $\tilde{\Theta}(n)$ states that are used to round the densities in each iteration (Algorithm 9). By Proposition 3.4.4, this procedure uses $\tilde{O}(n^{2.5})$ calls to the membership oracle in each iteration, resulting in a final query complexity of $\tilde{O}(n^3 + n^{2.5}/\epsilon)$. Since the affine transformation is an n-dimensional matrix-vector product, the number of additional arithmetic operations is hence $O(n^2)\cdot\tilde{O}(n^3+n^{2.5}/\epsilon) = \tilde{O}(n^5 + n^{4.5}/\epsilon)$. □

3.4.3 Proofs of lemmas in Section 3.4.2

We now prove the lemmas in Section 3.4.2 that establish Theorem 4.1.1.

3.4.3.1 From the pencil construction to the original convex body

Here we prove

Lemma 3.4.1. *If we have access to* $\widetilde{\mathrm{Vol}(\mathrm{K}')}$ *such that*

$$\frac{1}{1+\epsilon/2}\mathrm{Vol}(\mathrm{K}') \le \widetilde{\mathrm{Vol}(\mathrm{K}')} \le (1+\epsilon/2)\mathrm{Vol}(\mathrm{K}') \tag{3.96}$$

with probability at least 0.7, then we can return a value $\widetilde{\mathrm{Vol}(\mathrm{K})}$ *such that*

$$\frac{1}{1+\epsilon}\mathrm{Vol}(\mathrm{K}) \le \widetilde{\mathrm{Vol}(\mathrm{K})} \le (1+\epsilon)\mathrm{Vol}(\mathrm{K}) \tag{3.97}$$

holds with probability at least $2/3$*, using* $\tilde{O}(n^{2.5}/\epsilon)$ *quantum queries to the membership oracle* O_{K}.

Proof. We follow the same notation in Section 3.4.1, i.e., without loss of generality we assume that $r = 1$ and denote $D = R/r = R$. Since R and r are both given, D is a known value. The pencil construction is

$$\mathrm{K}' := ([0, 2D] \times \mathrm{K}) \cap \Big\{x \in \mathbb{R}^{n+1} : x_0 \geq 0, \sum_{i=1}^{n} x_i^2 \leq x_0^2\Big\}. \tag{3.98}$$

By the definition of D, for any $(x_1, \ldots, x_n) \in \mathrm{K}$ we have $\sum_{i=1}^{n} x_i^2 \leq D^2$, so $[D, 2D] \times \mathrm{K} \subseteq \mathrm{K}'$. This implies that $D\mathrm{Vol}(\mathrm{K}) \leq \mathrm{Vol}(\mathrm{K}') \leq 2D\mathrm{Vol}(\mathrm{K})$. In other words, letting $\xi_{\mathrm{K}} := \frac{\mathrm{Vol}(\mathrm{K}')}{2D\mathrm{Vol}(\mathrm{K})}$, we have $0.5 \leq \xi_{\mathrm{K}} \leq 1$.

Classically, we consider a Monte Carlo approach to approximating $\mathrm{Vol}(\mathrm{K})$: we take k (approximately) uniform samples $x_1, \ldots, x_k$ from $[0, 2D] \times \mathrm{K}$, and if k' of them are in K', we return $\frac{k'}{k} \cdot \widetilde{\mathrm{Vol}(\mathrm{K}')}$. For each $i \in [k]$, $\delta[x_i \in \mathrm{K}']$ is a boolean random variable with expectation $\xi_{\mathrm{K}} = \Theta(1)$. Any boolean random variable has variance $O(1)$. Therefore, by Chebyshev's inequality, taking $k = O(1/\epsilon^2)$ suffices to ensure that

$$\Pr\left[\left|\frac{k'}{k} - \xi_{\mathrm{K}}\right| \leq \frac{\epsilon \xi_{\mathrm{K}}}{2}\right] \geq 0.99. \tag{3.99}$$

Quantumly, we adopt the same Monte Carlo approach but we implement two steps using quantum techniques:

- We take an approximately uniform sample from $K' = [0, 2D] \times \mathrm{K}$ via the quantum hit-and-run walk. To obtain a quantum stationary state, we use a similar idea as in [65] to construct a sequence of $m = \lceil n \log_2(2D) \rceil$ convex bodies. Let $\hat{\mathrm{K}}_0 := \mathrm{B}_2^{n+1}(0,1)$ and $\hat{\mathrm{K}}_i := 2^{i/n}\mathrm{B}_2^{n+1}(0,1) \cap \mathrm{K}'$ for $i \in [m]$. As the length of the pencil is $2D$, $\hat{\mathrm{K}}_m = \mathrm{K}'$. The state $|\pi_0\rangle$ corresponding to $\hat{\mathrm{K}}_0$ is easy to prepare. It is straightforward to verify that $\langle \pi_i | \pi_{i+1} \rangle \geq c$ for some constant c, as $\mathrm{Vol}(\hat{\mathrm{K}}_{i+1}) \leq 2\mathrm{Vol}(\hat{\mathrm{K}}_i)$. To utilize the quantum speedup for MCMC framework (Theorem 3.2.1), it remains to lower bound the phase gap of the quantum walk operator for $\hat{\mathrm{K}}_i$. It can be shown that the mixing property of the hit-and-run walk in Theorem 3.2.5 implies that the phase gap of the quantum walk operator is $\tilde{\Omega}(n^{-1.5})$; see the proof of Lemma 3.4.8. Thus, by Theorem 3.2.1, $|\pi_m\rangle$ can be prepared using $\tilde{O}(n) \cdot \tilde{O}(n^{1.5}) = \tilde{O}(n^{2.5})$ quantum queries to O_{K}.

- We estimate ξ_{K} with multiplicative error $\epsilon/2$ using the quantum Chebyshev inequality (see Theorem 3.2.3) instead of its classical counterpart. This means that $O(1/\epsilon)$ executions of quantum sampling in the first step suffice.

Overall, $\tilde{O}(n^{2.5}/\epsilon)$ quantum queries to O_{K} suffice to ensure that we obtain an estimate of ξ_{K} within multiplicative error $\epsilon/2$ with success probability at least 0.99. Since (3.96) ensures that $\widetilde{\mathrm{Vol}(\mathrm{K}')}$ estimates $\mathrm{Vol}(\mathrm{K}')$ up to multiplicative error $\epsilon/2$ with probability at least 0.7, $\frac{\widetilde{\mathrm{Vol}(\mathrm{K}')}}{2D\xi_{\mathrm{K}}}$ estimates $\mathrm{Vol}(\mathrm{K})$ up to multiplicative error $\epsilon/2 + \epsilon/2 = \epsilon$ with success probability $0.99 \cdot 0.7 > 2/3$. $\square$

3.4.3.2 Inner product between stationary states of consecutive steps

We now show that the inner product between stationary states of consecutive steps is at least a constant. More precisely, we have the following:

Lemma 3.4.2. *Let $|\pi_i\rangle$ be the stationary distribution state of the quantum walk W_i for $i \in [m]$ defined in (3.86). For $n \geq 2$, we have $\langle \pi_i | \pi_{i+1} \rangle > 1/3$ for $i \in [m-1]$.*

Proof. Recall that the stationary distribution π_i of step i has density proportional to $e^{-a_i x_0}$ as discussed in Section 3.4.1. The corresponding stationary distribution state is $|\pi_i\rangle = \int_{\mathrm{K}'} \mathrm{d}x \sqrt{\frac{e^{-a_i x_0}}{Z(a_i)}} |x\rangle$. Lovász and Vempala [73, Lemma 3.2] proved that $a^{n+1} Z(a)$ is log-concave (noting that the dimension of K' is $n+1$). This implies that

$$\sqrt{a_i^{n+1} Z(a_i)} \sqrt{a_{i+1}^{n+1} Z(a_{i+1})} \leq \left(\frac{a_i + a_{i+1}}{2} \right)^{n+1} Z\left(\frac{a_i + a_{i+1}}{2} \right). \tag{3.100}$$

Now, we have

$$\begin{aligned}
\langle \pi_i | \pi_{i+1} \rangle &= \int_{\mathrm{K}'} \mathrm{d}x \frac{\exp(-\frac{a_i + a_{i+1}}{2} x_0)}{\sqrt{Z(a_i)}\sqrt{Z(a_{i+1})}} && (3.101)\\
&\geq \left(\frac{2\sqrt{a_i}\sqrt{a_{i+1}}}{a_i + a_{i+1}} \right)^{n+1} \frac{\int_{\mathrm{K}'} \mathrm{d}x \exp(-\frac{a_i + a_{i+1}}{2} x_0)}{Z\left(\frac{a_i + a_{i+1}}{2}\right)} && (3.102)\\
&= \left(\frac{2\sqrt{a_i}\sqrt{a_{i+1}}}{a_i + a_{i+1}} \right)^{n+1} && (3.103)\\
&= \left(\frac{2\sqrt{a_i}\sqrt{a_i(1 - \frac{1}{\sqrt{n}})}}{a_i + a_i(1 - \frac{1}{\sqrt{n}})} \right)^{n+1} = \left(\frac{2\sqrt{1 - \frac{1}{\sqrt{n}}}}{2 - \frac{1}{\sqrt{n}}} \right)^{n+1}, && (3.104)
\end{aligned}$$

where the inequality follows from (3.100). When $n = 2$ or $n = 3$, the above inequality holds. When $n \geq 4$, to lower bound the above quantity, we use the fact that $\sqrt{1 - 1/\sqrt{n}} \geq 1 - \frac{1}{2\sqrt{n}} - \frac{1}{2n}$. Hence, for $n \geq 2$ we have

$$\langle \pi_i | \pi_{i+1} \rangle \geq \left(\frac{2 - \frac{1}{\sqrt{n}} - \frac{1}{n}}{2 - \frac{1}{\sqrt{n}}} \right)^{n+1} = \left(1 - \frac{\frac{1}{n}}{2 - \frac{1}{\sqrt{n}}} \right)^{n+1} \geq \left(1 - \frac{1}{(2 - \frac{1}{\sqrt{2}})n} \right)^{n+1} > \frac{1}{3}$$

as claimed. □

3.4.3.3 Chebyshev cooling and nondestructive mean estimation

Now we briefly review the classical framework for Chebyshev cooling and discuss how to adapt it to quantum algorithms. Suppose we want to compute the expectation of a product

$$V = \prod_{i=1}^{m} V_i \tag{3.105}$$

of independent random variables. The following theorem of Dyer and Frieze [68] upper bounds the number of samples from the V_i that suffices to estimate $\mathbb{E}[V]$ with bounded relative error.

Proposition 3.4.1 ([68, Section 4.1])**.** *Let $V_1, \ldots, V_m$ be independent random variables such that $\frac{\mathbb{E}[V_i^2]}{\mathbb{E}[V_i]^2} \leq B$ for all $i \in [m]$. Let $X_j^{(1)}, \ldots, X_j^{(k)}$ be k samples of V_j for $j \in [m]$, and define $\overline{X}_j = \frac{1}{k}\sum_{\ell=1}^{k} X_j^{(\ell)}$. Let $V = \prod_{j=1}^{m} V_j$ and $X = \prod_{j=1}^{m} \overline{X}_j$. Then, taking $k = 16Bm/\epsilon^2$ ensures that*

$$\Pr\left[(1-\epsilon)\mathbb{E}[V] \leq X \leq (1+\epsilon)\mathbb{E}[V]\right] \geq \frac{3}{4}. \tag{3.106}$$

With standard techniques, the probability can be boosted to $1-\delta$ with a $\log(1/\delta)$ overhead.

In applications such as volume estimation [72] and estimating partition functions [112], the samples are produced by a random walk. Let the mixing time for each random walk be at most T. Then the total complexity for estimating $\mathbb{E}[V]$ with success probability $1-\delta$ is $\tilde{O}(TBm\log(1/\delta)/\epsilon^2)$. Replacing the random walk with a quantum walk can potentially improve the mixing time; see Section 3.1.3.2 for relevant literature. In particular, Montanaro [11] proposed a quantum algorithm for the simulated annealing framework with complexity $\tilde{O}(TBm\log(1/\delta)/\epsilon)$, which has a quadratic improvement in precision. Note that the dependence on T was not improved, as multiple copies of quantum states were prepared for the mean estimation (which uses measurements). In this paper, we use the quantum Chebyshev inequality (see Theorem 3.2.3) to estimate the expectation of V_i in a nondestructive manner which, together with Theorem 3.2.1, achieves complexity $\tilde{O}(\sqrt{T}Bm\log(1/\delta)/\epsilon)$.

Recall that the random variables V_i (determined by the cooling schedule) satisfy Eq. (3.95). The following lemma uses this property of the simulated annealing procedure to show that the quantum Chebyshev inequality can be used to estimate the mean of V_i on the distribution π_i, which gives an estimate of the ratio $\frac{Z(a_{i+1})}{Z(a_i)}$ in the volume estimation algorithm. We first show that our random variables can be made to satisfy the conditions of Theorem 3.2.3, and then we outline how the corresponding circuit can be implemented. A detailed error analysis is deferred to Section 3.4.3.4. To make the mean estimation nondestructive, we use the following theorem of Harrow and Wei.

Theorem 3.4.1 ([89, Theorem 6]). *Given state $|\psi\rangle$ and reflections $R_\psi = 2|\psi\rangle\langle\psi| - I$ and $R = 2P - I$, and any $\eta > 0$, there exists a quantum algorithm that outputs $\tilde{a}$, an approximation to $a = \langle\psi|P|\psi\rangle$, so that*

$$|\tilde{a} - a| \leq 2\pi\frac{a(1-a)}{M} + \frac{\pi^2}{M^2} \tag{3.107}$$

with probability at least $1-\eta$ and $O(\log(1/\eta)M)$ uses of R_ψ and R. Morover the algorithm restores the state ψ with probability at least $1-\eta$.

Lemma 3.4.3. *Given $\tilde{O}(\log(1/\delta)/\epsilon)$ copies of the quantum states $|\pi_{i-1}\rangle$, there exists a quantum* *algorithm that outputs an estimate of* $\mathbb{E}_{\pi_i}[V_i]$ *(in Eq. (3.94)) with relative error less than* ϵ *with probability at least* $1-O(\delta)$ *using* $\tilde{O}(\mathcal{C}\log(1/\delta)/\epsilon)$ *oracle calls, where* $\mathcal{C}$ *oracle calls are required to implement a sampler for* $|\pi_i\rangle$*. Moreover, this quantum algorithm is nondestructive, i.e., the initial copies of quantum states* $|\pi_{i-1}\rangle$ *are restored after the computation with probability at least* $1-O(\delta)$.

Proof. We apply the quantum Chebyshev inequality (Theorem 3.2.3). For the random variables V_i, we let μ_i denote their mean and σ_i^2 their variance. From (3.95), $\sqrt{\sigma_i^2+\mu_i^2}/\mu_i \leq \sqrt{8} < 3$. For a small constant c, we use $\log(1/\delta)/c^2$ copies of $|\pi_{i-1}\rangle$ to create copies of $|\pi_i\rangle$ using $\pi/3$-amplitude amplification. We now use a quantum circuit that given $|x\rangle|0\rangle$ computes $|x\rangle|e^{a_i x_0 - a_{i-1}x_0}\rangle$, and then apply a circuit U_{median} that computes the median of all the ancilla registers:

$$U_{\text{median}}|0\rangle|a_1\rangle\cdots|a_s\rangle = |\text{median}\{a_1,\ldots,a_s\}\rangle|a_1\rangle\cdots|a_s\rangle. \tag{3.108}$$

By the classical Chebyshev inequality, we measure $\hat{\mu}_i$ such that $|\hat{\mu}_i-\mu_i| \leq c\mu_i$ with probability at least $1-\delta$. Thus the probability that $\hat{\mu}_i/(1-c) < \mu$ is less than δ. Taking $H = \hat{\mu}_i/(1-c)$, our variables satisfy the conditions of the quantum Chebyshev inequality. In order to output an estimate of the mean with relative error at most ϵ, the quantum Chebyshev inequality now requires $\tilde{O}(\log(1/\delta)/\epsilon)$ calls to a sampler for the state $|\pi_i\rangle$, which we construct using $\pi/3$-amplitude amplification on copies of $|\pi_{i-1}\rangle$. By the union bound, the probability of failure of the whole procedure is $O(\delta)$.

To be more specific, we replace $U|0\rangle$ in BasicEst (Algorithm 6) by $U_{i-1,l}|\pi_{i-1}\rangle$ (where $U_{i-1,l}$ is the circuit transforming the l^{th} copy of $|\pi_{i-1}\rangle$ to $|\pi_i\rangle$), and replace $\mathcal{Q}$ by $-U_{i-1,l}(\Pi_{i-1} - \Pi_{i-1}^{\perp})U_{i-1,l}^{\dagger}(\Pi_i - \Pi_i^{\perp})$ (where $\Pi_i = |\pi_i\rangle\langle\pi_i|$ and $\Pi_i^{\perp} = I - \Pi_i$ for all $i \in [m]$). The quantum circuit for nondestructive BasicEst is shown in Figure 3.4. Here, we run $\Theta(\log(1/\delta))$ executions of amplitude estimation (Figure 3.2) in parallel. Note that by (3.21), each amplitude estimation returns a state $\frac{e^{i\theta_p}}{\sqrt{2}}|\tilde{\theta}_p\rangle - \frac{e^{-i\theta_p}}{\sqrt{2}}|-\tilde{\theta}_p\rangle$. We use an ancilla register and apply the unitary

$$U_{\sin^2}|\theta\rangle|0\rangle := |\theta\rangle|\sin^2\theta\rangle; \tag{3.109}$$

because $\sin^2(\tilde{\theta}_p) = \sin^2(-\tilde{\theta}_p) = \tilde{p}$, the ancilla register becomes $|\tilde{p}\rangle$, where $\tilde{p}$ estimates p well as claimed in Theorem 3.2.2. We then take the median of such $\Theta(\log(1/\delta))$ executions using (3.108), and finally run the inverse of $U_{\sin^2}$ gates and amplitude estimations. The correctness follows from the proof of Theorem 3.2.3 in [88].

To assure non-destructiveness, we replace every application of Quantum Amplitude Estimation with the Nondestructive Mean Estimation as in Theorem 3.4.1. The resulting guarantees on the error are the same as with the original amplitude estimation algorithm. To ensure an overall success probability of $1-O(\delta)$, it suffices to perform each instance of Nondestructive Amplitude Estimation with success probability $1-\tilde{O}(\delta)$. Note that since we estimate an unweighted mean, $2P - I$ with $P = H|0\rangle\langle 0|H$ can be implemented as HR_0H where R_0 is a reflection around the $|0\rangle$ state. Finally, we show in Corollary 3.4.1 that R_{π_i} (the reflection around $|\pi_i\rangle$) can be implemented using the same number of oracle calls and gates as that required to sample π_i (up to polylogarithmic factors). $\square$

A detailed error analysis is given in the next subsection (see Lemma 3.4.9).

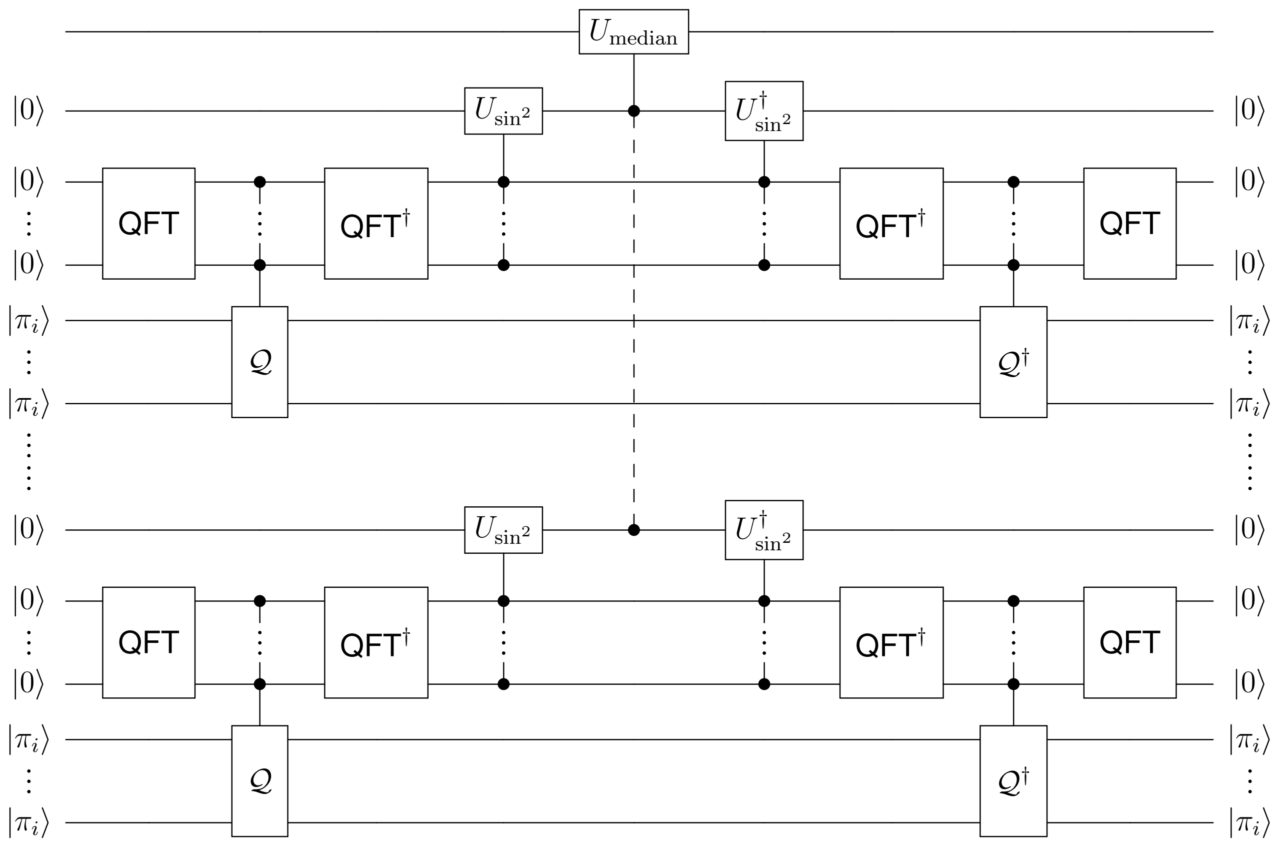


Figure 3.4: The quantum circuit for nondestructive BasicEst.

3.4.3.4 Error analysis

In this section, we analyze the error incurred by both the quantum Chebyshev inequality (Line 3) and $\pi/3$-amplitude amplification (Line 5) in Algorithm 8.

Lemma 3.4.4. *For $\epsilon_1 < 1$, given $\tilde{O}(\log(1/\delta)/\epsilon_1)$ copies of a state $|\tilde{\pi}_{i-1}\rangle$ such that $\||\tilde{\pi}_{i-1}\rangle - |\pi_{i-1}\rangle\| \leq \epsilon_1$, there exists a quantum procedure (using $\pi/3$-amplitude amplification and the quantum Chebyshev inequality) that outputs a $\tilde{V}_i$ such that $|\tilde{V}_i - \mathbb{E}_{\pi_i}[V_i]| \leq \epsilon_1 \mathbb{E}_{\pi_i}[V_i]$ (where $\mathbb{E}_{\pi_i}[V_i]$ is defined in Eq. (3.94)) with success probability $1 - \delta^4$ using $\tilde{O}(n^{3/2}\log(1/\delta)/\epsilon_1 + n^{3/2}\log(1/\epsilon'))$ calls to the membership oracle and returns $\tilde{O}(\log(1/\delta)/\epsilon_1)$ copies of final states $|\tilde{\pi}_i\rangle$ such that $\||\tilde{\pi}_i\rangle - |\pi_i\rangle\| = O(\epsilon_1 + \delta + \epsilon')$.*

We first show that $\pi/3$-amplitude amplification can be used to rotate $|\pi_i\rangle$ into $|\pi_{i-1}\rangle$ with error ϵ' using $\tilde{O}(\log(1/\epsilon))$ oracle calls. This procedure is used as a subroutine in a mean estimation circuit that estimates the mean of the random variable V_i using multiple approximate copies of $|\pi_{i-1}\rangle$. We ensure that the measurement probabilities are highly peaked so that the state is not disturbed very much. Finally $\pi/3$-amplitude estimation is used again to rotate the approximate copies of the state $|\pi_{i-1}\rangle$ to approximate copies of the state $|\pi_i\rangle$.

Large effective spectral gap Consider an ergodic, reversible Markov chain (Ω, p) with transition matrix P and a unique stationary distribution with density π. Let $a(x)$ be a probability measure over Ω such that the Markov chain mixes to its stationary distribution with a corresponding probability density $\pi(x)$ within a total variation distance of ϵ within t steps. Further let $a(x)$ be a warm start for $\pi(x)$. From the definition of the transition matrix $P(x,y) = \langle x|P|y\rangle = p_{y\to x}$.

The discriminant matrix D defined in (3.43) is related to the transition matrix as $P = D_\pi D D_\pi^{-1}$, as shown in (3.85). For a hit-and-run walk, the transition matrix P represents a convolution with an L_2 normalized function (corresponding to the square root of the density $p_{x\to y}$). Bounded subsets of $L_2(\Omega)$ are therefore mapped by P to other bounded subsets, and hence P is compact. Since D is connected to P by a similarity relation, D is a compact Hermitian operator over $L_2(\Omega)$ and thus has a countable set of real eigenvalues λ_i and corresponding orthonormal eigenvectors (eigenfunctions) $v_i \in L_2(\Omega)$. Orthonormality implies that $\int_\Omega v_i(x)v_j(x)\,\mathrm{d}x = \delta_{i,j}$. Notice that

$$PD_\pi v_i = D_\pi D(v_i) = \lambda_j D_\pi v_i; \tag{3.110}$$

thus $f_i = D_\pi v_i$ is an eigenvector of P' with eigenvalue λ_i. The eigenvectors f_i may not be orthogonal under the standard inner product on $L_2(\Omega)$. However, we can define an inner product

$$\langle f, g\rangle_\pi := \langle D_\pi^{-1} f, D_\pi^{-1} g\rangle = \int_\Omega \frac{f(x)g(x)}{\pi(x)}\,\mathrm{d}x \tag{3.111}$$

over the space $L_2(\Omega)$. It is easy to see that $\langle f_i, f_j\rangle_\pi = \langle v_i, v_j\rangle = \delta_{i,j}$. A corresponding norm can be defined as $\|f\|_\pi = \langle f, f\rangle_\pi$.

It can be verified that $\sqrt{\pi}(x)$ is an eigenfunction of D with eigenvalue 1. Thus the stationary state $\pi(x)$ is an eigenfunction of the transition operator P with eigenvalue 1. Since P is stochastic, this is the leading eigenvalue. The eigenvalues of P are thus $1, \lambda_1, \lambda_2, \ldots$ with corresponding eigenfunctions $\pi(x), f_2(x), f_3(x), \ldots$. From the orthonormality of the f under $\langle\cdot,\cdot\rangle_\pi$, for any function g in $L_2(\Omega)$ we have

$$g = \sum_{i=1}^{\infty}\langle g, f_i\rangle_\pi f_i = \langle g, \pi\rangle_\pi + \sum_{i=2}^{\infty}\langle g, f_i\rangle_\pi f_i \tag{3.112}$$

$$= \left(\int_\Omega \frac{g(x)\pi(x)}{\pi(x)}\,\mathrm{d}x\right)\pi + \sum_{i=2}^{\infty}\langle g, f_i\rangle_\pi f_i \tag{3.113}$$

$$= \left(\int_\Omega g(x)\,\mathrm{d}x\right)\pi + \sum_{i=2}^{\infty}\langle g, f_i\rangle_\pi f_i. \tag{3.114}$$

Since a is a probability density, $a = \pi + \sum_{i=2}^{\infty}\langle a, f_i\rangle_\pi f_i$. After t steps of the Markov chain M on a we obtain the state $P^t a = \pi + \sum_{i=2}^{\infty}\lambda_i^t\langle a, f_i\rangle_\pi f_i$. Since the walk mixes to total variation distance ϵ we have $\|P^t a - \pi\|_1 \le \epsilon$, and further since a is a warm start $\|P^t a - \pi\|_\pi$. Consequently, $\|\sum_{i=2}^{\infty}\lambda_i^t\langle a, f_i\rangle_\pi f_i\|_\pi \le \epsilon$ and from the orthonormality of f, $\langle a, f_i\rangle_\pi \lambda_i^t \le \epsilon$. If $1 > \lambda_i \ge 1 - \frac{1}{\Omega(t)}$ then $\lambda_i^t = \Omega(1)$ and $\langle a, f_i\rangle_\pi = \emptyset(\epsilon)$.

The above analysis indicates that if a probability density a (that is a warm start) mixes in t steps under a Markov chain (Ω, p), then it has small overlap with each of the "bad" eigenfunctions (with spectral gap less than $\frac{1}{\Omega(t)}$). Thus P effectively has a large spectral gap when it acts on a.

Corresponding to a, consider the quantum states

$$|a\rangle := \int_\Omega \sqrt{a(x)}|x\rangle \,\mathrm{d}x, \qquad |\phi_a\rangle := \int_\Omega \int_\Omega \sqrt{a_x p_{x\to y}}|x\rangle|y\rangle \,\mathrm{d}x \,\mathrm{d}y. \tag{3.115}$$

For an eigenvector v_i of D (with eigenvalue λ_i), define the state $|v_i\rangle := \int_\Omega v_i(x)\,\mathrm{d}x = \int_\Omega \frac{f_i(x)}{\sqrt{\pi(x)}}\,\mathrm{d}x$. Then the walk operator W has the corresponding eigenvector $|u_i\rangle = \big(I - (\lambda_i - i\sqrt{1-\lambda_i^2})S\big)T|v_i\rangle$ following the proof of Theorem 3.3.1. Let $C_i := \lambda_i - i\sqrt{1-\lambda_i^2}$; then $\langle\phi_a|u_i\rangle = \langle\phi_a|T|v_i\rangle - C_i\langle\phi_a|ST|u_i\rangle$. Furthermore,

$$\langle\phi_a|T|v_i\rangle = \langle a|v_i\rangle = \int_\Omega \frac{\sqrt{a(x)}f_i(x)}{\sqrt{\pi(x)}}\,\mathrm{d}x, \tag{3.116}$$

and

$$\begin{aligned}
\langle\phi_a|ST|v_i\rangle &= \left(\int_\Omega \sqrt{a_x p_{x\to y}}\langle y|\langle x|\right)\left(\int_\Omega \sqrt{v_{x'} p_{x'\to y'}}|x'\rangle|y'\rangle\right) && (3.117)\\
&= \int_\Omega \sqrt{a_x}\Big(\int_\Omega \sqrt{p_{x\to y}p_{y\to x}v_i(y)}\,\mathrm{d}y\Big)\,\mathrm{d}x && (3.118)\\
&= \int_\Omega \sqrt{a_x}(Dv_i)(x)\,\mathrm{d}x && (3.119)\\
&= \lambda_i \int_\Omega \sqrt{a_x}v_i(x)\,\mathrm{d}x && (3.120)\\
&= \lambda_i\langle a|v_i\rangle. && (3.121)
\end{aligned}$$

We have $\langle\phi_a|u_i\rangle = (1-\lambda_i C_i)\langle a|v_i\rangle$ and therefore

$$|\langle\phi_a|u_i\rangle| = (1-\lambda_i C_i)\langle a|v_i\rangle = \sqrt{(1-\lambda_i^2)^2 + (1-\lambda_i)^2}\langle a|v_i\rangle \le 2|\langle a|v_i\rangle|. \tag{3.122}$$

In addition,

$$\langle a|v_i\rangle = \int_\Omega \frac{\sqrt{a(x)}f_i(x)}{\sqrt{\pi(x)}}\,\mathrm{d}x = \int_\Omega \sqrt{\frac{\pi(x)}{a(x)}}\frac{a(x)f_i(x)}{\pi(x)}\,\mathrm{d}x. \tag{3.123}$$

The above discussion establishes the following proposition indicating that if a distribution with density $a(x)$ mixes fast and the stationary distribution with density $\pi(x)$ has a bounded L_2-norm with respect to $a(x)$, then the quantum walk operator W acting on the subspace spanned by $|\pi\rangle$ and $|a\rangle$ has a large effective spectral gap.

Proposition 3.4.2. *Let $M = (\Omega, p)$ be an ergodic reversible Markov chain with a transition operator P and unique stationary state with a corresponding density $\pi \in L_2(\Omega)$. Let $\{(\lambda_i, f_i)\}$ be the set of eigenvalues and eigenfunctions of P, and $|u_i\rangle$ be the eigenvectors of the corresponding quantum walk operator W. Let $a \in L_2(\Omega)$ be a probability density that is a warm start for π and mixes up to total variation distance ϵ in t steps of M. Furthermore, assume that $\int_\Omega \frac{\pi(x)}{a(x)}\,\pi(x)\mathrm{d}x \le c$ for some constant c. Define*

$$|a\rangle = \int_\Omega \sqrt{a(x)}|x\rangle\,\mathrm{d}x; \tag{3.124}$$

$$|\phi_a\rangle = \int_\Omega \sqrt{a(x)}\int_\Omega \sqrt{p_{x\to y}}|x\rangle|y\rangle\,\mathrm{d}x\,\mathrm{d}y. \tag{3.125}$$

Then $\langle\phi_a|u_i\rangle = O(\epsilon^{1/2})$ for all i such that $1 > \lambda_i \ge 1 - \frac{1}{\Omega(t)}$.

Proof. Define $S = \{x | \frac{\pi(x)}{a(x)} \geq \sqrt{\frac{c}{\epsilon}}\}$. Because $\int_\Omega \frac{\pi(x)^2}{a(x)^2} a(x)\mathrm{d}x = \int_\Omega \frac{\pi(x)}{a(x)} \pi(x)\mathrm{d}x \leq c$, Markov's inequality implies that $\int_S a(x)\mathrm{d}x \leq \epsilon$.

We now define the quantum state $|a'\rangle$ such that $\langle x|a'\rangle = \langle x|a\rangle$ if $x \notin S$ and $\langle a|x'\rangle = 0$ otherwise, and $|\phi_{a'}\rangle = T|a'\rangle$. Then

$$\||\phi_a\rangle - |\phi_{a'}\rangle\| = \left\| \int_S \sqrt{a(x)} T|x\rangle \, \mathrm{d}x \right\| = \sqrt{\int_\Omega a(x) \, \mathrm{d}x} = \sqrt{\epsilon}. \tag{3.126}$$

From (3.122) and (3.123), if $1 > \lambda_i \geq 1 - \frac{1}{O(t)}$, then

$$|\langle \phi_{a'}|u_i\rangle| \leq \left| 2 \int_\Omega \sqrt{\frac{\pi(x)}{a(x)}} \frac{a(x) f_i(x)}{\pi(x)} \, \mathrm{d}x \right| \leq \frac{2c^{1/4} \langle a, f_i\rangle_\pi}{\epsilon^{1/4}} \leq 2c^{1/4}\epsilon^{3/4}. \tag{3.127}$$

Finally,

$$\langle \phi_a|u_i\rangle = \langle \phi_{a'}|u_i\rangle + \langle \phi_a - \phi_{a'}|u_i\rangle \leq 2c^{1/4}\epsilon^{3/4} + \sqrt{\epsilon} = \emptyset(\sqrt{\epsilon}) \tag{3.128}$$

if $1 > \lambda_i \geq 1 - \frac{1}{\Omega(t)}$. Hence the result follows. $\square$

Warmness of π_{i+1} with respect to π_i We show that density π_i mixes to π_{i+1} under the walk W_{i+1} and vice versa. To apply Theorem 3.2.5, we show that the two distributions are warm with respect to each other.

The L_2-norm of a distribution with density $\pi_1 \in L_2(\Omega)$ with respect to another with density $\pi_2 \in L_2(\Omega)$ is defined as

$$\|\pi_1/\pi_2\| = \mathbb{E}_{X\sim\pi_1}\left[\frac{\pi_1(X)}{\pi_2(X)}\right] = \int_\Omega \frac{\pi_1(x)}{\pi_2(x)} \pi_1(x) \, \mathrm{d}x. \tag{3.129}$$

A density $\pi_1 \in L_2(\Omega)$ is said to be a warm start for $\pi_2 \in L_2(\Omega)$ if the L_2-norm $\|\pi_1/\pi_2\|$ is bounded by a constant.

Lemma 3.4.5 ([73, Lemma 4.4])**.** *The L_2-norm of the probability distribution with density $\pi_i = \frac{e^{-a_i x_0}}{Z(a_i)}$ with respect to that with density $\pi_{i+1} = \frac{e^{-a_{i+1} x_0}}{Z(a_{i+1})}$ is at most 8.*

Lemma 3.4.6. *The L_2-norm of the probability distribution with density $\pi_{i+1} = \frac{e^{-a_{i+1} x_0}}{Z(a_{i+1})}$ with respect to that with density $\pi_i = \frac{e^{-a_i x_0}}{Z(a_i)}$ is at most e.*

Proof. Since $a^n Z(a)$ is a log-concave function [73, Lemma 3.2], we have

$$\begin{aligned}
\mathbb{E}_{X\sim\pi_{i+1}}\left[\frac{\pi_{i+1}(X)}{\pi_i(X)}\right] &= \frac{\int_{\mathrm{K}'} e^{(a_i - a_{i+1})x_0} e^{-a_{i+1}x_0}\mathrm{d}x \int_{\mathrm{K}'} e^{-a_i x_0}\mathrm{d}x}{\int_{\mathrm{K}'} e^{-a_{i+1}x_0}\mathrm{d}x \int_{\mathrm{K}'} e^{-a_{i+1}x_0}\mathrm{d}x} \\
&= \frac{Z(2a_{i+1} - a_i)Z(a_i)}{Z(a_{i+1})^2} \qquad \text{(definition of } Z\text{)} && (3.130) \\
&\le \left(\frac{a_{i+1}^2}{a_i(2a_{i+1} - a_i)}\right)^n \qquad \text{(logconcavity of } a^n Z(a)\text{)} && (3.131) \\
&\le \left(\frac{\left(1 - \frac{1}{\sqrt{n}}\right)^2}{1 - \frac{2}{\sqrt{n}}}\right)^n \qquad \text{(definition of } a_i\text{)} && (3.132) \\
&\le \left(1 + \frac{2}{n}\right)^n < e^2, && (3.133)
\end{aligned}$$

where (3.133) holds because $1 + \frac{1}{n} - \frac{2}{\sqrt{n}} \le (1 + \frac{2}{n})(1 - \frac{2}{\sqrt{n}})$ as long as $n \ge 16$. □

Error analysis of $\pi/3$-amplitude amplification Consider a simulated annealing procedure that follows a sequence of Markov chains $M_1, M_2, \ldots$ with stationary states $\mu_1, \mu_2, \ldots$. Consider an alternate walk operator (used in [78]) of the form

$$W_i' = U_i^\dagger S U_i R_{\mathcal{A}} U_i^\dagger S U_i R_{\mathcal{A}} \tag{3.134}$$

where $R_{\mathcal{A}}$ denotes the reflection about the subspace $\mathcal{A} := \mathrm{span}\{|x\rangle|0\rangle : x \in \mathrm{K}\}$ and S is the swap operator. We have $U_i|x\rangle|0\rangle = \int_{y\in \mathrm{K}} \sqrt{p^{(i)}_{x\to y}}|x\rangle|y\rangle\, \mathrm{d}y$ where $p^{(i)}$ is the transition probability corresponding to the i^{th} chain.

The W_i' operator is related to the walk operator $W_i = S(2\Pi_i - I)$ via conjugation by U_i, i.e., $W_i = U_i W_i' U_i^{\dagger}$. Thus W_i' has the same eigenvalues as W_i, and if $|u_j\rangle$ is an eigenvector of W_i with eigenvalue λ_j, then $|v\rangle = U_i^{\dagger}|u_j\rangle$ is an eigenvector of W_i' with the same eigenvalue λ_j. For any classical distribution f, we define $|f\rangle = \int_{\Omega} \sqrt{f(x)}|x\rangle\, \mathrm{d}x$ and

$$|\phi^{(i)}_f\rangle = \int_{\Omega} \sqrt{f(x)}|x\rangle \int_{\Omega} \sqrt{p^{(i)}_{x\to y}}|y\rangle\, \mathrm{d}y\, \mathrm{d}x$$

. Since $|\phi^{(i)}_{\pi_i}\rangle$ is a stationary state of W_i with eigenvalue 1, it follows that $|\pi_i\rangle|0\rangle$ is an eigenvalue of W_i with eigenvalue 1.

In each stage of the volume estimation algorithm, we sample from a state with density $\pi_i(x) = \frac{e^{-a_i x_0}}{Z(a_i)}$. Each such distribution is the stationary state of a hit-and-run walk with the corresponding target density. Thus the corresponding state $|\pi_i\rangle$ is the stationary state of the corresponding walk operators W_i and W_i'. Both W_i and $W_{i'}$ can be implemented using a constant number of U_i gates.

From Lemma 3.4.2, we know that the inner product $\langle \pi_i | \pi_{i+1}\rangle$ between the states at any stage of the algorithm is at least $\frac{1}{3}$. This implies that the inner product between $|\pi_i\rangle|0\rangle$ and $|\pi_{i+1}\rangle|0\rangle$ is also at least $\frac{1}{3}$. In the following we abuse notation by sometimes writing only $|\pi_i\rangle$ to denote $|\pi_i\rangle|0\rangle$, as it is easy to tell from context whether the ancilla register should be present.

Lemma 3.2.1 in Section 3.2.2 indicates that $\pi/3$-amplitude amplification can be used to rotate the state $|\pi_i\rangle$ to $|\pi_{i+1}\rangle$ if we can implement the rotation unitaries

$$R_i = \omega|\pi_i\rangle\langle\pi_i| + (I - |\pi_i\rangle\langle\pi_i|) \quad \text{and} \quad R_{i+1} = \omega|\pi_{1+1}\rangle\langle\pi_{i+1}| + (I - |\pi_{i+1}\rangle\langle\pi_{i+1}|) \,.$$

To implement these rotations we use the fact that π_i and π_{i+1} are the eigenvectors of the operators W_i' and W_{i+1}' with eigenvalue 1, respectively. We show the following lemmas which are adapted variants of Lemma 2 and Corollary 2 in [78]:

Lemma 3.4.7. *Let W be a unitary operator with a unique leading eigenvector $|\psi_0\rangle$ with eigenvalue 1. Denote the remaining eigenvectors by $|\psi_j\rangle$ with corresponding eigenvalues $e^{2\pi i\xi_j}$. For any $\Delta \in (0,1]$ and $\epsilon_2 < 1/2$, define $a := \log(1/\Delta)$ and $c := \log(1/\sqrt{\epsilon_2})$. There exists a quantum circuit V that uses ac ancilla qubits and invokes the controlled-W gate $2^a c$ times such that*

$$V|\psi_0\rangle|0\rangle^{\otimes ac} = |\psi_0\rangle|0\rangle^{\otimes ac} \tag{3.135}$$

and

$$V|\psi_j\rangle|0\rangle^{\otimes ac} = \sqrt{1-\epsilon_2(j)}|\psi_j\rangle|\chi_j\rangle + \sqrt{\epsilon_2(j)}|\psi_j\rangle|0\rangle^{\otimes ac} \tag{3.136}$$

where $|\chi_j\rangle$ is orthogonal to $|0\rangle^{\otimes ac}$ for all $|\psi_j\rangle$ such that $\xi_j \geq \Delta$, and $\epsilon_2(j) \leq \epsilon_2$ for all j.

Proof. Consider a quantum phase estimation circuit U with a ancilla qubits that invokes the controlled-W gate 2^a times (see Figure 3.5). The phase estimation circuit first creates an equal superposition over a ancilla qubits using Hadamard gates. For $k = 0, \ldots, a-1$ we apply a controlled-W^k operator to the input register, controlled by the $(a-k)^{\text{th}}$ register. Finally the inverse quantum Fourier transform is applied on the ancilla registers. Then

$$U|\psi_j\rangle|0\rangle^{\otimes a} = |\psi_j\rangle \otimes \mathsf{QFT}^\dagger \left(\frac{1}{\sqrt{2^a}} \sum_{m=0}^{2^a-1} e^{2\pi i m \xi_j} |m\rangle \right) \tag{3.137}$$

$$= |\psi_j\rangle \otimes \frac{1}{2^a} \sum_{m,m'=0}^{2^a-1} e^{2\pi i m(\xi_j - m'/2^a)} |m'\rangle. \tag{3.138}$$

The amplitude corresponding to $|0\rangle$ on the ancilla registers is

$$a_{j,0} := \frac{1}{2^a} \sum_{m=0}^{2^a-1} e^{2\pi i m \xi_j} = \frac{1 - e^{2\pi i 2^a \xi_j}}{2^a(1 - e^{2\pi i \xi_j})} \tag{3.139}$$

for $j \neq 0$, and $a_{0,0} = 1$. If $j \neq 0$ then

$$|a_{j,0}| = \left| \frac{1 - e^{2\pi i 2^a \xi_j}}{2^a(1 - e^{2\pi i \xi_j})} \right| \leq \left| \frac{1}{2^{a-1}(1 - e^{2\pi i \xi_j})} \right| \leq \frac{1}{2^{a+1}|\xi_j|}. \tag{3.140}$$

Thus $|a_{j,0}| \leq \frac{1}{2}$ if $\xi_j \geq \Delta$. Using c copies of the circuit (resulting in ac ancilla registers and $2^a c$ controlled-W gates), the amplitude for 0 in all the ancilla registers if $\xi_j \geq \Delta$ is at most $\frac{1}{2^c} = \sqrt{\epsilon}$. $\square$

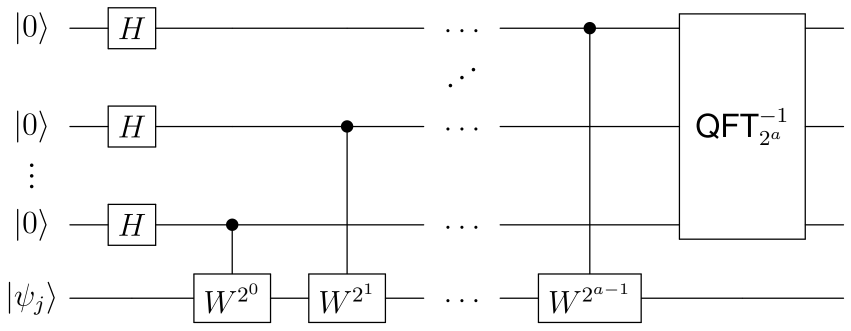


Figure 3.5: The quantum phase estimation circuit. Here W is a unitary operator with eigenvector $|\psi_j\rangle$; in $\pi/3$-amplitude estimation it is the quantum walk operator W_i' in (3.134).

Corollary 3.4.1. *Let W be a unitary operator with a unique leading eigenvector $|\psi_0\rangle$ with eigenvalue 1. Denote the remaining eigenvectors by $|\psi_j\rangle$ with corresponding eigenvalues $e^{2\pi i \xi_j}$. For any $\Delta \in (0,1]$ and $\epsilon_2 < 1/2$, define $a := \log(1/\Delta)$ and $c := \log(1/\sqrt{\epsilon_2})$. For any constant $\alpha \in \mathbb{C}$, there exists a quantum circuit $\tilde{R}$ that uses ac ancilla qubits and invokes the controlled-W gate $2^{a+1}c$ times such that*

$$\tilde{R}|\psi_0\rangle|0\rangle^{\otimes ac} = (R|\psi_0\rangle)|0\rangle^{\otimes ac} \tag{3.141}$$

(where $R = \alpha|\psi_0\rangle\langle\psi_0| - (I - |\psi_0\rangle\langle\psi_0|)$) and

$$\|\tilde{R}|\psi_j\rangle|0\rangle^{\otimes ac} - (R|\psi_j\rangle)|0\rangle^{\otimes ac}\| \leq \sqrt{\epsilon_2} \tag{3.142}$$

for $j \neq 0$ such that $\xi_j \geq \Delta$.

Proof. Let $\tilde{R} := V^\dagger(I \otimes Q)V$ where V is the quantum circuit in Lemma 3.4.7 and $Q := \alpha|0\rangle\langle 0|^{\otimes ac} + (I - |0\rangle\langle 0|^{\otimes ac})$. Then we have

$$\tilde{R}|\psi_0\rangle|0\rangle^{\otimes ac} = V^\dagger(I \otimes Q)|\psi_0\rangle|0\rangle^{\otimes ac} = \alpha|\psi_0\rangle|0\rangle^{\otimes ac} = R|\psi_0\rangle|0\rangle^{\otimes ac}. \tag{3.143}$$

For $j \neq 0$ such that $\xi_j \geq \Delta$,

$$\tilde{R}|\psi_j\rangle|0\rangle^{\otimes ac} = V^\dagger(I \otimes Q)(\sqrt{1-\epsilon_2}|\psi_j\rangle|\chi_j\rangle + \sqrt{\epsilon_2}|\psi_j\rangle|0\rangle^{\otimes ac}) \tag{3.144}$$

$$= V^\dagger(\sqrt{1-\epsilon_2}|\psi_j\rangle|\chi_j\rangle + \sqrt{\epsilon_2}\alpha|\psi_j\rangle|0\rangle^{\otimes ac}) \tag{3.145}$$

$$= V^\dagger(|\psi_j\rangle \otimes (\sqrt{1-\epsilon_2}|\chi_j\rangle + \sqrt{\epsilon_2}|0\rangle^{\otimes ac}) + \sqrt{\epsilon_2}(\alpha-1)|\psi_j\rangle|0\rangle^{\otimes ac}) \tag{3.146}$$

$$= |\psi_j\rangle|0_j\rangle + V^\dagger\sqrt{\epsilon_2}(\alpha-1)|\psi_j\rangle|0\rangle^{\otimes ac}. \tag{3.147}$$

Thus $\|\tilde{R}|\psi_j\rangle|0\rangle^{\otimes ac} - (R|\psi_j\rangle)|0\rangle^{\otimes ac}\| \leq \|V^\dagger\sqrt{\epsilon_2}(\alpha-1)|\psi_j\rangle|0\rangle^{\otimes ac}\| \leq \sqrt{\epsilon_2}$. □

Finally, we prove the following lemma for analyzing the error incurred by $\pi/3$-amplitude amplification in our quantum volume estimation algorithm:

Lemma 3.4.8. *Starting from $|\pi_i\rangle$, we can obtain a state $|\tilde{\pi}_{i+1}\rangle$ such that $\||\pi_{i+1}\rangle - |\tilde{\pi}_{i+1}\rangle\| \leq \epsilon$ using $\tilde{O}(n^{3/2}\log(1/\epsilon))$ calls to the controlled walk operators W'_i, W'_{i+1}. This results in $\tilde{O}(n^{3/2}\log(1/\epsilon))$ calls to the membership oracle O_{K}.*

Proof. From Theorem 3.2.5, Lemma 3.4.5, and Lemma 3.4.6, we find that

- $\pi_i(x)$ mixes up to total variation distance ϵ_1 in $O\big(n^3\log^5\frac{n}{\epsilon_1}\big)$ steps of the Markov chain M_{i+1}, and
- $\pi_{i+1}(x)$ mixes up to total variation distance ϵ_1 in $O\big(n^3\log^5\frac{n}{\epsilon_1}\big)$ steps of the Markov chain M_i.

From Proposition 3.4.2, we find the following:

- $|\pi_i\rangle = |\pi'_i\rangle + |e_1\rangle$ where $|\pi'_i\rangle$ lies in the space of eigenvectors $|v_j^{(i+1)}\rangle$ of W'_{i+1} such that $\lambda_j^{(i+1)} = 1$ or $\lambda_j^{(i+1)} \leq 1 - \frac{1}{O(n^3\log^5(n/\epsilon_1))}$, and $\||e_1\rangle\| \leq \epsilon_1$; and

- $|\pi_{i+1}\rangle = |\pi'_{i+1}\rangle + |e_2\rangle$ where $|\pi'_{i+1}\rangle$ lies in the space of eigenvectors $|v_j^{(i)}\rangle$ of W'_i such that $\lambda_j^{(i)} = 1$ or $\lambda_j^{(i)} \leq 1 - \frac{1}{O(n^3 \log^5(n/\epsilon_1))}$, and $\||e_2\rangle\| \leq \epsilon_1$.

Note that $|\pi_i\rangle$ and $|\pi_{i+1}\rangle$ are simply the leading eigenvectors of W_i and W_{i+1}, respectively. Thus both $|\pi_i\rangle$ and $|\pi_{i+1}\rangle$ lie ϵ_1 close to the "good" subspaces corresponding to W'_i (respectively W'_{i+1}) which are spanned by eigenvectors $|v_j^{(i)}\rangle$ (respectively $|v_j^{(i+1)}\rangle$) with eigenvalues $e^{2\pi i \xi_j^{(i)}}$ (respectively $e^{2\pi i \xi_j^{(i+1)}}$) such that $\xi_j^{(i)} = 0$ or $\xi_j^{(i)} \geq \frac{1}{O(n^{3/2} \log^{5/2}(n/\epsilon_1))}$. Each state that occurs during $\pi/3$-amplitude amplification to rotate $|\pi_i\rangle$ to $|\pi_{i+1}\rangle$ or vice versa is a linear combination of $|\pi_i\rangle$ and $|\pi_{i+1}\rangle$ and is thus also close to the good subspaces of W'_i and W'_{i+1}.

Applying Corollary 3.4.1 with $\Delta = \frac{1}{n^{3/2} \ln^{5/2}(n/\epsilon_1)}$ and $\epsilon_2 = \epsilon_1^2$, we can implement a quantum operators $\tilde{R}_i, \tilde{R}_{i+1}$ such that $\|R_i - \tilde{R}_i\| \leq 2\epsilon_1$ and $\|R_{i+1} - \tilde{R}_{i+1}\| \leq 2\epsilon_1$, using $O(n^{3/2} \log^{5/2}(n/\epsilon_1) \log(1/\epsilon_1))$ calls to the controlled-W'_i and controlled-W'_{i+1} operators, respectively.

The above shows how to approximately implement R_i and R_{i+1}. If these operators could be implemented perfectly, Lemma 3.2.1 and Lemma 3.4.2 show that we can prepare a state $|\tilde{\pi}_{i+i}\rangle$ such that $\langle \pi_{i+1} | \tilde{\pi}_{i+1}\rangle \leq 1 - (2/3)^{3^m}$ by applying m recursive levels of $\pi/3$-amplitude amplification to $|\pi_i\rangle$, using 3^m calls to $R_i, R_i^\dagger, R_{i+1}, R_{i+1}^\dagger$. Since
$\|\pi_{i+1} - \tilde{\pi}_{i+1}\| = \sqrt{2(1 - \langle \pi_{i+1} | \tilde{\pi}_{i+1}\rangle)}$, after $O(\log(1/\epsilon_2))$ calls to the rotation gates we obtain a final state with error ϵ_2. However, each rotation gate can cause an error of ϵ_1 by itself. By making $O(n^{3/2} \log^{5/2}(n/\epsilon_1) \log(1/\epsilon_1) \log(1/\epsilon_2))$ calls to controlled-W'_i and controlled-W'_{i+1} operators, we obtain a final error of $O(\epsilon_1 \log(1/\epsilon_2) + \epsilon_2)$. Choosing $\epsilon_2 = \epsilon/2$ and $\epsilon_1 = \epsilon/(2\ln(2/\epsilon))$ gives the result. $\square$

Error analysis for the quantum Chebyshev inequality We also analyze the error from the quantum Chebyshev inequality (Theorem 3.2.3), giving a robust version of Lemma 3.4.3.

Lemma 3.4.9. *Suppose we have $\tilde{O}(\log(1/\delta)/\epsilon)$ copies of a state $|\tilde{\pi}_{i-1}\rangle$ such that $\||\tilde{\pi}_{i-1}\rangle - |\pi_{i-1}\rangle\| \leq \epsilon$. Then the quantum Chebyshev inequality can be used to output $\tilde{V}_i$ such that $|\tilde{V}_i - \mathbb{E}_{\pi_i}[V_i]| \leq O(\epsilon)\mathbb{E}_{\pi_i}[V_i]$ with success probability $1-\delta^4$ using $\tilde{O}(n^{3/2}\log(1/\delta)/\epsilon)$ calls to the membership oracle. The output state $|\hat{\pi}_{i-1}\rangle$ satisfies $\||\hat{\pi}_{i-1}\rangle - |\pi_{i-1}\rangle\| = O(\epsilon+\delta)$.*

Proof. The error-free version of this lemma was proven in Lemma 3.4.3. Here we focus on the error analysis. The quantum Chebyshev inequality uses an implementation of $US_0U^\dagger S_i$ where U is a unitary operator satisfying $U|\pi_{i-1}\rangle = |\pi_i\rangle$. From Lemma 3.4.8, using $\log(1/\epsilon_2)$ iterations of $\pi/3$-amplitude amplification ($U_{\log 1/\epsilon_2}$ in (3.16)) instead of U induces an error of ϵ_2 and uses $O(n^{3/2}\log(1/\epsilon_2))$ oracle calls. Using approximate phase estimation as in Corollary 3.4.1 and Lemma 3.4.8, Π_{i-1} and Π_i can be implemented up to error ϵ_3 using $O(n^{3/2}\log(1/\epsilon_3))$ oracle calls. Thus each block corresponding to Theorem 3.2.2 induces an error of $O(\epsilon_2+\epsilon_3)$, and the final state before the median is measured has an error of $O(\epsilon+\epsilon_2+\epsilon_3)$. Therefore, using $O(\log(1/\delta_1)/\epsilon)$ copies of $|\tilde{\pi}_{i-1}\rangle$ returns a sample $\tilde{V}_i$ such that $|\tilde{V}_i - \mathbb{E}_{\pi_i}[V_i]| \leq O(\epsilon_2+\epsilon_3+\epsilon)\mathbb{E}_{\pi_i}[V_i]$ with success probability $1-\delta_1$. Performing a measurement with success probability $1-\delta_1$ implies that the posterior state has an overlap $\sqrt{1-\delta_1}$ with the initial state. This induces an error of magnitude at most $\sqrt{2(1-\sqrt{1-\delta_1})} = O(\delta_1^{1/4})$.

The measurement on the $\log(1/\delta)/c$ copies of $|\tilde{\pi}_{i-1}\rangle$ used to estimate $\hat{\mu}$ has relative error at most c with probability $1-\delta$. This causes an error $O(\delta_1^{1/4})$ in addition to the error ϵ_2 from $\pi/3$-amplitude amplification.

Finally, note that the basic amplitude estimation circuit (analyzed in Theorem 3.2.2) is a subroutine of the quantum Chebyshev inequality (Theorem 3.2.3), and uncomputing the block corresponding to Theorem 3.2.2 induces an error of $O(\epsilon_2 + \epsilon_3)$, giving an overall error of $O(\epsilon_2 + \epsilon_3 + \epsilon + \delta^{1/4})$. The result follows by taking $\epsilon_2 = \epsilon_3 = \epsilon$ and $\delta_1 = \delta^4$. □

We finally prove Lemma 3.4.4 here.

Proof. Lemma 3.4.9 is used to estimate the mean with $\epsilon = \epsilon_1$ and leaves a posterior state $|\hat{\pi}_{i-1}\rangle$ such that $\||\hat{\pi}_{i-1}\rangle - |\pi_{i-1}\rangle\| = O(\epsilon_1 + \delta)$. We can then use $\pi/3$-amplitude amplification to rotate this state into $|\tilde{\pi}_i\rangle$, adding error $O(\epsilon')$ at the cost of $O(n^{3/2}\log(1/\epsilon'))$. This completes the proof. □

3.4.4 Quantum algorithms for rounding logconcave densities

We first define roundedness of logconcave density functions as follows:

Definition 3.4.1. *A logconcave density function f is said to be c-rounded if*

1. *The level set of f of probability $1/8$ contains a ball of radius r;*

2. *$\mathbb{E}_f(|x - z_f|) \leq R^2$, where z_f is the centroid of f, i.e., $z_f := \mathbb{E}_f(x)$;*

and $R/r \leq c\sqrt{n}$.

In the previous section we assumed that the distributions π_i sampled during the hit-and-run walk are $O(1)$-rounded (i.e., well-rounded). From Theorem 3.2.5, this implies that the hit-and-run walk for the distribution π_i mixes from a warm start in time $\tilde{O}(n^3)$. In this subsection we show how the distributions π_i can be transformed to satisfy this condition.

Following the classical discussion in [40], we actually show a stronger condition: the distributions are transformed to be in "near-isotropic" position. A density function f is said to be in *isotropic* position if

$$\mathbb{E}_f[x] = 0 \quad \text{and} \quad \mathbb{E}_f[xx^T] = I. \tag{3.148}$$

The latter equation is equivalent to $\int_{\mathbb{R}^n} (u^T x)^2 f(x)\,\mathrm{d}x = |u|^2$ for every vector $u \in \mathbb{R}^n$. We say that K is *near-isotropic* up to a factor of c if

$$\frac{1}{c} \le \int_{\mathbb{R}^n} (u^T(x - z_f))^2 f(x)\,\mathrm{d}x \le c \tag{3.149}$$

for every unit vector $u \in \mathbb{R}^n$.

The following lemma shows that logconcave density functions in isotropic position are also $O(1)$-rounded:

Lemma 3.4.10 ([118, Lemma 5.13])**.** *Every isotropic logconcave density is $(1/e)$-rounded.*

The following lemma shows that any logconcave density function can be put into isotropic position by applying an affine transformation, generalizing the same result for uniform distributions by Rudelson [119]:

Lemma 3.4.11 ([40, Lemma 2.2])**.** *Let f be a logconcave function in $\mathbb{R}^n$ such that there is no linear subspace $\mathcal{S} \subseteq \mathbb{R}^n$ such that $\int_{\mathcal{S}} f(x)\,dx > 1/2$, and let $X^1, \ldots, X^k$ be independent random points from the corresponding distribution. There is a constant C_0 such that if $k > C_0 t^3 \ln n$, then the transformation $g(x) = T^{-1/2}x$ where*

$$\bar{X} = \frac{1}{k}\sum_{i=1}^{k} X^i, \qquad T = \frac{1}{k}\sum_{i=1}^{k} (X^i - \bar{X})(X^i - \bar{X})^T \tag{3.150}$$

puts f in 2-isotropic position with probability at least $1 - 1/2^t$.

From Lemma 3.4.11, $k = \lceil C_0 n \ln^5 n \rceil = \tilde{\Theta}(n)$ samples from a logconcave density f suffice to put it into near-isotropic position. However, efficiently obtaining samples from a density π_i requires it to be well-rounded to start with. To overcome this difficulty, we interlace the rounding with the stages of the volume estimation algorithm where in each stage, we obtain an affine transformation that puts the density to be sampled in the next stage into isotropic position. The density π_0 is very close to an exponential distribution (since it is concentrated inside the convex body) and can hence be sampled without resorting to a random walk.

To show that samples from π_i can be used to transform π_{i+1} into isotropic position, we use the following lemma:

Lemma 3.4.12 ([39, Lemma 4.3])**.** *Let f and g be logconcave densities over K with centroids z_f and z_g respectively. Then for any $u \in \mathbb{R}^n$,*

$$\mathbb{E}_f[(u \cdot (x - z_f))^2] \leq 16\mathbb{E}_f\left[\frac{f}{g}\right]\mathbb{E}_g[(u \cdot (x - z_g))^2]. \tag{3.151}$$

We now have the following proposition:

Proposition 3.4.3. *If affine transformation S_i puts π_i in near-isotropic position then it also puts π_{i+1} in near-isotropic position.*

Proof. Let S_i put π_i in 2-isotropic position. Applying Lemma 3.4.12 with $f = \pi_{i+1}, g = \pi_i$, we have that for any unit vector $u \in \mathbb{R}^n$,

$$\mathbb{E}_{\pi_{i+1}}[(u \cdot (x - z_{\pi_{i+1}}))^2] \leq 16\mathbb{E}_{\pi_{i+1}}\left[\frac{\pi_{i+1}}{\pi_i}\right]\mathbb{E}_{\pi_i}[(u \cdot (x - z_{\pi_i}))^2] \leq 32e^2 \tag{3.152}$$

since $\mathbb{E}_{\pi_{i+1}}\left[\frac{\pi_{i+1}}{\pi_i}\right] \leq e^2$ from Lemma 3.4.6. Again applying Lemma 3.4.12

$$\frac{1}{2} \leq \mathbb{E}_{\pi_i}[(u \cdot (x - z_{\pi_i}))^2] \leq \mathbb{E}_{\pi_i}\left[\frac{\pi_i}{\pi_{i+1}}\right] \mathbb{E}_{\pi_{i+1}}[(u \cdot (x - z_{\pi_{i+1}}))^2] \tag{3.153}$$

$\mathbb{E}_{\pi_i}\left[\frac{\pi_i}{\pi_{i+1}}\right] \leq 8$ from Lemma 3.4.5. Therefore,

$$\frac{1}{2} \leq 128e^2 \mathbb{E}_{\pi_{i+1}}[(u \cdot (x - z_{\pi_{i+1}}))^2] \tag{3.154}$$

Thus $E_{\pi_{i+1}}$ is also put in near-isotropic position. $\square$

We finally have the main result of this section:

Proposition 3.4.4. *At each stage i of Algorithm 9, the affine transformation puts the distribution π_{i+1} in near-isotropic position using an additional $\tilde{O}(n^{2.5})$ quantum queries to O_K.*

Proof. Since π_0 is nearly an exponential distribution, it can be sampled without using a random walk and thus the proposition is true for $i = 0$. Assume that the proposition is true for $1, 2, \ldots, k$. Then an affine transformation can be found to put π_k in near-isotropic position. Thus a classical hit-and-run walk starting from π_{k-1} converges to π_k in $\tilde{O}(n^3)$ steps. By the analysis in Section 3.4.3.4, a quantum sample $|\pi_{k-1}\rangle$ can be rotated to $|\pi_k\rangle$ using $\tilde{O}(n^{1.5})$ quantum queries. $\tilde{O}(n)$ such samples suffice to compute the covariance matrix T in (3.150), which puts π_k in 2-isotropic position. By Proposition 3.4.3, this also puts π_{k+1} in near-isotropic position. This concludes the proof. $\square$

Rounding the convex body as a preprocessing step Consider applying only the rounding part of Algorithm 9. By Proposition 3.4.4, the final affine transformation puts the density $\pi_m \propto$

Algorithm 9: Volume estimation of convex K with interlaced rounding.

```
Input: Membership oracle $O_{\mathrm{K}}$ for $\mathrm{K}$.
Output: $\epsilon$-multiplicative approximation of $\mathrm{Vol}(\mathrm{K})$.
1 Set $m = \Theta(\sqrt{n}\log(n/\epsilon))$ to be the number of iterations of simulated annealing and
    $a_i = 2n(1-\frac{1}{\sqrt{n}})^i$ for $i \in [m]$. Let $\pi_i$ be the probability distribution over $\mathrm{K}'$ with
    density proportional to $e^{-a_i x_0}$;
  Set error parameters $\delta, \epsilon' = \Theta(\epsilon/m^2), \epsilon_1 = \epsilon/2m$; let $k = \tilde{\Theta}(\sqrt{n}/\epsilon)$ be the number of
    copies of stationary states for applying the quantum Chebyshev inequality; let
    $l = \tilde{\Theta}(n)$ be the number of copies of stationary states needed to obtain the affine
    transformation $S_i$; Prepare $k+l$ (approximate) copies of $|\pi_0\rangle$, denoted
    $|\tilde{\pi}_0^{(1)}\rangle, \ldots, |\tilde{\pi}_0^{(k+l)}\rangle$;
  for $i \in [m]$ do
2     Use the quantum Chebyshev inequality on the $k$ copies of the state $|\tilde{\pi}_{i-1}\rangle$ with
        parameters $\epsilon_1, \delta$ to estimate the expectation $\mathbb{E}_{\pi_i}[V_i]$ (in Eq. (3.94)) as $\tilde{V}_i$
        (Lemma 3.4.9 and Figure 3.4). The post-measurement states are denoted
        $|\hat{\pi}_{i-1}^{(1)}\rangle, \ldots, |\hat{\pi}_{i-1}^{(k)}\rangle$;
3     Use the $l$ copies of the state $|\pi_{i-1}\rangle$ to nondestructively[15] obtain the affine
        transformation $S_i = T = \frac{1}{l}\sum_{q=1}^{l}(X^q - \bar{X})(X^q - \bar{X})^T$ where the $X_q$ are samples
        from the density $\pi_{i-1}$ and $\bar{X} = \frac{1}{l}\sum_{q=1}^{l} X^q$. The post-measurement states are
        denoted $|\hat{\pi}_{i-1}^{(k+1)}\rangle, \ldots, |\hat{\pi}_{i-1}^{(k+l)}\rangle$;
4     Apply $\pi/3$-amplitude amplification with error $\epsilon'$ (Section 3.2.2 and Lemma 3.4.8)
        and affine transformation $S_i$ to map $|S_i\hat{\pi}_{i-1}^{(1)}\rangle, \ldots, |S_i\hat{\pi}_{i-1}^{(k+l)}\rangle$ to
        $|S_i\tilde{\pi}_i^{(1)}\rangle, \ldots, |S_i\tilde{\pi}_i^{(k+l)}\rangle$, using the quantum hit-and-run walk;
5     Invert $S_i$ to get $k+l$ (approximate) copies of the stationary distribution $|\pi_i\rangle$ for
        use in the next iteration;
6 Compute an estimate $\widetilde{\mathrm{Vol}(\mathrm{K}')} = n!v_n(2n)^{-(n+1)}\tilde{V}_1 \cdots \tilde{V}_m$ of the volume of $\mathrm{K}'$, where
    $v_n$ is the volume of the $n$-dimensional unit ball;
7 Use $\widetilde{\mathrm{Vol}(\mathrm{K}')}$ to estimate the volume of $\mathrm{K}$ as $\widetilde{\mathrm{Vol}(\mathrm{K})}$ (Section 3.4.3.1).
```

$e^{-a_m x_0}$ in near-isotropic position. Since $a_m \leq \epsilon^2/n$, we have

$$(1-\epsilon^2)\mathbb{E}_{\mathrm{K}'}[|X-\bar{X}|]^2 \leq \int_{\mathrm{K}'} \frac{e^{-a_m x_0}|x-\bar{x}|^2}{Z(a_m)}\,\mathrm{d}x \leq 2n; \tag{3.155}$$

thus $\mathbb{E}_{\mathrm{K}'}[|X-\bar{X}|]^2 \leq 2n/(1-\epsilon^2)$. From [40, Lemma 3.3], all but an ϵ-fraction of the body is contained inside a ball of radius $O(\sqrt{n})$. Combined with our assumption that $B_2(0,1) \subseteq \mathrm{K}'$, this shows that S_{m+1} puts the convex body K' in well-rounded position.

3.5 Implementation of the quantum hit-and-run walk

Due to the precision of representing real numbers, the implementation of volume estimation algorithms in practice requires to walk in a discrete domain that is a subset of $\mathbb{R}^n$. It is known that walks only taking local steps within a short distance (such as the grid walk and the ball walk) can be discretized with good approximation by dividing $\mathbb{R}^n$ into small hypercubes and walking on their centers (see e.g. [87]), but such error analysis does not automatically apply to hit-and-run walks for which we did not find existing classical discretizations. We emphasize the discretization in contrast to most classical treatments for two reasons: (1) Quantum algorithms are typically presented in a circuit model, in contrast to the RAM model used by classical algorithms. Continuous variables in the circuit model correspond to registers of infinite size, preventing a clear analysis of the resources of the algorithm in terms of gate count. Specifically to obtain the performance of the algorithm in reality, we must show that $\mathrm{poly}(\log(1/\epsilon))$ bit registers suffice. (2) Standard methods of preparing walk operators corresponding to classical Markov Chains (see for eg. [85]) rely on the sparsity of the transition matrix. In the case of geometric random walks sparsity is not well-defined in the continuous case and may not hold even for discretizations (for example, the hit-and-run walk has a non-zero transition density to any point in the convex body.) The efficient preparation of quantum states corresponding to classical distributions is not always a trivial operation, and there has been research [120] about preparing common distributions

[15]Similar to Lemma 3.4.3, we do not directly measure the states; instead we use a quantum circuit to (classically) compute the affine transformation S_i and apply it to the convex body coherently for the next iteration. Note that the quantum register holding the affine transformation will be in some superposition, but by using $O(\log n)$ copies and taking the mean (as in Lemma 3.4.3), the amplitude of the correct affine transformation will be arbitrarily close to 1.

for quantum Monte-Carlo methods. Most existing general procedures come without provable guarantees on the resources required for sufficiently accurate samples; we provide here a simple analysis for the cost of implementing the hit-and-run walk via the Grover-Rudolph method [121].

In this section, we introduce a discretized quantum hit-and-run walk and give an explicit analysis of its implementation. The basic idea of the discretization is to represent the coordinates with rational numbers. We approximate K by a set of discretized points in K and define a Markov chain on these points (see Section 3.5.1). We use a two-level discretization: the hit-and-run process is performed with a coarser discretization and then a point in a finer discretization of the coarse grid is chosen uniformly at random as the actual point to jump to. This ensures that the starting and ending points (in the coarser discretization) of one jump are far from the boundary so that a small perturbation does not change the length of the chord induced by the two points significantly. Then in Section 3.5.2, the discrete conductance can be bounded by bounding the distance between the discrete and continuous transition probabilities as well as the distance between the discrete and continuous subset measures. In Section 3.5.3, we prove that the quantum gate complexity of implementing the discretized quantum hit-and-run walk is $\tilde{O}(n)$, the same overhead as for implementing classical hit-and-run walks.

3.5.1 Discretization of the hit-and-run walk

For a convex body $\mathrm{K} \subseteq \mathbb{R}^n$, we let K_ϵ denote the set of vectors in K whose coordinates can be represented by some fixed-point representation using $\log(1/\epsilon)$ bits[16] We can use the . We call K_ϵ an *ϵ-discretization* of K. The finite set K_ϵ provides an ϵ-net for K. We also define $(\mathbb{R}^n)_\epsilon$ as a

[16]Note that this ϵ is different from the multiplicative error in the problem definition. However, this ϵ is not the dominating error and the overhead is only logarithmic.

ϵ-discretization of $\mathbb{R}^n$.

We consider a Markov chain whose states are the points in K_ϵ. For any $v \in \mathbb{R}^n$, we define the ϵ-*box* $b_\epsilon(v) := \{x \in \mathbb{R}^n : x(i) \in [v(i) - \epsilon/2, v(i) + \epsilon/2],\ \forall i \in [n]\}$. Let $\overline{\mathrm{K}}_\epsilon$ be the continuous set formed by the ϵ-boxes of the points in K_ϵ, i.e., $\overline{\mathrm{K}}_\epsilon = \bigcup_{x \in \mathrm{K}_\epsilon} b_\epsilon(x)$. For two distinct points $u, v \in \mathbb{R}^n$, we denote by ℓ_{uv} the line through them. For a line $\ell \subseteq \mathbb{R}^n$, let $\ell(\overline{\mathrm{K}}_\epsilon)$ be the segment of ℓ contained in $\overline{\mathrm{K}}_\epsilon$, i.e., $\ell(\overline{\mathrm{K}}_\epsilon) = \{x \in \ell : x \in \overline{\mathrm{K}}_\epsilon\}$. In addition, for $u \in \ell$, we define $\ell(\overline{\mathrm{K}}_\epsilon, u, \epsilon')$ as the ϵ'-discretization of $\ell(\overline{\mathrm{K}}_\epsilon)$ starting from u, i.e., $\ell(\overline{\mathrm{K}}_\epsilon, u, \epsilon') = \{x \in \ell(\overline{\mathrm{K}}_\epsilon) : |x - u| = k\epsilon' \text{ for some } k \in \{0, 1, \ldots\}\}$. Analogous to the distribution π_f for the continuous-space case, we define its corresponding discrete distribution $\hat{\pi}_f$ with $\hat{\pi}_f(\mathrm{S}) = \sum_{x \in \mathrm{S}} f(x) / \sum_{x \in (\mathbb{R}^n)_\epsilon} f(x)$.

To implement the hit-and-run walk (see Section 3.2.4), we sample a uniformly random direction from a point u. We achieve this by sampling n coordinates according to the standard normal distribution from the corresponding coordinate of u and normalizing the new point to have unit length; the uniformity of such sampling is well known (see for example [122, 123]).[17] Let this normalized point be v, so that the sampled direction is ℓ_{uv}. Note that the coordinate we sample from is discrete. The directions we can sample form a discrete set denoted $\mathrm{L}(u, \epsilon')$, where ϵ' is the precision for sampling directions.

Now we compute the probability that a specific direction is sampled. After normalization, the point will "snap" to a point in $(\mathbb{R}^n)_\epsilon$. Consider $\bigcup_{v:\, b_{\epsilon'}(v) \cap \mathrm{B}_n \neq \varnothing} b_\epsilon(v)$, where B_n is the n-dimensional unit sphere. We use the $(n-1)$-dimensional volume (surface area) of this body to approximate that of B_n, with up to a $\sqrt{2}$ enlargement factor due to the fact that ϵ-boxes have sharp corners. Thus, the number of points that v can snap to is in the range $[n\mathrm{Vol}(\mathrm{B}_n)/\epsilon^{n-1}, \sqrt{2}n\mathrm{Vol}(\mathrm{B}_n)$

[17]A one-line proof is that this distribution is invariant under orthogonal transformations, but the uniform distribution on the n-dimensional unit sphere B_n is the unique distribution that satisfies this property. Although the Gaussian distributions we sample from are discretized, the invariance under orthogonal transformations holds approximately, so we have approximate uniformity.

$/\epsilon^{n-1}]$, which is also the range of $|\mathrm{L}(u, \epsilon')|$. To make the lines in $\mathrm{L}(u, \epsilon')$ cover every ϵ-box on the boundary of $\sqrt{n}\mathrm{B}_n$ (so that it is possible to sample all the points in K_ϵ), we need $\epsilon' \le \epsilon/\sqrt{n}$.

Let $L := |\mathrm{L}(u, \epsilon')|$. We label the lines in $\mathrm{L}(u, \epsilon')$ as $\{\ell_1, \dots, \ell_L\}$ (ordered arbitrarily). For each $i \in [L]$, let v_i be the point after normalization. Intuitively, v_i approximates a point on the "surface" of the unit ball around u (see Figure 3.6). There are *hyperfaces* of $b_{\epsilon'}(v_i)$ that are out-facing and not adjacent to any ϵ'-box in $\bigcup_{v:\, b_{\epsilon'}(v) \cap \mathrm{B}_n \neq \varnothing} b_\epsilon(v)$ (as an illustration, see dashed edges in Figure 3.6). For all points v'' in these hyperfaces, the line segments from u through v'' of length $\sqrt{n}$ form a set, which we refer to as a *hyperpyramid*, denoted by P_i. The apex of each hyperpyramid is u, and the base of each hyperpyramid is a subset of the hyperspherical surface. Intuitively, the bases of $\mathrm{P}_1, \dots, \mathrm{P}_L$ form a partition of the "surface" of the ball of radius $\sqrt{n}$ around u, and therefore $\{\mathrm{P}_1, \dots, \mathrm{P}_L\}$ forms a partition of the ball of radius $\sqrt{n}$ around u.

3.5.2 Conductance lower bound on the discretized hit-and-run walk

The discretized hit-and-run walk on K_ϵ described above can be summarized as Algorithm 10.

Algorithm 10: One step of the discretized hit-and-run walk.

```
  Input: Current point $u \in \mathrm{K}_\epsilon$.
1 Uniformly sample a line $\ell \in \mathrm{L}(u, \epsilon)$ by independently sampling $n$ coordinates around
    $u$ according to the standard normal distribution and then normalizing to unit length;
2 Sample a point $v'$ in $\ell(\overline{\mathrm{K}}_\epsilon, u, \epsilon')$ according to $f$;
3 Let $v'' \in \mathrm{K}_{\sqrt{\epsilon}n^{1/4}}$ that is closest to $v'$;
4 Output a uniform sample $v$ in $b_{\sqrt{\epsilon}n^{1/4}}(v'') \cap (\mathbb{R}^n)_\epsilon$;
```

Note that we have used a two-level discretization of K, as illustrated in Figure 3.7. The first level is a coarser discretization $\mathrm{K}_{\sqrt{\epsilon}n^{1/4}}$ and the second level is a finer discretization K_ϵ. We first choose a temporary point v'' in $\mathrm{K}_{\sqrt{\epsilon}n^{1/4}}$. Then we choose a point v uniformly at random in $b_{\sqrt{\epsilon}n^{1/4}}(v'') \cap (\mathbb{R}^n)_\epsilon$ to jump to. The purpose of this two-level discretization is to avoid having a

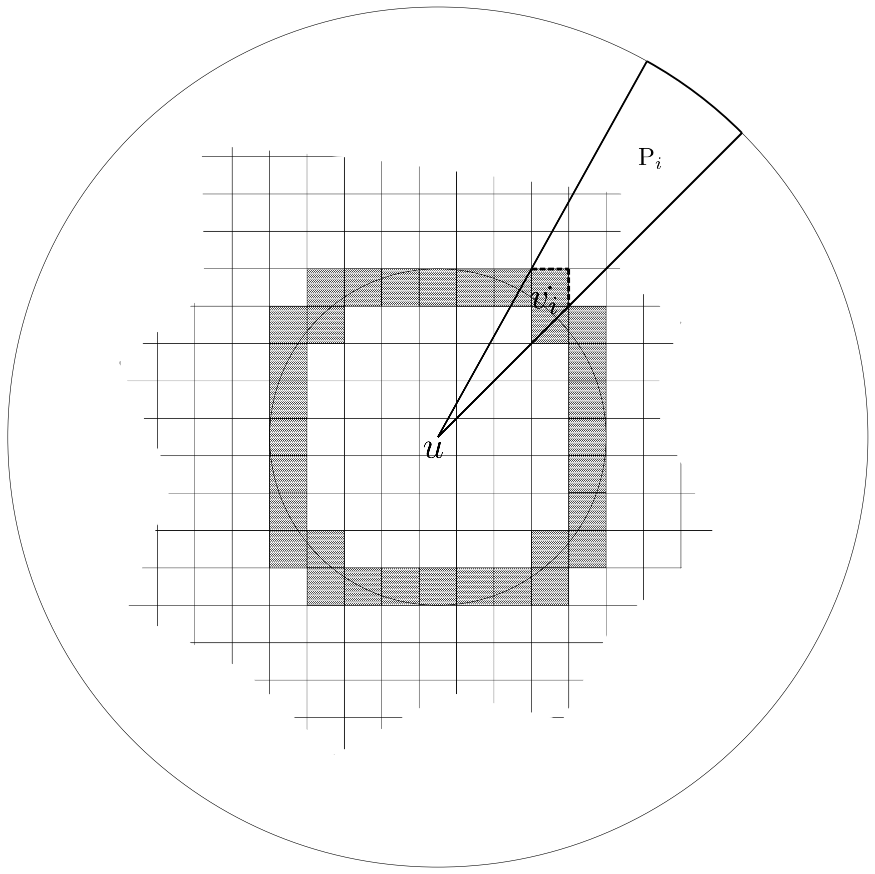


Figure 3.6: Constructing a hyperpyramid. The inner circle represents the unit ball and the outer circle represents the ball of radius $\sqrt{n}$. The grids represents the ϵ'-discretization of $\mathbb{R}^n$; each grid is an ϵ'-box. The shaded boxes are points where a direction "snaps" to after normalization, and the dashed edges of $b_{\epsilon'}(v_i)$ is its "outer face." The hyperpyramid P_i is represented by a circular sector.

small change of the original point u cause a huge difference in $\ell_{uv}(\overline{\mathrm{K}}_\epsilon)$ (when u is very close to the boundary).

We first compute the transition probability of the discretized hit-and-run walk.

Lemma 3.5.1. *The transition probabilities defined by Algorithm 10 satisfy*

$$P_{uv} \geq \sum_{\substack{v' \in \ell(\overline{\mathrm{K}}_\epsilon u, \epsilon'), \ell \in \mathrm{L}(x,\epsilon): \\ \ell(\overline{\mathrm{K}}_\epsilon, u, \epsilon') \cap b_{\sqrt{\epsilon} n^{1/4}}(v) \neq \varnothing}} \frac{\epsilon^{n-1}(\sqrt{\epsilon})^n f(v')}{\sqrt{2} n^{1+n/4} \mathrm{Vol}(\mathrm{B}_n)(\sqrt{n})^{n-1} \hat{\mu}_f(\ell(\overline{\mathrm{K}}_\epsilon, u, \epsilon'))}, \tag{3.156}$$

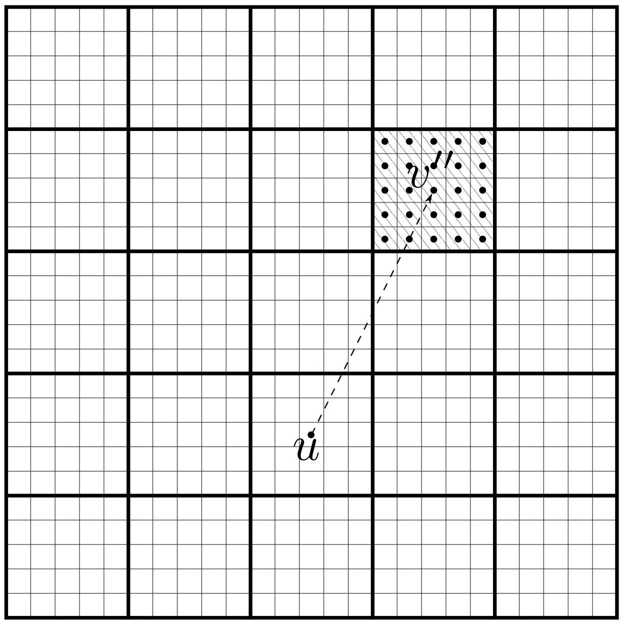


Figure 3.7: A demonstration of the 2-level discretization of K. The thicker grid represents the coarser discretization $\mathrm{K}_{\sqrt{\epsilon}n^{1/4}}$ and the thinner grid represents the finer discretization K_ϵ. When v'' is chosen from $\mathrm{K}_{\sqrt{\epsilon}n^{1/4}}$, an actual point v to jump is chosen uniformly at random in $b_{\sqrt{\epsilon}n^{1/4}}(v'') \cap (\mathbb{R}^n)_\epsilon$ marked by the points in the shaded region.

where for any $\mathrm{S} \subseteq \mathbb{R}^n$*, we define*

$$\hat{\mu}_f(\mathrm{S}) := \sum_{x \in \mathrm{S}} f(x). \tag{3.157}$$

Proof. First note that the probability of a line $\ell \in \mathrm{L}(u, \epsilon)$ being sampled is at least $\frac{\epsilon^{n-1}}{\sqrt{2}n\mathrm{Vol}(\mathrm{B}_n)(\sqrt{n})^{n-1}}$. Along ℓ, the probability of sampling v' is $f(v')/\hat{\mu}_f(\ell(\overline{\mathrm{K}}_\epsilon, u, \epsilon'))$, and the probability of choosing v in $b_{\sqrt{\epsilon}n^{1/4}}(v'') \cap \mathrm{K}_\epsilon$ is $(\sqrt{\epsilon})^n/n^{n/4}$. □

According to the definition in (3.12), the conductance of any subset $\mathrm{S} \subseteq \mathrm{K}_\epsilon$ is

$$\phi(\mathrm{S}) = \frac{\sum_{u \in \mathrm{S}} \sum_{v \in \mathrm{K}_\epsilon \setminus \mathrm{S}} P_{uv} \hat{\pi}_f(u)}{\min\{\hat{\pi}_f(\mathrm{S}), \hat{\pi}_f(\mathrm{K}_\epsilon \setminus \mathrm{S})\}}, \tag{3.158}$$

where $\hat{\pi}_f$ is defined as $\hat{\pi}_f(\mathrm{A}) = \sum_{x\in\mathrm{A}} f(x)$. The conductance of the Markov chain is then

$$\phi = \min_{\mathrm{S}\subseteq\mathrm{K}_\epsilon} \phi(\mathrm{S}). \tag{3.159}$$

Now we prove the main theorem of this section, which shows that the conductance of the discretized hit-and-run walk does not differ significantly from that of the continuous hit-and-run walk.

Theorem 3.5.1. *Let* K_ϵ *be the discretization of convex body* K *that contains a unit ball and is contained in a ball with radius* $R \le \sqrt{n}$*. Let the density function be* $f(x) = e^{-a^T x}$ *having support* K *where* $a = (1, 0, \dots, 0)$*. Let* $\epsilon' \le \sqrt{\epsilon} n^{-3/4}$*. For* $\mathrm{S} \subseteq \mathrm{K}_\epsilon$ *such that* $\hat{\pi}_f(\mathrm{S}) \le 1/2$*, we have*

$$\phi(\mathrm{S}) \ge \frac{1}{10^{16} n\sqrt{n}\ln(\frac{2n\sqrt{n}}{\hat{\pi}_f(\mathrm{S})})} - \epsilon. \tag{3.160}$$

Proof. This proof closely follows that of [72, Theorem 6.9]. We first consider the transition probability for the continuous hit-and-run walk in K. For $u, v \in \mathrm{K}$, recall that

$$P'_u(b_\epsilon(v)) = \frac{2}{n\mathrm{Vol}(\mathrm{B}_n)} \int_{b_\epsilon(v)} \frac{f(x)\,\mathrm{d}x}{\mu_f(u,x)|x-u|^{n-1}}, \tag{3.161}$$

where $\mu_f(u,x)$ is a shorthand for $\mu_f(\ell_{ux}(\overline{\mathrm{K}}_\epsilon))$. We compare $P'_u(b_{\sqrt{\epsilon}n^{1/4}}(v))$ with P_{uv} for $u \in \mathrm{K}_\epsilon$ and $v \in \mathrm{K}_{\sqrt{\epsilon}n^{1/4}}$. To this end, we use $\hat{\mu}_f$ to approximate μ_f: for each ℓ, we have

$$\epsilon'\hat{\mu}_f(\ell(\overline{\mathrm{K}}_\epsilon, u, \epsilon')) \le e^{\epsilon'}\mu_f(\ell(\overline{\mathrm{K}}_\epsilon)). \tag{3.162}$$

Consider each hyperpyramid P_i defined in Section 3.5.1 whose associated line through its apex is ℓ_i and $\ell_i(\overline{\mathrm{K}}_\epsilon, u, \epsilon') \cap b_{\sqrt{\epsilon}n^{1/4}}(v) \neq \varnothing$. Note that the distance between each $u \in \mathrm{K}_\epsilon$ and the boundary of $\overline{\mathrm{K}}_\epsilon$ is at least $\epsilon/2$. Inside each hyperpyramid, the length of the chords through u can differ by a factor at most 2. For each $\ell \subset \mathrm{P}_i$, $\hat{\mu}_f(\ell_i(\overline{\mathrm{K}}_\epsilon, u, \epsilon')) \leq 2e^{\epsilon'}\hat{\mu}_f(\ell(\overline{\mathrm{K}}_\epsilon, u, \epsilon'))$. Together with (3.162), it follows that

$$\epsilon'\hat{\mu}_f(\ell_i(\overline{\mathrm{K}}_\epsilon, u, \epsilon')) \leq 2e^{2\epsilon'}\mu_f(\ell(\overline{\mathrm{K}}_\epsilon)) \tag{3.163}$$

for all $\ell \subset \mathrm{P}_i$. Define $c_i := |\ell_i(\overline{\mathrm{K}}_\epsilon, u, \epsilon') \cap b_{\sqrt{\epsilon}n^{1/4}}(v)|$ (the number of points in this set) and $d_i := |\ell_i(\overline{\mathrm{K}}_\epsilon) \cap b_{\sqrt{\epsilon}n^{1/4}}(v)|$ (the length of this line). Note that $c_i \leq d_i/\epsilon'$. We further partition P_i into c_i sets $\mathrm{Q}_{i,1}, \ldots, \mathrm{Q}_{i,c_i}$ along the direction of ℓ_i so that the distance between the hyperplanes that separate adjacent sets is at most ϵ'. For each $j \in [c_i]$, we have

$$\begin{aligned} \frac{\epsilon^{n-1}f(v')}{n\mathrm{Vol}(\mathrm{B}_n)(\sqrt{n})^{n-1}\hat{\mu}_f(\ell_i(\overline{\mathrm{K}}_\epsilon, u, \epsilon'))} &= \frac{\epsilon^{n-1}f(v')\epsilon'|v'-u|^{n-1}}{\epsilon' n\mathrm{Vol}(\mathrm{B}_n)(\sqrt{n})^{n-1}\hat{\mu}_f(\ell_i(\overline{\mathrm{K}}_\epsilon, u, \epsilon'))|v'-u|^{n-1}} \\ &\geq \frac{f(v')\mathrm{Vol}(\mathrm{Q}_{i,j} \cap b_{\sqrt{\epsilon}n^{1/4}}(v))}{2\epsilon' n\mathrm{Vol}(\mathrm{B}_n)\hat{\mu}_f(\ell_i(\overline{\mathrm{K}}_\epsilon, u, \epsilon'))|v'-u|^{n-1}}, \end{aligned} \tag{3.164}$$

where we have used the fact that the distance between adjacent $\mathrm{Q}_{i,j}$ and $\mathrm{Q}_{i,j+1}$ can be bounded from below by $|\mathrm{Q}_{i,j} \cap \ell_i|/(1+\epsilon'/2) \geq |\mathrm{Q}_{i,j} \cap \ell_i|/2$.

Now we consider the integration in $\mathrm{Q}_{i,j} \cap b_{\sqrt{\epsilon}n^{1/4}}(v)$. We use $f(v)$ to approximate $f(v')$ which causes a relative error at most $e^{\sqrt{\epsilon}n^{1/4}}$, and use $|v'-u|^{n-1}$ to approximate $|x-u|^{n-1}$ for all $x \in \mathrm{Q}_{i,j} \cap b_{\sqrt{\epsilon}n^{1/4}}(v)$ which causes a relative error at most e provided $\epsilon' \leq \sqrt{\epsilon}n^{-3/4}$ (noting that the distance between x and u is at most $\sqrt{\epsilon}n^{1/4}$). We have

$$\begin{aligned}
&\int_{\mathrm{Q}_{i,j}\cap b_{\sqrt{\epsilon}n^{1/4}}(v)} \frac{f(x)\,\mathrm{d}x}{n\mathrm{Vol}(\mathrm{B}_n)\mu_f(u,x)|x-u|^{n-1}} \\
&\qquad \le \frac{2e^{\sqrt{\epsilon}n^{1/4}+2\epsilon'+1}f(v')}{\epsilon' n\mathrm{Vol}(\mathrm{B}_n)\hat{\mu}_f(\ell_i(\overline{\mathrm{K}}_\epsilon,u,\epsilon'))|v'-u|^{n-1}} \int_{\mathrm{Q}_{i,j}\cap b_{\sqrt{\epsilon}n^{1/4}}(v)} \mathrm{d}x && (3.165) \\
&\qquad = \frac{2e^{\sqrt{\epsilon}n^{1/4}+2\epsilon'+1}f(v')\mathrm{Vol}(\mathrm{Q}_{i,j}\cap b_{\sqrt{\epsilon}n^{1/4}}(v))}{\epsilon' n\mathrm{Vol}(\mathrm{B}_n)\hat{\mu}_f(\ell_i(\overline{\mathrm{K}}_\epsilon,u,\epsilon'))|v'-u|^{n-1}}, && (3.166)
\end{aligned}$$

where the inequality follows from (3.163). Let $i_1,\dots,i_t$ be the indices such that $\mathrm{P}_{i_j}\cap b_{\sqrt{\epsilon}n^{1/4}}(v)\neq\varnothing$ for $j\in[t]$. We use $\bigcup_{j\in[t]}\mathrm{P}_{i_j}\cap b_{\sqrt{\epsilon}n^{1/4}}(v)$ as a partition to approximate $b_{\sqrt{\epsilon}n^{1/4}}(v)$, which causes a relative error at most $(1+\epsilon)^n$ for $\mathrm{Vol}(b_{\sqrt{\epsilon}n^{1/4}}(v))$. We have

$$\int_{b_{\sqrt{\epsilon}n^{1/4}}(v)} \frac{f(x)\,\mathrm{d}x}{\mu_f(u,x)|x-u|^{n-1}} \le (1+\epsilon)^n \sum_{j\in[t]} \int_{b_{\sqrt{\epsilon}n^{1/4}}(v)\cap\mathrm{P}_{i_j}} \frac{f(x)\,\mathrm{d}x}{\mu_f(u,x)|x-u|^{n-1}}.$$

Hence,

$$\begin{aligned}
&\frac{(\sqrt{\epsilon})^n}{n^{n/4}} P'_u(b_{\sqrt{\epsilon}n^{1/4}}(v)) \\
&= \frac{2(\sqrt{\epsilon})^n}{n^{1+n/4}\mathrm{Vol}(\mathrm{B}_n)} \int_{b_{\sqrt{\epsilon}n^{1/4}}(v)} \frac{f(x)\,\mathrm{d}x}{\mu_f(u,x)|x-u|^{n-1}} && (3.167)\\
&\le \frac{2(\sqrt{\epsilon})^n(1+\epsilon)^n}{n\mathrm{Vol}(\mathrm{B}_n)} \sum_{j\in[t]} \int_{b_{\sqrt{\epsilon}n^{1/4}}(v)\cap \mathrm{P}_{i_j}} \frac{f(x)\,\mathrm{d}x}{\mu_f(u,x)|x-u|^{n-1}} && (3.168)\\
&= \frac{2(\sqrt{\epsilon})^n(1+\epsilon)^n}{n\mathrm{Vol}(\mathrm{B}_n)} \sum_{j\in[t]} \sum_{k\in[c_{i_j}]} \int_{b_{\sqrt{\epsilon}n^{1/4}}(v)\cap \mathrm{Q}_{i_j,k}} \frac{f(x)\,\mathrm{d}x}{\mu_f(u,x)|x-u|^{n-1}} && (3.169)\\
&\le \sum_{j\in[t]} \sum_{k\in[c_{i_j}]} \frac{4(1+\epsilon)^n e^{\sqrt{\epsilon}n^{1/4}+2\epsilon'+1}(\sqrt{\epsilon})^n f(v')\mathrm{Vol}(\mathrm{Q}_{i,j}\cap b_{\sqrt{\epsilon}n^{1/4}}(v))}{\epsilon' n\mathrm{Vol}(\mathrm{B}_n)\hat{\mu}_f(\ell_{i_j}(\overline{\mathrm{K}}_\epsilon,u,\epsilon'))|u-v'|^{n-1}} && (3.170)\\
&\le 4(1+\epsilon)^n e^{\sqrt{\epsilon}n^{1/4}+2\epsilon'+1} \sum_{j\in[t]} \sum_{k\in[c_{i_j}]} \frac{2\epsilon^{n-1}(\sqrt{\epsilon})^n f(v')}{n\mathrm{Vol}(\mathrm{B}_n)(\sqrt{n})^{n-1}\hat{\mu}(\ell_{i_j}(\overline{\mathrm{K}}_\epsilon,u,\epsilon'))} && (3.171)\\
&= 4(1+\epsilon)^n e^{\sqrt{\epsilon}n^{1/4}+2\epsilon'+1} P_{uv} \le e^{5+2\epsilon'} P_{uv}, && (3.172)
\end{aligned}$$

where the last inequality holds when $\epsilon \le 1/n$.

For $u \in \mathrm{K}_\epsilon$ and $v \in \mathrm{K}_{\sqrt{\epsilon}n^{1/4}}$, we approximate $\int_{x\in b_\epsilon(u)} P'_u(b_{\sqrt{\epsilon}n^{1/4}}(v))\,\mathrm{d}\pi_f(x)$ by $\epsilon^n P_{uv}$. Note that for all $u' \in b_\epsilon(u)$, we have $|u'-v|^n \le e|u-v|^n$. Also, the lengths of ℓ_{uv} and $\ell_{u'v}$ can differ by at most a factor of 2. As a result, $\hat{\pi}_f(\ell_{u'v}(\overline{\mathrm{K}}_\epsilon,u,\epsilon') \le 2\hat{\pi}_f(\ell_{uv}(\overline{\mathrm{K}}_\epsilon,u',\epsilon')$. It follows that $P_{uv} \ge P_{u'v}/(2e)$. Therefore,

$$\begin{aligned}
\int_{x\in b_\epsilon(u)} P'_u(b_{\sqrt{\epsilon}n^{1/4}}(v))\,\mathrm{d}\pi_f(x) &\le \int_{x\in b_\epsilon(u)} \frac{2e^{5+2\epsilon'}n^{n/4}}{(\sqrt{\epsilon})^n} P_{xv}\,\mathrm{d}\pi_f(x) && (3.173)\\
&\le \frac{2e^{5+2\epsilon'+\epsilon}n^{n/4}}{(\sqrt{\epsilon})^n} P_{uv}\hat{\pi}_f(u)\epsilon^n. && (3.174)
\end{aligned}$$

Next, for the relationship between $\hat{\pi}_f$ and π_f, we consider the sets $\overline{\mathrm{K}}_\epsilon \cap \mathrm{K}$, $\overline{\mathrm{K}}_\epsilon \setminus \mathrm{K}$, and

$\mathrm{K} \setminus \overline{\mathrm{K}}_\epsilon$ separately. Without loss of generality, assume $\hat{\pi}_f(\mathrm{S}) \le \hat{\pi}_f(\mathrm{K}_\epsilon \setminus \mathrm{S})$. We partition S as $\mathrm{S}_1 \cup \mathrm{S}_2$, where $\mathrm{S}_1 = \{x \in \mathrm{S} : b_\epsilon(x) \subseteq \mathrm{K}\}$ and $\mathrm{S}_2 = \mathrm{S} \setminus \mathrm{S}_1$. We also define $\overline{\mathrm{S}}_1 := \bigcup_{x \in \mathrm{S}_1} b_\epsilon(x)$ and $\overline{\mathrm{S}}_2 := \bigcup_{x \in \mathrm{S}_2} b_\epsilon(x)$. For $\overline{\mathrm{S}}_1$, we use $f(v)$ to approximate $f(x)$ for all $x \in b_\epsilon(v)$; it follows that

$$\hat{\pi}_f(\mathrm{S}_1) \le e^{2\epsilon} \pi_f(\overline{\mathrm{S}}_1) \quad \text{and} \quad \pi_f(\overline{\mathrm{S}}_1) \le e^{2\epsilon} \hat{\pi}_f(\mathrm{S}_1). \tag{3.175}$$

For S_2, we have

$$\hat{\pi}_f(\mathrm{S}_2) \le 2e^{2\epsilon} \pi_f(\overline{\mathrm{S}}_2 \cap \mathrm{K}), \tag{3.176}$$

so

$$\begin{aligned} \hat{\pi}_f(\mathrm{S}) &= \hat{\pi}_f(\mathrm{S}_1) + \hat{\pi}_f(\mathrm{S}_2) && (3.177) \\ &\le e^{2\epsilon} \pi_f(\overline{\mathrm{S}}_1) + 2e^{2\epsilon} \pi_f(\overline{\mathrm{S}}_2 \cap \mathrm{K}) \le 3\pi_f(\mathrm{K} \cap \overline{\mathrm{S}}). && (3.178) \end{aligned}$$

Now we bound the numerator of the conductance: $\sum_{u \in \mathrm{S}} \sum_{v \in \mathrm{K}_\epsilon \setminus \mathrm{S}} P_{uv} \hat{\pi}_f(u)$. For $u \in \mathrm{S}$ and $v \in \mathrm{K} \setminus \mathrm{S}$, we consider four cases. First, when $b_\epsilon(u), b_\epsilon(v) \subseteq \mathrm{K}$, we have

$$P_{uv} \hat{\pi}_f(u) \ge \frac{(\sqrt{\epsilon})^n}{2e^{5+2\epsilon'+\epsilon} n^{n/4}} \int_{x \in b_\epsilon(u)} P'_u(b_{\sqrt{\epsilon} n^{1/4}}(v)) \, \mathrm{d}\pi_f(x). \tag{3.179}$$

Second, when $b_\epsilon(u) \subseteq \mathrm{K}$ and $b_\epsilon(v) \nsubseteq \mathrm{K}$, we have

$$P_{uv}\hat{\pi}_f(u) \geq \frac{(\sqrt{\epsilon})^n}{2e^{5+2\epsilon'+\epsilon}n^{n/4}} \int_{x \in b_\epsilon(u)} P'_u(b_{\sqrt{\epsilon}n^{1/4}}(v)) \,\mathrm{d}\pi_f(x) \tag{3.180}$$

$$\geq \frac{(\sqrt{\epsilon})^n}{2e^{5+2\epsilon'+\epsilon}n^{n/4}} \int_{x \in b_\epsilon(u)} P'_u(b_{\sqrt{\epsilon}n^{1/4}}(v) \cap \mathrm{K}) \,\mathrm{d}\pi_f(x). \tag{3.181}$$

Third, when $b_\epsilon(u) \nsubseteq \mathrm{K}$ and $b_\epsilon(v) \subseteq \mathrm{K}$, we have

$$P_{uv}\hat{\pi}_f(u) \geq \frac{(\sqrt{\epsilon})^n}{2e^{5+2\epsilon'+\epsilon}n^{n/4}} \int_{x \in b_\epsilon(u)} P'_u(b_{\sqrt{\epsilon}n^{1/4}}(v)) \,\mathrm{d}\pi_f(x) \tag{3.182}$$

$$\geq \frac{(\sqrt{\epsilon})^n}{2e^{5+2\epsilon'+\epsilon}n^{n/4}} \int_{x \in b_\epsilon(u) \cap \mathrm{K}} P'_u(b_{\sqrt{\epsilon}n^{1/4}}(v)) \,\mathrm{d}\pi_f(x). \tag{3.183}$$

Fourth, when $b_\epsilon(u) \nsubseteq \mathrm{K}$ and $b_\epsilon(v) \nsubseteq \mathrm{K}$, we have

$$P_{uv}\hat{\pi}_f(u) \geq \frac{(\sqrt{\epsilon})^n}{2e^{5+2\epsilon'+\epsilon}n^{n/4}} \int_{x \in b_\epsilon(u)} P'_u(b_{\sqrt{\epsilon}n^{1/4}}(v)) \,\mathrm{d}\pi_f(x) \tag{3.184}$$

$$\geq \frac{(\sqrt{\epsilon})^n}{2e^{5+2\epsilon'+\epsilon}n^{n/4}} \int_{x \in b_\epsilon(u) \cap \mathrm{K}} P'_u(b_{\sqrt{\epsilon}n^{1/4}}(v) \cap \mathrm{K}) \,\mathrm{d}\pi_f(x). \tag{3.185}$$

We also need to consider the set $\mathrm{K} \setminus \overline{\mathrm{K}}_\epsilon$. There exists a small subset $\mathrm{E} \subseteq \mathrm{K} \setminus \overline{\mathrm{K}}_\epsilon$ such that $\pi_f(\mathrm{E}) \leq \epsilon\pi_f(\mathrm{S})$. We need to consider the transition from E to $\subseteq \mathrm{K} \setminus \overline{\mathrm{K}}_\epsilon \setminus \mathrm{E}$: we have $\int_{x \in \mathrm{E}} P'_x(\mathrm{K} \setminus \overline{\mathrm{K}}_\epsilon \setminus \mathrm{E}) \,\mathrm{d}\pi_f(x) \leq \pi_f(\mathrm{E}) \leq \epsilon\pi_f(\mathrm{S})$. Putting everything together, we have

$$\sum_{u \in \mathrm{S}} \sum_{v \in \mathrm{K}_\epsilon \setminus \mathrm{S}} P_{uv}\hat{\pi}_f(u) + \int_{x \in \mathrm{E} \cap \mathrm{K}} P'_x(\mathrm{K} \setminus \overline{\mathrm{K}}_\epsilon \setminus \mathrm{E}) \,\mathrm{d}\pi_f(x)$$

$$\geq \frac{1}{2e^{5+2\epsilon'+\epsilon}} \int_{x \in \overline{\mathrm{S}} \cap \mathrm{K} \cup \mathrm{E}} P'_x(\mathrm{K} \setminus (\overline{\mathrm{S}} \cap \mathrm{K} \cup \mathrm{E})) \,\mathrm{d}\pi_f(x), \tag{3.186}$$

which further implies that

$$\sum_{u\in \mathrm{S}}\sum_{v\in \mathrm{K}_\epsilon\setminus \mathrm{S}} P_{uv}\hat{\pi}_f(u) \geq \frac{1}{2e^{5+2\epsilon'+\epsilon}}\int_{x\in\overline{\mathrm{S}}\cap\mathrm{K}\cup\mathrm{E}} P'_x(\mathrm{K}\setminus(\overline{\mathrm{S}}\cap\mathrm{K}\cup\mathrm{E}))\,\mathrm{d}\pi_f(x) - \epsilon\pi_f(\mathrm{S})$$

$$\geq \frac{1}{2e^{5+2\epsilon'+\epsilon}}\int_{x\in\overline{\mathrm{S}}\cap\mathrm{K}\cup\mathrm{E}} P'_x(\mathrm{K}\setminus(\overline{\mathrm{S}}\cap\mathrm{K}\cup\mathrm{E}))\,\mathrm{d}\pi_f(x) - \epsilon e^{\epsilon}\hat{\pi}_f(\mathrm{S}). \quad (3.187)$$

By Proposition 3.2.3, we have

$$\phi(\mathrm{S}) = \frac{\sum_{u\in\mathrm{S}}\sum_{v\in\mathrm{K}_\epsilon\setminus\mathrm{S}} P_{uv}\hat{\pi}_f(u)}{\hat{\pi}_f(\mathrm{S})} \quad (3.188)$$

$$\geq \frac{1}{2e^{5+2\epsilon'+\epsilon}}\frac{\int_{x\in\overline{\mathrm{S}}\cap\mathrm{K}\cup\mathrm{E}} P'_x(\mathrm{K}\setminus(\overline{\mathrm{S}}\cap\mathrm{K}\cup\mathrm{E}))\,\mathrm{d}\pi_f(x)}{\hat{\pi}_f(\mathrm{S})} - \frac{\epsilon}{2e^{4+2\epsilon'}} \quad (3.189)$$

$$\geq \frac{1}{6e^{5+2\epsilon'+\epsilon}}\frac{\int_{x\in\overline{\mathrm{S}}\cap\mathrm{K}\cup\mathrm{E}} P'_x(\mathrm{K}\setminus(\overline{\mathrm{S}}\cap\mathrm{K}\cup\mathrm{E}))\,\mathrm{d}\pi_f(x)}{\pi_f(\overline{\mathrm{S}}\cap\mathrm{K})+\pi_f(\mathrm{E})} - \frac{\epsilon}{2e^{5+2\epsilon'}} \quad (3.190)$$

$$\geq \frac{1}{10^{14}e^{5+2\epsilon'+\epsilon}n\sqrt{n}\ln(\frac{n\sqrt{n}}{\pi_f(\overline{\mathrm{S}}\cap\mathrm{K})})} - \frac{\epsilon}{2e^{5+2\epsilon'}} \quad (3.191)$$

$$\geq \frac{1}{10^{14}e^{5+2\epsilon'+\epsilon}n\sqrt{n}\ln(\frac{n\sqrt{n}}{(1-e^{-\epsilon}/2)e^{\epsilon}\hat{\pi}_f(\mathrm{S})})} - \frac{\epsilon}{2e^{5+2\epsilon'}}, \quad (3.192)$$

where the third inequality follows from (3.178). The above inequality can then be simplified to

$$\phi(\mathrm{S}) \geq \frac{1}{10^{16}n\sqrt{n}\ln(\frac{2n\sqrt{n}}{\hat{\pi}_f(\mathrm{S})})} - \epsilon, \quad (3.193)$$

which is exactly the claim in Theorem 3.5.1. □

The mixing time for the discrete hit-and-run walk can be bounded by the following corollary.

Corollary 3.5.1. *Let* K_ϵ *be the discretization of convex body* K *that contains a unit ball and* *is contained in a ball with radius $R \leq \sqrt{n}$. Let the density function be $f(x) = e^{-a^T x}$ having support* K *where $a = (1, 0, \ldots, 0)$. Let $\epsilon' \leq \sqrt{\epsilon} n^{-3/4}$. Let the initial distribution be σ and the distribution after m steps be σ^m. If $\sum_{x \in \mathrm{K}_\epsilon} \frac{\sigma(x)}{\hat{\pi}_f(x)} \sigma(x) \leq M$ then, after*

$$m \geq 10^{33} n^3 \ln^2 \frac{Mn\sqrt{n}}{\epsilon} \ln \frac{M}{\epsilon} \tag{3.194}$$

steps, we have $d_{\mathrm{TV}}(\sigma^m, \hat{\pi}_f) \leq \epsilon$.

Proof. First note that, since $\sum_{x \in \mathrm{K}_\epsilon} \frac{\sigma(x)}{\hat{\pi}_f(x)} \sigma(x) \leq M$, the set $\mathrm{S} = \{x : \frac{\sigma(x)}{\hat{\pi}_f(X)} > \frac{2M}{\epsilon}\}$ has measure $\sigma(\mathrm{S}) \leq \epsilon/2$. Then a random point in K_ϵ can be thought of as being generated with probability $1 - \epsilon/2$ from a distribution σ' satisfying $\frac{\sigma'(\mathrm{S}')}{\hat{\pi}_f(\mathrm{S}')} \leq 2M/\epsilon$ for any subset $\mathrm{S}' \subseteq \mathrm{K}_\epsilon$ and with probability $\epsilon/2$ from some other distribution. As a consequence of Theorem 3.5.1, for any such subset S' with $\hat{\pi}_f(\mathrm{S}') = p$, the conductance of S' is at least

$$\Phi_p = \frac{1}{10^{16} n\sqrt{n} \ln(2n\sqrt{n}/p)} - \epsilon. \tag{3.195}$$

For the purpose of analysis, we use $p = \frac{\epsilon^2}{8M}$. When ϵ is reasonably small (say, $\epsilon \leq \frac{1}{2 \cdot 10^{16} n\sqrt{n} \ln(Mn\sqrt{n}/\epsilon)}$), the ϵ term in the conductance bound can be ignored with an additional $1/2$ factor. Then we have $\Phi_p \geq \frac{1}{2 \cdot 10^{16} n\sqrt{n} \ln(2n\sqrt{n}/p)}$. By the condition that $\sigma'(\mathrm{S}') \leq (2M/\epsilon)\hat{\pi}_f(\mathrm{S}')$, as well as the way a random point in K_ϵ is generated, Proposition 3.2.2 implies that

$$d_{\mathrm{TV}}(\sigma^{(m)}, \hat{\pi}_f) \leq \frac{\epsilon}{2} + \left(1 - \frac{\epsilon}{2}\right) \left(\frac{\epsilon}{2} + \frac{4M}{\epsilon} \left(1 - \frac{\Phi_p^2}{2}\right)^m \right). \tag{3.196}$$

Therefore, after the claimed number of steps, the total variation distance is at most ϵ. □

As the uniform distribution is a special case of a log-concave distribution, the proof of Theorem 3.5.1 also applies to this case. More specifically, we use Proposition 3.2.4 in (3.191), which yields the following stronger corollary.

Corollary 3.5.2. *Let* K_ϵ *be the discretization of a convex body* K *that contains a unit ball and is contained in a ball with radius* $R \leq \sqrt{n}$*. Let* $\epsilon' \leq \sqrt{\epsilon} n^{-3/4}$*. The conductance of the hit-and-run walk in* K_ϵ *with uniform distribution satisfies*

$$\phi \geq \frac{1}{2^{26} n \sqrt{n}} - \epsilon. \tag{3.197}$$

Note that Corollary 3.5.2 is stronger than Theorem 3.5.1 because (3.197) is independent of $\mathrm{S} \subseteq \mathrm{K}_\epsilon$. This corollary is informative and is not used in this paper.

3.5.3 Implementing the quantum walk operators

We now describe how to implement the discretized quantum walk. Following (3.2), consider a convex body K such that $\mathrm{B}_2(0, r) \subseteq \mathrm{K} \subseteq \mathrm{B}_2(0, R)$. Each stage of the volume estimation algorithm involves a hit-and-run walk over the convex body with target density e^{-ax_0}. In order to use techniques from [78] to obtain a speedup in mixing time, we implement the quantum walk operator W corresponding to an ϵ-discretized version of this walk Algorithm 10.

Let $|x\rangle$ be the register for the state of the walk, and U be a unitary that satisfies $U|x\rangle|0\rangle = |x\rangle|p_x\rangle$ for all $|x\rangle$ (recall that $|p_x\rangle = \sum_{y \in \mathrm{K}_\epsilon} \sqrt{p_{x \to y}}|y\rangle$ where $p_{x \to y}$ is the probability of a transition from x to y). Since the state of the hit-and-run walk is given by points on an ϵ-grid that can be restricted to $\mathrm{B}_2(0, R)$, there are $\left(\frac{2R}{\epsilon}\right)^n$ possible values of x and thus $|x\rangle$ can be represented using $n \log\left(\frac{2R}{\epsilon}\right)$ qubits. In the rest of the section, we abuse notation by letting x refer to both a point on the grid and its corresponding bit representation. Then the quantum walk operator [78] can be realized as

$$W' = U^\dagger S U R_{\mathcal{A}} U^\dagger S U R_{\mathcal{A}} \tag{3.198}$$

where $R_{\mathcal{A}}$ is the reflection around the subspace $\mathcal{A} = \operatorname{span}\{|x\rangle|0\rangle \mid x \in \mathrm{K}_\epsilon\}$ and S is the swap operator. It thus remains to implement the operator U.

Continuous case We first explain a continuous version of the implementation before explaining how it can be discretized. Given an input $|x\rangle$, consider n real ancilla registers, each in the state $\int_0^1 |z\rangle\, \mathrm{d}z$. Given a pair of uniformly distributed random variables ξ_1, ξ_2, the Box-Muller transform

$$\phi_1 = \sqrt{-2\delta^2 \ln \xi_1}\cos 2\pi\xi_2 \tag{3.199}$$

$$\phi_2 = \sqrt{-2\delta^2 \ln \xi_1}\sin 2\pi\xi_2 \tag{3.200}$$

yields two variables ϕ_1, ϕ_2 that are distributed according to a univariate normal distribution with mean 0 and variance δ^2. Thus applying the unitary mapping

$$|\xi_1\rangle|\xi_2\rangle \mapsto \left|\sqrt{-4\ln\xi_1}\cos 2\pi\xi_2\right\rangle\left|\sqrt{-4\ln\xi_1}\sin 2\pi\xi_2\right\rangle \tag{3.201}$$

to $\int_0^1 |z\rangle\, \mathrm{d}z \otimes \int_0^1 |z\rangle\, \mathrm{d}z$ yields the state $\int_{\mathbb{R}} \frac{1}{\sqrt{4\pi}} e^{-z^2/4}|z\rangle\, \mathrm{d}z \otimes \int_{\mathbb{R}} \frac{1}{\sqrt{4\pi}} e^{-z^2/4}|z\rangle\, \mathrm{d}z$. With n such registers, we have the state

$$\int_{\mathbb{R}^n} \frac{1}{\sqrt{4\pi}} e^{-(\sum_{i=1}^n z_i^2/4)}|z\rangle\, \mathrm{d}z. \tag{3.202}$$

We now compute the unit vector (direction) corresponding to each x in a different ancilla register, and uncompute the Gaussian registers. Since $\frac{1}{\sqrt{4\pi}}e^{-(\sum_{i=1}^{n} x_i^2/4)}$ is independent of the direction of the vector z, we obtain a uniform distribution over all the directions on the n-dimensional sphere $\mathcal{S}^n$ given by

$$\sqrt{\frac{n\pi^{n/2}}{\Gamma\left(n+\frac{1}{2}\right)}}\int_{\mathcal{S}^n}|u\rangle\,\mathrm{d}u. \tag{3.203}$$

Corresponding to each direction u, the line $\{x+tu : t\in\mathbb{R}\}$ intersects the convex body K at two points with parameters t_1, t_2. These points as well as the length $l(u)=|t_1-t_2|$ can be determined within error ϵ using $O(\log\frac{1}{\epsilon})$ calls to the membership oracle. We must now map each direction $|u\rangle$ to a superposition proportional to $\int_{t_1}^{t_2} e^{a^T(x+tu)/2}|x+tu\rangle\,\mathrm{d}t = \int_{t_1}^{t_2} e^{a_0(x_0+tu_0)/2}|x+tu\rangle\,\mathrm{d}t$. Since the exponential distribution is efficiently integrable, this can be easily effected by making a variable change starting from the state $\int_0^1|z\rangle\,\mathrm{d}z$. The normalization factor is

$$A := \sqrt{\frac{a_0u_0}{e^{-a_0x_0}\left(e^{-a_0t_1}-e^{-a_0t_2}\right)}}. \tag{3.204}$$

Consider the variable change $f\colon[0,1]\to[t_1,t_2]$ such that $\frac{\mathrm{d}f^{-1}(t)}{\mathrm{d}t}=Ae^{a_0(x_0+tu_0)/2}$, $f(0)=t_1$, $f(1)=t_2$. Applying f to $\int_0^1|z\rangle\,\mathrm{d}z$ produces $\int_{t_1}^{t_2}Ae^{a_0(x_0+tu_0)/2}|t\rangle\,\mathrm{d}t$, which can be transformed to $\int_{t_1}^{t_2}e^{a_0(x_0+tu_0)/2}|x+tu\rangle\,\mathrm{d}t$ with an operation controlled on the input register x. This produces the appropriate superposition over points corresponding to each direction.

Discrete case The operator U can be implemented in a discrete setting using a similar process to the continuous case with two main changes:

- Instead of a continuous uniform variable $\int_0^1|z\rangle\,\mathrm{d}z$ we use a discrete uniform distribution.

We can create a uniform distribution on a grid with spacing ϵ as follows. We take n sets of ancilla registers, each consisting of $\log(1/\epsilon)$ registers initialized to the state 0. We apply Hadamard gates to each of these registers, giving the superposition $\bigotimes_{i=1}^{n} \sqrt{\epsilon} \sum_{z_i=0}^{1/\epsilon-1} |z_i\rangle$. Each $|z\rangle$ can be mapped to $|z\epsilon\rangle$, producing the required uniform distribution over the grid.

- Applying a bijective mapping to a discrete uniform distribution simply relabels the states, so the change of variable methods used in the continuous setting cannot be used to construct the Gaussian and exponential superpositions. We use instead the Grover-Rudolph method [121] that prepares states with amplitudes corresponding to efficiently integrable probability distributions. Exponential distributions can be analytically integrated, and an n-dimensional Gaussian variable is a product of n univariate standard normal distributions, each of which can be efficiently integrated by Monte Carlo methods.

Given a point $u \in K_\epsilon$ and a line $l(u,\epsilon)$ to be approximately uniformly sampled, we determine the range of points in $l(\overline{\mathrm{K}}_\epsilon, u, \epsilon')$ using binary search with the membership oracle and prepare an exponential superposition as described above. We apply a unitary mapping to compute the closest point $v'' \in \mathrm{K}_{\sqrt{\epsilon}n^{1/4}}$. Finally, corresponding to each point v'', we generate a uniform distribution over an ϵ grid in $b_{\sqrt{\epsilon}n^{1/4}} \cap (\mathbb{R}_n)_\epsilon$ by applying the Hadamard transform to $\log(n^{1/4}/\sqrt{\epsilon})$ qubits.

Overall, this implementation of the discretized quantum hit-and-run walk operator gives the following.

Theorem 3.5.2. *The gate complexity of implementing an operator $\tilde{U}$ such that $\|\tilde{U} - U\| = O(\epsilon)$ where $U|x\rangle|0\rangle = |x\rangle \sum_{y\in \mathrm{K}_\epsilon} \sqrt{p_{x\to y}}|y\rangle$ is $\tilde{O}\left(n \log\left(\frac{1}{\epsilon}\right)\right)$. The correspondsing quantum walk operator W can be implemented using a constant number of calls to U.*

3.6 Quantum lower bounds for volume estimation

3.6.1 A quantum lower bound in n

In this subsection, we prove the following quantum query lower bound in n for volume estimation:

Theorem 3.6.1. *Suppose $0 < \epsilon < \sqrt{2} - 1$. Estimating the volume of K with multiplicative precision ϵ requires $\Omega(\sqrt{n})$ quantum queries to the membership oracle O_{K} defined in (4.8).*

Proof. We prove Theorem 3.6.1 by reduction from the Hamming weight problem. In [92] by Nayak and Wu, it is shown that if we are given an oracle $O_s\colon |i, b\rangle \mapsto |i, b \oplus s_i\rangle$ for an input n-bit string $s = (s_1, \ldots, s_n) \in \{0,1\}^n$, and given the promise that the Hamming weight of s is either 0 or 1, it takes $\Omega(\sqrt{n})$ quantum queries to decide which is the case.

To establish an $\Omega(\sqrt{n})$ lower bound for volume estimation, for an n-bit string $s \in \{0,1\}^n$ with Hamming weight $|s|_{\text{Ham}} \leq 1$, we consider the convex body $\mathrm{K} = \bigtimes_{i=1}^{n} [0, 2^{s_i}]$. The volume of K is $2^{|s|_{\text{Ham}}} \in \{1, 2\}$, and membership in K is determined by the function

$$\textsc{mem}_s(x) := \begin{cases} 1 & \text{if for each } i \in [n], 0 \leq x_i \leq 2^{s_i}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.205}$$

The corresponding membership oracle O_{K} (defined in (4.8)) can be simulated by querying O_s using Algorithm 11.

We now prove that for any positive integer k and $s \in \{0,1\}^n$ with $|s|_{\text{Ham}} \leq 1$, if there is a k-query algorithm that computes the volume with access to $\textsc{mem}_s$, then there is a k-query algorithm

Algorithm 11: Simulating MEM_s with one query to O_s.

```
Input: A vector $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$.
Output: $\text{MEM}_s(x)$.
1  for $i = 1, \ldots, n$ do
2      if $x_i > 2$ or $x_i < 0$ then
3          Return 0;
4      Set $y_i = 1$ if $x_i > 1$ and 0 otherwise;
5  if $|y|_{\text{Ham}} > 1$ then
6      Return 0;
7  else
8      if $|y|_{\text{Ham}} = 1$ then
9          Find $i$ such that $y_i = 1$. Return $O_s(i)$;
10     else
11         Return 1;
```

for deciding whether $|s|_{\text{Ham}} > 0$ with access to O_s. We first show that Algorithm 11 simulates the oracle MEM_s. In the for loop of Line 1, we know that $y_i = 1$ if and only if $1 < x_i \leq 2$, which is inside the convex body if $s_i = 1$. The case $|y|_{\text{Ham}} > 1$ implies that there exist two distinct coordinates i, j such that $x_i, x_j > 1$, which implies that x lies outside the convex body. Now we are left with the cases $|y|_{\text{Ham}} = 1$ or 0. In Line 9, $y_i = 1$ implies $1 < x_i \leq 2$, which lies in the convex body if and only if $s_i = O_s(i) = 1$. Also, $|y| = 0$ implies that for every coordinate i, $0 \leq x_i \leq 1$, which lies in the body for all s.

Finally, if there is a k-query algorithm that computes an estimate $\widetilde{\text{Vol}(\text{K})}$ of the volume of K up to multiplicative precision $0 < \epsilon < \sqrt{2} - 1$, then $s = \lceil \log_2 \widetilde{\text{Vol}(\text{K})} \rfloor$ where $\lceil \cdot \rfloor$ returns the nearest integer. This immediately gives a k-query algorithm that decides whether $|s|_{\text{Ham}} = 0$ or 1. Since there is an $\Omega(\sqrt{n})$ quantum query lower bound for this task, the $\Omega(\sqrt{n})$ lower bound on volume estimation follows. □

Remark 3.6.1. *The proof of Theorem 3.6.1 has similarity to [79, Section 5].*

3.6.2 An optimal quantum lower bound in $1/\epsilon$

In this subsection, we prove:

Theorem 3.6.2. *Suppose $1/n \leq \epsilon \leq 1/3$. Estimating the volume of K with multiplicative precision ϵ requires $\Omega(1/\epsilon)$ quantum queries to the membership oracle O_{K} defined in (4.8).*

Comparing with Theorem 4.1.1, this shows that our quantum algorithm for volume estimation is optimal in $1/\epsilon$ up to poly-logarithmic factors.

The proof constructs a convex body whose volume encodes the Hamming weight of a string. A membership oracle for this convex body can be implemented by querying the bits of the string. Then the tight lower bound of Nayak and Wu on the quantum query complextiy of approximating the Hamming weight [92] implies a lower bound on the query complexity of volume estimation.

We construct the convex body by attaching hyperpyramids to the faces of the n-dimensional unit hypercube. The axis of each hyperpyramid is aligned with the axis of the face of the hypercube it corresponds to, and the height of the hyperpyramid is $1/2$. More concretely, if the unit hypercube is $H_n := [-1/2, 1/2]^n$, then the two hyperpyramids on the face perpendicular to the ith axis are

$$P_{i,+} := \big\{x : x_i \geq 1/2, |x_k| + |x_i| \leq 1 \,\forall\, k \in [n]/\{i\}\big\}; \tag{3.206}$$

$$P_{i,-} := \big\{x : x_i \leq -1/2, |x_k| + |x_i| \leq 1 \,\forall\, k \in [n]/\{i\}\big\}. \tag{3.207}$$

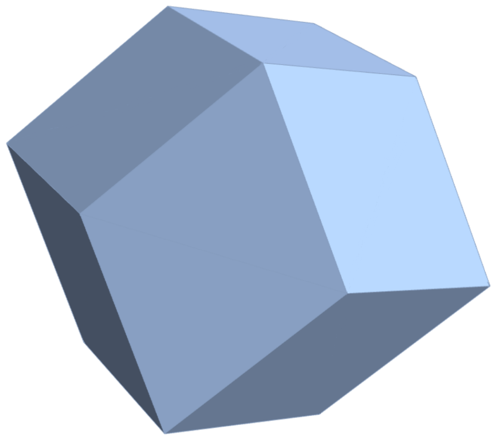


Figure 3.8: The convex body C_3.

We denote the convex body with all hyperpyramids attached by

$$\mathrm{C}_n := H_n \cup \Big(\bigcup_{i=1}^{n}(P_{i,+} \cup P_{i,-})\Big). \tag{3.208}$$

For illustration, the 3-dimensional convex body C_3 is shown in Figure 3.8.

We first prove:

Lemma 3.6.1. *C_n is convex for all $n \in \mathbb{N}$.*

Proof. It suffices to show that if $x, y \in \mathrm{C}_n$ and $\alpha \in [0,1]$, then $\alpha x + (1-\alpha)y \in \mathrm{C}_n$. We consider three cases:

Case 1: $x, y \in H_n$ This case is straightforward as H_n is convex, hence $\alpha x + (1-\alpha)y \in H_n \subset \mathrm{C}_n$.

Case 2: $x \in \bigcup_{i=1}^{n}(P_{i,+} \cup P_{i,-}), y \in H_n$ Let $i^* \in [n]$ such that $x \in P_{i^*,+} \cup P_{i^*,-}$. Then by (3.206) and (3.207), $|x_{i^*}| \geq 1/2$ and $|x_i| + |x_{i^*}| \leq 1 \; \forall i \in [n] \setminus \{i^*\}$, which implies $|x_i| \leq$

$1/2\ \forall i \in [n] \setminus \{i^*\}$. Also note that $y \in H_n$ implies $|y_i| \le 1/2\ \forall i \in [n]$. Therefore,

$$|\alpha x_i + (1-\alpha)y_i| \le \alpha|x_i| + (1-\alpha)|y_i| \le \frac{\alpha}{2} + \frac{1-\alpha}{2} = \frac{1}{2} \quad \forall\, i \in [n]/\{i^*\}. \tag{3.209}$$

If $|\alpha x_{i^*} + (1-\alpha)y_{i^*}| \le 1/2$, then $\alpha x + (1-\alpha)y \in H_n \subseteq \mathrm{C}_n$. If $|\alpha x_{i^*} + (1-\alpha)y_{i^*}| > 1/2$, then

$$|\alpha x_{i^*} + (1-\alpha)y_{i^*}| + |\alpha x_i + (1-\alpha)y_i| \le \alpha(|x_{i^*}| + |x_i|) + (1-\alpha)(|y_{i^*}| + |y_i|) \tag{3.210}$$

$$\le \alpha + (1-\alpha) = 1 \quad \forall\, i \in [n] \setminus \{i^*\}. \tag{3.211}$$

Therefore, by (3.206) and (3.207) we have $\alpha x + (1-\alpha)y \in P_{i^*,+} \cup P_{i^*,-} \subset \mathrm{C}_n$. In any case, we always have $\alpha x + (1-\alpha)y \in \mathrm{C}_n$.

Case 3: $x, y \in \bigcup_{i=1}^{n}(P_{i,+} \cup P_{i,-})$ Let $i^*, j^* \in [n]$ such that $x \in P_{i^*,+} \cup P_{i^*,-}$ and $y \in P_{j^*,+} \cup P_{j^*,-}$. If $i^* = j^*$, the proof is identical to that of Case 2 and we omit the details here. It remains to consider the case $i^* \neq j^*$. Then we have $|x_i|, |y_i| \le 1/2\ \forall i \in [n] \setminus \{i^*, j^*\}$. In addition,

$$|\alpha x_{i^*} + (1-\alpha)y_{i^*}| + |\alpha x_{j^*} + (1-\alpha)y_{j^*}| \le \alpha(|x_{i^*}| + |x_{j^*}|) + (1-\alpha)(|y_{j^*}| + |y_{i^*}|)$$

$$\le \alpha + (1-\alpha) = 1 \tag{3.212}$$

by (3.206) and (3.207). This means that at most one of $|\alpha x_{i^*} + (1-\alpha)y_{i^*}|$ and $|\alpha x_{j^*} + (1-\alpha)y_{j^*}|$ can be more than $1/2$. If neither of them is more than $1/2$, then $\alpha x + (1-\alpha)y \in H_n \subset \mathrm{C}_n$. If exactly one of them is more than $1/2$, say $|\alpha x_{i^*} + (1-\alpha)y_{i^*}| > 1/2$ and $|\alpha x_{j^*} + (1-\alpha)y_{j^*}| \le 1/2$, then $\alpha x + (1-\alpha)y \in P_{i^*,+} \cup P_{i^*,-} \subset \mathrm{C}_n$. In any case, we always have $\alpha x + (1-\alpha)y \in \mathrm{C}_n$. $\square$

We use the following lower bound on the quantum query complexity of approximating the

Hamming weight:

Proposition 3.6.1 ([92]). *Suppose we are given the quantum oracle $O_s|i\rangle|0\rangle = |i\rangle|s_i\rangle\ \forall i \in [n]$ for some $s \in \{0,1\}^n$. Let $0 \le l < l' \le n$ be two integers, $\Delta = |l - l'|$, and $m \in \{l, l'\}$ such that $|\frac{n}{2} - m|$ is maximized. Then the quantum query complexity of determining whether s has Hamming weight at most l or at least l' is $\Theta(\sqrt{n/\Delta} + \sqrt{m(n-m)}/\Delta)$.*

Now we can prove Theorem 3.6.2.

Proof. Given a binary string $s \in \{0,1\}^n$, we consider the convex body

$$\mathrm{C}_s := H_n \cup \Big(\bigcup_{i\colon s_i = 1} (P_{i,+} \cup P_{i,-}) \Big). \tag{3.213}$$

By Lemma 3.6.1 and the fact that each hyperpyramid is the intersection of C_n and the convex spaces $\{x : x_i \ge 1/2\}$ or $\{x : x_i \ge 1/2\}$, C_s is also convex. Furthermore, a query to the membership oracle in (4.8) for C_s can be implemented using one query to the binary string oracle O_s: queries to points outside C_n or inside H_n are trivially answered with 0 and 1, respectively, whereas queries to points in $P_{i,+} \cup P_{i,-}$ should return s_i. Also note that for each $i \in [n]$, the volume of the hyperpyramid $P_{i,+}$ is

$$\mathrm{Vol}(P_{i,+}) = \int_0^{1/2} (1-2t)^{n-1}\mathrm{d}t = \frac{1}{2n} \tag{3.214}$$

since the intersection of $P_{i,+}$ and $\{x : x_i = 1/2 + t\}$ is an $(n-1)$-dimensional hypercube with side-length $1 - 2t$ and hence volume $(1-2t)^{n-1}$. By symmetry, we also have $\mathrm{Vol}(P_{i,-}) = \frac{1}{2n}$.

Therefore

$$\mathrm{Vol}(\mathrm{C}_s) = \mathrm{Vol}(H_n) + \sum_{i:s_i=1} \big(\mathrm{Vol}(P_{i,+}) + \mathrm{Vol}(P_{i,-})\big) \tag{3.215}$$

$$= 1 + |s|_{\text{Ham}} \cdot \frac{2}{2n} = 1 + \frac{|s|_{\text{Ham}}}{n}. \tag{3.216}$$

In other words, estimating the volume of C_s with multiplicative error ϵ is equivalent to the Hamming distance problem with $\Delta = 4\epsilon n$. Taking $m = \frac{n}{2} + \Delta$ in Proposition 3.6.1, we find that the quantum query complexity of estimating the volume of C_s is at least

$$\Omega\Big(\sqrt{\frac{n}{\epsilon n}} + \frac{\sqrt{n^2/4 - \epsilon^2 n^2}}{\epsilon n}\Big) = \Omega\Big(\frac{1}{\epsilon}\Big) \tag{3.217}$$

for any $1/n \leq \epsilon \leq 1/3$. □

Remark 3.6.2. *The same proof strategy implies a classical lower bound of $\Omega(1/\epsilon^2)$ for volume estimation if we replace Proposition 3.6.1 by its folklore classical counterpart. In particular, this shows that our quantum algorithm in Theorem 4.1.1 achieves a provable quadratic quantum speedup in $1/\epsilon$.*

Remark 3.6.3. *Although the proofs of both theorems consider well-rounded convex bodies, this assumption can be simply waived by assuming known multiplicative rescaling factors $c_1, \ldots, c_n$ along all the n directions. The proofs follow from the same arguments.*

Chapter 4: Sublinear Quantum Algorithms for Linear Classification via Matrix Games

In this section we first present a sublinear algorithm with quantum speedup for linear classification and some associated problems. We then view this problem as a matrix game and extend our algorithms to more general matrix games; with applications to problems such as the well known caratheodory problem. The results discussed here were first established in [26] and [27].

4.1 Introduction

Motivations. Classification is a fundamental problem of supervised learning, which takes a training set of data points of known classes as inputs and aims to training a model for predicting the classes of future data points. It is also ubiquitous due to its broad connections and applications to computer vision, natural language processing, statistics, etc.

A fundamental case of classification is *linear classification*, where we are given n data points $X_1, \ldots, X_n$ in $\mathbb{R}^d$ and a label vector $y \in \{-1, 1\}^n$. The goal is to find a separating hyperplane, i.e., a unit vector w in $\mathbb{R}^d$, such that

$$y_i \cdot X_i^\top w \geq 0 \quad \forall\, i \in [n]. \tag{4.1}$$

By taking $X_i \leftarrow (-1)^{y_i} X_i$, it reduces to a *maximin* problem, i.e., $\max_w \min_i X_i^\top w \geq 0$. The approximation version of linear classification is to find a unit vector $\bar{w} \in \mathbb{R}^d$ so that

$$X_i^\top \bar{w} \geq \max_{w \in \mathbb{R}_d} \min_{i' \in [n]} X_{i'}^\top w - \epsilon \quad \forall\, i \in [n], \tag{4.2}$$

i.e., $\bar{w}$ approximately solves the maximin problem. More generally, we can regard a (nonlinear) classifier as a *kernel-based* classifier by replacing X_i by $\Psi(X_i)$ (Ψ being a kernel function). We will focus on algorithms finding approximate classifiers (in the sense of (4.2)) with *provable guarantees*.

The Perceptron Algorithm for linear classification is one of the oldest algorithms studied in machine learning [124, 125], which runs in time $O(nd/\epsilon^2)$ for finding an $\bar{w} \in \mathbb{R}^d$ satisfying (4.2). The state-of-the-art classical result along this line [126] solves linear classification in time $\tilde{O}((n+d)/\epsilon^2)$. A careful reader might notice that the input to linear classification is n d-dimensional vectors with total size $O(nd)$. Hence, the result of [126] is *sub-linear* in its input size. To make it possible, [126] assumes the following entry-wise input model:

Input model: given any $i \in [n]$ and $j \in [d]$, the j-th entry of X_i can be recovered in $O(1)$ time.

The output of [126] is an *efficient* classical representation of $\bar{w}$ in the sense that every entry of $\bar{w}$ can be recovered with $\tilde{O}(1)$ cost. It is no surprise that $\bar{w}$ per se gives such a representation. However, there could be more *succinct and efficient representations* of $\bar{w}$, which could be reasonable alternatives of $\bar{w}$ for sub-linear algorithms that run in time less the dimension of $\bar{w}$ (as we will see in the quantum case). The complexity of [126] is also optimal (up to poly-logarithmic factors) in the above input/output model as shown by the same paper.

Recent developments in quantum computation, especially in the emerging topic of "quantum machine learning" (see the surveys [127, 128, 129]), suggest that quantum algorithms might offer significant speed-ups for optimization and machine learning problems. In particular, a quantum counterpart of the Perceptron algorithm has been proposed in [130] with improved time complexity from $O(nd/\epsilon^2)$ to $\tilde{O}(\sqrt{n}d/\epsilon^2)$ (details in related works). Motivated both by the significance of classification and the promise of quantum algorithms, we investigate the *optimal* quantum algorithm for classification. Specifically, we aim to design a quantum counterpart of [126].

It is natural to require that quantum algorithms make use of the classical input/output model as much as possible to make the comparison fair. In particular, it is favorable to avoid the use of too powerful input data structure which might render any finding of quantum speedup inconclusive, especially in light of a recent development of quantum-inspired classical machine learning algorithms (e.g., [131]). Our choice of input/output models for quantum algorithms is hence almost the same as the classical one, except we allow *coherent* queries to the entries of X_i:

Quantum input model: given any $i \in [n]$ and $j \in [d]$, the j-th entry of X_i can be recovered in $O(1)$ time *coherently*.

Coherent queries allow the quantum algorithm to query many locations in super-position, which is a *standard* assumption that accounts for many quantum speed-ups (e.g., Grover's algorithm [132]). A more precise definition is given in Section 4.2.

On the other side, our output is exactly the same as classical algorithms, which guarantees no overhead when using our quantum algorithms as subroutines for any applications.

Contributions. Our main contribution is a *tight* characterization (up to poly-log factors) of quantum algorithms for various classification problems in the aforementioned input/output model.

Theorem 4.1.1 (Main theorem)**.** *Given $\epsilon = \Theta(1)$, we have quantum algorithms that return an efficient representation of $\bar{w} \in \mathrm{B}_d$ for the following problems*[1]*, respectively, with complexity $\tilde{O}(\sqrt{n} + \sqrt{d})$ and high success probability:*

- *Linear classification (Section 4.3):*

$$\min_{i\in[n]} X_i^\top \bar{w} \geq \max_{w\in\mathrm{B}_d} \min_{i\in[n]} X_i^\top w - \epsilon. \tag{4.3}$$

- *Kernel-based classification (Section 4.4.1):*

$$\min_{i\in[n]} \langle \Psi(X_i), \bar{w}\rangle \geq \max_{w\in\mathrm{B}_d} \min_{i\in[n]} \langle \Psi(X_i), w\rangle - \epsilon, \tag{4.4}$$

  where $k(a,b) := \langle \Psi(a), \Psi(b)\rangle$ can be the polynomial kernel $k_q(a,b) = (a^\top b)^q$ or the Gaussian kernel $k_{Gauss}(a,b) = \exp(-\|a-b\|^2)$.

- *Minimum enclosing ball (Section 4.4.2.1):*

$$\max_{i\in[n]} \|\bar{w} - X_i\|^2 \leq \min_{w\in\mathbb{R}^d} \max_{i\in[n]} \|w - X_i\|^2 + \epsilon. \tag{4.5}$$

- *ℓ^2-margin SVM (Section 4.4.2.2):*

$$\min_{i\in[n]} (X_i^\top \bar{w})^2 \geq \max_{w\in\mathbb{R}^d} \min_{i\in[n]} 2X_i^\top w - \|w\|^2 - \epsilon. \tag{4.6}$$

[1]Here B_d is the unit ball in $\mathbb{R}^d$, *i.e.*, $\mathrm{B}_d := \left\{a \in \mathbb{R}^d \mid \sum_{i\in[d]} |a_i|^2 \leq 1\right\}$.

On the other hand, we show that it requires $\Omega(\sqrt{n}+\sqrt{d})$ queries to the quantum input model to prepare such $\bar{w}$ for these classification problems (Section 4.5).

Our matching upper and lower bounds $\sqrt{n}+\sqrt{d}$ give a *quadratic* improvement in both n and d comparing to the classical state-of-the-art results in [126].

Technically, our result is also inspired by the recent development of quantum semidefinite program (SDP) solvers (e.g., [18]) which provide quantum speed-ups for approximating zero-sum games for the purpose of solving SDPs. Note that such a connection was leveraged classically in another direction in a follow-up work of [126] for solving SDPs [133]. However, our algorithm is even simpler because we only use simple quantum state preparation instead of complicated quantum operations in quantum SDP solvers; this is because quantum state preparation is a direct counterpart of the ℓ^2 sampling used in [126] (see Section 4.3.1 for details). In a nutshell, our result is a demonstration of quantum speed-ups for sampling-based classical algorithms.

Moreover, our algorithms are hybrid classical-quantum algorithms where the quantum part is isolated pieces of state preparation connected by classical processing. In addition, special instances of these state preparation might be physically realizable as suggested by some work-in-progress [134]. All of the above suggest the possibility of implementing these algorithms on near-term quantum machines [135].

In general, we deem our result as a proposal of one end-to-end quantum application in machine learning, with both provable guarantees and the perspective of implementation (at least in prototype) on near-term quantum machines.

Application to matrix zero-sum games. As a side result, our techniques can be applied to solve matrix zero-sum games. To be more specific, the input of the zero-sum game is a matrix

$X \in \mathbb{R}^{n_1 \times n_2}$ and an $\epsilon > 0$, and the goal is to find $a \in \mathbb{R}^{n_1}$ and $b \in \mathbb{R}^{n_2}$ such that[2]

$$a^\dagger X b \geq \max_{p \in \Delta_{n_1}} \min_{q \in \Delta_{n_2}} p^\dagger X q - \epsilon. \tag{4.7}$$

If we are given the quantum input model of A, we could output such a and b as classical vectors[3] with complexity $\tilde{O}(\sqrt{n_1 + n_2}/\epsilon^4)$ (see Theorem 4.4.3). When $\epsilon = \Theta(1)$, our quantum algorithm is optimal as we prove an $\Omega(\sqrt{n_1 + n_2})$ quantum lower bound (see Theorem 4.5.3).

Related works. We make the following comparisons with existing literatures in quantum machine learning.

- The most relevant result is the quantum perceptron models in [130]. The classical perceptron method [124, 125] is a pivotal linear classification algorithm. In each iteration, it checks whether (4.1) holds; if not, then it searches for a violated constraint i_0 (i.e., $y_{i_0} X_{i_0}^\top \bar{w} < 0$) and update $\bar{w} \leftarrow \bar{w} + X_{i_0}$ (up to normalization). This classical perceptron method has complexity $\tilde{O}(nd/\epsilon^2)$; the quantum counterpart in [130] improved the complexity to $\tilde{O}(\sqrt{n}d/\epsilon^2)$ by applying Grover search [132] to find a violated constraint. In contrast, we quantize the sublinear algorithm for linear classification in [126] with techniques inspired by quantum SDP solvers [18]. As a result, we establish a better quantum complexity $\tilde{O}(\sqrt{n} + \sqrt{d})$.

  In addition, [130] relies on an unusual input model where a data point in $\mathbb{R}^d$ is represented by concatenating the the binary representations of the d floating point numbers; if we were only given standard inputs with entry-wise queries to the coordinates of data points, we need a cost of $\Omega(d)$ to transform the data into their input form, giving the total complexity $\tilde{O}(\sqrt{n}d)$.

[2]Here Δ_n is the set of probability distributions on $[n]$, i.e., $\Delta_n := \{a \in \mathbb{R}^n \mid a_i \geq 0 \ \forall i \in [n], \sum_{i \in [n]} a_i = 1\}$.
[3]In fact, x and y are classical vectors with succinct representations; see more details at Remark 4.4.1.

The same group of authors also gave a quantum algorithm for nearest-neighbor classification with complexity $\tilde{O}(\sqrt{n})$ [136]. This complexity also depends on the sparsity of the input data; in the worst case where every data point has $\Theta(d)$ nonzero entries, the complexity becomes $\tilde{O}(\sqrt{n}d^2)$.

- There have been rich developments on quantum algorithms for linear algebraic problems. One prominent example is the quantum algorithm for solving linear systems [10, 20]; in particular, they run in time $\mathrm{poly}(\log d)$ for any sparse d-dimensional linear systems. These linear system solvers are subsequently applied to machine learning applications such as cluster assignment [137], support vector machine (SVM) [138], etc.

  However, these quantum algorithms have two drawbacks. First, they require the input matrix to be *sparse* with efficient access to nonzero elements, i.e., every row/column of the matrix has at most $\mathrm{poly}(\log d)$ nonzero elements and their indexes can be queried in $\mathrm{poly}(\log d)$ time. Second, the outputs of these algorithms are quantum states instead of classical vectors, and it takes $\Omega(d)$ copies of the quantum state to reveal one entry of the output in the worst case. More caveats are listed in [139].

  In contrast, our quantum algorithms do not have the sparsity constraint and work for arbitrary input data, and the outputs of our quantum algorithms are succinct but efficient classical representations of vectors in $\mathbb{R}^d$, which can be directly used for classical applications.

- There are two lines of quantum machine learning algorithms with different input requirements. One of them is based on quantum principal component analysis [140] and requires purely quantum inputs.

  Another line is the recent development of quantum-inspired classical poly-logarithmic time algorithms for various machine learning tasks such as recommendation systems [131], principal component analysis [141], solving linear systems [142, 143], SDPs [144], and so on. These algorithms follow a Monte-Carlo approach for low-rank matrix approximation [145] and assume the ability to take samples according to the spectral norms of all rows. In other words, these results enforce additional requirements on their input: the input matrix should not only be low-rank but also be preprocessed as the sampling data structure.

- There are also a few heuristic quantum machine learning approaches for classification [12, 146, 147] without theoretical guarantees. We, however, look forward to further experiments based on their proposals.

4.2 Preliminaries

Quantum oracle. Quantum access to the input data (referred as quantum oracles) needs to be reversible and allows access to different parts of the input data in *superposition* (the essence of quantum speed-ups). Specifically, to access elements in an $n \times d$ matrix X, we exploit an oracle O_X (a unitary on $\mathbb{C}^n \otimes \mathbb{C}^d \otimes \mathbb{C}^{d_{\text{acc}}}$) such that

$$O_X(|i\rangle \otimes |j\rangle \otimes |z\rangle) = |i\rangle \otimes |j\rangle \otimes |z \oplus X_{ij}\rangle \tag{4.8}$$

for any $i \in [n]$, $j \in [d]$ and $z \in \mathbb{C}^{d_{\text{acc}}}$ such that X_{ij} can be represented in $\mathbb{C}^{d_{\text{acc}}}$. Intuitively, O_X reads the entry X_{ij} and stores it in the third register. However, to make O_X reversible (and unitary), O_X applies the XOR operation ($\oplus$) on the third register. Note that O_X is a natural unitary generalization of classical random access to X, or in cases when any entry of X can be efficiently read. However, it is potentially stronger when queries become linear combinations of basis vectors, e.g., $\sum_k \alpha_k |i_k\rangle \otimes |j_k\rangle$. This is technically how to make superposition of different queries in quantum.

We summarize the quantum notations as follows.

	Classical	Quantum
Ket and bra	$\vec{e}_i$ and $\vec{e}_i^{\top}$	$\|i\rangle$ and $\langle i\|$
Basis	$\{\vec{e}_0, \ldots, \vec{e}_{d-1}\}$	$\{\|0\rangle, \ldots, \|d-1\rangle\}$
State	$\vec{v} = (v_0, \ldots, v_{d-1})^{\top}$	$\|v\rangle = \sum_{i=0}^{d-1} v_i \|i\rangle$
Tensor	$\vec{u} \otimes \vec{v}$	$\|u\rangle \otimes \|v\rangle$ or $\|u\rangle\|v\rangle$
Oracle	$w = (X_{ij})_{i,j=1}^{n}$	$O_X\|i\rangle\|j\rangle\|z\rangle = \|i\rangle\|j\rangle\|z \oplus X_{ij}\rangle$

Table 4.1: Summary of quantum notations used in this paper.

Quantum complexity measure. We assume that a single query to the oracle O_X has a unit cost. *Quantum query complexity* is defined as the total counts of oracle queries, and *quantum gate complexity* is defined as the total counts of oracle queries and two-qubit gates.

Notations. Throughout this paper, we denote $\mathbf{1}_n$ to be the n-dimensional all-one vector, and $X \in \mathbb{R}^{n\times d}$ to be the matrix whose entry in the intersection of its i^{th} row and j^{th} column is $X_i(j)$ for all $i \in [n]$, $j \in [d]$. Without loss of generality, we assume $X_1, \ldots, X_n \in \mathrm{B}_d$, i.e., all the n data points (also the n rows of X) are normalized to have ℓ^2-norm at most 1.

4.3 Linear classification

4.3.1 Techniques

At a high level, our quantum algorithm leverages ideas from both classical and quantum algorithm design. We use a primal-dual approach under the multiplicative weight framework

[148], in particular its improved version in [126] by sampling the update of weight vectors. An important observation of ours is that such classical algorithms can be accelerated significantly in quantum computation, which relies on a seminal technique in quantum algorithm design: amplitude amplification and estimation [57, 132].

Multiplicative weight under a primal-dual approach. Note that linear classification is essentially a minimax problem (zero-sum game); by strong duality, we have

$$\sigma = \max_{w \in \mathbb{R}_d} \min_{p \in \Delta_n} p^\top X w = \min_{p \in \Delta_n} \max_{w \in \mathbb{R}_d} p^\top X w. \tag{4.9}$$

To find its equilibrium point, we adopt an online primal-dual approach with T rounds; at round $t \in [T]$, the primal computes $p_t \in \Delta_n$ and the dual computes $w_t \in \mathbb{R}_d$, both based on p_τ and w_τ for all $\tau \in [t-1]$. After T rounds, the average solution $\bar{w} = \frac{1}{T}\sum_{t=1}^{T} w_t$ approximately solves the zero-sum game with high probability, i.e., $\min_{p \in \Delta_n} p^\top X \bar{w} \geq \sigma - \epsilon$.

For the primal problem, we pick p_t by the *multiplicative weight* (MW) method. Given a sequence of vectors $r_1, \ldots, r_T \in \mathbb{R}^n$, MW sets $w_1 := \mathbf{1}_n$ and for all $t \in [T]$, $p_t := w_t / \|w_t\|_1$ and $w_{t+1}(i) := w_t(i) f_{\mathrm{w}}(-\eta r_t(i))$ for all $i \in [n]$, where f_{w} is a weight function and η is the parameter representing the step size. MW promises an upper bound on $\sum_{t=1}^{T} p_t^\top r_t$, whose precise form depends on the choice of the weight function f_{w}. The most common update is the *exponential weight update*: $f_{\mathrm{w}}(x) = e^{-x}$ [148], but in this paper we use a quadratic weight update suggested by [126], where $w_{t+1}(i) := w_t(i)(1 - \eta r_t(i) + \eta^2 r_t(i)^2)$. In our primal problem, we set $r_t = X w_t$ for all $t \in [T]$ to find p_t.

For the dual problem, we pick w_t by the *online gradient descent* method [149]. Given a set of vectors $q_1, \ldots, q_T \in \mathbb{R}^d$ such that $\|q_i\|_2 \leq 1$. Let $w_0 := \mathbf{0}_d$, and $y_{t+1} := w_t + \frac{1}{\sqrt{T}} q_t$, $w_{t+1} := \frac{y_{t+1}}{\max\{1, \|y_{t+1}\|\}}$. Then

$$\max_{w \in \mathrm{B}_d} \sum_{t=1}^{T} q_t^\top w - \sum_{t=1}^{T} q_t^\top w_t \leq 2\sqrt{T}. \tag{4.10}$$

This can be regarded as a *regret* bound, i.e., $\sum_{t=1}^{T} q_t^\top w_t$ has at most a regret of $2\sqrt{T}$ compared to the best possible choice of w. In our dual problem, we set q_t as a sample of rows of X following the distribution p_t.

This primal-dual approach gives a correct algorithm with only $T = \tilde{O}(1/\epsilon^2)$ iterations. However, the primal step runs in $\Theta(nd)$ time to compute Xw_t. To obtain an algorithm that is *sublinear* in the size of X, a key observation by [126] is to replace the precise computation of Xw_t by an unbiased random variable. This is achieved via ℓ^2 sampling of w: we pick $j_t \in [d]$ by $j_t = j$ with probability $w_t(j)^2/\|w_t\|^2$, and for all $i \in [n]$ we take $\tilde{v}_t(i) = X_i(j_t)\|w_t\|^2/w_t(j_t)$. The expectation of the random variable $\tilde{v}_t(i)$ satisfies

$$\mathbb{E}[\tilde{v}_t(i)] = \sum_{j=1}^{d} \frac{w_t(j)^2}{\|w_t\|^2} \frac{X_i(j)\|w_t\|^2}{w_t(j)} = X_i w_t. \tag{4.11}$$

In a nutshell, the update of weight vectors in each iteration need not to be precisely computed because an ℓ^2 sample from w suffices to promise the provable guarantee of the framework. This trick improves the running time of MW to $O(n)$ and online gradient descent to $O(d)$; since there are $\tilde{O}(1/\epsilon^2)$ iterations, the total complexity is $\tilde{O}(\frac{n+d}{\epsilon^2})$ as claimed in [126].

Amplitude amplification and estimation. Consider a search problem where we are given a function $f_\omega \colon [n] \to \{-1, 1\}$ such that $f_\omega(i) = 1$ iff $i \neq \omega$. To search for ω, classically we need

$\Omega(n)$ queries to f_ω as checking all n positions is the only method.

Quantumly, given a unitary U_ω such that $U_\omega|i\rangle = |i\rangle$ for all $i \neq \omega$ and $U_\omega|\omega\rangle = -|\omega\rangle$, Grover's algorithm [132] finds ω with complexity $\tilde{O}(\sqrt{n})$. Denote $|s\rangle = \frac{1}{\sqrt{n}}\sum_{i\in[n]}|i\rangle$ (the uniform superposition), $|s'\rangle = \frac{1}{\sqrt{n-1}}\sum_{i\in[n]/\{\omega\}}|i\rangle$, and $U_s = 2|s\rangle\langle s| - I$, the unitary U_ω reflects a state with respect to $|s'\rangle$ and the unitary U_s reflects a state with respect to $|s\rangle$. If we start with $|s\rangle$ and denote $\theta = 2\arcsin(1/\sqrt{n})$ (the angle between $U_\omega|s\rangle$ and $|s\rangle$), then the angle between $U_\omega|s\rangle$ and $U_sU_\omega|s\rangle$ is *amplified* to 2θ, and in general the angle between $U_\omega|s\rangle$ and $(U_sU_\omega)^k|s\rangle$ is $2k\theta$. To find ω, it suffices to take $k = \Theta(\sqrt{n})$ in this quantum algorithm. See Figure 4.1 for an illustration.

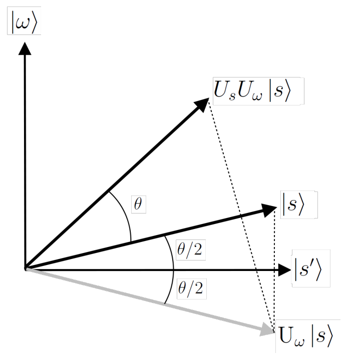


Figure 4.1: Geometric interpretation of Grover's algorithm. This figure is copied from Wikipedia.

This trick of alternatively applying these two unitaries is called *amplitude amplification*; in general, this provides a quadratic speedup for search problems. For the quantitative version of estimating θ (not only finding ω), quadratic quantum speedup also holds via an improved version of amplitude amplification called *amplitude estimation* [57].

Our **main technical contribution** is the implementations of amplitude amplification and estimation in the primal-dual approach for solving minimax problems. On the one hand, we achieve quadratic quantum speedup for multiplicative weight update, i.e., we improve the complexity from $\tilde{O}(n)$ to $\tilde{O}(\sqrt{n})$. This is because the ℓ^2 sampling of w is identical to measuring the quantum state $|w\rangle$ in the computational basis; furthermore, we prepare the state $|w\rangle$ by amplitude amplification (see Section 4.3.2.1).[4]

On the other hand, we also achieve quadratic quantum speedup for online gradient descent (improving $\tilde{O}(d)$ to $\tilde{O}(\sqrt{d})$). This is because the main cost of online gradient descent comes from estimating the norms $\|y_t\|$, which can be regarded as an amplitude estimation problem; details are given in Section 4.3.3.

Comparison between classical and quantum results. Although our quantum algorithms enjoy quadratic speedups in n and d, their executions incur a larger dependence in ϵ: we have worst case $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4} + \frac{\sqrt{d}}{\epsilon^8}\big)$ compared to the classical complexity $\tilde{O}\big(\frac{n}{\epsilon^2} + \frac{d}{\epsilon^2}\big)$ in [126]. The main reason of having a larger ϵ-dependence in quantum is because we cannot prepare the weight states in MW via those in previous iterations (i.e., the quantum state $|w_t\rangle$ cannot be prepared by $|w_{t-1}\rangle$), and we have to start over every time; this is an intrinsic difficulty due to quantum state preparation.

Therefore, there is a trade-off between [126] and our results for arbitrary ϵ: we provide faster training of the classifiers if we allow a constant error, while the classical algorithms in [126] might work better if we require high-accuracy classifiers.

4.3.2 Quantum speedup for multiplicative weights

First, we give a quantum algorithm for linear classification with complexity $\tilde{O}(\sqrt{n})$:

Theorem 4.3.1. *With success probability at least $2/3$, Algorithm 12 returns a succinct classical*

[4] Another common method to prepare quantum states is via quantum random access memory (QRAM). This is incomparable to our approach because preparing the data structure for QRAM takes $\Omega(n)$ cost (though after that one read takes $\tilde{O}(1)$ cost). Here we use amplitude amplification for giving sublinear algorithms. See also Section 4.3.2.1.

representation of a vector $\bar{w} \in \mathbb{R}^d$ such that

$$X_i \bar{w} \geq \max_{w \in \mathrm{B}_d} \min_{i' \in [n]} X_{i'} w - \epsilon \quad \forall\, i \in [n], \tag{4.12}$$

using $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4} + \frac{d}{\epsilon^2}\big)$ quantum gates.

Algorithm 12: Quantum linear classification algorithm.

```
Input: $\epsilon > 0$, a quantum oracle $O_X$ for $X \in \mathbb{R}^{n\times d}$.
Output: $\bar{w}$ that satisfies (4.12).
1 Let $T = 23^2\epsilon^{-2}\log n$, $y_1 = \mathbf{0}_d$, $\eta = \sqrt{\frac{\log n}{T}}$, $u_1 = \mathbf{1}_n$, $|p_1\rangle = \frac{1}{\sqrt{n}}\sum_{i\in[n]}|i\rangle$;
2 for $t = 1$ to $T$ do
3     Define[5] $w_t := \frac{y_t}{\max\{1,\|y_t\|\}}$;
4     Measure the state $|p_t\rangle$ in the computational basis and denote the output as $i_t \in [n]$;
5     Define $y_{t+1} := y_t + \frac{1}{\sqrt{2T}}X_{i_t}$;
6     Choose $j_t \in [d]$ by $j_t = j$ with probability $\frac{w_t(j)^2}{\|w_t\|^2}$;
7     For all $i \in [n]$, denote $\tilde{v}_t(i) = X_i(j_t)\frac{\|w_t\|^2}{w_t(j_t)}$, $v_t(i) = \min\{1/\eta, \max\{-1/\eta, \tilde{v}_t(i)\}\}$,
       and $u_{t+1}(i) = u_t(i)(1 - \eta v_t(i) + \eta^2 v_t(i)^2)$. Implement a quantum oracle $O_t$ such
       that for all $i \in [n]$, $O_t|i\rangle|0\rangle = |i\rangle|u_{t+1}(i)\rangle$ by Algorithm 14 in Section 4.3.2.2;
8     Prepare $|p_{t+1}\rangle = \frac{1}{\|u_{t+1}\|_2}\sum_{i\in[n]} u_{t+1}(i)|i\rangle$ by applying Algorithm 13 to $O_t$;
9 Return $\bar{w} = \frac{1}{T}\sum_{t=1}^{T} w_t$;
```

Note that Algorithm 12 is inspired by the classical sublinear algorithm [126] by using online gradient descent in Line 5 and ℓ^2 sampling in Line 6 and Line 7. However, to achieve the $\tilde{O}(\sqrt{n})$ quantum complexity we use two quantum building blocks: a state preparation procedure in Line 7, and an oracle implementation procedure in Line 8; their details are covered in , respectively. The full proof of Theorem 4.3.1 is given in Section 4.3.2.3.

[5]By defining w_t here, we do not write down the whole vector but we construct any query to its entries in $O(1)$ time. For example, the ith coordinate of w_t is $w_t(i) = \frac{y_t(i)}{\max\{1,\|y_t\|\}}$, constructed by one query to $y_t(i)$. The y_{t+1} in Line 5 is defined in the same sense.

4.3.2.1 Quantum state preparation with oracles

We use the following result for quantum state preparation (see, e.g., [150]):

Proposition 4.3.1. *Assume that $a \in \mathbb{C}^n$, and we are given a unitary oracle O_a such that $O|i\rangle|0\rangle = |i\rangle|a_i\rangle$ for all $i \in [n]$. Then Algorithm 13 takes $O(\sqrt{n})$ calls to O_a for preparing the quantum state $\frac{1}{\|a\|_2}\sum_{i\in[n]} a_i|i\rangle$ with success probability $1 - O(1/n)$.*

Algorithm 13: Prepare a pure state given an oracle to its coefficients.

```
1 Apply Dürr-Høyer's algorithm [151] to find $a_{\max} := \max_{i\in[n]} |a_i|$ in $O(\sqrt{n})$ time;
2 Prepare the uniform superposition $\frac{1}{\sqrt{n}}\sum_{i\in[n]} |i\rangle$;
3 Perform the following unitary transformations:
```

$$\frac{1}{\sqrt{n}}\sum_{i\in[n]}|i\rangle \xmapsto{O_a} \frac{1}{\sqrt{n}}\sum_{i\in[n]}|i\rangle|a_i\rangle \mapsto \frac{1}{\sqrt{n}}\sum_{i\in[n]}|i\rangle|a_i\rangle\Big(\frac{a_i}{a_{\max}}|0\rangle + \sqrt{1-\frac{|a_i|^2}{a_{\max}^2}}|1\rangle\Big)$$
$$\xmapsto{O_a^{-1}} \frac{1}{\sqrt{n}}\sum_{i\in[n]}|i\rangle|0\rangle\Big(\frac{a_i}{a_{\max}}|0\rangle + \sqrt{1-\frac{|a_i|^2}{a_{\max}^2}}|1\rangle\Big); \tag{4.13}$$

```
4 Delete the second system in Eq. (4.13), and rewrite the state as
```

$$\frac{\|a\|_2}{\sqrt{n}a_{\max}} \cdot \Big(\frac{1}{\|a\|_2}\sum_{i\in[n]} a_i|i\rangle\Big)|1\rangle + |a^{\perp}\rangle|0\rangle, \tag{4.14}$$

```
  where $|a^{\perp}\rangle := \frac{1}{\sqrt{n}}\sum_{i\in[n]}\sqrt{1-\frac{|a_i|^2}{a_{\max}^2}}|i\rangle$ is a garbage state;
5 Apply amplitude amplification [57] for the state in (4.14) conditioned on the second
  system being 1. Return the output;
```

Note that the coefficient in (4.14) satisfies $\frac{\|a\|_2}{\sqrt{n}a_{\max}} \geq \frac{1}{\sqrt{n}}$; therefore, applying amplitude amplification for $O(\sqrt{n})$ times indeed promises that we obtain $|1\rangle$ on the second system with success probability $1 - O(1/n)$, i.e., the state $\frac{1}{\|a\|_2}\sum_{i\in[n]} a_i|i\rangle$ is prepared in the first system.

Remark 4.3.1. *Algorithm 13 is incomparable to state preparation via quantum random access* *memory (QRAM). QRAM relies on the weak assumption that we start from zero, and every added datum is processed in poly-logarithmic time. In total, this takes at least linear time in the size of the data (see, for instance, [22]). For the task of Proposition 4.3.1, QRAM takes at least $\Omega(n)$ cost.*

In this paper, we use the standard model where the input is formulated as an oracle, also widely assumed and used in existing quantum algorithm literatures (e.g., [10, 18, 20, 132]). Under the standard model, Algorithm 13 prepares states with only $O(\sqrt{n})$ cost.

Nevertheless, it is an interesting question to ask whether there is a $\mathrm{poly}(\log(nd))$-time quantum algorithm for linear classification given the existence of a pre-loaded QRAM of X. This would require the ability to take summations of the vectors $\frac{1}{\sqrt{2T}}X_{i_t}$ in Line 5 of Algorithm 12 in $\mathrm{poly}(\log(nd))$-time as well as the ability to update the weight state u_{t+1} in Line 8 in $\mathrm{poly}(\log(nd))$-time, both using QRAM. These two tasks are plausible as suggested by classical poly-log time sample-based algorithms for matrix arithmetics under multiplicative weight frameworks [144], which can potentially be combined with the analysis of QRAM data structures in [22]; we leave this possibility as an open question.

4.3.2.2 Implementation of the quantum oracle for updating the weight vectors

The quantum oracle O_t in Line 7 of Algorithm 12 is implemented by Algorithm 14. For convenience, we denote $\mathrm{clip}(v, 1/\eta) := \min\{1/\eta, \max\{-1/\eta, v\}\}$ for all $v \in \mathbb{R}$.

Because we have stored w_s and j_s, we could construct classical oracles $O_{s,j}(0) = j_s$, $O_{s,w}(j_s) = \frac{\|w_s\|^2}{w_s(j_s)}$ with $O(1)$ complexity. In the algorithm, we first call $O_{s,j}$ to compute j_s and store it into the second register in (4.15). In (4.16), we call the quantum oracle O_X for the value

Algorithm 14: Quantum oracle for updating the weight state.

```
  Input: $w_1, \dots, w_t \in \mathbb{R}^d$, $j_1, \dots, j_t \in [d]$.
  Output: An oracle $O_t$ such that $O_t|i\rangle|0\rangle = |i\rangle|u_{t+1}(i)\rangle$ for all $i \in [n]$.
1 Define three classical oracles: $O_{s,j}(0) = j_s$, $O_{s,w}(j_s) = \frac{\|w_s\|^2}{w_s(j_s)}$, and
    $O_{\text{clip}}(a,b,c) = c \cdot \big(1 - \eta\,\text{clip}(ab, 1/\eta) + \eta^2\,\text{clip}(ab, 1/\eta)^2\big)$;
2 for $s = 1$ to $t$ do
3     Perform the following maps:
```

$$\begin{aligned}
|i\rangle|0\rangle|0\rangle|0\rangle|u_s(i)\rangle &\xmapsto{O_{s,j}} |i\rangle|j_s\rangle|0\rangle|0\rangle|u_s(i)\rangle &\quad (4.15)\\
&\xmapsto{O_X} |i\rangle|j_s\rangle|X_i(j_s)\rangle|0\rangle|u_s(i)\rangle &\quad (4.16)\\
&\xmapsto{O_{s,w}} |i\rangle|j_s\rangle|X_i(j_s)\rangle\Big|\frac{\|w_s\|^2}{w_s(j_s)}\Big\rangle|u_s(i)\rangle &\quad (4.17)\\
&\xmapsto{O_{\text{clip}}} |i\rangle|j_s\rangle|X_i(j_s)\rangle\Big|\frac{\|w_s\|^2}{w_s(j_s)}\Big\rangle|u_{s+1}(i)\rangle &\quad (4.18)\\
&\xmapsto{O_{s,w}^{-1}} |i\rangle|j_s\rangle|X_i(j_s)\rangle|0\rangle|u_{s+1}(i)\rangle &\quad (4.19)\\
&\xmapsto{O_X^{-1}} |i\rangle|j_s\rangle|0\rangle|0\rangle|u_{s+1}(i)\rangle &\quad (4.20)\\
&\xmapsto{O_{s,j}^{-1}} |i\rangle|0\rangle|0\rangle|0\rangle|u_{s+1}(i)\rangle. &\quad (4.21)
\end{aligned}$$

$X_i(j_s)$, which is stored into the third register. In (4.17), we call $O_{s,w}$ to compute $\frac{\|w_s\|^2}{w_s(j_s)}$ and store it into the fourth register. In (4.18), because we have $X_i(j_s)$ and $\frac{\|w_s\|^2}{w_s(j_s)}$ at hand, we could use $\tilde{O}(1)$ arithmetic computations to compute $\tilde{v}_s(i) = X_i(j_s)\|w_s\|^2/w_t(j_s)$ and

$$u_{s+1}(i) = u_s(i)\big(1 - \eta\,\mathrm{clip}(\tilde{v}_s(i), 1/\eta) + \eta^2\,\mathrm{clip}(\tilde{v}_s(i), 1/\eta)^2\big). \tag{4.22}$$

We then store $u_{s+1}(i)$ into the fifth register. In (4.19), (4.20), and (4.21), we uncompute the steps in (4.17), (4.16), and (4.15), respectively (we need these steps in Algorithm 14 to keep its unitarity).

In total, between (4.15)-(4.21) we use 2 queries to O_X and $\tilde{O}(1)$ additional arithmetic computations. Because s goes from 1 to t, in total we use $2t$ queries to O_X and $\tilde{O}(t)$ additional arithmetic computations.

4.3.2.3 Proof of Theorem 4.3.1

To prove Theorem 4.3.1, we use the following five lemmas proved in [126] for analyzing the online gradient gradient descent and ℓ^2 sampling outcomes:

Lemma 4.3.1 (Lemma A.2 of [126])**.** *The updates of w in Line 3 and y in Line 5 satisfy*

$$\max_{w\in\mathrm{B}_n} \sum_{t\in[T]} X_{i_t} w \leq \sum_{t\in[T]} X_{i_t} w_t + 2\sqrt{2T}. \tag{4.23}$$

Lemma 4.3.2 (Lemma 2.3 of [126])**.** *For any $t \in [T]$, denote p_t to be the unit vector in $\mathbb{R}^n$ such* *that $(p_t)_i = |\langle i|p_t\rangle|^2$ for all $i \in [n]$. Then the update for p_{t+1} in Line 8 satisfies*

$$\sum_{t\in[T]} p_t^\top v_t \le \min_{i\in[n]} \sum_{t\in[T]} v_t(i) + \eta \sum_{t\in[T]} p_t^\top v_t^2 + \frac{\log n}{\eta}, \tag{4.24}$$

where v_t^2 is defined as $(v_t^2)_i := (v_t)_i^2$ for all $i \in [n]$.

Lemma 4.3.3 (Lemma 2.4 of [126])**.** *With probability at least $1 - O(1/n)$,*

$$\max_{i\in[n]} \sum_{t\in[T]} \left[v_t(i) - X_i w_t\right] \le 4\eta T. \tag{4.25}$$

Lemma 4.3.4 (Lemma 2.5 of [126])**.** *With probability at least $1 - O(1/n)$,*

$$\Big|\sum_{t\in[T]} X_{i_t} w_t - \sum_{t\in[T]} p_t^\top v_t\Big| \le 10\eta T. \tag{4.26}$$

Lemma 4.3.5 (Lemma 2.6 of [126])**.** *With probability at least $3/4$,*

$$\sum_{t\in[T]} p_t^\top v_t^2 \le 8T. \tag{4.27}$$

Proof. We first prove the correctness of Algorithm 12. By Lemma 4.3.1, we have

$$\sum_{t\in[T]} X_{i_t} w_t \ge \max_{w\in\mathrm{B}_n} \sum_{t\in[T]} X_{i_t} w - 2\sqrt{2T} \ge T\sigma - 2\sqrt{2T}. \tag{4.28}$$

On the other hand, Lemma 4.3.3 implies that for any $i \in [n]$,

$$\sum_{t\in[T]} X_i w_t \ge \sum_{t\in[T]} v_t(i) - 4\eta T. \tag{4.29}$$

Together with Lemma 4.7.1, we have

$$\sum_{t\in[T]} p_t^\top v_t \le \min_{i\in[n]} \sum_{t\in[T]} X_i w_t + \eta \sum_{t\in[T]} p_t^\top v_t^2 + \frac{\log n}{\eta} + 4\eta T. \tag{4.30}$$

Plugging Lemma 4.3.4, Lemma 4.3.5, and (4.28) into (4.30), with probability at least $\frac{3}{4} - 2 \cdot O(\frac{1}{n}) \ge \frac{2}{3}$,

$$\min_{i\in[n]} \sum_{t\in[T]} X_i w_t \ge -\frac{\log n}{\eta} - 8\eta T - 4\eta T + T\sigma - 2\sqrt{2T} - 10\eta T \ge T\sigma - 22\eta T - \frac{\log n}{\eta}. \tag{4.31}$$

Since $T = 23^2\epsilon^{-2}\log n$ and $\eta = \sqrt{\frac{\log n}{T}}$, we have

$$\min_{i\in[n]} X_i \bar{w} = \frac{1}{T}\min_{i\in[n]} \sum_{t=1}^{T} X_i w_t \ge \sigma - 23\sqrt{\frac{\log n}{T}} \ge \sigma - \epsilon \tag{4.32}$$

with probability at least $2/3$, which is exactly (4.12).

Now we analyze the gate complexity of Algorithm 12. To run Line 3 and Line 5, we need d time and space to compute and store w_t and y_{t+1}; for all $t \in [T]$, this takes total complexity $O(dT) = \tilde{O}(\frac{d}{\epsilon^2})$. It takes another $O(dT) = \tilde{O}(\frac{d}{\epsilon^2})$ cost to compute j_t for all $t \in [T]$ in Line 6.

The quantum part of Algorithm 12 mainly happens at Line 7 and Line 8, where we prepare the quantum state $|p_{t+1}\rangle$ instead of computing the coefficients $u_{t+1}(i)$ one by one for all $i \in [n]$. To be more specific, we construct an oracle O_t such that $O_t|i\rangle|0\rangle = |i\rangle|u_{t+1}(i)\rangle$ for all $i \in [n]$. This is achieved iteratively, i.e., at iteration s we map $|i\rangle|u_s(i)\rangle$ to $|i\rangle|u_{s+1}(i)\rangle$. The full details are given in Algorithm 14 in Section 4.3.2.2; in total, one query to O_t is implemented by $2t$ queries to O_X and $\tilde{O}(t)$ additional arithmetic computations.

Finally, we prepare the state $|p_{t+1}\rangle = \frac{1}{\|u_{t+1}\|_2} \cdot \sum_{i\in[n]} u_{t+1}(i)|i\rangle$ in Line 8 using $O(\sqrt{n})$ calls to O_t, which are equivalent to $O(\sqrt{n}t)$ calls to O_X by Line 7 and $\tilde{O}(\sqrt{n}t)$ additional arithmetic computations. Therefore, the total complexity of Line 8 for all $t \in [T]$ is

$$\sum_{t=1}^{T} \tilde{O}(\sqrt{n}t) = \tilde{O}(\sqrt{n}T^2) = \tilde{O}\Big(\frac{\sqrt{n}}{\epsilon^4}\Big). \tag{4.33}$$

In all, the total complexity of Algorithm 12 is $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4} + \frac{d}{\epsilon^2}\big)$, establishing our statement.

Finally, the output $\bar{w}$ has a succinct classical representation with space complexity $O(\log n/\epsilon^2)$. To achieve this, we save $2T = O(\log n/\epsilon^2)$ values in Algorithm 12: $i_1, \ldots, i_T$ and $\|y_1\|, \ldots, \|y_T\|$; it then only takes $O(\log n/\epsilon^2)$ cost to recover any coordinate of $\bar{w}$ by Line 3 and Line 5. □

Remark 4.3.2. *Theorem 4.3.1 could also be applied to the PAC model. For the case where there exists a hyperplane classifying all data points correctly with margin σ, and assume that the margin is not small in the sense that $\frac{1}{\sigma^2} < d$, PAC learning theory implies that the number of examples needed for training a classifier of error δ is $O(1/\sigma^2\delta)$. As a result, we have a quantum algorithm that computes a $\sigma/2$-approximation to the best classifier with cost*

$$\tilde{O}\Big(\frac{\sqrt{1/\sigma^2\delta}}{\sigma^4} + \frac{d}{\sigma^2}\Big) = \tilde{O}\Big(\frac{1}{\sigma^5\sqrt{\delta}} + \frac{d}{\sigma^2}\Big). \tag{4.34}$$

This is better than the classical complexity $O(\frac{1}{\sigma^4\delta} + \frac{d}{\sigma^2})$ in [126] as long as $\delta \leq \sigma^2$, which is plausible under the assumption that the margin σ is large.

4.3.3 Quantum speedup for online gradient descent

Norm estimation by amplitude estimation. We further improve the dependence in d to $\tilde{O}(\sqrt{d})$. To achieve this, we cannot update w_t and y_t in Line 3 and Line 5 by each coordinate because storing w_t or y_t would already take cost at least d. We solve this issue by not updating w_t and y_t explicitly and instead only computing $\|y_t\|$ for all $i \in [T]$. This norm estimation is achieved by the following lemma:

Lemma 4.3.6. *Assume that $F\colon [d] \to [0,1]$ with a quantum oracle $O_F|i\rangle|0\rangle = |i\rangle|F(i)\rangle$ for all $i \in [d]$. Denote $m = \frac{1}{d}\sum_{i=1}^{d} F(i)$. Then for any $\delta > 0$, there is a quantum algorithm that uses $O(\sqrt{d}/\delta)$ queries to O_F and returns an $\tilde{m}$ such that $|\tilde{m} - m| \leq \delta m$ with probability at least 2/3.*

Our proof of Lemma 4.3.6 is based on amplitude estimation:

Theorem 4.3.2 (Theorem 15 of [57])**.** *For any $0 < \epsilon < 1$ and Boolean function $f\colon [d] \to \{0,1\}$ with quantum oracle $O_f|i\rangle|0\rangle = |i\rangle|f(i)\rangle$ for all $i \in [d]$, there is a quantum algorithm that outputs an estimate $\hat{t}$ to $t = |f^{-1}(1)|$ such that*

$$|\hat{t} - t| \leq \epsilon t \tag{4.35}$$

with probability at least $8/\pi^2$, using $O(\frac{1}{\epsilon}\sqrt{\frac{d}{t}})$ evaluations of O_f. If $t = 0$, the algorithm outputs $\hat{t} = 0$ with certainty and O_f is evaluated $O(\sqrt{d})$ times.

Proof. Assume that $F(i)$ has l bits for precision for all $i \in [d]$ (in our paper, we take $l = O(1)$, say $l = 64$ for double float precision), and for all $k \in [l]$ denote $F_k(i)$ as the k^{th} bit of $F(i)$; denote $n_k = \sum_{i\in[d]} F_k(i)$.

We apply Theorem 4.3.2 to all the l bits of n_k using $O(\sqrt{d}/\delta)$ queries (taking $\epsilon = \delta/2$), which gives an approximation $\hat{n}_k$ of n_k such that with probability at least $8/\pi^2$ we have $|n_k - \hat{n}_k| \leq \delta n_k/2$ if $n_k \geq 1$, and $\hat{n}_k = 0$ if $n_k = 0$. Running this procedure for $\Theta(\log l)$ times and take the median of all returned $\hat{n}_k$, and do this for all $k \in [l]$, Chernoff's bound promises that with probability $2/3$ we have

$$|n_k - \hat{n}_k| \leq \delta n_k \quad \forall\, k \in [l]. \tag{4.36}$$

As a result, if we take $\tilde{m} = \frac{1}{d}\sum_{k\in[l]} \frac{\hat{n}_k}{2^k}$, and observe that $m = \frac{1}{d}\sum_{k\in[l]} \frac{n_k}{2^k}$, with probability at least $2/3$ we have

$$|\tilde{m} - m| \leq \frac{1}{d}\sum_{k\in[l]} \left|\frac{\hat{n}_k}{2^k} - \frac{n_k}{2^k}\right| \leq \frac{1}{d}\sum_{k\in[l]} \frac{\delta n_k}{2^k} = \delta m. \tag{4.37}$$

The total quantum query complexity is $O(l \log l \cdot \sqrt{d}/\delta) = O(\sqrt{d}/\delta)$. □

Quantum algorithm with $\tilde{O}(\sqrt{d})$ cost. Instead of updating y_t explicitly in Line 5 of Algorithm 12, we save the i_t for all $t \in [T]$ in Line 4, which only takes $\tilde{O}(1/\epsilon^2)$ cost in total but we can directly generate y_t given $i_1, \ldots, i_t$. Furthermore, notice that the probabilities in the ℓ^2 sampling in Line 6 do not change because $w_t(j)^2/\|w_t\|^2 = y_t(j)^2/\|y_t\|^2$; it suffices to replace $\tilde{v}_t(i) = X_i(j_t)\|w_t\|^2/w_t(j_t)$ by $\tilde{v}_t(i) = X_i(j_t)\|y_t\|^2/(y_t(j_t)\max\{1, \|y_t\|\})$ in Line 7. These observations result in Algorithm 15 with the following result:

Theorem 4.3.3. *With success probability at least $2/3$, there is a quantum algorithm that returns* *a succinct classical representation of a vector $\bar{w} \in \mathbb{R}^d$ such that*

$$X_i \bar{w} \geq \max_{w \in \mathrm{B}_d} \min_{i' \in [n]} X_{i'} w - \epsilon \quad \forall\, i \in [n], \tag{4.38}$$

using $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4} + \frac{\sqrt{d}}{\epsilon^8}\big)$ quantum gates.

Algorithm 15: Quantum linear classification algorithm with $\tilde{O}(\sqrt{d})$ cost.

```
    Input: $\epsilon > 0$, a quantum oracle $O_X$ for $X \in \mathbb{R}^{n\times d}$.
    Output: $\bar{w}$ that satisfies (4.12).
1   Let $T = 27^2\epsilon^{-2}\log n$, $y_1 = \mathbf{0}_d$, $\eta = \sqrt{\frac{\log n}{T}}$, $u_1 = \mathbf{1}_n$, $|p_1\rangle = \frac{1}{\sqrt{n}}\sum_{i\in[n]}|i\rangle$;
2   for $t = 1$ to $T$ do
3       Measure the state $|p_t\rangle$ in the computational basis and denote the output as $i_t \in [n]$;
4       Define[6] $y_{t+1} := y_t + \frac{1}{\sqrt{2T}} X_{i_t}$;
5       Apply Lemma 4.3.6 for $2\lceil\log T\rceil$ times to estimate $\|y_t\|^2$ with precision $\delta = \eta^2$,
          and take the median of all the $2\lceil\log T\rceil$ outputs, denoted $\widetilde{\|y_t\|}^2$;
6       Choose $j_t \in [d]$ by $j_t = j$ with probability $y_t(j)^2/\|y_t\|^2$, which is achieved by
          applying Algorithm 13 to prepare the quantum state $|y_t\rangle$ and measure in the
          computational basis;
7       For all $i \in [n]$, denote $\tilde{v}_t(i) = X_i(j_t)\widetilde{\|y_t\|}^2/\big(y_t(j_t)\max\{1, \widetilde{\|y_t\|}\}\big)$,
          $v_t(i) = \mathrm{clip}(\tilde{v}_t(i), 1/\eta)$, and $u_{t+1}(i) = u_t(i)(1 - \eta v_t(i) + \eta^2 v_t(i)^2)$. Apply
          Algorithm 14 to prepare an oracle $O_t$ such that $O_t|i\rangle|0\rangle = |i\rangle|u_{t+1}(i)\rangle$ for all
          $i \in [n]$, using $2t$ queries to $O_X$ and $\tilde{O}(t)$ additional arithmetic computations;
8       Prepare the state $|p_{t+1}\rangle = \frac{1}{\|u_{t+1}\|_2}\sum_{i\in[n]} u_{t+1}(i)|i\rangle$ using Algorithm 13 and $O_t$;
9   Return $\bar{w} = \frac{1}{T}\sum_{t=1}^{T}\frac{y_t}{\max\{1, \widetilde{\|y_t\|}\}}$;
```

Proof. For clarification, we denote

$$\tilde{v}_{t,\text{approx}}(i) = \frac{X_i(j_t)\widetilde{\|y_t\|}^2}{y_t(j_t)\max\{1, \widetilde{\|y_t\|}\}}, \qquad \tilde{v}_{t,\text{true}}(i) = \frac{X_i(j_t)\|y_t\|^2}{y_t(j_t)\max\{1, \|y_t\|\}} \quad \forall\, i \in [n]. \tag{4.39}$$

[6]The meaning of the definition here is the same as Footnote 12 in Algorithm 12.

In other words, the $\tilde{v}_t$ in Line 7 of Algorithm 15 is $\tilde{v}_{t,\text{approx}}$, an approximation of $\tilde{v}_{t,\text{true}}$. We prove:

$$|\tilde{v}_{t,\text{approx}}(i) - \tilde{v}_{t,\text{true}}(i)| \le \eta \quad \forall\, i \in [n]. \tag{4.40}$$

Without loss generality, we can assume that $\tilde{v}_{t,\text{true}}(i), \tilde{v}_{t,\text{approx}}(i) \le 1/\eta$; otherwise, they are both truncated to $1/\eta$ by the clip function in Line 7 and no error occurs. For convenience, we denote $m = \|y_t\|^2$ and $\tilde{m} = \widetilde{\|y_t\|}^2$. Then

$$|\tilde{v}_{t,\text{approx}}(i) - \tilde{v}_{t,\text{true}}(i)| = \tilde{v}_{t,\text{true}}(i) \cdot \left|\frac{\tilde{v}_{t,\text{approx}}(i)}{\tilde{v}_{t,\text{true}}(i)} - 1\right| \le \frac{1}{\eta} \cdot \left|\frac{\tilde{v}_{t,\text{approx}}(i)}{\tilde{v}_{t,\text{true}}(i)} - 1\right|. \tag{4.41}$$

When $\|y_t\| \ge 1$ we have $\frac{\tilde{v}_{t,\text{approx}}(i)}{\tilde{v}_{t,\text{true}}(i)} = \frac{\tilde{m}}{m}$; when $\|y_t\| \le 1$ we have $\frac{\tilde{v}_{t,\text{approx}}(i)}{\tilde{v}_{t,\text{true}}(i)} = \sqrt{\frac{\tilde{m}}{m}}$. Because in Line 5 $\widetilde{\|y_t\|}^2$ is the median of $2\lceil \log T \rceil$ executions of Lemma 4.3.6, with failure probability at most $1 - (2/3)^{2\log T} = O(1/T^2)$ we have $|\frac{\tilde{m}}{m} - 1| \le \delta$; given there are T iterations in total, the probability that Line 5 always succeeds is at least $1 - T \cdot O(1/T^2) = 1 - o(1)$, and we have

$$\left|\frac{\tilde{m}}{m} - 1\right|, \left|\sqrt{\frac{\tilde{m}}{m}} - 1\right| \le \delta. \tag{4.42}$$

Plugging this into (4.41), we have

$$|\tilde{v}_{t,\text{approx}}(i) - \tilde{v}_{t,\text{true}}(i)| \le \frac{\delta}{\eta} = \eta, \tag{4.43}$$

which proves (4.40).

Now we prove the correctness of Algorithm 15. By (4.40) and Lemma 4.3.3, with probability at least $1 - O(1/n)$ we have

$$\max_{i\in[n]} \sum_{t\in[T]} \left[v_t(i) - X_i w_t\right] \leq 4\eta T + \eta T = 5\eta T, \tag{4.44}$$

where $w_t = \frac{y_t}{\max\{1, \widetilde{\|y_t\|}\}}$ for all $t \in [T]$. By (4.40) and Lemma 4.3.4, with probability at least $1 - O(1/n)$ we have

$$\Big|\sum_{t\in[T]} X_{i_t} w_t - \sum_{t\in[T]} p_t^\top v_t\Big| \leq 10\eta T + \eta T = 11\eta T; \tag{4.45}$$

by (4.40) and Lemma 4.3.5, with probability at least $3/4$ we have

$$\sum_{t\in[T]} p_t^\top v_t^2 \leq 8T + 2T = 10T. \tag{4.46}$$

As a result, similar to the proof of Theorem 4.3.1, we have

$$\min_{i\in[n]} \sum_{t\in[T]} X_i w_t \geq -\frac{\log n}{\eta} - 10\eta T - 5\eta T + T\sigma - 2\sqrt{2T} - 11\eta T \geq T\sigma - 26\eta T - \frac{\log n}{\eta}. \tag{4.47}$$

Since $T = 27^2\epsilon^{-2}\log n$ and $\eta = \sqrt{\frac{\log n}{T}}$, we have

$$\min_{i\in[n]} X_i \bar{w} = \frac{1}{T}\min_{i\in[n]} \sum_{t=1}^{T} X_i w_t \geq \sigma - 27\sqrt{\frac{\log n}{T}} \geq \sigma - \epsilon \tag{4.48}$$

with probability at least $2/3$, which is exactly (4.38).

It remains to analyze the time complexity. Same as the proof of Theorem 4.3.1, the complexity in n is $\tilde{O}(\frac{\sqrt{n}}{\epsilon^4})$. It remains to show that the complexity in d is $\tilde{O}(\frac{\sqrt{n}}{\epsilon^8})$. The cost in d in Algorithm 12 and Algorithm 15 differs at Line 5 and Line 6. We first look at Line 5; because

$$y_t = \frac{1}{\sqrt{2T}} \sum_{\tau=1}^{T} X_{i_\tau}, \tag{4.49}$$

one query to a coefficient of y_t takes $t = \tilde{O}(1/\epsilon^2)$ queries to O_X. Next, since $X_i \in \mathrm{B}_n$ for all $i \in [n]$, we know that $X_{ij} \in [-1, 1]$ for all $i \in [n]$, $j \in [d]$; to apply Lemma 4.3.6 (F should have image domain in $[0, 1]$) we need to renormalize y_t by a factor of $t = \tilde{O}(1/\epsilon^2)$. In addition, notice that $\delta = \eta^2 = \Theta(\epsilon^2)$; as a result, the query complexity of executing Lemma 4.3.6 is $\tilde{O}(\sqrt{d}/\epsilon^2)$. Finally, there are in total $T = \tilde{O}(1/\epsilon^2)$ iterations. Therefore, the total complexity in Line 5 is

$$\tilde{O}\Big(\frac{1}{\epsilon^2}\Big) \cdot \tilde{O}\Big(\frac{1}{\epsilon^2}\Big) \cdot \tilde{O}\Big(\frac{\sqrt{d}}{\epsilon^2}\Big) \cdot \tilde{O}\Big(\frac{1}{\epsilon^2}\Big) = \tilde{O}\Big(\frac{\sqrt{d}}{\epsilon^8}\Big). \tag{4.50}$$

Regarding the complexity in d in Line 6, the cost is to prepare the pure state $|y_t\rangle$ whose coefficient is proportional to y_t. To achieve this, we need $t = \tilde{O}(1/\epsilon^2)$ queries to O_X (for summing up the rows $X_{i_1}, \ldots, X_{i_t}$) such that we have an oracle O_{y_t} satisfying $O_{y_t}|j\rangle|0\rangle = |j\rangle|y_t(j)\rangle$ for all $j \in [d]$. By Algorithm 13, the query complexity of preparing $|y_t\rangle$ using O_{y_t} is $O(\sqrt{d})$. Because there are in total $T = \tilde{O}(1/\epsilon^2)$ iterations, the total complexity in Line 6 is

$$\tilde{O}\Big(\frac{1}{\epsilon^2}\Big) \cdot O(\sqrt{d}) \cdot \tilde{O}\Big(\frac{1}{\epsilon^2}\Big) = \tilde{O}\Big(\frac{\sqrt{d}}{\epsilon^4}\Big). \tag{4.51}$$

In all, the total complexity in d is $\tilde{O}(\sqrt{d}/\epsilon^8)$ as dominated by (4.50). Finally, $\bar{w}$ has a succinct classical representation: using $i_1, \ldots, i_T$ obtained from Line 3 and $\widetilde{\|y_1\|}^2, \ldots, \widetilde{\|y_T\|}^2$

obtained from Line 5, we could restore a coordinate of $\bar{w}$ in time $T = \tilde{O}(1/\epsilon^2)$. $\square$

Remark 4.3.3. *For practical applications of linear classification, typically the number of data points n is larger than the dimension d, so in practice Theorem 4.3.1 might perform better than Theorem 4.3.3. Nevertheless, the $\tilde{O}(\sqrt{d})$ complexity in Theorem 4.3.3 matches our quantum lower bound (see Theorem 4.5.1).*

4.4 Applications

As introduced in Section 4.3.1, the ℓ^2 sampling of w picks $j_t \in [d]$ by $j_t = j$ with probability $w(j)^2/\|w\|^2$, and the expectation of the random variable $X_i(j_t)\|w\|^2/w(j_t)$ is $X_i w$. Here, if we consider some alternate random variables, we could give unbiased estimators of nonlinear functions of X. We first look at the general case of applying kernel functions [152] in Section 4.4.1. We then look at the special case of quadratic problems in Section 4.4.2 as they enjoy simple forms that can be applied to finding minimum enclosing balls [153] and ℓ^2-margin support vector machines [154]. Finally, we follow this methodology to give a sublinear quantum algorithm for solving matrix zero-sum games in Section 4.4.3.

4.4.1 Kernel methods

Having quantum algorithms for solving linear classification at hand, it is natural to consider linear classification under kernels. Let $\Psi\colon \mathbb{R}^d \mapsto \mathcal{H}$ be a mapping into a reproducing kernel Hilbert space (RKHS), and the problem is to find the classifier $h \in \mathcal{H}$ that solves the maximin problem

$$\sigma = \max_{h\in\mathcal{H}} \min_{i\in[n]} \langle h, \Psi(X_i)\rangle, \tag{4.52}$$

where the kernel is defined as $k(a,b) := \langle \Psi(a), \Psi(b)\rangle$ for all $a, b \in \mathbb{R}^d$.

Classically, [126] gave the following result for classification under efficiently-computable kernels, following the linear classification algorithm therein:

Theorem 4.4.1 (Lemma 5.3 of [126])**.** *Denote T_k as the time cost for computing $k(X_i, X_j)$ for some $i, j \in [n]$, and denote L_k as the time cost for computing a random variable $\tilde{k}(X_i, X_j)$ for some $i, j \in [n]$ such that $\mathbb{E}[\tilde{k}(X_i, X_j)] = k(X_i, X_j)$ and $\mathrm{Var}[k(X_i, X_j)] \leq 1$. Then there is a classical algorithm that runs in time*

$$\tilde{O}\Big(\frac{L_k n + d}{\epsilon^2} + \min\Big\{\frac{T_k}{\epsilon^4}, \frac{L_k}{\epsilon^6}\Big\}\Big) \tag{4.53}$$

and returns a vector $\bar{h} \in \mathcal{H}$ such that with high success probability $\langle \bar{h}, \Psi(X_i)\rangle \geq \sigma - \epsilon$ for all $i \in [n]$.

Quantumly, we give an algorithm for classification under kernels based on Algorithm 15:

Theorem 4.3.3 and Theorem 4.4.1 imply that our quantum kernel-based classifier has time complexity

$$\tilde{O}\Big(\frac{L_k\sqrt{n}}{\epsilon^4} + \frac{\sqrt{d}}{\epsilon^8} + \min\Big\{\frac{T_k}{\epsilon^4}, \frac{L_k}{\epsilon^6}\Big\}\Big). \tag{4.54}$$

For polynomial kernels of degree q, i.e., $k_q(x,y) = (x^\top y)^q$, we have $L_{k_q} = q$ by taking the

Algorithm 16: Quantum algorithm for kernel-based classification.

```
Input: $\epsilon > 0$, a quantum oracle $O_X$ for $X \in \mathbb{R}^{n\times d}$.
Output: $\bar{w}$ that satisfies (4.12).
1 Let $T = 27^2\epsilon^{-2}\log n$, $y_1 = \mathbf{0}_d$, $\eta = \sqrt{\frac{\log n}{T}}$, $u_1 = \mathbf{1}_n$, $|p_1\rangle = \frac{1}{\sqrt{n}}\sum_{i\in[n]}|i\rangle$;
2 for $t = 1$ to $T$ do
3     Measure the state $|p_t\rangle$ in the computational basis and denote the output as $i_t \in [n]$;
4     Define $y_{t+1} := y_t + \frac{1}{\sqrt{2T}}\Psi(X_{i_t})$;
5     Apply Lemma 4.3.6 for $2\lceil\log T\rceil$ times to estimate $\|y_t\|^2$ with precision $\delta = \eta^2$,
       and take the median of all the $2\lceil\log T\rceil$ outputs, denoted $\widetilde{\|y_t\|}^2$;
6     Choose $j_t \in [d]$ by $j_t = j$ with probability $y_t(j)^2/\|y_t\|^2$, which is achieved by
       applying Algorithm 13 to prepare the quantum state $|y_t\rangle$ and measure in the
       computational basis;
7     For all $i \in [n]$, denote $\tilde{v}_t(i) = \Psi(X_i)(j_t)\widetilde{\|y_t\|}^2/\big(y_t(j_t)\max\{1, \widetilde{\|y_t\|}\}\big)$,
       $v_t(i) = \mathrm{clip}(\tilde{v}_t(i), 1/\eta)$, and $u_{t+1}(i) = u_t(i)(1 - \eta v_t(i) + \eta^2 v_t(i)^2)$. Apply
       Algorithm 14 to prepare an oracle $O_t$ such that $O_t|i\rangle|0\rangle = |i\rangle|u_{t+1}(i)\rangle$ for all
       $i \in [n]$, using $2t$ queries to $O_X$ and $\tilde{O}(t)$ additional arithmetic computations;
8     Prepare the state $|p_{t+1}\rangle = \frac{1}{\|u_{t+1}\|_2}\sum_{i\in[n]} u_{t+1}(i)|i\rangle$ using Algorithm 13 and $O_t$;
9 Return $\bar{w} = \frac{1}{T}\sum_{t=1}^{T} \frac{y_t}{\max\{1, \widetilde{\|y_t\|}\}}$;
```

product of q independent ℓ^2 samples (this is an unbiased estimator of $(x^\top y)^q$ and the variance of each sample is at most 1). As a result of (4.54),

Corollary 4.4.1. *For the polynomial kernel of degree q, there is a quantum algorithm that solves the classification task within precision ϵ with gate complexity $\tilde{O}\big(\frac{q\sqrt{n}}{\epsilon^4} + \frac{q\sqrt{d}}{\epsilon^8}\big)$.*

Compared to the classical complexity $\tilde{O}\big(\frac{q(n+d)}{\epsilon^2} + \min\big\{\frac{d\log q}{\epsilon^4}, \frac{q}{\epsilon^6}\big\}\big)$ in Corollary 5.4 of [126], our quantum algorithm gives quadratic speedups in n and d.

For Gaussian kernels, i.e., $k_{\text{Gauss}}(x, y) = \exp(-\|x - y\|^2)$, Corollary 5.5 of [126] proved that $L_{k_{\text{Gauss}}} = 1/s^4$ if the Gaussian has standard deviation s. As a result,

Corollary 4.4.2. *For the polynomial kernel of degree q, there is a quantum algorithm that solves the classification task within precision ϵ with gate complexity $\tilde{O}\big(\frac{\sqrt{n}}{s^4\epsilon^4} + \frac{\sqrt{d}}{s^4\epsilon^8}\big)$.*

This still gives quadratic speedups in n and d compared to the classical complexity $\tilde{O}\big(\frac{n+d}{s^4\epsilon^2} +$

$\min\left\{\frac{d}{\epsilon^4}, \frac{1}{s^4\epsilon^6}\right\})$ in Corollary 5.5 of [126].

4.4.2 Quadratic machine learning problems

We consider the maximin problem of a quadratic function:

$$\max_{w\in\mathbb{R}^d}\min_{p\in\Delta_n} p^\top(b + 2Xw - \mathbf{1}_n\|w\|^2) = \max_{w\in\mathbb{R}^d}\min_{i\in[n]} b_i + 2X_i w - \|w\|^2, \tag{4.55}$$

where $b \in \mathbb{R}^n$ and $X \in \mathbb{R}^{n\times d}$. Note that the function $b_i + 2X_i w - \|w\|^2$ in Eq. (4.55) is 2-strongly convex; as a result, the regret of the online gradient descent after T rounds can be improved to $O(\log T)$ by [155] instead of $O(\sqrt{T})$ as in Eq. (4.10). In addition, ℓ^2 sampling of the w in Algorithm 12 and Algorithm 15 still works: consider the random variable $w = b_i + \frac{2X_i(j)\|w\|^2}{w(j)} - \|w\|^2$ where $j = k$ with probability $\frac{w(k)^2}{\|w\|^2}$. Then the expectation of w is

$$\mathbb{E}[X] = \sum_{j=1}^{d}\frac{w(j)^2}{\|w\|^2}\Big(b_i + \frac{2X_i(j)\|w\|^2}{w(j)} - \|w\|^2\Big) = b_i + 2X_i w - \|w\|^2, \tag{4.56}$$

i.e., w is an unbiased estimator of the quadratic form in (4.55). As a result, given the quantum oracle O_X in (4.8), we could give sublinear quantum algorithms for such problems; these include two important problems: minimum enclosing balls (MEB) and ℓ^2-margin supper vector machines (SVM).

4.4.2.1 Minimum enclosing ball

In the minimum enclosing ball (MEB) problem we have $b_i = -\|X_i\|^2$ for all $i \in [n]$; Eq. (4.55) then becomes $\max_{w\in\mathbb{R}^d}\min_{i\in[n]} -\|X_i\|^2 + 2X_i w - \|w\|^2 = -\min_{w\in\mathbb{R}^d}\max_{i\in[n]}\|w - X_i\|^2$, which is the smallest radius of the balls that contain all the n data points $X_1, \ldots, X_n$.

Denote $\sigma_{\text{MEB}} = \min_{w\in\mathbb{R}^d} \max_{i\in[n]} \|w - X_i\|^2$, we have:

Theorem 4.4.2. *There is a quantum algorithm that returns a vector $\bar{w} \in \mathbb{R}^d$ such that with probability at least $2/3$,*

$$\max_{i\in[n]} \|\bar{w} - X_i\|^2 \leq \sigma_{\text{MEB}} + \epsilon, \tag{4.57}$$

using $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4} + \frac{d}{\epsilon}\big)$ quantum gates; the quantum gate complexity can also be improved to $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4} + \frac{\sqrt{d}}{\epsilon^7}\big)$.

We omit the proof of Theorem 4.4.2 because it directly follows from Theorem 4.3.1 (see also Theorem 3.1 in [126]) and Theorem 4.3.3. For the $\tilde{O}(\sqrt{d})$ complexity result, the same idea of Algorithm 15 is applied to estimate the norm $\|y_t\|$ by amplitude estimation; the error dependence becomes $1/\epsilon^7$ because with high probability, the number of iterations that we obtain a new y_t in Line 4 is $O(\alpha T) = \tilde{O}(1/\epsilon)$, and the other overheads in ϵ is still $\tilde{O}(\sqrt{d}/\epsilon^6)$ (see Eq. (4.50)).

4.4.2.2 ℓ^2-margin SVM

To estimate the margin of a support vector machine (SVM) in ℓ^2-norm, we take $b_i = 0$ for all $i \in [n]$; Eq. (4.55) then becomes solving $\sigma_{\text{SVM}} := \max_{w\in\mathbb{R}^d} \min_{i\in[n]} 2X_i w - \|w\|^2$.

Notice that $\sigma_{\text{SVM}} \geq 0$ because $2X_i w - \|w\|^2 = 0$ for all $i \in [n]$ when $w = 0$. For the case $\sigma_{\text{SVM}} > 0$ and taking $0 < \epsilon < \sigma_{\text{SVM}}$, similar to Theorem 4.4.2 we have:

Corollary 4.4.3. *There is a quantum algorithm that returns a vector $\bar{w} \in \mathbb{R}^d$ such that with* *probability at least* $2/3$*,*

$$\min_{i\in[n]} 2X_i\bar{w} - \|\bar{w}\|^2 \geq \sigma_{\mathrm{SVM}} - \epsilon > 0, \tag{4.58}$$

using $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4} + \frac{d}{\epsilon}\big)$ *quantum gates; the quantum gate complexity can also be improved to* $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4} + \frac{\sqrt{d}}{\epsilon^7}\big)$.

Note that (4.135) implies that $X_i\bar{w} > 0$ for all $i \in [n]$; furthermore, by the AM-GM inequality we have $\frac{(X_i\bar{w})^2}{\|\bar{w}\|^2} + \|\bar{w}\|^2 \geq 2X_i\bar{w}$, and hence

$$\min_{i\in[n]} \Big(\frac{X_i\bar{w}}{\|\bar{w}\|}\Big)^2 \geq \min_{i\in[n]} 2X_i\bar{w} - \|\bar{w}\|^2 \geq \sigma_{\mathrm{SVM}} - \epsilon. \tag{4.59}$$

If we denote $\hat{w} = \bar{w}/\|\bar{w}\|$, then $X_i\hat{w} \geq \sqrt{\sigma_{\mathrm{SVM}} - \epsilon} > 0$ for all $i \in [n]$. Consequently, if the data X is from an SVM, we obtain a normalized direction $\hat{w}$ (in ℓ^2-norm) such that all data points have a margin of at least $\sqrt{\sigma_{\mathrm{SVM}} - \epsilon}$. Classically, this task takes time $\tilde{O}(n+d)$ for constant σ_{SVM} by [126], but our quantum algorithm only takes time $\tilde{O}(\sqrt{n} + \sqrt{d})$.

4.4.3 Matrix zero-sum games

Our ℓ^2-sampling technique can also be adapted to solve matrix zero-sum games as an application. To be more specific, the input of a zero-sum game is a matrix $X \in \mathbb{R}^{n_1\times n_2}$, and the goal is to find $a \in \mathbb{R}^{n_1}$ and $b \in \mathbb{R}^{n_2}$ such that

$$a^\dagger X b \geq \max_{p\in\Delta_{n_1}} \min_{q\in\Delta_{n_2}} p^\dagger X q - \epsilon \tag{4.60}$$

for some $\epsilon > 0$; such (a, b) is called an *ϵ-optimal strategy*. It is shown in [156, Proposition 1] that for $0 < \epsilon < 0.1$, an ϵ-optimal strategy for the $(n_1 + n_2 + 1)$-dimensional anti-symmetric matrix

$$X' = \begin{pmatrix} 0 & X & -\mathbf{1}_{n_1} \\ -X^\dagger & 0 & \mathbf{1}_{n_2} \\ \mathbf{1}_{n_1} & -\mathbf{1}_{n_2} & 0 \end{pmatrix} \tag{4.61}$$

implies an 18ϵ-optimal strategy for X. Therefore, without loss of generality, we could assume that X is an n-dimensional anti-symmetric matrix (by taking $n = n_1 + n_2 + 1$). In this case, the game value $\max_{p\in\Delta_n} \min_{q\in\Delta_n} p^\dagger X q$ in (4.60) equals to 0, and due to symmetry finding an ϵ-optimal strategy reduces to find an $w \in \Delta_n$ such that

$$Xw \leq \epsilon \cdot \mathbf{1}_n, \tag{4.62}$$

where $\leq$ applies to each coordinate. As a normalization, we assume that $\max_{i,j\in[n]} |X_{i,j}| \leq 1$.

Classically, one query to X is to ask for one entry in the matrix, whereas quantumly we assume the oracle in (4.8). Inspired by Ref. [156, Theorem 1], we give the following result for solving the zero-sum game:

Theorem 4.4.3. *With success probability at least $2/3$, Algorithm 17 returns a vector $\bar{w} \in \mathbb{R}^n$ such that $X\bar{w} \leq \epsilon \cdot \mathbf{1}_n$, using $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4}\big)$ quantum gates.*

Proof. We first prove the correctness of Algorithm 17. We denote $P_i(t) := \exp[\epsilon \sum_{\tau=1}^{t} X_{i,k_\tau}/2]$ and $p_i(t) = P_i(t)/\sum_{j=1}^{n} P_j(t)$ for all $i \in [n]$ and $t \in [T]$. Then $|p_{t+1}\rangle = \sum_{i=1}^{n} \sqrt{p_i(t)}|i\rangle$. We

Algorithm 17: Sublinear quantum algorithm for solving zero-sum games.

```
Input: $\epsilon > 0$, a quantum oracle $O_X$ for $X \in \mathbb{R}^{n\times n}$.
Output: $\bar{w} \in \Delta_n$ that satisfies (4.62).
1 Let $T \leftarrow 4\epsilon^{-2}\log n$, $A \leftarrow \mathbf{0}_n$, $|p_1\rangle \leftarrow \frac{1}{\sqrt{n}}\sum_{i\in[n]}|i\rangle$;
2 for $t = 1$ to $T$ do
3     Measure the state $|p_t\rangle$ in the computational basis and denote the output as $k_t \in [n]$;
4     Update the $k_t$-th coordinate of $A$: $A_{k_t} \leftarrow A_{k_t} + 1$;
5     Prepare the state
          $$|p_{t+1}\rangle = \frac{\sum_{i\in[n]}\exp[\epsilon\sum_{\tau=1}^{t}X_{i,k_\tau}/4]|i\rangle}{\sqrt{\sum_{j\in[n]}\exp[\epsilon\sum_{\tau=1}^{t}X_{j,k_\tau}/2]}} \quad (4.63)$$
       using Algorithm 13;
6 Return $\bar{w} = A/T$;
```

also denote the potential function $\Phi(t) = \sum_{i=1}^{n} P_i(t)$. It satisfies

$$\Phi(t) = \sum_{i=1}^{n} P_i(t) = \sum_{i=1}^{n} P_i(t-1)\exp[\epsilon X_{i,k_t}/2] = \Phi(t-1)\sum_{i=1}^{n} p_i(t-1)\exp[\epsilon X_{i,k_t}/2]. \quad (4.64)$$

Since Line 3 selects k_t with probability $p_{k_t}(t-1)$, Eq. (4.64) implies

$$\mathbb{E}[\Phi(t)] = \Phi(t-1)\sum_{i,k=1}^{n} p_i(t-1)p_k(t-1)\exp[\epsilon X_{i,k}/2]. \quad (4.65)$$

Because $|X_{i,k}| \leq 1$, we have

$$\exp[\epsilon X_{i,k}/2] \leq 1 - \frac{\epsilon X_{i,k}}{2} + \frac{\epsilon^2}{6}. \quad (4.66)$$

Also because X is skew-symmetric, we have $\sum_{i,k=1}^{n} p_i(t-1)p_k(t-1)X_{i,k} = 0$. Plugging this and (4.66) into (4.65), we have $\mathbb{E}[\Phi(t)] \leq \mathbb{E}[\Phi(t-1)]\big(1+\frac{\epsilon^2}{6}\big)$. As a result of induction,

$$\mathbb{E}[\Phi(T)] \leq \Phi(0)\Big(1+\frac{\epsilon^2}{6}\Big)^T \leq n\exp[T\epsilon^2/6] \leq n^{5/3}. \tag{4.67}$$

By Markov's inequality, we have $\Phi(T) \leq 3n^{5/3} \leq n^2$ with probability at least $2/3$. Notice that $\Phi(T) \leq n^2$ implies $P_i(T) \leq n^2$ for all $i \in [n]$, i.e., $\epsilon\sum_{\tau=1}^{T} X_{i,k_\tau}/2 \leq 2\ln n$ for all $i \in [n]$. The i-th coordinate of $X\bar{w}$ satisfies

$$(X\bar{w})_i = \frac{1}{T}(XA)_i = \frac{1}{T}\sum_{\tau=1}^{T} X_{i,k_\tau} \leq \frac{\epsilon^2}{4\ln n}\cdot\frac{4\ln n}{\epsilon} = \epsilon; \tag{4.68}$$

since this is true for all $i \in [n]$, we have $X\bar{w} \leq \epsilon\cdot\mathbf{1}_n$.

It remains to prove the complexity claim. The measurement in Line 3 takes $O(\log n)$ gates, and the update in Line 4 also takes $O(\log n)$ gates because it only adds 1 to one of the n coordinates. The complexity of the algorithm thus mainly comes from Line 5 for preparing $|p_{t+1}\rangle$. Notice that $\arg\max \exp[\epsilon\sum_{\tau=1}^{t} X_{i,k_\tau}/4] = \arg\max \sum_{\tau=1}^{t} X_{i,k_\tau}$, which can be computed in $O(t\sqrt{n})$ queries to the oracle O_X. Similarly, the amplitude amplification in Algorithm 13 can also be done with cost $O(t\sqrt{n})$. In total, the time complexity of Algorithm 17 is

$$\sum_{t=1}^{T} O(t\sqrt{n}) = O(T^2\sqrt{n}) = \tilde{O}\Big(\frac{\sqrt{n}}{\epsilon^4}\Big). \tag{4.69}$$

□

Remark 4.4.1. *The output of Algorithm 17 is a classical vector in Δ_n; furthermore, it has a succinct representation of $O(\log^2 n/\epsilon^2)$ bits: Line 4 in each iteration add 1 to one of the n*

coordinates and hence can be stored in $\lceil \log_2 n \rceil$ bits, and there are in total $O(\log n/\epsilon^2)$ rounds. Therefore, such output can be directly useful for classical applications, which distinguishes from many quantum machine learning algorithms that output a quantum state (whose applications are more subtle).

4.5 Quantum lower bounds

All quantum algorithms (upper bounds) above have matching lower bounds in n and d. Assuming $\epsilon = \Theta(1)$ and given the oracle O_X in (4.8), we prove quantum lower bounds on linear classification, minimum enclosing ball, and matrix zero-sum games in Section 4.5.1, Section 4.5.2, and Section 4.5.3, respectively.

4.5.1 Linear classification

Recall that the input of the linear classification problem is a matrix $X \in \mathbb{R}^{n\times d}$ such that $X_i \in \mathrm{B}_d$ for all $i \in [n]$ (X_i being the i^{th} row of X), and the goal is to approximately solve

$$\sigma := \max_{w\in \mathrm{B}_d} \min_{p\in\Delta_n} p^\top X w = \max_{w\in\mathrm{B}_d} \min_{i\in[n]} X_i w. \tag{4.70}$$

Given the quantum oracle O_X such that $O_X|i\rangle|j\rangle|0\rangle = |i\rangle|j\rangle|X_{ij}\rangle\ \forall\, i \in [n], j \in [d]$, Theorem 4.3.3 solves this task with high success probability with cost $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4} + \frac{\sqrt{d}}{\epsilon^8}\big)$. We prove a quantum lower bound that matches this upper bound in n and d for constant ϵ:

Theorem 4.5.1. *Assume* $0 < \epsilon < 0.04$. *Then to return an* $\bar{w} \in \mathrm{B}_d$ *satisfying*

$$X_j \bar{w} \geq \max_{w \in \mathrm{B}_d} \min_{i \in [n]} X_i w - \epsilon \quad \forall j \in [n] \tag{4.71}$$

with probability at least $2/3$, *we need* $\Omega(\sqrt{n} + \sqrt{d})$ *quantum queries to* O_X.

Proof. Assume we are given the promise that X is from one of the two cases below:

1. There exists an $l \in \{2, \ldots, d\}$ such that $X_{11} = -\frac{1}{\sqrt{2}}$, $X_{1l} = \frac{1}{\sqrt{2}}$; $X_{21} = X_{2l} = \frac{1}{\sqrt{2}}$; there exists a unique $k \in \{3, \ldots, n\}$ such that $X_{k1} = 1$, $X_{kl} = 0$; $X_{ij} = \frac{1}{\sqrt{2}}$ for all $i \in \{3, \ldots, n\}/\{k\}$, $j \in \{1, l\}$, and $X_{ij} = 0$ for all $i \in [n]$, $j \notin \{1, l\}$.

2. There exists an $l \in \{2, \ldots, d\}$ such that $X_{11} = -\frac{1}{\sqrt{2}}$, $X_{1l} = \frac{1}{\sqrt{2}}$; $X_{21} = X_{2l} = \frac{1}{\sqrt{2}}$; $X_{ij} = \frac{1}{\sqrt{2}}$ for all $i \in \{3, \ldots, n\}$, $j \in \{1, l\}$, and $X_{ij} = 0$ for all $i \in [n]$, $j \notin \{1, l\}$.

Notice that the only difference between these two cases is a row where the first entry is 1 and the

l^{th} entry is 0; they have the following pictures, respectively:

$$\text{Case 1: } X = \begin{pmatrix} -\frac{1}{\sqrt{2}} & 0 & \cdots & 0 & \frac{1}{\sqrt{2}} & 0 & \cdots & 0 \\ \frac{1}{\sqrt{2}} & 0 & \cdots & 0 & \frac{1}{\sqrt{2}} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{\sqrt{2}} & 0 & \cdots & 0 & \frac{1}{\sqrt{2}} & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \frac{1}{\sqrt{2}} & 0 & \cdots & 0 & \frac{1}{\sqrt{2}} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{\sqrt{2}} & 0 & \cdots & 0 & \frac{1}{\sqrt{2}} & 0 & \cdots & 0 \end{pmatrix}; \tag{4.72}$$

and

$$\text{Case 2: } X = \begin{pmatrix} -\frac{1}{\sqrt{2}} & 0 & \cdots & 0 & \frac{1}{\sqrt{2}} & 0 & \cdots & 0 \\ \frac{1}{\sqrt{2}} & 0 & \cdots & 0 & \frac{1}{\sqrt{2}} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{\sqrt{2}} & 0 & \cdots & 0 & \frac{1}{\sqrt{2}} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{\sqrt{2}} & 0 & \cdots & 0 & \frac{1}{\sqrt{2}} & 0 & \cdots & 0 \end{pmatrix}. \tag{4.73}$$

We denote the maximin value in (4.70) of these cases as σ_1 and σ_2, respectively. We have:

- $\sigma_2 = \frac{1}{\sqrt{2}}$.

On the one hand, consider $\bar{w} = \vec{e_l} \in \mathrm{B}_d$ (the vector in $\mathbb{R}^d$ with the l^{th} coordinate being 1 and all other coordinates being 0). Then $X_i\bar{w} = \frac{1}{\sqrt{2}}$ for all $i \in [n]$, and hence $\sigma_2 \geq \min_{i\in[n]} X_i\bar{w} = \frac{1}{\sqrt{2}}$. On the other hand, for any $w = (w_1, \dots, w_d) \in \mathrm{B}_d$, we have

$$\min_{i\in[n]} X_i w = \min\Big\{ -\frac{1}{\sqrt{2}}w_1 + \frac{1}{\sqrt{2}}w_l, \frac{1}{\sqrt{2}}w_1 + \frac{1}{\sqrt{2}}w_l \Big\} \leq \frac{1}{\sqrt{2}}w_l \leq \frac{1}{\sqrt{2}}, \tag{4.74}$$

where the first inequality comes from the fact that $\min\{a, b\} \leq \frac{a+b}{2}$ for all $X, b \in \mathbb{R}$ and the second inequality comes from the fact that $w \in \mathrm{B}_d$ and $|w_l| \leq 1$. As a result, $\sigma_2 = \max_{w\in\mathrm{B}_d} \min_{i\in[n]} X_i w \leq \frac{1}{\sqrt{2}}$. In conclusion, we have $\sigma_2 = \frac{1}{\sqrt{2}}$.

- $\sigma_1 = \frac{1}{\sqrt{4+2\sqrt{2}}}$.

On the one hand, consider $\bar{w} = \frac{1}{\sqrt{4+2\sqrt{2}}}\vec{e_1} + \frac{\sqrt{2}+1}{\sqrt{4+2\sqrt{2}}}\vec{e_l} \in \mathrm{B}_d$. Then

$$X_1\bar{w} = -\frac{1}{\sqrt{2}} \cdot \frac{1}{\sqrt{4+2\sqrt{2}}} + \frac{1}{\sqrt{2}} \cdot \frac{\sqrt{2}+1}{\sqrt{4+2\sqrt{2}}} = \frac{1}{\sqrt{4+2\sqrt{2}}}; \tag{4.75}$$

$$X_i\bar{w} = \frac{1}{\sqrt{2}} \cdot \frac{1}{\sqrt{4+2\sqrt{2}}} + \frac{1}{\sqrt{2}} \cdot \frac{\sqrt{2}+1}{\sqrt{4+2\sqrt{2}}} = \frac{\sqrt{2}+1}{\sqrt{4+2\sqrt{2}}} > \frac{1}{\sqrt{4+2\sqrt{2}}} \quad \forall\, i \in [n]/\{1, k\}; \tag{4.76}$$

$$X_k\bar{w} = 1 \cdot \frac{1}{\sqrt{4+2\sqrt{2}}} + 0 \cdot \frac{\sqrt{2}+1}{\sqrt{4+2\sqrt{2}}} = \frac{1}{\sqrt{4+2\sqrt{2}}}. \tag{4.77}$$

In all, $\sigma_1 \geq \min_{i\in[n]} X_i\bar{w} = \frac{1}{\sqrt{4+2\sqrt{2}}}$.

On the other hand, for any $w = (w_1, \dots, w_d) \in \mathrm{B}_d$, we have

$$\min_{i\in[n]} X_i w = \min\Big\{ -\frac{1}{\sqrt{2}}w_1 + \frac{1}{\sqrt{2}}w_l, \frac{1}{\sqrt{2}}w_1 + \frac{1}{\sqrt{2}}w_l, w_1 \Big\}. \tag{4.78}$$

If $w_1 \leq \frac{1}{\sqrt{4+2\sqrt{2}}}$, then (4.78) implies that $\min_{i\in[n]} X_i w \leq \frac{1}{\sqrt{4+2\sqrt{2}}}$; if $w_1 \geq \frac{1}{\sqrt{4+2\sqrt{2}}}$, then

$$w_l \leq \sqrt{1-w_1^2} = \sqrt{1-\frac{1}{4+2\sqrt{2}}} = \frac{\sqrt{2}+1}{\sqrt{4+2\sqrt{2}}}, \tag{4.79}$$

and hence by (4.78) we have

$$\min_{i\in[n]} X_i w \leq -\frac{1}{\sqrt{2}}w_1 + \frac{1}{\sqrt{2}}w_l \leq -\frac{1}{\sqrt{2}}\cdot\frac{1}{\sqrt{4+2\sqrt{2}}} + \frac{1}{\sqrt{2}}\cdot\frac{\sqrt{2}+1}{\sqrt{4+2\sqrt{2}}} = \frac{1}{\sqrt{4+2\sqrt{2}}}. \tag{4.80}$$

In all, we always have $\min_{i\in[n]} X_i w \leq \frac{1}{\sqrt{4+2\sqrt{2}}}$. As a result, $\sigma_1 = \max_{w\in\mathrm{B}_d}\min_{i\in[n]} X_i w \leq \frac{1}{\sqrt{4+2\sqrt{2}}}$. In conclusion, we have $\sigma_1 = \frac{1}{\sqrt{4+2\sqrt{2}}}$.

Now, we prove that an $\bar{w} \in \mathrm{B}_d$ satisfying (4.71) would simultaneously reveal whether X is from Case 1 or Case 2 as well as the value of $l \in \{2,\ldots,d\}$, by the following algorithm:

1. Check if one of $\bar{w}_2,\ldots,\bar{w}_d$ is larger than 0.94; if there exists an $l' \in \{2,\ldots,d\}$ such that $\bar{w}_{l'} > 0.94$, return 'Case 2' and $l = l'$;
2. Otherwise, return 'Case 1' and $l = \arg\max_{i\in\{2,\ldots,d\}} \bar{w}_i$.

We first prove that the classification of X (between Case 1 and Case 2) is correct. On the one hand, assume that X comes from Case 1. If we wrongly classified X as from Case 2, we would have $\bar{w}_{l'} > 0.94$ and $\bar{w}_1 < \sqrt{1-0.94^2} < 0.342$; this would imply

$$\min_{i\in[n]} X_i\bar{w} = \min\Big\{-\frac{1}{\sqrt{2}}\bar{w}_1 + \frac{1}{\sqrt{2}}\bar{w}_l, \frac{1}{\sqrt{2}}\bar{w}_1 + \frac{1}{\sqrt{2}}\bar{w}_l, \bar{w}_1\Big\} \leq \bar{w}_1 < \frac{1}{\sqrt{4+2\sqrt{2}}} - 0.04 \leq \sigma_1 - \epsilon \tag{4.81}$$

by $0.342 < \frac{1}{\sqrt{4+2\sqrt{2}}} - 0.04$, contradicts with (4.71). Therefore, for this case we must make correct classification that X comes from Case 1.

On the other hand, assume that X comes from Case 2. If we wrongly classified X as from Case 1, we would have $\bar{w}_l \leq \max_{i\in\{2,\ldots,d\}} \bar{w}_i \leq 0.94$; this would imply

$$\min_{i\in[n]} X_i\bar{w} = \min\left\{-\frac{1}{\sqrt{2}}\bar{w}_1 + \frac{1}{\sqrt{2}}\bar{w}_l, \frac{1}{\sqrt{2}}\bar{w}_1 + \frac{1}{\sqrt{2}}\bar{w}_l\right\} \leq \frac{1}{\sqrt{2}}\bar{w}_l < \frac{1}{\sqrt{2}} - 0.04 \leq \sigma_2 - \epsilon \tag{4.82}$$

by $\frac{0.94}{\sqrt{2}} < \frac{1}{\sqrt{2}} - 0.04$, contradicts with (4.71). Therefore, for this case we must make correct classification that X comes from Case 2. In all, our classification is always correct.

It remains to prove that the value of l is correct. If X is from Case 1, we have

$$\sigma_1 - \epsilon \leq \min_{i\in[n]} X_i\bar{w} = \min\left\{-\frac{1}{\sqrt{2}}\bar{w}_1 + \frac{1}{\sqrt{2}}\bar{w}_l, \frac{1}{\sqrt{2}}\bar{w}_1 + \frac{1}{\sqrt{2}}\bar{w}_l, \bar{w}_1\right\}; \tag{4.83}$$

as a result, $\bar{w}_1 \geq \sigma_1 - \epsilon > 0.38 - 0.04 = 0.34$, and

$$-\frac{1}{\sqrt{2}}\bar{w}_1 + \frac{1}{\sqrt{2}}\bar{w}_l > 0.34 \quad\Longrightarrow\quad \bar{w}_l > 0.34\sqrt{2} + \bar{w}_1 > 0.34(\sqrt{2}+1) > 0.82. \tag{4.84}$$

Because $2\cdot 0.82^2 > 1$, $\bar{w}_l$ must be the largest among $\bar{w}_2, \ldots, \bar{w}_d$ (otherwise $l' = \arg\max_{i\in\{2,\ldots,d\}} \bar{w}_i$ and $l \neq l'$ would imply $\|\bar{w}\|^2 = \sum_{i\in[d]} |\bar{w}_i|^2 \geq \bar{w}_l^2 + \bar{w}_{l'}^2 \geq 2\bar{w}_l^2 > 1$, contradiction). Therefore, Line 2 of our algorithm correctly returns the value of l.

If X is from Case 2, we have

$$\sigma_2 - \epsilon \leq \min_{i\in[n]} X_i\bar{w} = \min\left\{-\frac{1}{\sqrt{2}}\bar{w}_1 + \frac{1}{\sqrt{2}}\bar{w}_l, \frac{1}{\sqrt{2}}\bar{w}_1 + \frac{1}{\sqrt{2}}\bar{w}_l\right\} \leq \frac{1}{\sqrt{2}}\bar{w}_l, \tag{4.85}$$

and hence $\bar{w}_l \geq \sqrt{2}(\sigma_2-\epsilon) \geq \sqrt{2}(\frac{1}{\sqrt{2}}-0.04) > 0.94$. Because $2\cdot 0.94^2 > 1$, only one coordinate of $\bar{w}$ could be at least 0.94 and we must have $l = l'$. Therefore, Line 1 of our algorithm correctly returns the value of l.

In all, we have proved that an ϵ-approximate solution $\bar{w} \in \mathrm{B}_d$ for (4.71) would simultaneously reveal whether X is from Case 1 or Case 2 as well as the value of $l \in \{2, \ldots, d\}$. On the one hand, notice that distinguishing these two cases requires $\Omega(\sqrt{n-2}) = \Omega(\sqrt{n})$ quantum queries to O_X for searching the position of k because of the quantum lower bound for search [91]; therefore, it gives an $\Omega(\sqrt{n})$ quantum lower bound on queries to O_X for returning an $\bar{w}$ that satisfies (4.71). On the other hand, finding the value of l is also a search problem on the entries of X, which requires $\Omega(\sqrt{d-1}) = \Omega(\sqrt{d})$ quantum queries to O_X also due to the quantum lower bound for search [91]. These observations complete the proof of Theorem 4.5.1. $\square$

Because the kernel-based classifier in Section 4.4.1 contains the linear classification in Section 4.3 as a special case, Theorem 4.5.1 implies an $\Omega(\sqrt{n}+\sqrt{d})$ quantum lower bound on the kernel method.

4.5.2 Minimum enclosing ball (MEB)

Similarly, the input of the MEB problem is a matrix $X \in \mathbb{R}^{n\times d}$ such that $X_i \in \mathrm{B}_d$ for all $i \in [n]$, and we are given the quantum oracle O_X such that $O_X|i\rangle|j\rangle|0\rangle = |i\rangle|j\rangle|X_{ij}\rangle\ \forall\, i \in [n], j \in [d]$. The goal of MEB is to approximately solve

$$\sigma_{\mathrm{MEB}} = \min_{w\in\mathbb{R}^d} \max_{i\in[n]} \|w - X_i\|^2. \tag{4.86}$$

Theorem 4.4.2 solves this task with high success probability with cost $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4}+\frac{\sqrt{d}}{\epsilon^7}\big)$. In this subsection, we prove a quantum lower bound that matches this upper bound in n and d for constant ϵ:

Theorem 4.5.2. *Assume* $0<\epsilon<0.01$. *Then to return an* $\bar{w}\in\mathbb{R}_d$ *satisfying*

$$\max_{i\in[n]}\|\bar{w}-X_i\|^2\leq\min_{w\in\mathbb{R}^d}\max_{i\in[n]}\|w-X_i\|^2+\epsilon \tag{4.87}$$

with probability at least $2/3$, *we need* $\Omega(\sqrt{n}+\sqrt{d})$ *quantum queries to* O_X.

By Section 4.4.2.2, Theorem 4.5.2 also implies an $\Omega(\sqrt{n}+\sqrt{d})$ quantum lower bound on ℓ^2-margin SVMs.

Proof. We also assume that X is from one of the two cases in Theorem 4.5.1; see also (4.72) and (4.73). We denote the maximin value in (4.86) of these cases as $\sigma_{\text{MEB},1}$ and $\sigma_{\text{MEB},2}$, respectively. We have:

- $\sigma_{\text{MEB},2}=\frac{1}{2}$.

  On the one hand, consider $\bar{w}=\frac{1}{\sqrt{2}}\vec{e}_l$. Then

$$\|\bar{w}-X_1\|^2=\Big(w_1+\frac{1}{\sqrt{2}}\Big)^2+\Big(w_l-\frac{1}{\sqrt{2}}\Big)^2+\sum_{i\neq 1,l}w_i^2=\Big(\frac{1}{\sqrt{2}}\Big)^2=\frac{1}{2}; \tag{4.88}$$

$$\|\bar{w}-X_i\|^2=\Big(w_1-\frac{1}{\sqrt{2}}\Big)^2+\Big(w_l-\frac{1}{\sqrt{2}}\Big)^2+\sum_{i\neq 1,l}w_i^2=\Big(\frac{1}{\sqrt{2}}\Big)^2=\frac{1}{2}\quad\forall\, i\in\{2,\ldots,n\}. \tag{4.89}$$

Therefore, $\|\bar{w}-X_i\|^2=\frac{1}{2}$ for all $i\in[n]$, and hence $\sigma_{\text{MEB},2}\leq\max_{i\in[n]}\|\bar{w}-X_i\|^2=\frac{1}{2}$.

On the other hand, for any $w = (w_1, \ldots, w_d) \in \mathbb{R}_d$, we have

$$\begin{aligned}
&\max_{i\in[n]} \|w - X_i\|^2 \\
&= \max\Big\{\Big(w_1 - \frac{1}{\sqrt{2}}\Big)^2 + \Big(w_l - \frac{1}{\sqrt{2}}\Big)^2 + \sum_{i\neq 1,l} w_i^2, \Big(w_1 + \frac{1}{\sqrt{2}}\Big)^2 + \Big(w_l - \frac{1}{\sqrt{2}}\Big)^2 + \sum_{i\neq 1,l} w_i^2\Big\} \qquad (4.90) \\
&\geq \frac{1}{2}\Big[\Big(w_1 - \frac{1}{\sqrt{2}}\Big)^2 + \Big(w_l - \frac{1}{\sqrt{2}}\Big)^2\Big] + \frac{1}{2}\Big[\Big(w_1 + \frac{1}{\sqrt{2}}\Big)^2 + \Big(w_l - \frac{1}{\sqrt{2}}\Big)^2\Big] + \sum_{i\neq 1,l} w_i^2 \qquad (4.91) \\
&= w_1^2 + \Big(w_l - \frac{1}{\sqrt{2}}\Big)^2 + \sum_{i\neq 1,l} w_i^2 + \frac{1}{2} \qquad (4.92) \\
&\geq \frac{1}{2}, \qquad (4.93)
\end{aligned}$$

where (4.91) comes from the fact that $\max\{a, b\} \geq \frac{1}{2}(a + b)$ and $\sum_{i\neq 1,l} w_i^2 \geq 0$. Therefore, $\sigma_{\text{MEB},2} \geq \frac{1}{2}$. In all, we must have $\sigma_{\text{MEB},2} = \frac{1}{2}$.

- $\sigma_{\text{MEB},1} = \frac{2+\sqrt{2}}{4}$.

On the one hand, consider $\bar{w} = \big(\frac{1}{2} - \frac{\sqrt{2}}{4}\big)\vec{e}_1 + \frac{\sqrt{2}}{4}\vec{e}_l$. Then

$$\|\bar{w} - X_1\|^2 = \Big(w_1 + \frac{1}{\sqrt{2}}\Big)^2 + \Big(w_l - \frac{1}{\sqrt{2}}\Big)^2 + \sum_{i\neq 1,l} w_i^2 = \Big(\frac{1}{2} + \frac{\sqrt{2}}{4}\Big)^2 + \Big(\frac{\sqrt{2}}{4}\Big)^2 = \frac{2+\sqrt{2}}{4}; \qquad (4.94)$$

$$\|\bar{w} - X_k\|^2 = (w_1 - 1)^2 + w_l^2 + \sum_{i\neq 1,l} w_i^2 = \Big(\frac{1}{2} + \frac{\sqrt{2}}{4}\Big)^2 + \Big(\frac{\sqrt{2}}{4}\Big)^2 = \frac{2+\sqrt{2}}{4}; \qquad (4.95)$$

$$\|\bar{w} - X_i\|^2 = \Big(w_1 - \frac{1}{\sqrt{2}}\Big)^2 + \Big(w_l - \frac{1}{\sqrt{2}}\Big)^2 + \sum_{i\neq 1,l} w_i^2 = \frac{6 - 3\sqrt{2}}{4} < \frac{2+\sqrt{2}}{4} \quad \forall\, i \in [n]/\{1, k\}. \qquad (4.96)$$

In all, $\sigma_{\mathrm{MEB},1} \le \max_{i\in[n]} \|\bar{w} - X_i\|^2 = \frac{2+\sqrt{2}}{4}$.

On the other hand, for any $w = (w_1, \ldots, w_d) \in \mathbb{R}_d$, we have

$$\max_{i\in[n]} \|w - X_i\|^2 \ge \max\Big\{\Big(w_1 + \frac{1}{\sqrt{2}}\Big)^2 + \Big(w_l - \frac{1}{\sqrt{2}}\Big)^2 + \sum_{i\neq 1,l} w_i^2, (w_1-1)^2 + w_l^2 + \sum_{i\neq 1,l} w_i^2\Big\} \tag{4.97}$$

$$\ge \frac{1}{2}\Big[\Big(w_1 + \frac{1}{\sqrt{2}}\Big)^2 + \Big(w_l - \frac{1}{\sqrt{2}}\Big)^2\Big] + \frac{1}{2}\Big[(w_1-1)^2 + w_l^2\Big] + \sum_{i\neq 1,l} w_i^2 \tag{4.98}$$

$$= \Big[w_1 - \Big(\frac{1}{2} - \frac{\sqrt{2}}{4}\Big)\Big]^2 + \Big(w_l - \frac{\sqrt{2}}{4}\Big)^2 + \sum_{i\neq 1,l} w_i^2 + \frac{2+\sqrt{2}}{4} \tag{4.99}$$

$$\ge \frac{2+\sqrt{2}}{4}. \tag{4.100}$$

Therefore, $\sigma_{\mathrm{MEB},2} \ge \frac{2+\sqrt{2}}{4}$. In all, we must have $\sigma_{\mathrm{MEB},2} = \frac{2+\sqrt{2}}{4}$.

Now, we prove that an $\bar{w} \in \mathbb{R}_d$ satisfying (4.87) would simultaneously reveal whether X is from Case 1 or Case 2 as well as the value of $l \in \{2, \ldots, d\}$, by the following algorithm:

1. Check if one of $\bar{w}_2, \ldots, \bar{w}_d$ is larger than $\frac{3\sqrt{2}}{8}$; if there exists an $l' \in \{2, \ldots, d\}$ such that $\bar{w}_{l'} > \frac{3\sqrt{2}}{8}$, return 'Case 1' and $l = l'$;
2. Otherwise, return 'Case 2' and $l = \arg\max_{i\in\{2,\ldots,d\}} \bar{w}_i$.

We first prove that the classification of X (between Case 1 and Case 2) is correct. On the one hand, assume that X comes from Case 1. If we wrongly classified X as from Case 2, we would have $\bar{w}_l \le \max_{i\in\{2,\ldots,d\}} \bar{w}_i \le \frac{3\sqrt{2}}{8}$. By (4.92), this would imply

$$\max_{i\in[n]} \|\bar{w} - X_i\|^2 \ge \Big(\bar{w}_l - \frac{1}{\sqrt{2}}\Big)^2 + \frac{1}{2} \ge \frac{1}{32} + \frac{1}{2} > \sigma_{\mathrm{MEB},1} + \epsilon, \tag{4.101}$$

contradicts with (4.87). Therefore, for this case we must make correct classification that X comes from Case 2.

On the other hand, assume that X comes from Case 2. If we wrongly classified X as from Case 1, we would have $\bar{w}_{l'} > \frac{3\sqrt{2}}{8}$. If $l = l'$, then (4.99) implies that

$$\max_{i\in[n]} \|\bar{w} - X_i\|^2 \geq \left(\bar{w}_l - \frac{\sqrt{2}}{4}\right)^2 + \frac{2+\sqrt{2}}{4} \geq \frac{1}{32} + \frac{2+\sqrt{2}}{4} > \sigma_{\text{MEB},2} + \epsilon, \tag{4.102}$$

contradicts with (4.87). If $l \neq l'$, then (4.99) implies that

$$\max_{i\in[n]} \|\bar{w} - X_i\|^2 \geq \bar{w}_{l'}^2 + \frac{2+\sqrt{2}}{4} \geq \frac{9}{32} + \frac{2+\sqrt{2}}{4} > \sigma_{\text{MEB},2} + \epsilon, \tag{4.103}$$

also contradicts with (4.87). Therefore, for this case we must make correct classification that X comes from Case 1.

In all, our classification is always correct. It remains to prove that the value of l is correct. If X is from Case 1, by (4.92) we have

$$\frac{1}{2} + 0.01 \geq \max_{i\in[n]} \|\bar{w} - X_i\|^2 \geq \bar{w}_1^2 + \left(\bar{w}_l - \frac{1}{\sqrt{2}}\right)^2 + \sum_{i\neq 1,l} \bar{w}_i^2 + \frac{1}{2}. \tag{4.104}$$

As a result, $\bar{w}_i \leq 0.1 < \frac{3\sqrt{2}}{8}$ for all $i \in [n]/\{l\}$ and $\bar{w}_l \geq \frac{1}{\sqrt{2}} - 0.1 > \frac{3\sqrt{2}}{8}$. Therefore, we must have $l = l'$, i.e., Line 1 of our algorithm correctly returns the value of l.

If X is from Case 2, by (4.99) we have

$$\frac{2+\sqrt{2}}{4}+0.01 \geq \max_{i\in[n]} \|\bar{w}-X_i\|^2 \tag{4.105}$$

$$\geq \left[\bar{w}_1-\left(\frac{1}{2}-\frac{\sqrt{2}}{4}\right)\right]^2+\left(\bar{w}_l-\frac{\sqrt{2}}{4}\right)^2+\sum_{i\neq 1,l}\bar{w}_i^2+\frac{2+\sqrt{2}}{4}. \tag{4.106}$$

As a result, $\bar{w}_i \leq 0.1 < 0.25$ for all $i \in [n]/\{1,l\}$, $\bar{w}_1 \leq \frac{1}{2}-\frac{\sqrt{2}}{4}+0.1 < 0.25$, and $\bar{w}_l \geq \frac{\sqrt{2}}{4}-0.1 > 0.25$. Therefore, we must have $\bar{w}_l = \max_{i\in\{2,\dots,d\}} \bar{w}_i$, i.e., Line 1 of our algorithm correctly returns the value of l.

In all, we have proved that an ϵ-approximate solution $\bar{w} \in \mathbb{R}_d$ for (4.87) would simultaneously reveal whether X is from Case 1 or Case 2 as well as the value of $l \in \{2,\dots,d\}$. On the one hand, notice that distinguishing these two cases requires $\Omega(\sqrt{n-2}) = \Omega(\sqrt{n})$ quantum queries to O_X for searching the position of k because of the quantum lower bound for search [91]; therefore, it gives an $\Omega(\sqrt{n})$ quantum lower bound on queries to O_X for returning an $\bar{w}$ that satisfies (4.87). On the other hand, finding the value of l is also a search problem on the entries of X, which requires $\Omega(\sqrt{d-1}) = \Omega(\sqrt{d})$ quantum queries to O_X also due to the quantum lower bound for search [91]. These observations complete the proof of Theorem 4.5.2. □

4.5.3 Zero-sum games

Recall that for zero-sum games, we are given an n-dimensional anti-symmetric matrix X normalized by $\max_{i,j\in[n]} |X_{i,j}| \leq 1$, and the goal is to return an $w \in \Delta_n$ such that

$$Xw \leq \epsilon \cdot \mathbf{1}_n. \tag{4.107}$$

Given the quantum oracle O_X in (4.8), i.e., $O_X|i\rangle|j\rangle|0\rangle = |i\rangle|j\rangle|X_{ij}\rangle\ \forall\, i,j \in [n]$, Theorem 4.3.1 solves this task with high success probability with cost $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4}\big)$. We prove a matching quantum lower bound in n:

Theorem 4.5.3. *Assume $0 < \epsilon < 1/3$. Then to return an $w \in \Delta_n$ satisfying (4.107) with probability at least $2/3$, we need $\Omega(\sqrt{n})$ quantum queries to O_X.*

Proof. Assume that there exists an $k \in [n]$ such that

$$X_{ki} = \begin{cases} 1 & \text{if } i \neq k \\ 0 & \text{if } i = k \end{cases} \qquad X_{ik} = \begin{cases} -1 & \text{if } i \neq k \\ 0 & \text{if } i = k \end{cases} \qquad X_{ij} = 0 \quad \text{if } i,j \neq k. \tag{4.108}$$

Denote $w = (w_1, \ldots, w_n)^\dagger$. Then (4.108) implies that

$$\forall\, i \neq k,\ (Xw)_i = \sum_{j=1}^{n} X_{ij} w_j = X_{ik} w_k = -w_k; \tag{4.109}$$

$$(Xw)_k = \sum_{j=1}^{n} X_{kj} w_j = \sum_{j \neq k} w_j. \tag{4.110}$$

In order to have (4.107), we need to have $-w_k \leq \epsilon$ and $\sum_{j\neq k} w_j \leq \epsilon$ by (4.109) and (4.110), respectively. Because $w_j \geq 0$ for all $j \in [n]$ and $\sum_{j=1}^{n} w_j = 1$, they imply that $1 - \epsilon \leq w_k \leq 1$ and $0 \leq w_j \leq \epsilon$ for all $j \in [n]/\{k\}$. Therefore, if we could return an $w \in \Delta_n$ satisfying (4.107) with probability at least $2/3$, then we become aware of the value of k. On the other hand, this is a search problem on the entries of X, which requires $\Omega(\sqrt{n})$ quantum queries to O_X by the quantum lower bound for search [91]. In all, this implies that the cost of solving the zero-sum game takes at least $\Omega(\sqrt{n})$ quantum queries or gates. □

4.6 Generalization to Matrix Games

Minimax games between two parties, i.e., $\min_x \max_y f(x, y)$, is a basic model in game theory and has ubiquitous connections and applications to economics, optimization and machine learning, theoretical computer science, etc. Among minimax games, one of the most fundamental cases is the bilinear minimax game, also known as the *matrix game*, with the following form:

$$\min_{x\in\mathcal{X}} \max_{y\in\mathcal{Y}} y^\top Ax, \text{ where } A \in \mathbb{R}^{n\times d}, \mathcal{X} \subset \mathbb{R}^d, \mathcal{Y} \subset \mathbb{R}^n. \tag{4.111}$$

Matrix games are fundamental in algorithm design due to their equivalence to linear programs [157], and also in machine learning because they contain classification [124, 158] as a special case, and many other important problems.

For many common domains $\mathcal{X}$ and $\mathcal{Y}$, matrix games can be solved efficiently within approximation error ϵ, i.e., to output $x' \in \mathcal{X}$ and $y' \in \mathcal{Y}$ such that $(y')^\top Ax'$ is ϵ-close to the optimum in (4.114). For some specific choices of $\mathcal{X}$ and $\mathcal{Y}$, the matrix game can even be solved in *sublinear time* in the size nd of A. When $\mathcal{X}$ and $\mathcal{Y}$ are both ℓ_1-norm unit balls, [156] can solve the matrix game in time $O((n + d) \log(n + d)/\epsilon^2)$. When $\mathcal{X}$ is the ℓ_2-norm unit ball in $\mathbb{R}^d$ and $\mathcal{Y}$ is the ℓ_1-norm unit ball in $\mathbb{R}^n$, [126] can solve the matrix game in time $O((n + d) \log n/\epsilon^2)$.

As far as we know, the ℓ_1-ℓ_1 and ℓ_2-ℓ_1 matrix games are the only two cases where sublinear algorithms are known. However, there is general interest of solving matrix games with general norms. For instance, matrix games are closely related to the Carathéodory problem for finding a sparse linear combination in the convex hull of given data points, where all the ℓ_p-metrics with $p \geq 2$ have been well-studied [159, 160, 161]. In addition, matrix games are common in machine learning especially support vector machines (SVMs), and general ℓ_p-margin SVMs have also been considered by previous literature, see e.g. the book by [162]. In all, it is a natural question to investigate *sublinear algorithms for general matrix games*. In addition, quantum computing has been rapidly advancing and current technology has reached ”quantum supremacy” for some specific tasks [163]; since previous works have given sublinear quantum algorithms for ℓ_1-ℓ_1 matrix games [164, 165] and ℓ_2-ℓ_1 matrix games [164] with running time $(\sqrt{n}+\sqrt{d})\,\mathrm{poly}(1/\epsilon)$, it is also natural to explore *sublinear quantum algorithms for general matrix games*.

Contributions. We conduct a systematic study of ℓ_q-ℓ_1 matrix games for any $q \in (1,2]$ which corresponds to ℓ_q-margin SVMs and the ℓ_p-Carathéodory problem for any $p \geq 2$. We use the following entry-wise input model, the standard assumption in the sublinear algorithms in [126, 156]:

Theorem 4.6.1 (Main Theorem)**.** *Given $q \in (1,2]$. Define $p \geq 2$ such that $\frac{1}{p}+\frac{1}{q}=1$. Consider the ℓ_q-ℓ_1 matrix game*[7]*:*

$$\sigma := \max_{x \in \mathrm{B}_q^d} \min_{\mathbf{p} \in \Delta_n} \mathbf{p}^\top A x, \tag{4.112}$$

where B_q^d is the ℓ_q-unit ball in $\mathbb{R}^d$ and Δ_n is the ℓ_1-simplex in $\mathbb{R}^n$. Then we can find an $\bar{x} \in \mathrm{B}_q^d$ s.t.[8]

$$\min_{i \in [n]} A_i \bar{x} \geq \sigma - \epsilon \tag{4.113}$$

[7]Throughout the paper, we use the bold font $\mathbf{p}$ to denote a vector and the math font p to denote a real number.
[8]$\bar{x} \in \mathrm{B}_q^d$ is the standard objective quantity under the ℓ_q-norm. Also note that once we have the $\bar{x}$ in (4.113), any $\mathbf{p} \in \Delta_n$ satisfies $\mathbf{p}^\top A\bar{x} \geq \sigma - \epsilon$.

with success probability at least $2/3$*, using*

- $O\big(\frac{(n+d)(p+\log n)}{\epsilon^2}\big)$ *classical queries (Theorem 4.7.1); or*

- $\tilde{O}\big(\frac{p^2\sqrt{n}}{\epsilon^4}+\frac{p^{3.5}\sqrt{d}}{\epsilon^7}\big)$ *quantum queries*[9] *(Theorem 4.8.1).*

When $p=\Omega(\log d/\epsilon)$*, the above bounds can be improved (by Lemma 4.6.1) to respectively*

- $O\big(\frac{(n+d)(\frac{\log d}{\epsilon}+\log n)}{\epsilon^2}\big)$ *queries to the classical input model;*

- $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^6}+\frac{\sqrt{d}}{\epsilon^{10.5}}\big)$ *queries to the quantum input model.*

Both results are optimal in n *and* d *up to poly-log factors as we show* $\Omega(n+d)$ *and* $\Omega(\sqrt{n}+\sqrt{d})$ *classical and quantum lower bounds respectively when* $\epsilon=\Theta(1)$.

Conceptually, our classical and quantum algorithms for general matrix games enjoy quite a few nice properties. On the one hand, they can be directly applied to

- **Convex geometry:** We give the *first* sublinear classical and quantum algorithms for the approximate Carathéodory problem (Corollary 4.9.1), improving the previous linear-time algorithms of [160, 161];

- **Supervised learning:** We provide the *first* sublinear algorithms for general ℓ_q-margin support vector machines (SVMs) (Corollary 4.9.2).

On the other hand, our quantum algorithm is **friendly for near-term applications**. It uses the *standard quantum input model* and needs not to use any sophisticated quantum data structures. It is *classical-quantum hybrid* where the quantum part is isolated by pieces of state preparations connected by classical processing. Its output is completely *classical*.

[9]Here $\tilde{O}$ omits poly-logarithmic factors.

Technique-wise, we are deeply inspired by [126], which serves as the starting point of our algorithm design. At a high level, Clarkson et al.'s algorithm follows a primal-dual framework where the primal part applies (ℓ_2-norm) online gradient descent (OGD) by [149], and the dual part applies multiplicative weight updates (MWU) by ℓ_2-sampling. The choice of the ℓ_2-norm metric greatly facilitates the design and analysis of the algorithms for both parts. However, it is conceivable that *more sophisticated design and analysis* will be required to handle general ℓ_q-ℓ_1 matrix games.

Our **main technical contribution** in this section is to expand the primal-dual approach of [126] to work for more general metrics for the ℓ_q-ℓ_1 matrix game. Specifically, in the primal we replace OGD by a generalized p-norm OGD due to [166], and in the dual we replace the ℓ_2-sampling by ℓ_q-sampling. We conduct a careful algorithm design and analysis to ensure that this strategy only incurs an $O(p/\epsilon^2)$ overhead in the number of iterations, and the error of the ℓ_q-ℓ_1 matrix game is still bounded by ϵ as in (4.113). In a nutshell, our algorithm can be viewed as an *interpolation* between the ℓ_2-ℓ_1 matrix game [126] and the ℓ_1-ℓ_1 matrix game [156]: when q is close to 2 the algorithm is more similar to [126], whereas when q is close to 1, p is large and the p-norm GD becomes closer to the normalized exponentiated gradient [166], which is exactly the update rule in **(author?)** [156].

Quantumly, we also provide a corresponding quantum speedup of our new classical algorithm, inspired by the previous quantum speedup by [164]. They achieved a quantum speedup of $\tilde{O}(\sqrt{n}+\sqrt{d})$ for solving ℓ_2-ℓ_1 matrix games by levering *quantum amplitude amplification* and observing that ℓ_2-sampling can be readily accomplished by *quantum state preparation* as quantum states refer to ℓ_2 unit vectors. For general ℓ_q-ℓ_1 matrix game ($q \in (1,2]$), we likewise upgrade both primal and dual parts as in our classical algorithm: specifically, in the primal, we apply the p-norm OGD in $\tilde{O}(\sqrt{d})$ time, whereas in the dual, we apply the multiplicative weight update via an ℓ_q-sampling in $\tilde{O}(\sqrt{n})$ time. To that end, we contribute to the following technical improvements:

- In our algorithm, we cannot directly leverage quantum state preparation in the ℓ_q metric because it corresponds to ℓ_2-normalized vectors. Instead, we propose Algorithm 19 for **quantum ℓ_q-sampling** with $O(\sqrt{n})$ oracle calls which works with states whose amplitudes follow ℓ_q-norm proportion. Measuring such states is equivalent to performing ℓ_q-sampling.

- When $p = q = 2$, we improved the **ϵ-dependence** from the $1/\epsilon^8$ in [164] to $1/\epsilon^7$, which is achieved by deriving a better upper bound on the entries of the vectors in the p-norm OGD.

These improvements together result in Theorem 4.6.1.

Related works on Matrix Games. Matrix games were probably first studied as zero-sum games by [167]. The seminal work [43] proposed the mirror descent method and gave an algorithm for solving matrix games in time $\tilde{O}(nd/\epsilon^2)$. This was later improved to $\tilde{O}(nd/\epsilon)$ by the prox-method due to [168] and the dual extrapolation method due to [169]. To further improve the cost, there have been two main focuses:

- Sampling-based methods: They focus on achieving sublinear cost in nd, the size of the matrix A. [126, 156] mentioned above are seminal examples; these sublinear algorithms can also be used to solve semidefinite programs [170], SVMs [171], etc.

- Variance-reduced methods: They focus on the cost in $1/\epsilon$, in particular its decoupling with nd. [172] showed how to apply the standard SVRG [173] technique for solving ℓ_2-ℓ_2 matrix games; this idea can also be extended to smooth functions using general Bregman divergences [174]. Variance-reduced methods for solving matrix games culminate in [175], where they show how to solve ℓ_1-ℓ_1 and ℓ_2-ℓ_1 matrix games in time $\tilde{O}(\mathrm{nnz}(A) + \sqrt{\mathrm{nnz}(A) \cdot (n+d)}/\epsilon)$, where $\mathrm{nnz}(A)$ is the number of nonzero elements in A.

There have been relatively few quantum results for solving matrix games. [130] solved the ℓ_2-ℓ_1 matrix game with cost $\tilde{O}(\sqrt{n}d/\epsilon^2)$ using an unusual input model where the representation of a data point in $\mathbb{R}^d$ is the concatenation of d floating point numbers. More recently, [165] was able to solve the ℓ_1-ℓ_1 matrix game with cost $\tilde{O}(\sqrt{n}/\epsilon^3 + \sqrt{d}/\epsilon^3)$ using the standard input model above, and [164] solved the ℓ_2-ℓ_1 matrix game with cost $\tilde{O}(\sqrt{n}/\epsilon^4 + \sqrt{d}/\epsilon^8)$ also using the standard input model.

Interpolation for large p. If p is large, we prove the following lemma showing that we can restrict without loss of generality to cases where p such that $\frac{1}{p} + \frac{1}{q} = 1$ is $O(\log d/\epsilon)$, since in this case the ℓ_q-ℓ_1 matrix game is ϵ-close to the ℓ_1-ℓ_1 matrix game in the following sense:

Lemma 4.6.1. *An ℓ_q-ℓ_1 matrix game where p such that $\frac{1}{p} + \frac{1}{q} = 1$ is greater than $\log d/\epsilon$ can be solved using an algorithm for solving ℓ_1-ℓ_1 games. This introduces an error $O(\epsilon)$ in the objective value.*

Proof. Assume without loss of generality that $\epsilon \leq 1/2$. Let $p \geq \log d/\epsilon \geq \log d/(-\log(1-\epsilon))$. It can be easily verified that $\mathrm{B}_1^d \subset \mathrm{B}_q^d \subset \mathrm{B}_1^d + \left(1 - d^{-1/p}\right) \mathrm{B}_q^d$. Thus $\mathrm{B}_q^d \subset \mathrm{B}_1^d + \epsilon \mathrm{B}_q^d$.

Consider applying an algorithm to solve an ℓ_1-ℓ_1 matrix game instead of the ℓ_q-ℓ_1 matrix game as required in (4.112). Let the optimal solution to (4.112) be $x^* \in \mathrm{B}_q^d, p^* \in \Delta_n$. By the previous analysis, there is a point $x \in \mathrm{B}_1^d$ such that $\|x - x^*\|_q \leq \epsilon$. Thus the solution x, p^* has an error at most $O(\epsilon)$ from the true objective, and the algorithm for solving ℓ_1-ℓ_1 games finds a solution at least as good as this. $\square$

Notations. Throughout the paper, we denote $p, q > 1$ to be two real numbers such that $\frac{1}{p} + \frac{1}{q} = 1$; $p \in [2, +\infty)$ and $q \in (1, 2]$. For any $s > 1$, we use B_s^d to denote the d-dimensional unit ball in ℓ_s-norm, i.e., $\mathrm{B}_s^d := \{x : \sum_{i \in [d]} |x_i|^s \leq 1\}$; we use Δ_n to denote the n-dimensional unit simplex $\{p \in \mathbb{R}^n : p_i \geq 0, \sum_i p_i = 1\}$, and use $\mathbf{1}_n$ to denote the n-dimensional all-one vector. We denote $A \in \mathbb{R}^{n \times d}$ to be the matrix whose ith row is $A_i^\top$ for all $i \in [n]$. We define $\operatorname{sgn}\colon \mathbb{R} \to \{-1, 0, 1\}$ such that $\operatorname{sgn}(x) = -1$ if $x < 0$, $\operatorname{sgn}(x) = 1$ if $x > 0$, and $\operatorname{sgn}(0) = 0$.

4.7 A sublinear classical algorithm for general matrix games

For any $q \in (1, 2]$, we consider the ℓ_q-ℓ_1 matrix game:

$$\sigma := \max_{x \in \mathrm{B}_q^d} \min_{\mathbf{p} \in \Delta_n} \mathbf{p}^\top A x. \tag{4.114}$$

The goal is to find a $\bar{x}$ that approximates the equilibrium of the matrix game within additive error ϵ:

$$\min_{i \in [n]} A_i \bar{x} \geq \sigma - \epsilon. \tag{4.115}$$

Throughout the paper, we assume $A_1, \ldots, A_n \in \mathrm{B}_p^d$, i.e., all the n data points are normalized to have ℓ_p-norm at most 1.

Theorem 4.7.1. *The output of Algorithm 18 satisfies (4.115) with probability at least $2/3$, and*

Algorithm 18: A sublinear algorithm for ℓ_q-ℓ_1 games.

```
    Input: $\epsilon > 0$; $p \in [2, +\infty), q \in (1, 2]$ such that $\frac{1}{p} + \frac{1}{q} = 1$; $A \in \mathbb{R}^{n \times d}$ with
           $A_i \in \mathrm{B}_p^d\ \forall i \in [n]$.
    Output: $\bar{x}$ that satisfies (4.115).
 1  Let $T = \lceil \frac{895 \log n + 4p}{\epsilon^2} \rceil$, $y_1 = \mathbf{0}_d$, $\eta = \sqrt{\frac{11 \log n}{12T}}$, $w_1 = \mathbf{1}_n$;
 2  for $t = 1$ to $T$ do
 3      $p_t \leftarrow \frac{w_t}{\|w_t\|_1}$, $x_t \leftarrow \frac{y_t}{\max\{1, \|y_t\|_q\}}$;
 4      Choose $i_t \in [n]$ by $i_t \leftarrow i$ with probability $p_t(i)$;
 5      Define $y_{t+1}$ where for any $j \in [d]$, $y_{t+1,j} \leftarrow y_t + \sqrt{\frac{q-1}{2T}} \frac{\mathrm{sgn}(A_{i_t,j})|A_{i_t,j}|^{p-1}}{\|A_{i_t}\|_p^{p-2}}$;
 6      Choose $j_t \in [d]$ by $j_t \leftarrow j$ with probability $\frac{x_t(j)^q}{\|x_t\|_q^q}$;
 7      for $i = 1$ to $n$ do
 8          $\tilde{v}_t(i) \leftarrow A_i(j_t)\|x_t\|_q^q / x_t(j_t)^{q-1}$ ;
 9          $v_t(i) \leftarrow \mathrm{clip}(\tilde{v}_t(i), \frac{1}{\eta})$ where $\mathrm{clip}(v, M) := \min\{M, \max\{-M, v\}\}\ \forall v, M \in \mathbb{R}$;
10          $w_{t+1}(i) \leftarrow w_t(i)(1 - \eta v_t(i) + \eta^2 v_t(i)^2)$;
11  Return $\bar{x} = \frac{1}{T} \sum_{t=1}^{T} x_t$.
```

its total running time is $O(\frac{(n+d)(p+\log n)}{\epsilon^2})$ *where* $p \geq 2$ *such that* $\frac{1}{p} + \frac{1}{q} = 1$.

Our sublinear algorithm follows the primal-dual approach of Algorithm 1 of [126], which solves ℓ_1-ℓ_2 matrix games. Here for ℓ_q-ℓ_1 matrix games, the solution vector x now lies in $\mathbb{B}_q^d$. Hence, the most natural adaptations are to use ℓ_q-sampling instead of ℓ_2-sampling in the primal updates, and to use a p-norm OGD by [166] which generalizes the online gradient descent by [149] in ℓ_2-norm. In the following, we use various technical tools to show these natural adaptations actually work.

Proposition 4.7.1 ((author?) 166, Corollary 2.18). *Consider a set of vectors* $u_1, \ldots, u_T \in \mathbb{R}^d$ *such that* $\|u_i\|_p \leq 1$. *Set* $\iota = \sqrt{\frac{q-1}{2T}}$. *Let* $x_0 \leftarrow \mathbf{0}_d$, $\tilde{x}_{t+1,i} \leftarrow x_{t,i} + \iota \frac{\mathrm{sgn}(u_{t,i})|u_{t,i}|^{p-1}}{\|u_t\|_p^{p-2}}$ *for all* $i \in [d]$, *and* $x_{t+1} \leftarrow \frac{\tilde{x}_{t+1}}{\max\{1, \|\tilde{x}_{t+1}\|_q\}}$. *Then*

$$\max_{x \in \mathrm{B}_q^d} \sum_{t=1}^{T} u_t^\top x - \sum_{t=1}^{T} u_t^\top x_t \leq \sqrt{\frac{2T}{q-1}}. \tag{4.116}$$

The analysis of Algorithm 18 uses the following lemma, adapted from the variance multiplicative weight lemma and martingale tail bounds in [126][10]:

Lemma 4.7.1 (Section 2 of [126])**.** *In Algorithm 18, the parameters p_t in Line 3 and v_t in Line 9 satisfy*

$$\sum_{t\in[T]} p_t^\top v_t \le \min_{i\in[n]} \sum_{t\in[T]} v_t(i) + \eta \sum_{t\in[T]} p_t^\top v_t^2 + \frac{\log n}{\eta} \tag{4.117}$$

where v_t^2 is defined as $(v_t^2)_i := (v_t)_i^2$ for all $i \in [n]$, as long as the update rule of w_t is as in Line 10 and $\mathrm{Var}[v_t(i)^2] \le 1$ for all $t \in [T]$ and $i \in [n]$. Furthermore, with probability at least $1 - O(1/n)$,

$$\max_{i\in[n]} \sum_{t\in[T]} \big[v_t(i) - A_i x_t\big] \le 4\eta T; \tag{4.118}$$

$$\Big|\sum_{t\in[T]} A_{i_t} x_t - \sum_{t\in[T]} p_t^\top v_t\Big| \le 10\eta T, \tag{4.119}$$

with probability at least $5/7$, $\sum_{t\in[T]} p_t^\top v_t^2 \le 7T$.

We also need to prove the following inequality on different moments of random variables.

Lemma 4.7.2. *Suppose that X is a random variable on $\mathbb{R}$, and $p \ge 2$. If $\mathbb{E}[|X|^p] \le 1$, then $\mathbb{E}[X^2] \le 1$.*

Proof. Denote the probability density of X as μ. Then $\int_{-\infty}^{+\infty} |x|^p \mathrm{d}\mu_x = \mathbb{E}[|X|^p] \le 1$. By Hölder's

[10]The proof follows from the proofs of Lemmas 2.3, 2.4, 2.5, and 2.6 in Section 2 and Appendix B of [126], with only small modifications to fit our new parameter choices. For instance, the original statement requires that $\eta \ge \sqrt{\frac{\log n}{T}}$, but the proofs actually work for $\eta \ge \sqrt{\frac{11 \log n}{12T}}$.

inequality, we have

$$1 \geq \left(\int_{-\infty}^{+\infty} |x|^p \mathrm{d}\mu_x\right)^{2/p} \left(\int_{-\infty}^{+\infty} 1 \mathrm{d}\mu_x\right)^{1-2/p}$$
$$\geq \int_{-\infty}^{+\infty} |x|^2 \cdot 1^{1-2/p} \mathrm{d}\mu_x = \int_{-\infty}^{+\infty} x^2 \mathrm{d}\mu_x, \tag{4.120}$$

hence the result follows. □

Now we are ready to prove our main theorem.

Proof of Theorem 4.7.1. First, $\tilde{v}_t(i)$ is an unbiased estimator of $A_i x_t$ as

$$\mathbb{E}[\tilde{v}_t(i)] = \sum_{j_t=1}^{d} \frac{x_t(j_t)^q}{\|x_t\|_q^q} \cdot \frac{A_i(j_t)\|x_t\|_q^q}{x_t(j_t)^{q-1}} = A_i x_t. \tag{4.121}$$

Furthermore,

$$\mathbb{E}[|\tilde{v}_t(i)|^p] = \sum_{j_t=1}^{d} \frac{x_t(j_t)^q}{\|x_t\|_q^q} \cdot \frac{|A_i(j_t)|^p \|x_t\|_q^{pq}}{x_t(j_t)^{p(q-1)}}$$
$$= \|A_i\|_p^p \|x_t\|_q^p \leq 1, \tag{4.122}$$

where the second equality follows from the identities $q = p(q-1)$ and $p = q(p-1)$, and the last inequality follows from the assumption that $A_i \in \mathrm{B}_p^d \ \forall i \in [n]$. By Lemma 4.7.2, $\mathbb{E}[\tilde{v}_t(i)^2] \leq 1$. Because the clip function in Line 9 only makes variance smaller, this means that the conditions of Lemma 4.7.1 are satisfied and we hence have (4.117), rewritten below:

$$\sum_{t\in[T]} p_t^\top v_t \leq \min_{i\in[n]} \sum_{t\in[T]} v_t(i) + \eta \sum_{t\in[T]} p_t^\top v_t^2 + \frac{\log n}{\eta}. \tag{4.123}$$

Furthermore, Lemma 4.7.1 implies that with probability $5/7 - O(1/n)$ we have

$$\sum_{t\in[T]} A_{i_t}x_t \le \min_{i\in[n]} \sum_{t\in[T]} v_t(i) + 17\eta T + \frac{\log n}{\eta}. \tag{4.124}$$

Moreover, (4.118) gives $\sum_{t\in[T]} \left[v_t(i) - A_i x_t\right] \le 4\eta T$, and hence $\min_{i\in[n]} \sum_{t\in[T]} v_t(i) \le 4\eta T + \min_{i\in[n]} \sum\limits_{t\in[T]} A_i x_t$. Plugging this into (4.124), we have

$$\begin{aligned} \sum_{t\in[T]} A_{i_t}x_t &\le \sum_{t\in[T]} p_t^\top v_t + 10\eta T \\ &\le \min_{i\in[n]} \sum_{t\in[T]} A_i x_t + 21\eta T + \frac{\log n}{\eta} \end{aligned} \tag{4.125}$$

with probability $(5/7 - O(1/n)) \cdot (1 - O(1/n)) \ge 2/3$.

On the other hand, by taking $u_t = A_{i_t}$ in Proposition 4.7.1,

$$T\sigma \le \max_{x\in \mathrm{B}_q^d} \sum_{t=1}^{T} A_{i_t}x \le \sum_{t=1}^{T} A_{i_t}x_t + \sqrt{2Tp} \tag{4.126}$$

since $\frac{1}{q-1} = \frac{p}{q} \le p$. Combining (4.125) and (4.126), we have

$$\min_{i\in[n]} \sum_{t\in[T]} A_i x_t \ge T\sigma - \sqrt{2Tp} - 21\eta T - \frac{\log n}{\eta}. \tag{4.127}$$

Consequently, the return $\bar{x} = \frac{1}{T}\sum_{t=1}^{T} x_t$ of Algorithm 18 in Line 11 satisfies

$$\min_{i\in[n]} A_i \bar{x} \ge \sigma - \sqrt{\frac{2p}{T}} - 21\eta - \frac{\log n}{\eta T}. \tag{4.128}$$

To prove (4.115), it remains to show that $\sqrt{\frac{2p}{T}} + 21\eta + \frac{\log n}{\eta T} \leq \epsilon$, which is equivalent to $\sqrt{2p} + 21\sqrt{\frac{11\log n}{12}} + \sqrt{\frac{12\log n}{11}} \leq \sqrt{T}\epsilon$ by the definition of η. This is true because the AM-GM inequality implies that that LHS is at most $2(\sqrt{2p})^2 + 2\big(21\sqrt{\frac{11\log n}{12}} + \sqrt{\frac{12\log n}{11}}\big)^2 \leq 4p + 895\log n \leq T\epsilon^2$. □

Lemma 4.6.1 combined with Theorem 4.7.1 yields the classical result in Theorem 4.6.1.

4.8 A sublinear quantum algorithm for general matrix games

In this section, we give a quantum algorithm for solving the general ℓ_q-ℓ_1 matrix game. It closely follows our classical algorithm because they both use a primal-dual approach, where the primal part is composed of p-norm online gradient descent and the dual part is composed of multiplicative weight updates. However, we adopt quantum techniques to achieve speedup on both.

The intuition behind the quantum algorithm and the quantum speedup is that we measure quantum states to obtain random samples. These quantum states can be efficiently prepared (with cost $\tilde{O}(\sqrt{n})$ and $\tilde{O}(\sqrt{d})$). A quantum state can be mathematically represented by an ℓ_2-normalized complex vector ψ in the sense that measuring this quantum states yields outcome i with probability $|\psi_i|^2$ (thus there is a quantum state corresponding to every probability distribution). Let us denote the quantum state for sampling from w by $|w\rangle$ and the quantum state for sampling from x by $|x\rangle$ (different from the notation in the paper). If we can maintain $|w\rangle$ and $|x\rangle$ in each iteration, then there is no need for classical updates, and preparing $|w\rangle$ and $|x\rangle$ becomes the bottleneck of the quantum algorithm. We design Algorithm 19 for such state preparation and we showed (in Proposition 4.10.1) that preparing $|w\rangle$ costs $\tilde{O}(\sqrt{n})$ and preparing $|x\rangle$ costs $\tilde{O}(\sqrt{d})$.

This is the source of our quantum speedup.

As an important subroutine, Algorithm 19 is used to prepare states for ℓ_q-sampling. It uses standard Grover-based techniques to prepare states but we carefully keep track of the normalizing factor to facilitate ℓ_q-sampling. This subroutine is summarized in Proposition 4.10.1 (in the supplementary material). We give its high-level ideas as follows:

1. We first create a quantum state corresponding to the uniform distribution, which is easy.

2. For each entry, we create a state with the desired amplitude associated with 0, and an undesired amplitude associated to 1 (the unitarity of quantum operations necessitates the existence of this undesired term).

3. Finally we use a technique called amplitude amplification to amplify the portion of the state corresponding to 0 for each entry, to get a state with only the desired amplitudes.

The details of our quantum algorithm is rather technical. To simplify the presentation, we postpone its pseudocode (Algorithm 20) to the final section and highlight how it is different from Algorithm 18 in the following.

- For the primal part, we prepare a quantum state $|y_t\rangle$ for the q-norm OGD and measure it (in Line 7) to obtain a sample $j_t \in [d]$. The subtlety here is that we need to perform the ℓ_q-sampling to the vector y_t; this is different from the ℓ_2-sampling in [164] which uses the fact that pure quantum states are ℓ_2-normalized. To this end, we design Algorithm 19 for *ℓ_q-quantum state sampling*, which may be of independent interest; this algorithm is built upon a clever use of *quantum amplitude amplification*, the technique behind the Grover search [132]. Note that sampling according to y_t is equivalent to sampling according to

Algorithm 19: Prepare an ℓ_q-pure state given an oracle to its coefficients.

```
1 Apply the minimum finding algorithm [151] to find $a_{\|q\|} := \max_{i\in[n]} |a_i|^{q/2}$ in $O(\sqrt{n})$
   time;
2 Prepare the uniform superposition $\frac{1}{\sqrt{n}}\sum_{r\in[n]}|i\rangle$;
3 Perform the following unitary transformations:
```

$$\begin{aligned} &\frac{1}{\sqrt{n}}\sum_{i\in[n]}|i\rangle \overset{O_a}{\longmapsto} \frac{1}{\sqrt{n}}\sum_{i\in[n]}|i\rangle|a_i\rangle \\ &\mapsto \frac{1}{\sqrt{n}}\sum_{i\in[n]}|i\rangle|a_i\rangle\left(\frac{a_i^{q/2}}{a_{\|q\|}}|0\rangle+\sqrt{1-\frac{|a_i|^q}{a_{\|q\|}^2}}|1\rangle\right) \\ &\overset{O_a^{-1}}{\longmapsto} \frac{1}{\sqrt{n}}\sum_{i\in[n]}|i\rangle|0\rangle\left(\frac{a_i^{q/2}}{a_{\|q\|}}|0\rangle+\sqrt{1-\frac{|a_i|^q}{a_{\|q\|}^2}}|1\rangle\right); \end{aligned}$$

```
4 Discard the second register above and rewrite the state as
```

$$\frac{\|a\|_q^{q/2}}{\sqrt{n}a_{\|q\|}}\left(\frac{1}{\|a\|_q^{q/2}}\sum_{i\in[n]}a_i^{q/2}|i\rangle\right)|0\rangle+|a^{\perp}\rangle|1\rangle \tag{4.129}$$

```
   where $|a^{\perp}\rangle := \frac{1}{\sqrt{n}}\sum_{i\in[n]}\sqrt{1-\frac{|a_i|^q}{a_{\|q\|}^2}}|i\rangle$;
5 Apply amplitude amplification [57] for the state in (4.129) conditioned on the second
   register being 0. Return the output.
```

x_t in Algorithm 18, because $x_t(j)^q/\|x_t\|_q^q = y_t(j)^q/\|y_t\|_q^q$. Moreover, it suffices to replace $\|x_t\|_q^q/x_t(j_t)^{q-1}$ with $\|y_t\|_q^q/(y_t(j_t)^{q-1}\max\{1,\|y_t\|_q\})$ in Line 8 of Algorithm 20. Similar to preparing $|p_t\rangle$, we use $\tilde{O}(\sqrt{d})$ queries to O_A to prepare y_t, while classically we need to compute all the entries of y_t, which takes $O(d)$ queries.

- For the dual part, we prepare the multiplicative weight vector as a quantum state $|p_t\rangle$ and measure it (in Line 3) to obtain a sample $i_t \in [n]$. This adaption enables us to achieve the $\tilde{O}(\sqrt{n})$ dependence by using quantum amplitude amplification in the quantum state preparation: in Line 8, we implement the oracle O_t and in Line 9 we use $\tilde{O}(\sqrt{n})$ queries to O_t to prepare the state $|p_{t+1}\rangle$ for the next iteration. In contrast, classically we need to compute all the entries of w_{t+1} to obtain the probability distribution p_{t+1} for the next iteration, which takes $O(n)$ queries.

In general, Algorithm 20 can be viewed as a template for achieving quantum speedups for online mirror descent methods: In this work, we focus on the general matrix games where the primal and dual are in the special relationship of ℓ_p and ℓ_q norms, but in principle it may be applicable to study other dualities in online learning.

We summarize the main quantum result as the following theorem, which states the correctness and time complexity of Algorithm 20. The relevant technical proofs are deferred to the supplementary material.

Theorem 4.8.1. *Algorithm 20 returns a succinct classical representation*[11] *of a vector* $\bar{w} \in \mathbb{R}^d$

[11]The algorithm stores $T = \tilde{O}(p/\epsilon^2)$ real numbers classically: $i_1, \ldots, i_T$ obtained from Line 3 and $\widetilde{\|y_1\|_q}, \ldots, \widetilde{\|y_T\|_q}$ obtained from Line 6. After that, each coordinate of $\bar{x}$ can be computed in time $\tilde{O}(p/\epsilon^2)$.

such that

$$A_i\bar{x} \geq \max_{x\in\mathrm{B}_q^d}\min_{i'\in[n]} A_{i'}x - \epsilon \quad \forall i \in [n], \tag{4.130}$$

with probability at least $2/3$*, and its total running time is* $\tilde{O}\big(\frac{p^2\sqrt{n}}{\epsilon^4}+\frac{p^{3.5}\sqrt{d}}{\epsilon^7}\big)$*. We can also assume* $p = O(\log d/\epsilon)$ *(Lemma 4.6.1) and result in running time* $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^6}+\frac{\sqrt{d}}{\epsilon^{10.5}}\big)$*.*

Moreover, Algorithm 20 enjoys the following features:

- **Simple quantum input:** Algorithm 20 uses the standard quantum input model and needs not to use any sophisticated quantum data structures, such as quantum random access memory (QRAM) in some other quantum machine learning applications, to achieve speedups.
- **Hybrid classical-quantum feature:** Algorithm 20 is also highly classical-quantum hybrid: the quantum part is isolated by pieces of state preparations connected by classical processing. In addition, it only has $O(\frac{\log n+p}{\epsilon^2})$ iterations, which implies that the corresponding quantum circuit is shallow and can potentially be implemented even on near-term quantum machines [135].
- **Classical output:** The output of Theorem 4.8.1 is completely *classical*. Compared to quantum algorithms whose output is a quantum state and may incur overheads [139], Algorithm 20 guarantees minimal overheads and can be directly used for classical applications.

4.9 Applications

We give two applications that generically follow from our classical and quantum ℓ_q-ℓ_1 matrix game solvers.

4.9.1 Approximate Carathéodory problem

The exact Carathéodory problem is a fundamental result in linear algebra and convex geometry: every point $u \in \mathbb{R}^d$ in the convex hull of a vertex set $S \subset \mathbb{R}^d$ can be expressed as a convex combination of $d+1$ vertices in S. Recently, a breakthrough result by [159] shows that if $S \subset \mathrm{B}_p^d$, i.e., S is in the ℓ_p-norm unit ball, then there exists a point u' s.t. $\|u-u'\|_p \leq \epsilon$ and u' is a convex combination of $O(p/\epsilon^2)$ vertices in S. The follow-up work by **(author?)** [160] proved a matching lower bound $\Omega(p/\epsilon^2)$, and **(author?)** [161] can give better bounds under stronger assumptions on S or u.

Currently, the best-known time complexity of solving the approximate Carathéodory problem is $O(ndp/\epsilon^2)$ by Theorem 3.5 of [160]. We give classical and quantum sublinear algorithms:

Corollary 4.9.1. *Suppose that $S \subset \mathrm{B}_p^d$, $|S| = n$, and u is in the convex full of S. Then we can find a convex combination $\sum_{i=1}^k x_i v_i$ such that $v_i \in S$ for all $i \in [k]$, $k = O((p+\log n)/\epsilon^2)$, and $\|\sum_{i=1}^k x_i v_i\|_p \leq \epsilon$, using a classical algorithm with running time $O\big(\frac{(n+d)(p+\log n)}{\epsilon^2}\big)$ or a quantum algorithm with running time $\tilde{O}\big(\frac{p^2\sqrt{n}}{\epsilon^4} + \frac{p^{3.5}\sqrt{d}}{\epsilon^7}\big)$. We can also assume $p = O(\log d/\epsilon)$ (Lemma 4.6.1) and result in running time $O\big(\frac{(n+d)(\log d/\epsilon+\log n)}{\epsilon^2}\big)$ and $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^6} + \frac{\sqrt{d}}{\epsilon^{10.5}}\big)$, respectively.*

Proof. We denote the matrix $V := (v_1; v_2; \cdots; v_n)$ where v_i is the i^{th} element in S. Note that the approximate Carathéodory problem can be formed as $\min_{\mathbf{p}\in\Delta_n} \|V^\top \mathbf{p} - u\|_p$. In addition, by Hölder's inequality $\|y\|_p = \max_{x:\|x\|_q\leq 1} y^\top x$; therefore, we obtain the following minimax matrix game:

$$\min_{\mathbf{p}\in\Delta_n} \max_{x\in\mathrm{B}_q^d} (\mathbf{p}^\top V - u^\top)x. \tag{4.131}$$

We denote $U = (u; u; \cdots; u) \in \mathbb{R}^{n\times d}$, i.e., all the n rows of U are u. Then we have $(\mathbf{p}^\top V - u^\top)x = 2\mathbf{p}^\top \frac{V-U}{2} x$. Furthermore, since $u, v_i \in \mathrm{B}_p^d$ for all $i \in [n]$, each row of $\frac{V-U}{2}$ is also in $u, v_i \in \mathrm{B}_p^d$. Finally, by the Sion's Theorem [176] we can switch the order of the min and max in (4.131). In all, to solve the approximate Carathéodory problem with precision ϵ, it suffices to solve the maximin game

$$\max_{x\in\mathrm{B}_q^d} \min_{\mathbf{p}\in\Delta_n} \mathbf{p}^\top \frac{V-U}{2} x \tag{4.132}$$

with precision $\frac{\epsilon}{2}$. This is exactly (4.114), thus the result follows from Theorem 4.1.1 and Theorem 4.8.1.

$\square$

Compared to [160], we pay a $\log n$ overhead in the cardinality of the convex combination, but in time complexity the dominating term nd is significantly improved to $n + d$. We also give the first sublinear quantum algorithm. Note that as **(author?)** [160] pointed out, the approximate Carathéodory problem has wide applications in machine learning and optimization, including support vector machines (SVMs), rounding in polytopes, submodular function minimization, etc. We elaborate the details of SVMs below, and leave out the details of other applications as the reductions are direct.

4.9.2 ℓ_q-margin support vector machine (SVM)

When we solve the ℓ_q-ℓ_1 matrix game in Algorithm 18, we apply ℓ_q-sampling where $j_t = j$ with probability $w(j)^q/\|w\|_q^q$ for any $j \in [d]$. The key reason of the success of Algorithm 18 is because the expectation of the random variable $A_i(j_t)\|x_t\|_q^q/x_t(j_t)^{q-1}$ in Line 8 is $X_i w$, which is unbiased.

If we consider some alternate random variables, we can potentially solve a maximin game in ℓ_q-ℓ_1 norm with respect to some nonlinear functions of the matrix. A specific problem of significant interest is the ℓ_q-margin support vector machine (SVM), where we are given n data points $X_1, \ldots, X_n$ in $\mathbb{R}^d$ and a label vector $y \in \{1, -1\}^n$. The goal is to find a separating hyperplane $w \in \mathbb{R}^d$ of these data points with the largest margin under the ℓ_q-norm loss, i.e.,

$$\sigma_{\mathrm{SVM}} := \max_{w \in \mathbb{R}^d} \min_{i \in [n]} 2y_i \cdot X_i^\top w - \|w\|_q^q. \tag{4.133}$$

Without loss of generality, we assume $y_i = 1$ for all $i \in [n]$, otherwise we take $X_i \leftarrow (-1)^{y_i} \cdot X_i$. In this case, the random variable $2X_i(j)\|w\|_q^q / w(j)^{q-1} - \|w\|_q^q$ is unbiased under ℓ_q-sampling on j:

$$\mathbb{E}\Big[\frac{2X_i(j)\|w\|_q^q}{w(j)^{q-1}} - \|w\|_q^q\Big] = 2X_i^\top w - \|w\|_q^q. \tag{4.134}$$

Note that $\sigma_{\mathrm{SVM}} \geq 0$ since $2X_i^\top w - \|w\|_q^q = 0$ for all $i \in [n]$ when $w = 0$. For the case $\sigma_{\mathrm{SVM}} > 0$ and taking $0 < \epsilon < \sigma_{\mathrm{SVM}}$, similar to Theorem 4.1.1 and Theorem 4.8.1 we have:

Corollary 4.9.2. *To return a vector $\bar{w} \in \mathbb{R}^d$ such that with probability at least $2/3$,*

$$\min_{i \in [n]} 2X_i\bar{w} - \|\bar{w}\|_q^q \geq \sigma_{\mathrm{SVM}} - \epsilon > 0, \tag{4.135}$$

there is a classical algorithm that achieves this with $O\big(\frac{(n+d)(p+\log n)}{\epsilon^2}\big)$ time and a quantum algorithm that achieves this with $\tilde{O}\big(\frac{p^2\sqrt{n}}{\epsilon^4} + \frac{p^{3.5}\sqrt{d}}{\epsilon^7}\big)$ time. We can also assume $p = O(\log d/\epsilon)$ (Lemma 4.6.1) and result in running time $O\big(\frac{(n+d)(\log d/\epsilon + \log n)}{\epsilon^2}\big)$ and $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^6} + \frac{\sqrt{d}}{\epsilon^{10.5}}\big)$, respectively.

Notice that classical sublinear algorithms for ℓ_2-SVMs have been given [126, 171], and there is also a sublinear quantum algorithm for ℓ_2-SVMs in [164]. We essentially generalize their results to the l_q-norm cases based on our new general matrix game solvers in Theorem 4.7.1 and Theorem 4.8.1.

4.10 Deferred Technical Details

We first give the details of our quantum algorithm.

Algorithm 20: A sublinear quantum algorithm for ℓ_q-ℓ_1 matrix games.

```
Input: $\epsilon > 0$; $p \in [2, +\infty), q \in (1, 2]$ such that $\frac{1}{p} + \frac{1}{q} = 1$; $A \in \mathbb{R}^{n\times d}$ with
       $A_i \in \mathrm{B}_p^d\ \forall i \in [n]$.
Output: $\bar{x}$ that satisfies (4.115).
1  Let $T = \lceil \frac{1346 \log n + 4p}{\epsilon^2} \rceil$, $y_1 = \mathbf{0}_d$, $\eta = \sqrt{\frac{11 \log n}{12T}}$, $w_1 = \mathbf{1}_n$, $|p_1\rangle = \frac{1}{\sqrt{n}} \sum_{i\in[n]} |i\rangle$;
2  for $t = 1$ to $T$ do
3      Measure the state $|p_t\rangle$ in the computational basis and denote the output as $i_t \in [n]$;
4      For each $i \in [t]$, estimate $\|A_{i_t}\|_p^p$ by Lemma 4.10.1 with precision $\delta = \eta^2$.
         Output:= $\widetilde{\|A_{i_t}\|}_p^p$;
5      Define[12] $y_{t+1}$ by $y_{t+1,j} \leftarrow y_t + \sqrt{\frac{q-1}{2T}} \frac{\mathrm{sgn}(A_{i_t,j})|A_{i_t,j}|^{p-1}}{\widetilde{\|A_{i_t}\|}_p^{p-2}}$ for all $j \in [d]$;
6      Apply Lemma 4.10.1 $2\lceil \log T \rceil$ times to estimate $\|y_t\|_q^q$ with precision $\delta = \eta^2$, and
         take the median of the $2\lceil \log T \rceil$ outputs, denoted by $\widetilde{\|y_t\|}_q^q$;
7      Choose $j_t \in [d]$ by $j_t = j$ with probability $y_t(j)^q / \|y_t\|_q^q$, which is achieved by
         applying Algorithm 19 to prepare the quantum state $|y_t\rangle$ and measure in the
         computational basis;
8      For all $i \in [n]$, denote $\tilde{v}_t(i) = A_i(j_t)\widetilde{\|y_t\|}_q^q / \big(y_t(j_t)^{q-1} \max\{1, \widetilde{\|y_t\|}_q\}\big)$,
         $v_t(i) = \mathrm{clip}(\tilde{v}_t(i), 1/\eta)$, and $u_{t+1}(i) = u_t(i)(1 - \eta v_t(i) + \eta^2 v_t(i)^2)$. Prepare an
         oracle $O_t$ such that $O_t|i\rangle|0\rangle = |i\rangle|u_{t+1}(i)\rangle$ for all $i \in [n]$, using $2t$ queries to $O_X$ and
         $\tilde{O}(t)$ additional arithmetic computations;
9      Prepare $|p_{t+1}\rangle = \frac{1}{\|u_{t+1}\|_2} \sum_{i\in[n]} u_{t+1}(i)|i\rangle$ using Algorithm 19 (with $q = 2$ therein)
         and $O_t$;
10 Return $\bar{w} = \frac{1}{T} \sum_{t=1}^{T} \frac{y_t}{\max\{1, \widetilde{\|y_t\|}_q\}}$.
```

We need the following lemma to estimate the norm of a vector:

[12]Here we do not write down the whole vector y_{t+1}, but we construct any query to its entries in $O(1)$ time.

Lemma 4.10.1 ((author?) 164, Lemma 6). *Given a function $F : [d] \to [0,1]$ with a quantum oracle $O_F : |i\rangle|0\rangle \mapsto |i\rangle|F(i)\rangle$ for all $i \in [d]$, let $m = \frac{1}{d}\sum_{i=1}^{d} F(i)$. Then for any $\delta < 0$, there is a quantum algorithm that uses $O(\sqrt{d}/\delta)$ queries to O_F and returns an $\tilde{m}$ such that $|m - \tilde{m}| \leq \delta m$ with probability at least $2/3$.*

We use the procedure below for preparing a quantum state given an oracle to a power of its coefficients:

Proposition 4.10.1. *Assume that $a \in \mathbb{C}^n$, and we are given a unitary oracle O_a such that $O|i\rangle|0\rangle = |i\rangle|a_i\rangle$ for all $i \in [n]$. Then Algorithm 19 takes $O(\sqrt{n})$ calls to O_a for preparing the quantum state $\frac{1}{\|a\|_q^{q/2}}\sum_{i\in[n]} a_i^{q/2}|i\rangle$ with success probability $1 - O(1/n)$.*

Proof. Note that Algorithm 2 of [164] had given a quantum algorithm for preparing an ℓ_2-norm pure state given an oracle to its coefficients, and Algorithm 19 essentially generalize this result to the ℓ_q-norm case by replacing all a_i by $a_i^{q/2}$ as in Algorithm 19. Note that the coefficient in (4.129) satisfies $\frac{\|a\|_q^{q/2}}{\sqrt{n}a_{\|q\|}} \geq \frac{1}{\sqrt{n}}$. As a result, applying amplitude amplification for $O(\sqrt{n})$ times indeed promises that we obtain 0 in the second system with success probability $1 - O(1/n)$, i.e., the state $\frac{1}{\|a\|_q^{q/2}}\sum_{i\in[n]} a_i^{q/2}|i\rangle$ is prepared. □

We need the following lemma.

Lemma 4.10.2. *For all $i \in [n]$, Define*

$$\tilde{v}_{t,\text{approx}}(i) := \frac{A_i(j_t)\widetilde{\|y_t\|}_q^q}{y_t(j_t)^{q-1}\max\{1, \widetilde{\|y_t\|_q}\}}, \quad \tilde{v}_{t,\text{true}}(i) := \frac{A_i(j_t)\|y_t\|_q^q}{y_t(j_t)^{q-1}\max\{1, \|y_t\|_q\}}. \qquad (4.136)$$

where $\widetilde{\|y_t\|}_q^q$ *and* $\|y_t\|_q^q$ *satisfy*

$$\left|\widetilde{\|y_t\|}_q^q - \|y_t\|_q^q\right| \leq \delta\|y_t\|_q^q \tag{4.137}$$

with probability at least $1 - o(1)$*. Also assume that* $\tilde{v}_{t,\text{approx}}(i), \tilde{v}_{t,\text{true}}(i) \leq 1/\eta$*. Then, it holds that for all* $i \in [n]$*,*

$$|\tilde{v}_{t,\text{approx}}(i) - \tilde{v}_{t,\text{true}}(i)| \leq \frac{\delta}{\eta} \quad \forall i \in [n], \tag{4.138}$$

with probability at least $1 - o(1)$*.*

Proof. First note that

$$|\tilde{v}_{t,\text{approx}}(i) - \tilde{v}_{t,\text{true}}(i)| = \tilde{v}_{t,\text{true}}(i)\left|\frac{\tilde{v}_{t,\text{approx}}(i)}{\tilde{v}_{t,\text{true}}(i)} - 1\right| \leq \frac{1}{\eta}\left|\frac{\tilde{v}_{t,\text{approx}}(i)}{\tilde{v}_{t,\text{true}}(i)} - 1\right|. \tag{4.139}$$

When $\|y_t\|_q \geq 1$, we have $\frac{\tilde{v}_{t,\text{approx}}(i)}{\tilde{v}_{t,\text{true}}(i)} = \frac{\widetilde{\|y_t\|}_q^{q-1}}{\|y_t\|_q^{q-1}}$, and when $\|y_t\|_q \leq 1$, we have $\frac{\tilde{v}_{t,\text{approx}}(i)}{\tilde{v}_{t,\text{true}}(i)} = \frac{\widetilde{\|y_t\|}_q^q}{\|y_t\|_q^q}$. By assumption, with probability at least $1 - o(1)$, it holds that $\left|\frac{\widetilde{\|y_t\|}_q^q}{\|y_t\|_q^q} - 1\right| \leq \delta$. Since $1 \leq \frac{\widetilde{\|y_t\|}_q^{q-1}}{\|y_t\|_q^{q-1}} \leq \frac{\widetilde{\|y_t\|}_q^q}{\|y_t\|_q^q}$ when $\widetilde{\|y_t\|}_q \geq \|y_t\|_q$, and $1 \geq \frac{\widetilde{\|y_t\|}_q^{q-1}}{\|y_t\|_q^{q-1}} \geq \frac{\widetilde{\|y_t\|}_q^q}{\|y_t\|_q^q}$ when $\widetilde{\|y_t\|}_q < \|y_t\|_q$, it also holds that $\left|\frac{\widetilde{\|y_t\|}_q^{q-1}}{\|y_t\|_q^{q-1}} - 1\right| \leq \delta$. Putting this into (4.139), we have the desired inequality. □

Now, we are ready to prove the main quantum result.

Proof of Theorem 4.8.1. First note that in Line 4, we use an estimation $\widetilde{\|A_{i_t}\|}_p^p$ of $\|A_{i_t}\|_p^p$ with relative error at most δ. Then in Line 5, $\widetilde{\|A_{i_t}\|}_p^{p-2}$ is an estimation of $\|A_{i_t}\|_p^{p-2}$ with relative error at most δ because $p \geq 2$ and $\widetilde{\|A_{i_t}\|}_p^{p-2} = (\widetilde{\|A_{i_t}\|}_p^p)^{(p-2)/p}$. Hence, y_{t+1} has a relative error of at most δ compared to its true value defined by

$$y_t + \sqrt{\frac{q-1}{2T}} \frac{\operatorname{sgn}(A_{i_t,j})|A_{i_t,j}|^{p-1}}{\|A_{i_t}\|_p^{p-2}}. \tag{4.140}$$

Consider Line 6. The estimate $\widetilde{\|y_t\|_q^q}$ is the median of $2\lceil \log T \rceil$ executions of Lemma 4.10.1. It implies that, with failure probability is at most $1-(2/3)^{2\log T} = 1 - T^2$, (4.137) holds. Since there are T iterations in total, the probability that (4.137) holds is at least $1 - T \cdot O(1/T^2) = 1 - o(1)$. Also consider (4.136). It is easy to see that $\tilde{v}_{t,\text{approx}}(i), \tilde{v}_{t,\text{true}}(i) \leq 1/\eta$ because of Line 8. Therefore, the conditions of Lemma 4.10.2 hold and its result follows.

As $\delta = \eta^2$, by Lemma 4.10.2 and Lemma 4.7.1, we have that with probability at least $5/7 - O(1/n)$,

$$\sum_{t\in[T]} A_{i_t} x_t \leq \sum_{t\in[T]} p_t^\top v_t + 11\eta T \leq \min_{i\in[n]} \sum_{t\in[T]} v_t(i) + 21\eta T + \frac{\log n}{\eta}. \tag{4.141}$$

Moreover, by Lemma 4.10.2 and Eq. (4.118), we have $\min_{i\in[n]} \sum_{t\in[T]} v_t(i) \leq 4\eta T + \eta T + \min_{i\in[n]} \sum\limits_{t\in[T]} A_i x_t$. Plugging this into (4.141), we have

$$\sum_{t\in[T]} A_{i_t} x_t \leq \sum_{t\in[T]} p_t^\top v_t + 11\eta T \leq \min_{i\in[n]} \sum_{t\in[T]} A_i x_t + 26\eta T + \frac{\log n}{\eta} \tag{4.142}$$

with probability $(5/7 - O(1/n)) \cdot (1 - O(1/n)) \geq 2/3$.

Similar to the proof of Theorem 4.1.1, we have

$$\min_{i\in[n]} A_i \bar{x} \geq \sigma - \sqrt{\frac{2p}{T}} - 26\eta - \frac{\log n}{\eta T}. \tag{4.143}$$

By the choices of p and η in Algorithm 20, the desired error bound for (4.130) holds because

$$\left(\sqrt{\frac{2p}{T}}+26\eta+\frac{\log n}{\eta T}\right)^2 \leq 2\left(\frac{2p}{T}\right)+2\left(26\eta+\frac{\log n}{\eta T}\right)^2 \leq \frac{4p+1346\log n}{T} \leq \epsilon^2, \quad (4.144)$$

where the first inequality follows from the AM-GM inequality and the last inequality follows from the choice of T in Algorithm 20.

Now, we analyze the time complexity. In Line 4 of Algorithm 20, the number of queries to O_A for Lemma 4.10.1 is $O(\sqrt{d}/\delta)=\tilde{O}(p\sqrt{d}/\epsilon^2)$. In Line 5, we have

$$y_{t,j}=\sqrt{\frac{q-1}{2T}}\sum_{\tau=1}^{t}\frac{\operatorname{sgn}(A_{i_\tau,j})|A_{i_\tau,j}|^{p-1}}{\widetilde{\|A_{i_\tau}\|_p}^{\,p-2}}. \quad (4.145)$$

An oracle for y_t can be implemented with $\tilde{O}(p/\epsilon^2)$ queries to O_A. To estimate $\|y_t\|_q$, we first need to normalize y_t. The summand in (4.145) is in the range $[-1,1]$, to see this, note that

$$\frac{|A_{i_\tau,j}|^{p-1}}{\|A_{i_\tau}\|_p^{p-2}} \leq \frac{|A_{i_\tau,j}|^{p-1}}{(|A_{i_\tau,j}|^p)^{(p-2)/p}} = |A_{i_\tau,j}| \leq 1. \quad (4.146)$$

Therefore, $y_{t,j}=\tilde{O}(\sqrt{pq}/\epsilon)=\tilde{O}(\sqrt{p}/\epsilon)$. Since the precision is $\delta=\eta^2=\tilde{\Theta}(\epsilon^2/p)$, the cost for amplitude estimation is $\tilde{O}(p\sqrt{d}/\epsilon^2)$. Finally, there are $T=\tilde{O}(p/\epsilon^2)$ iterations in total. The total complexity in Line 5 is

$$\tilde{O}\left(\frac{p}{\epsilon^2}\right)\cdot\tilde{O}\left(\frac{\sqrt{p}}{\epsilon}\right)\cdot\tilde{O}\left(\frac{p\sqrt{d}}{\epsilon^2}\right)\cdot\tilde{O}\left(\frac{p}{\epsilon^2}\right)=\tilde{O}\left(\frac{p^{3.5}\sqrt{d}}{\epsilon^7}\right). \quad (4.147)$$

For Line 6, we need to prepare the state $|y_t\rangle$. To simulate a query to an coefficient of y_t, we need $\tilde{O}(p/\epsilon^2)$ queries to O_A. The query complexity for Algorithm 19 is $O(\sqrt{d})$, and there are

$T = \tilde{O}(p/\epsilon)$ iterations in total. The total complexity in Line 6 is

$$\tilde{O}\left(\frac{p}{\epsilon^2}\right) \cdot O(\sqrt{d}) \cdot \tilde{O}\left(\frac{p}{\epsilon^2}\right) = \tilde{O}\left(\frac{p^2\sqrt{d}}{\epsilon^4}\right), \tag{4.148}$$

which is dominated by (4.147).

For Line 8, to implement one query to O_t, we need $2t$ queries to O_A with $\tilde{O}(t)$ additional arithmetic computations. For Line 9, to prepare the state $|p_{t+1}\rangle$, we need $O(\sqrt{n})$ queries to O_t, which can be implemented by $O(\sqrt{n}t)$ queries to O_A by Line 8 and $\tilde{O}(\sqrt{n}t)$ additional arithmetic computations. Therefore, the total complexity for Line 9 is

$$\sum_{t=1}^{T} \tilde{O}(\sqrt{n}t) = \tilde{O}(\sqrt{n}T^2) = \tilde{O}\left(\frac{p^2\sqrt{n}}{\epsilon^4}\right). \tag{4.149}$$

The time complexity of this algorithm is established by (4.147) and (4.149).

Finally, $\bar{x}$ has a succinct classical representation: using $i_1, \ldots, i_\tau$ obtained from Line 3 and $\widetilde{\|y_1\|}_q, \ldots, \widetilde{\|y_T\|}_q$ obtained from Line 6, a coordinate of $\bar{x}$ can be restored in time $T = \tilde{O}(p/\epsilon^2)$.

□

Chapter 5: Quantum Wasserstein GANs

This chapter presents a formulation of a variational algorithm for the stable and scalable learning of quantum states. The results presented were first obtained in [31].

5.1 Introduction

Generative adversarial networks (GANs) [177] represent a power tool of training deep *generative* models, which have a profound impact on machine learning. In GANs, a generator tries to generate fake samples resembling the true data, while a discriminator tries to discriminate between the true and the fake data. The learning process for generator and discriminator can be deemed as an adversarial game that converges to some equilibrium point under reasonable assumptions.

Inspired by the success of GANs and classical generative models, developing their quantum counterparts is a natural and important topic in the emerging field of quantum machine learning [127, 129]. There are at least two appealing reasons for which quantum GANs are extremely interesting. First, quantum GANs could provide potential quantum speedups due to the fact that quantum generators and discriminators (i.e., parameterized quantum circuits) cannot be efficiently simulated by classical generators/discriminators. In other words, there might exist distributions that can be efficiently generated by quantum GANs, while otherwise impossible with classical GANs.

Second, simple prototypes of quantum GANs (i.e., executing simple parameterized quantum circuits), similar to those of the variational methods (e.g., [178, 179, 180]), are likely to be implementable on near-term noisy-intermediate-scale-quantum (NISQ) machines [135]. Since the seminal work of [13], there are quite a few proposals (e.g, [181, 182, 183, 184, 185, 186, 187]) of constructions of quantum GANs on how to encode quantum or classical data into this framework. Furthermore, [185, 187] also demonstrated proof-of-principle implementations of small-scale quantum GANs on actual quantum machines.

A lot of existing quantum GANs focus on using quantum generators to generate classical distributions. For truly quantum applications such as investigation of quantum systems in condensed matter physics or quantum chemistry, the ability to generate *quantum data* is also important. In contrast to the case of classical distributions, where the loss function measuring the difference between the real and the fake distributions can be borrowed directly from the classical GANs, the design of the loss function between real and fake quantum data as well as the efficient training of the corresponding GAN is much more challenging. The only existing results on quantum data either have a unique design specific to the 1-qubit case [181, 185], or suffer from robust training issues discussed below [182].

More importantly, classical GANs are well known for being delicate and somewhat unstable in training. In particular, it is known [188] that the choice of the metric between real and fake distributions will be critical for the stability of the performance in the training. A few widely used ones such as the Kullback-Leibler (KL) divergence, the Jensen-Shannon (JS) divergence, and the total variation (or statistical) distance are not sensible for learning distributions supported by low dimensional generative models. The shortcoming of these metrics will likely carry through to their quantum counterparts and hence quantum GANs based on these metrics will likely suffer from the same weaknesses in training. This training issue was not significant in the existing numerical study of quantum GANs in the 1-qubit case [181, 185]. However, as observed by [182] and us, the training issue becomes much more significant when the quantum system scales up, even just in the case of a few qubits.

To tackle the training issue of classical GANs, a lot of research has been conducted on the convergence of training GANs in classical machine learning. A seminal work [188] used *Wasserstein distance* (or, *optimal transport* distance) [189] as a metric for measuring the distance between real and fake distributions. Comparing to other measures (such as KL and JS), Wasserstein distance is more appealing from optimization perspective because of its continuity and smoothness. As a result, the corresponding Wasserstein GAN (WGAN) is promising for improving the training stability of GANs. There are a lot of subsequent studies on various modifications of the WGAN, such as GAN with regularized Wasserstein distance [190], WGAN with entropic regularizers [191, 192], WGAN with gradient penalty [193, 194], relaxed WGAN [195], etc. It is known [196] that WGAN and its variants such as [193] have demonstrated improved training stability compared to the original GAN formulation.

Contributions. Inspired by the success of classical Wasserstein GANs and the need of smooth, robust, and scalable training methods for quantum GANs on quantum data, we propose the first design of quantum Wasserstein GANs (qWGANs). To this end, our technical contributions are multi-folded.

In Section 5.3, we propose a quantum counterpart of the Wasserstein distance, denoted by $\mathrm{qW}(P, Q)$ between quantum data P and Q, inspired by [188, 189]. We prove that $\mathrm{qW}(\cdot, \cdot)$ is a semi-metric (i.e., a metric without the triangle inequality) over quantum data and inherits nice properties such as continuity and smoothness of the classical Wasserstein distance. We will discuss a few other proposals of quantum Wasserstein distances such as [197, 198, 199, 200, 201, 202, 203, 204] and in particular why most of them are not suitable for the purpose of generating quantum data in GANs. We will also discuss the limitation of our proposal of quantum Wasserstein semi-metric and hope its successful application in quantum GANs could provide another perspective and motivation to study this topic.

In Section 5.4, we show how to add the quantum *entropic* regularization to $\mathrm{qW}(\cdot,\cdot)$ to further smoothen the loss function in the spirit of the classical case (e.g., [190]). We then show the construction of our regularized quantum Wasserstein GAN (qWGAN) in Figure 5.3 and describe the configuration and the parameterization of both the generator and the discriminator. Most importantly, we show that the evaluation of the loss function and the evaluation of the gradient of the loss function can be in principle efficiently implemented on quantum machines. This enables direct applications of classical training methods of GANs, such as alternating gradient-based optimization, to the quantum setting. It is a wide belief that classical computation cannot efficiently simulate quantum machines, in our case, the evaluation of the loss function and its gradient. Hence, the ability of evaluating them efficiently on quantum machines is *critical* for its scalability.

In Section 5.5, we supplement our theoretical results with experimental validations via classical simulation of qWGAN. Specifically, we demonstrate numerical performance of our qWGAN for quantum systems up to 8 qubits for pure states and up to 3 qubits for mixed states (i.e., mixture of pure states). Comparing to existing results [181, 182, 185], our numerical performance is more favorable in both the system size and its numerical stability. To give a rough sense, a single step in the classical simulation of the 8-qubit system involves multiple multiplications of $2^8 \times 2^8$ matrices. Learning a mixed state is much harder than learning pure states (a reasonable classical analogue of their difference is the one between learning a Gaussian distribution and learning a mixture of Gaussian distributions [205]). We present the only result for learning a true mixed state up to 3 qubits.

Furthermore, following a specific 4-qubit generator that is recently implemented on an ion-trap quantum machine [2] and a reasonable noise model on the same machine [206], we simulate the performance of our qWGAN with noisy quantum operations. Our result suggests that qWGAN can tolerant a reasonable amount of noise in quantum systems and still converge. This shows the possibility of implementing qWGAN on near-term (NISQ) machines [135].

Finally, we demonstrate a real-world application of qWGAN to approximate useful quantum application with large circuits by small ones. qWGAN can be used to approximate a potentially complicated unknown quantum state by a simple one when using a reasonably simple generator. We leverage this property and the Choi-Jamiołkowski isomorphism [207] between quantum operations and quantum states to generate a simple state that approximates another Choi-Jamiołkowski state corresponding to potentially complicated circuits in real quantum applications. The closeness in two Choi-Jamiołkowski states of quantum circuits will translate to the average output closeness between two quantum circuits over random input states. Specifically, we show that the quantum Hamiltonian simulation circuit for 1-d 3-qubit Heisenberg model in [208] can be approximated by a circuit of 52 gates with an average output fidelity over 0.9999 and a worst-case error 0.15. The best-known circuit based on the product formula will need $\sim$11900 gates, however, with a worst-case error 0.001.

Related results. All existing quantum GANs [13, 181, 182, 183, 184, 185, 186, 187], no matter dealing with classical or quantum data, have not investigated the possibility of using the Wasserstein distance. The most relevant works to ours are [181, 182, 185] with specific GANs dealing with quantum data. As we discussed above, [181, 185] only discussed the 1-qubit case (both pure and mixed) and [182] discussed the pure state case (up to 6 qubits) but with the loss function being the quantum counterpart of the total variation distance. Moreover, different from ours, the mixed state case in [181] is a labeled one: in addition to observing their mixture, one also gets a label of which pure state it is sampled from.

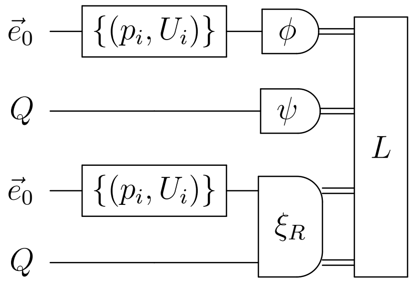


Figure 5.1: (1) $\{(p_i, U_i)\}$ refers to the generator with initial state $\vec{e}_0$ and its parameterization; (2) ϕ, ψ, ξ_R refers to the discriminator; (3) the figure shows how to evaluate the loss function L by measuring ϕ, ψ, ξ_R on the generated state and the real state Q with post-processing.

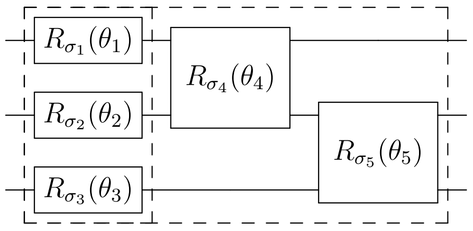


Figure 5.2: An example of a parameterized 3-qubit quantum circuit for U_i in the generator. $R_{\sigma_i}(\theta_i) = \exp(\frac{1}{2}\theta_i\sigma_i)$ denotes a Pauli rotation with angle θ_i. It could be a 1-qubit or 2-qubit gate depending on the specific Pauli matrix σ_i. The circuit consists of many such gates.

Figure 5.3: The Architecture of Quantum Wasserstein GAN.

5.2 Classical Wasserstein Distance & Wasserstein GANs

Let us first review the definition of Wasserstein distance and how it is used in classical WGANs.

Wasserstein distance Consider two probability distributions p and q given by corresponding density functions $p\colon \mathcal{X} \to \mathbb{R}, q\colon \mathcal{Y} \to \mathbb{R}$. Given a cost function $c\colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, the optimal transport cost between p and q, known as the *Kantorovich's* formulation [189], is defined as

$$d_c(p,q) := \min_{\pi \in \Pi(p,q)} \int_{\mathcal{X}} \int_{\mathcal{Y}} \pi(x,y) c(x,y) \,\mathrm{d}x \,\mathrm{d}y \tag{5.1}$$

where $\Pi(p,q)$ is the set of joint distributions π having marginal distributions p and q, i.e., $\int_{\mathcal{Y}} \pi(x,y)\,\mathrm{d}y = p(x)$ and $\int_{\mathcal{X}} \pi(x,y)\,\mathrm{d}x = q(y)$.

Wasserstein GAN The Wasserstein distance $d_c(p,q)$ can be used as an objective for learning a real distribution q by a parameterized function G_θ that acts on a base distribution p. Then the objective becomes learning parameters θ such that $d_c(G_\theta(p), q)$ is minimized as follows:

$$\min_{\theta} \min_{\pi \in \Pi(\mathcal{P},\mathcal{Q})} \int_{\mathcal{X}} \int_{\mathcal{Y}} \pi(x,y) c(G_\theta(x), y) \,\mathrm{d}x \,\mathrm{d}y. \tag{5.2}$$

In [188], Arjovsky et al. propose using the dual of (5.2) to formulate the original min-min problem into a min-max problem, i.e., a generative adversarial network, with the following form:

$$\begin{aligned} \min_{\theta} \max_{\alpha,\beta} \quad & \mathbb{E}_{x\sim\mathcal{P}}[\phi_\alpha(x)] - \mathbb{E}_{y\sim\mathcal{Q}}[\psi_\beta(y)], & (5.3)\\ \text{s.t} \quad & \phi_\alpha(G_\theta(x)) - \psi_\beta(y) \le c(G_\theta(x), y), \ \forall x, y, & (5.4) \end{aligned}$$

where ϕ, ψ are functions parameterized by α, β respectively. This is advantageous because it is usually easier to parameterize functions rather than joint distributions. The constraint (5.4) is usually enforced by a regularizer term for actual implementation. Out of many choices of regularizers, the most relevant one to ours is the entropy regularizer in [190]. In the case that $c(x,y) = \|x-y\|_2$ and $\phi = \psi$ in (5.3), the constraint is that ϕ must be a 1-Lipschitz function.

This is often enforced by the gradient penalty method in a neural network used to parameterize ϕ.

5.3 Quantum Wasserstein Semimetric

Mathematical formulation of quantum data Any quantum data (or quantum states) over space $\mathcal{X}$ (e.g., $\mathcal{X} = \mathbb{C}^d$) can be mathematically described by a *density operator* ρ that is a *positive semidefinite* matrix (i.e., $\rho \succeq 0$) with trace one (i.e., $\mathrm{Tr}(\rho) = 1$), and the set of which is denoted by $\mathcal{D}(\mathcal{X})$.

A quantum state ρ is *pure* if $\mathrm{rank}(\rho) = 1$; otherwise it is a *mixed* state. For a pure state ρ, it can be represented by the outer-product of a *unit* vector $\vec{v} \in \mathbb{C}^d$, i.e., $\rho = \vec{v}\vec{v}^\dagger$, where $\dagger$ refers to conjugate transpose. We can also use $\vec{v}$ to directly represent pure states. Mixed states are a classical mixture of pure states, e.g., $\rho = \sum_i p_i \vec{v}_i \vec{v}_i^\dagger$ where p_is form a classical distribution and $\vec{v}_i$s are all unit vectors.

Quantum states in a composed system of $\mathcal{X}$ and $\mathcal{Y}$ are represented by density operators ρ over the Kronecker-product space $\mathcal{X} \otimes \mathcal{Y}$ with dimension $\dim(\mathcal{X}) \dim(\mathcal{Y})$. 1-qubit systems refer to $\mathcal{X} = \mathbb{C}^2$. A 2-qubit system has dimension 4 ($\mathcal{X}^{\otimes 2}$) and an n-qubit system has dimension 2^n. The partial trace operation $\mathrm{Tr}_{\mathcal{X}}(\cdot)$ (resp. $\mathrm{Tr}_{\mathcal{Y}}(\cdot)$) is a linear mapping from ρ to its marginal state on $\mathcal{Y}$ (resp. $\mathcal{X}$).

From classical to quantum data Classical distributions p, q in (5.1) can be viewed as special mixed states $\mathcal{P} \in \mathcal{D}(\mathcal{X}), \mathcal{Q} \in \mathcal{D}(\mathcal{Y})$ where $\mathcal{P}, \mathcal{Q}$ are diagonal and p, q (viewed as density vectors) are the diagonals of $\mathcal{P}$, $\mathcal{Q}$ respectively. Note that this is different from the conventional meaning of samples from classical distributions, which are random variables with corresponding distributions.

This distinction is important to understand quantum data as the former (i.e., density operators) rather than the latter (i.e., samples) actually represents the entity of quantum data. This is because there are multiple ways (different quantum measurements) to read out classical samples out of quantum data for one fixed density operator. Mathematically, this is because density operators in general can have off-diagonal terms and quantum measurements can happen along arbitrary bases.

Consider $\mathcal{X}$ and $\mathcal{Y}$ from (5.1) being finite sets. We can express the classical Wasserstein distance (5.1) as a special case of the matrix formulation of quantum data. Precisely, we can replace the integral in (5.1) by summation, which can be then expressed by the trace of πC where C is a diagonal matrix with $c(x, y)$ in the diagonal. π is also a diagonal matrix expressing the coupling distribution $\pi(x, y)$ of p, q. Namely, π's diagonal is $\pi(x, y)$ and satisfies the coupling marginal condition $\mathrm{Tr}_{\mathcal{Y}}(\pi) = P$ and $\mathrm{Tr}_{\mathcal{X}}(\pi) = Q$ where P, Q are diagonal matrices with the distribution of p, q in the diagonal, respectively. As a result, the Kantorovich's optimal transport in (5.1) can be reformulated as

$$\begin{aligned} & d_c(p,q) := \min_{\pi} \mathrm{Tr}(\pi C) \qquad (5.5) \\ \text{s.t.} \quad & \mathrm{Tr}_{\mathcal{Y}}(\pi) = \mathrm{diag}\{p(x)\},\ \mathrm{Tr}_{\mathcal{X}}(\pi) = \mathrm{diag}\{q(y)\},\ \pi \in \mathcal{D}(\mathcal{X} \otimes \mathcal{Y}), \end{aligned}$$

where $C = \mathrm{diag}\{c(x, y)\}$. Note that (5.5) is effectively a linear program.

Quantum Wasserstein semimetric Our matrix reformulation of the classical Wasserstein distance (5.1) suggests a naive extension to the quantum setting as follows. Let $\mathrm{qW}(\mathcal{P}, \mathcal{Q})$ denote the quantum Wasserstein semimetric between $\mathcal{P} \in \mathcal{D}(\mathcal{X}), \mathcal{Q} \in \mathcal{D}(\mathcal{Y})$, which is defined by

$$\begin{aligned} \mathrm{qW}(\mathcal{P}, \mathcal{Q}) &:= \min_{\pi} \mathrm{Tr}(\pi C) \\ \text{s.t.} \quad \mathrm{Tr}_{\mathcal{Y}}(\pi) &= \mathcal{P},\ \mathrm{Tr}_{\mathcal{X}}(\pi) = \mathcal{Q},\ \pi \in \mathcal{D}(\mathcal{X} \otimes \mathcal{Y}), \end{aligned} \tag{5.6}$$

where C is a matrix over $\mathcal{X} \otimes \mathcal{Y}$ that should refer to some cost-type function. The choice of C is hence critical to make sense of the definition. First, matrix C needs to be Hermitian (i.e., $C = C^\dagger$) to make sure that $\mathrm{qW}(\cdot, \cdot)$ is real. A natural attempt is to use $C = \mathrm{diag}\{c(x, y)\}$ from (5.5), which turns out to be significantly wrong. This is because $\mathrm{qW}(\vec{v}\vec{v}^\dagger, \vec{v}\vec{v}^\dagger)$ will be strictly greater than zero for random choice of unit vector $\vec{v}$ in that case. This demonstrates a crucial difference between classical and quantum data: *while classical information is always stored in the diagonal (or computational basis) of the space, quantum information can be stored off-diagonally (or in an arbitrary basis of the space)*. Thus, choosing a diagonal C fails to detect the off-diagonal information in quantum data.

Our proposal is to leverage the concept of *symmetric subspace* in quantum information [209] to make sure that $\mathrm{qW}(P, P) = 0$ for any P. The projection onto the symmetric subspace is defined by

$$\Pi_{\text{sym}} := \frac{1}{2}(\mathbb{I}_{\mathcal{X}\otimes\mathcal{Y}} + \mathrm{SWAP}), \tag{5.7}$$

where $\mathbb{I}_{\mathcal{X}\otimes\mathcal{Y}}$ is the identity operator over $\mathcal{X} \otimes \mathcal{Y}$ and SWAP is the operator such that $\mathrm{SWAP}(\vec{x} \otimes \vec{y}) = (\vec{y} \otimes \vec{x}), \forall \vec{x} \in \mathcal{X}, \vec{y} \in \mathcal{Y}$.[1] It is well known that $\Pi_{\text{sym}}(\vec{u} \otimes \vec{u}) = \vec{u} \otimes \vec{u}$ for all unit vectors u.

[1] One needs that $\mathcal{X}$ is isometric to $\mathcal{Y}$ to well define Π_{sym}. However, this is without loss of generality by choosing appropriate and potentially larger spaces $\mathcal{X}$ and $\mathcal{Y}$ to describe quantum data.

With this property and by choosing C to be the complement of Π_{sym}, i.e.,

$$C := \mathbb{I}_{\mathcal{X}\otimes\mathcal{Y}} - \Pi_{\text{sym}} = \frac{1}{2}(\mathbb{I}_{\mathcal{X}\otimes\mathcal{Y}} - \text{SWAP}), \tag{5.8}$$

we can show $\mathrm{qW}(P, P) = 0$ for any P. This is achieved by choosing $\pi = \sum_i \lambda_i(\vec{v_i}\vec{v_i}^\dagger \otimes \vec{v_i}\vec{v_i}^\dagger)$ given P's spectral decomposition $P = \sum_i \lambda_i \vec{v_i}\vec{v_i}^\dagger$. Moreover, we can show

Theorem 5.3.1 (Proof in Appendix 5.7.2)**.** $\mathrm{qW}(\cdot,\cdot)$ *forms a semimetric over* $\mathcal{D}(\mathcal{X})$ *over any space* $\mathcal{X}$*, i.e., for any* $\mathcal{P}, \mathcal{Q} \in \mathcal{D}(\mathcal{X})$*,*

1. $\mathrm{qW}(\mathcal{P}, \mathcal{Q}) \geq 0$*,*
2. $\mathrm{qW}(\mathcal{P}, \mathcal{Q}) = \mathrm{qW}(\mathcal{Q}, \mathcal{P})$*,*
3. $\mathrm{qW}(\mathcal{P}, \mathcal{Q}) = 0$ *iff* $\mathcal{P} = \mathcal{Q}$*.*

Even though our definition of $\mathrm{qW}(\cdot,\cdot)$, especially the choice of C, does not directly come from a cost function $c(x, y)$ over $\mathcal{X}$ and $\mathcal{Y}$, it however still encodes some geometry of the space of quantum states. For example, let $P = \vec{v}\vec{v}^\dagger$ and $Q = \vec{u}\vec{u}^\dagger$, $\mathrm{qW}(P, Q)$ becomes 0.5 $(1 - |\vec{u}^\dagger\vec{v}|^2)$ where $|\vec{u}^\dagger\vec{v}|$ depends on the angle between $\vec{u}$ and $\vec{v}$ which are unit vectors representing (pure) quantum states.

The dual form of $\mathrm{qW}(\cdot,\cdot)$ The formulation of $\mathrm{qW}(\cdot,\cdot)$ in (5.6) is given by a semidefinite program (SDP), opposed to the classical form in (5.5) given by a linear program. Its dual form is as follows.

$$\begin{aligned} \max_{\phi,\psi} \quad & \mathrm{Tr}(Q\psi) - \mathrm{Tr}(P\phi) \\ \text{s.t.} \quad & \mathbb{I}_{\mathcal{X}} \otimes \psi - \phi \otimes \mathbb{I}_{\mathcal{Y}} \preceq C, \phi \in \mathcal{H}(\mathcal{X}),\ \psi \in \mathcal{H}(\mathcal{Y}), \end{aligned} \tag{5.9}$$

where $\mathcal{H}(\mathcal{X}), \mathcal{H}(\mathcal{Y})$ denote the set of Hermitian matrices over space $\mathcal{X}$ and $\mathcal{Y}$. We further show the *strong duality* for this SDP in Appendix 5.7.2. Thus, both the primal (5.6) and the dual (5.9) can be used as the definition of $\mathrm{qW}(\cdot,\cdot)$.

Comparison with other quantum Wasserstein metrics There have been a few different proposals that introduce matrices into the original definition of classical Wasserstein distance. We will compare these definitions with ours and discuss whether they are appropriate in our context of quantum GANs.

A few of these proposals (e.g., [201, 202, 210]) extend the dynamical formulation of Benamou and Brenier [211] in optimal transport to the matrix/quantum setting. In this formulation, couplings are defined not in terms of joint density measures, but in terms of smooth paths $t \to \rho(x,t)$ in the space of densities that satisfy some continuity equation with some time dependent vector field $v(x,t)$ inspired by physics. A pair $\{\rho(\cdot,\cdot), v(\cdot,\cdot)\}$ is said to couple P and Q, the set of which is denoted $C(P,Q)$, if $\rho(x,t)$ is a smooth path with $\rho(\cdot,0) = P$ and $\rho(\cdot,1) = Q$. The 2-Wasserstein distance is

$$\mathrm{W}_2(P,Q) = \inf_{\{\rho(\cdot,\cdot),v(\cdot,\cdot)\}\in C(P,Q)} \frac{1}{2}\int_0^1 \int_{R^n} |v(x,t)|^2 \rho(x,t)\,\mathrm{d}x\,\mathrm{d}t. \tag{5.10}$$

The above formulation seems difficult to manipulate in the context of GAN. It is unclear (a) whether the above definition has a favorable duality to admit the adversarial training and (b) whether the physics-inspired quantities like $v(x,t)$ are suitable for the purpose of generating fake quantum data.

A few other proposals (e.g., [197, 199]) introduce the matrix-valued mass defined by a function $\mu : X \to C^{n\times n}$ over domain X, where $\mu(x)$ is positive semidefinite and satisfies $\mathrm{Tr}(\int_X \mu(x)dx) = 1$. Instead of considering transport probability masses from X to Y, one considers transporting a matrix-valued mass $\mu_0(x)$ on X to another matrix-valued mass $\mu_1(y)$ on Y. One can similarly define the Kantorovich's coupling $\pi(x,y)$ of $\mu_0(x)$ and $\mu_1(y)$, and define the Wasserstein distance based on a slight different combination of $\pi(x,y)$ and $c(x,y)$ comparing to (5.1). This definition, however, fails to derive a new metric between two matrices. This is because the defined Wasserstein distance still measures the distance between X and Y based on some induced measure ($\|\cdot\|_F$) on the dimension-n matrix space. This is more evident when $X = \{P\}$ and $Y = \{Q\}$. The Wasserstein distance reduces to $c(x,y) + \|P-Q\|_F^2$ where the Frobenius norm ($\|\cdot\|_F$) is directly used in the definition.

The proposals in [198, 204] are very similar to us in the sense they define the same coupling in the Kantorovich's formulation as ours. However, their definition of the Wasserstein distance motivated by physics is induced by unbounded operator applied on continuous space, e.g., ∇_x, div_x. This makes their definition only applicable to continuous space, rather than qubits in our setting.

The closest result to ours is [203], although the authors haven't proposed one concrete quantum Wasserstein metric. Instead, they formulate a general form of reasonable quantum Wasserstein metrics between finite-dimensional quantum states and prove that Kantorovich-Rubinstein theorem does not hold under this general form. Namely, they show the trace distance between quantum states cannot be determined by any quantum Wasserstein metric out of their general form.

Limitation of our $\mathrm{qW}(\cdot,\cdot)$ Although we have successfully implemented qWGAN based on our $\mathrm{qW}(\cdot,\cdot)$ and observed improved numerical performance, there are a few perspectives about $\mathrm{qW}(\cdot,\cdot)$ worth further investigation. First, numerical study reveals that $\mathrm{qW}(\cdot,\cdot)$ does not satisfy the triangle inequality. Second, our $\mathrm{qW}(\cdot,\cdot)$ does not come from an explicit cost function, even though it encodes some geometry of the quantum state space. We conjecture that there could be a concrete underlying cost function and our $\mathrm{qW}(\cdot,\cdot)$ (or a related form) could be emerged as the 2-Wasserstein metric of that cost function. We hope our work provides an important motivation to further study this topic.

5.4 Quantum Wasserstein GAN

We describe the specific architecture of our qWGAN (Figure 5.3) and its training. Similar to (5.2) with the fake state P from a parameterized quantum generator G, consider

$$\begin{aligned} \min_{G} \min_{\pi} \quad & \mathrm{Tr}(\pi C) \\ \text{s.t.} \quad & \mathrm{Tr}_{\mathcal{Y}}(\pi) = P, \mathrm{Tr}_{\mathcal{X}}(\pi) = Q, \pi \in \mathcal{D}(\mathcal{X} \otimes \mathcal{Y}), \end{aligned} \tag{5.11}$$

or similar to (5.3) by taking the dual from (5.9),

$$\begin{aligned} \min_{G} \max_{\phi,\psi} \quad & \mathrm{Tr}(Q\psi) - \mathrm{Tr}(P\phi) = \mathbb{E}_Q[\psi] - \mathbb{E}_P[\phi] \\ \text{s.t.} \quad & \mathbb{I}_{\mathcal{X}} \otimes \psi - \phi \otimes \mathbb{I}_{\mathcal{Y}} \preceq C, \phi \in \mathcal{H}(\mathcal{X}),\ \psi \in \mathcal{H}(\mathcal{Y}), \end{aligned} \tag{5.12}$$

where we abuse the notation of $\mathbb{E}_Q[\psi] := \mathrm{Tr}(Q\psi)$, which refers to the expectation of the outcome of measuring Hermitian ψ on quantum state Q. We hence refer ϕ, ψ as the discriminator.

Regularized Quantum Wasserstein GAN

The dual form (5.12) is inconvenient for optimizing directly due to the constraint $\mathbb{I}_{\mathcal{X}} \otimes \psi - \phi \otimes \mathbb{I}_{\mathcal{Y}} \preceq C$. Inspired by the entropy regularizer in the classical setting (e.g., [190]), we add a *quantum-relative-entropy-based* regularizer between π and $P \otimes Q$ with a tunable parameter λ to (5.11) to obtain

$$\begin{aligned} \min_G \min_\pi \quad & \mathrm{Tr}(\pi C) + \lambda\,\mathrm{Tr}(\pi \log(\pi) - \pi \log(P \otimes Q)) \\ \text{s.t.} \quad & \mathrm{Tr}_{\mathcal{Y}}(\pi) = P, \mathrm{Tr}_{\mathcal{X}}(\pi) = Q, \pi \in \mathcal{D}(\mathcal{X} \otimes \mathcal{Y}). \end{aligned} \tag{5.13}$$

Using duality and the Golden-Thomposon inequality [212, 213], we can approximate (5.13) by

$$\min_G \max_{\phi,\psi} \quad \mathbb{E}_Q[\psi] - \mathbb{E}_P[\phi] - \mathbb{E}_{P\otimes Q}[\xi_R] \quad \text{s.t. } \phi \in \mathcal{H}(\mathcal{X}),\ \psi \in \mathcal{H}(\mathcal{Y}), \tag{5.14}$$

where ξ_R refers to the regularizing Hermitian

$$\xi_R = \frac{\lambda}{e} \exp\left(\frac{-C - \phi \otimes \mathbb{I}_{\mathcal{Y}} + \mathbb{I}_{\mathcal{X}} \otimes \psi}{\lambda}\right). \tag{5.15}$$

Similar to [190], we prove that this entropic regularization ensures that the objective for the outer minimization problem (5.14) is *differentiable* in P. (Proofs are given in Section 5.7.3.)

Parameterization of the Generator and the Discriminator

Generator G is a quantum operation that generates P from a fixed initial state ρ_0 (e.g., the classical all-zero state $\vec{e}_0$). Specifically, generator G can be described by an ensemble $\{(p_1, U_1), \ldots, (p_r, U_r)\}$ that means applying the unitary U_i with probability p_i. The distribution $\{p_1, \ldots, p_r\}$ can be parameterized directly or through some classical generative network. The rank of the generated state is r ($r = 1$ for pure states and $r > 1$ for mixed states). Our experiments include the cases $r = 1, 2$.

Each unitary U_i refers to a quantum circuit consisting of simple parameterized 1-qubit and 2-qubit Pauli-rotation quantum gates (see the right of Figure 5.3). These Pauli gates can be implemented on near-term machines (e.g., [2]) and also form a universal gate set for quantum computation. Hence, this generator construction is widely used in existing quantum GANs. The jth gate in U_i contains an angle $\theta_{i,j}$ as the parameter. All variables p_i, $\theta_{i,j}$ constitute the set of parameters for the generator.

Discriminator ϕ, ψ can be parameterized at least in two ways. The first approach is to represent ϕ, ψ as linear combinations of tensor products of Pauli matrices, which form a basis of the matrix space. Let $\phi = \sum_k \alpha_k A_k$ and $\psi = \sum_l \beta_l B_l$, where A_k, B_l are tensor products of Pauli matrices. To evaluate $\mathbb{E}_P[\phi]$ (similarly for $\mathbb{E}_Q[\psi]$), by linearity it suffices to collect the information of $\mathbb{E}_P[A_k]$s, which are simply Pauli measurements on the quantum state P and amenable to experiments. Hence, α_k and β_l can be used as the parameters of the discriminator. The second approach is to represent ϕ, ψ as parameterized quantum circuits (similar to the G) with a measurement in the computational basis. The set of parameters of ϕ (respectively ψ) could be the parameters of the circuit and values associated with each measurement outcome.

Our implementation mostly uses the first parameterization.

Training the Regularized Quantum Wasserstein GAN

For the scalability of the training of the Regularized Quantum Wasserstein GAN, one must be able to evaluate the loss function $L = \mathbb{E}_Q[\psi] - \mathbb{E}_P[\phi] - \mathbb{E}_{P\otimes Q}[\xi_R]$ or its gradient efficiently on a quantum computer. Ideally, one would hope to directly approximate gradients by quantum computers to facilitate the training of qWGAN, e.g., by using the alternating gradient descent method. We show that it is indeed possible and outline the key steps. More details are in Section 5.8.

Computing the loss function: Each unitary operation U_i that refers to an actual quantum circuit can be efficiently evaluated on quantum machines in terms of the circuit size. It can be shown that L is a linear function of P and can be computed by evaluating each $L_i = \mathbb{E}_Q[\psi] - \mathbb{E}_{U_i\rho_0 U_i^\dagger}[\phi] - \mathbb{E}_{U_i\rho_0 U_i^\dagger\otimes Q}[\xi_R]$ where $U_i\rho_0 U_i^\dagger$ refers to the state after applying U_i on ρ_0. Similarly, one can show that L is a linear function of the Hermitian matrices ϕ, ψ, ξ_R. Our parameterization of ϕ and ψ readily allows the use of efficient Pauli measurements to evaluate $\mathbb{E}_P[\phi]$ and $\mathbb{E}_Q[\psi]$. To handle the tricky part $\mathbb{E}_{P\otimes Q}[\xi_R]$, we relax ξ_R and use a Taylor series to approximate $\mathbb{E}_{P\otimes Q}[\xi_R]$; the result form can again be evaluated by Pauli measurements composed with simple SWAP operations. As the major computation (e.g., circuit evaluation and Pauli measurements) is efficient on quantum machines, the overall implementation is efficient with possible overhead of sampling trials.

Computing the gradients: The parameters of the qWGAN are $\{p_i\} \cup \{\theta_{i,j}\} \cup \{\alpha_k\} \cup \{\beta_l\}$. L is a linear function of p_i, α_k, β_l. Thus it can be shown that the partial derivatives w.r.t. p_i can be computed by evaluating the loss function on a generated state $U_i\rho_0 U_i^\dagger$ and the partial derivatives w.r.t. α_k, β_l can be computed by evaluating the loss function with ϕ, ψ replaced with A_k, B_l

respectively. The partial derivatives w.r.t. $\theta_{i,j}$ can be evaluated using techniques due to [214] via a simple yet elegant modification of the quantum circuits used to evaluate the loss function. The complexity analysis is similar to above. The only new ingredient is the quantum circuits to evaluate the partial derivatives w.r.t. $\theta_{i,j}$ due to [214], which are again efficient on quantum machines.

Summary of the training complexity: A rough complexity analysis above suggests that one step of the evaluation of the loss function (or the gradients) of our qWGAN can be efficiently implemented on quantum machines. (A careful analysis is in Appendix 5.8.5.) Given this ability, the rest of the training of qWGAN is similar to the classical case and will share the same complexity. It is worthwhile mentioning that quantum circuit evaluation and Pauli measurements are not known to be efficiently computable by classical machines; the best known approach will cost exponential time.

5.5 Experimental Results

We supplement our theoretical findings with numerical results by classical simulation of quantum WGANs of learning *pure* states (up to 8 qubits) and *mixed* states (up to 3 qubits) as well as its performance on noisy quantum machines. We use quantum fidelity between the generated and target states to track the progress of our quantum WGAN. If the training is successful, the fidelity will approach 1. Our quantum WGAN is trained using the alternating gradient descent method.

In most of the cases, the target state is generated by a circuit sharing the same structure with the generator but with randomly chosen parameters. We also demonstrate a special target

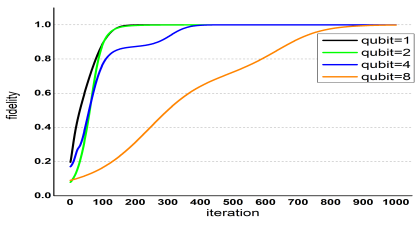


Figure 5.4: Fidelity vs Training Epochs

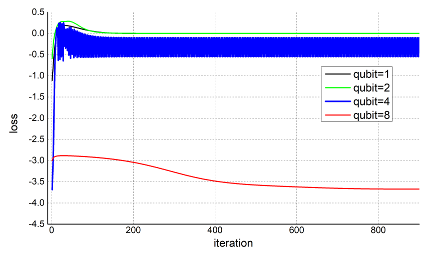


Figure 5.5: Training Loss

Figure 5.6: A typical performance of learning pure states (1,2,4, and 8 qubits).

state corresponding to useful quantum unitaries via the Choi-Jamiołkowski isomorphism. More details of the following experiments (e.g., parameter choices) can be found in Appendix 5.9.

Most of the simulations were run on a dual core Intel I5 processor with 8G memory. The 8-qubit pure state case was run on a Dual Intel Xeon E5-2697 v2 @ 2.70GHz processor with 128G memory. All source codes are publicly available at `https://github.com/yiminghwang/qWGAN`.

Pure states We demonstrate a typical performance of quantum WGAN of learning $1, 2, 4$, and 8 qubit pure states in Figure 5.6. We also plot the average fidelity for 10 runs with random initializations in Figure 5.11 which shows the numerical stability of qWGAN.

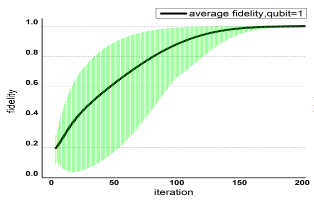


Figure 5.7: 1 qubit

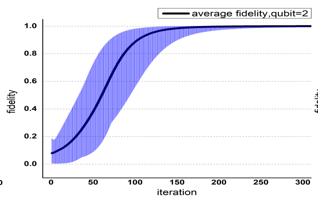


Figure 5.8: 2 qubits

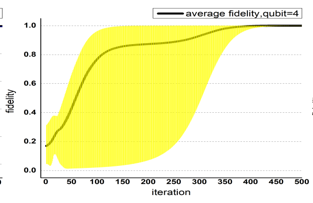


Figure 5.9: 4 qubits

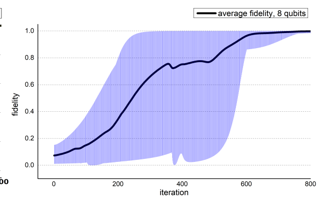


Figure 5.10: 8 qubits

Figure 5.11: Average performance of learning pure states (1, 2, 4, 8 qubits) where the black line is the average fidelity over multi-runs with random initializations and the shaded area refers to the range of the fidelity.

Mixed states We also demonstrate a typical learning of mixed quantum states of rank 2 with $1, 2$, and 3 qubits in Figure 5.15. The generator now consists of 2 unitary operators and 2 real

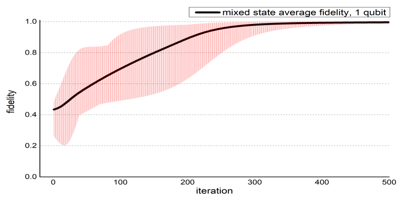


Figure 5.12: 1 qubit

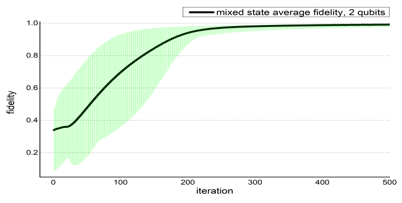


Figure 5.13: 2 qubits

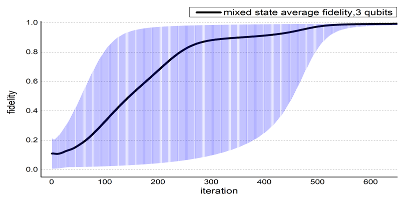


Figure 5.14: 3 qubits

Figure 5.15: Average performance of learning mixed states (1, 2, 3 qubits) where the black line is the average fidelity over multi-runs with random initializations and the shaded area refers to the range of the fidelity.

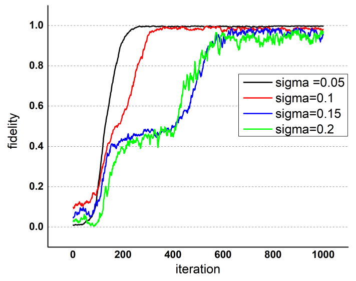


Figure 5.16: Learning 4-qubit pure states with noisy quantum operations.

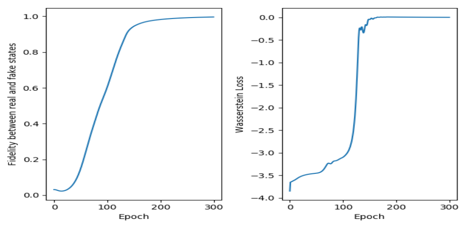


Figure 5.17: Learning to approximate the 3-qubit Hamiltonian simulation circuit of the 1-d Heisenberg model.

probability parameters p_1, p_2 which are normalized to form a probability distribution using a softmax layer.

Learning pure states with noise To investigate the possibility of implementing our quantum WGAN on near-term machines, we perform a numerical test on a practically implementable 4-qubit generator on the ion-trap machine [2] with an approximate noise model [206]. We deem this as the closest example that we can simulate to an actual physical experiment. In particular, we add a Gaussian sampling noise with standard deviation $\sigma = 0.2, 0.15, 0.1, 0.05$ to the measurement outcome of the quantum system. Our results (in Figure 5.16) show that the quantum WGAN can still learn a 4-qubit pure state in the presence of this kind of noise. As expected, noise of higher degrees (higher σ) increases the number of epochs before the state is learned successfully.

Comparison with existing experimental results We will compare to quantum GANs with quantum data [181, 182, 185]. It is unfortunate that there is neither precise figure nor public data in their papers which makes a precise comparison infeasible. However, we manage to give a rough comparison as follows. Ref. [181] studies the pure state and the labeled mixed state case for 1 qubit. It can be inferred from the plots of their results (Figure 8.b in [181]) that the relative entropy for both labels converges to 10^{-10} after $\sim$ 5000 iterations, and it takes more than 1000 iterations for the relative entropy to significantly decrease from 1. Ref. [185] performs experiments to learn 1-qubit pure and mixed states using a quantum GAN on a superconducting quantum circuit. However, the specific design of their GAN is very unique to the 1-qubit case. They observe that the fidelity between the fake state and the real state approaches 1 after 220 iterations for the pure state, and 120 iterations for the mixed state. From our figures, qWGAN can quickly converge for 1-qubit pure states after $150-160$ iterations and for a 1-qubit mixed state after ~ 120 iterations.

Ref. [182] studies only pure states but with numerical results up to 6 qubits. In particular, they demonstrate (in Figure 6 from [182]) in the case of 6-qubit that the normal gradient descent approach, like the one we use here, won't make much progress at all after 600 iterations. Hence they introduce a new training method. This is in sharp contrast to our Figure 5.6 where we demonstrate smooth convergence to fidelity 1 with the simple gradient descent for 8-qubit pure states within 900 iterations.

Application: approximating quantum circuits To approximate any quantum circuit U_0 over n-qubit space $\mathcal{X}$, consider Choi-Jamiołkowski state Ψ_0 over $\mathcal{X}\otimes\mathcal{X}$ defined as $(U_0\otimes\mathbb{I}_{\mathcal{X}})\Phi$ where Φ is the maximally entangled state $\frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}\vec{e_i}\otimes\vec{e_i}$ and $\{\vec{e_i}\}_{i=0}^{2^n-1}$ forms an orthonormal basis of $\mathcal{X}$. The generator is the normal generator circuit U_1 on the first $\mathcal{X}$ and identity on the second $\mathcal{X}$, i.e., $U_1\otimes\mathbb{I}$. In order to learn for the 1-d 3-qubit Heisenberg model circuit (treated as U_0) in [208], we simply run our qWGAN to learn the 6-qubit Choi-Jamiołkowski state Ψ_0 in Figure 5.17 and obtain the generator (i.e., U_1). We use the gate set of single or 2-qubit Pauli rotation gates. Then U_1 only has 52 gates, while using the best product-formula (2nd order) U_0 has $\sim$11900 gates. It is worth noting that U_1 achieves an average output fidelity over 0.9999 and a worst-case error 0.15, whereas U_0 has a worst-case error 0.001. However, the worst-case input of U_1 is not realistic in current experiments and hence the high average fidelity implies very reasonable approximation in practice.

5.6 Conclusion & Open Questions

We provide the first design of quantum Wasserstein GANs, its performance analysis on realistic quantum hardware through classical simulation, and a real-world application in this paper. At the technical level, we propose a counterpart of Wasserstein metric between quantum data. We believe that our result opens the possibility of quite a few future directions, for example:

- Can we implement our quantum WGAN on an actual quantum computer? Our noisy simulation suggests the possibility at least on an ion-trap machine.

- Can we apply our quantum WGAN to even larger and noisy quantum systems? In particular, can we approximate more useful quantum circuits using small ones by quantum WGAN? It seems very likely but requires more careful numerical analysis.

- Can we better understand and build a rich theory of quantum Wasserstein metrics in light of [189]?

5.7 Deferred Techinical Details

5.7.1 Matrix Arithmetics

Unless otherwise mentioned, the matrices we consider are *Hermitian*, defined as all matrices A such that $A^\dagger = A$. For any two Hermitian matrices $A, B \in \mathbb{C}^{n\times n}$, we say $A \succeq B$ iff $A - B$ is a positive semidefinite matrix (i.e., $A - B$ only has nonnegative eigenvalues), and $A \succ B$ iff $A - B$ is a positive definite matrix (i.e., $A - B$ only has positive eigenvalues).

A function of a Hermitian matrix is computed by taking summations of matrix powers under its Taylor expansion; for instance, for any Hermitian A we have

$$\exp(A) := \sum_{k=0}^{\infty} \frac{A^k}{k!}, \tag{5.16}$$

and for any $0 \prec B \prec 2I$ we have

$$\log(B) := \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k}(B - I)^k. \tag{5.17}$$

Furthermore, we introduce two tools for matrix arithmetics that we frequently use throughout the paper. The first is a rule for taking gradients of matrix functions:

Lemma 5.7.1 ([215]). *Given a Hermitian matrix $W \in \mathbb{C}^{n\times n}$ and a function $f : \mathbb{R} \to \mathbb{R}$, we define the gradient $\nabla_W f(W)$ as the entry-wise derivatives, i.e., $\nabla_W f(W) := (\frac{\partial f(W)_{ij}}{\partial W_{ij}})_{i,j=1}^n$. Then we have*

$$\nabla_W \operatorname{Tr}(W \log(W)) = [\log(W) + (W)]^\dagger = \log(W) + W. \tag{5.18}$$

For exponentiations of Hermitian matrices, we use the Golden-Thompson inequality stated as follows:

Lemma 5.7.2 ([212, 213])**.** *For any Hermitian matrices $A, B \in \mathbb{C}^{n\times n}$,*

$$\mathrm{Tr}(\exp(A+B)) \le \mathrm{Tr}(\exp(A)\exp(B)). \tag{5.19}$$

5.7.2 Properties of the Quantum Wasserstein Semimetric

5.7.2.1 Proofs

Lemma 5.7.3. *Strong Duality holds for the semidefinite program (5.6).*

Proof. Note that $\pi = \mathcal{P} \otimes \mathcal{Q}$ is a feasible solution to the primal program (5.6).

Consider the solution $\psi = -\mathbb{I}_{\mathcal{Y}}$, $\phi = \mathbb{I}_{\mathcal{X}}$ for the dual program (5.6). Then $\mathbb{I}_{\mathcal{X}} \otimes \psi - \phi \otimes \mathbb{I}_{\mathcal{Y}} - C = -2\mathbb{I}_{\mathcal{X}} \otimes \mathbb{I}_{\mathcal{Y}} - C$. For any vector $v \in \mathcal{X} \otimes \mathcal{Y}$, $v^\dagger(-2\mathbb{I}_{\mathcal{X}} \otimes \mathbb{I}_{\mathcal{Y}} - C)v = -2 - v^\dagger C v \le -2 < 0$. Therefore $\mathbb{I}_{\mathcal{X}} \otimes \psi - \phi \otimes \mathbb{I}_{\mathcal{Y}} \prec C$ and the solution is strictly feasible. Since a strictly feasible solution exists to the dual program and the primal feasible set is non-empty, Slater's conditions are satisfied and the lemma holds [216, Theorem 1 (1)]. □

Lemma 5.7.3 shows that the primal and dual SDPs have the same optimal value and thus (5.9) can be taken as an alternate definition of the Quantum Wasserstein distance.

The following theorem establishes some properties of the Quantum Wasserstein distance.

Theorem 5.7.1. $\mathrm{qW}(\cdot,\cdot)$ *forms a semimetric over the set of density matrices $\mathcal{D}(\mathcal{X})$ over any space $\mathcal{X}$, i.e., for any $\mathcal{P}, \mathcal{Q} \in \mathcal{D}(\mathcal{X})$,*

1. $\mathrm{qW}(\mathcal{P}, \mathcal{Q}) \ge 0$, *2.* $\mathrm{qW}(\mathcal{P}, \mathcal{Q}) = \mathrm{qW}(\mathcal{Q}, \mathcal{P})$,

3. $\mathrm{qW}(\mathcal{P}, \mathcal{Q}) = 0$ *iff* $\mathcal{P} = \mathcal{Q}$.

Proof. We will use the definition of $\mathrm{qW}(\cdot, \cdot)$ from (5.6) with $\mathcal{Y}$ being an isometric copy of $\mathcal{X}$.

1. Consider the matrix $C = \frac{\mathbb{I}-\mathrm{SWAP}}{2}$. Let $\vec{u} = \sum_{i,j\in\Gamma} u_{ij}\vec{e}_i\vec{e}_j$ be any vector in $\mathcal{X}\otimes\mathcal{Y} = \mathbb{C}^{|\Gamma|}\otimes\mathbb{C}^{|\Gamma|}$. By simple calculation,

$$\vec{u}^\dagger C\vec{u} = \sum_{i,j} u_{ij}^*(u_{ij} - u_{ji}) = \sum_{i\le j}(u_{ij}^* - u_{ji}^*)(u_{ij} - u_{ji}) = \sum_{i\le j} |u_{ij} - u_{ji}|^2 \ge 0; \tag{5.20}$$

thus C is positive semidefinite. As a result, $\mathrm{Tr}(\pi C) \ge 0$ for all $\pi \succeq 0$, and $\mathrm{qW}(\mathcal{P}, \mathcal{Q}) \ge 0$ for all density matrices $\mathcal{P}, \mathcal{Q} \in \mathcal{D}(\mathcal{X})$.

2. This property trivially holds because of the definition in (5.6) is symmetric in $\mathcal{P}$ and $\mathcal{Q}$.

3. Suppose that $P = Q$ have spectral decomposition $\sum_i \lambda_i\vec{v}_i\vec{v}_i^\dagger$. Consider $\pi_0 = \sum_i \lambda_i(\vec{v}_i\vec{v}_i^\dagger \otimes \vec{v}_i\vec{v}_i^\dagger)$. Then, $\mathrm{Tr}(\pi_0 C) = \mathrm{Tr}(\sum_i \lambda_i(\vec{v}_i\vec{v}_i^\dagger \otimes \vec{v}_i\vec{v}_i^\dagger)C) = \mathrm{Tr}(\sum_i \lambda_i(\vec{v}_i^\dagger \otimes \vec{v}_i^\dagger)C(\vec{v}_i \otimes \vec{v}_i))$. Since $C = \frac{I-\mathrm{SWAP}}{2}$, $C(\vec{v}_i \otimes \vec{v}_i) = 0$. Thus $\mathrm{Tr}(\pi_0 C) = 0$ and since C is positive semidefinite, this must be the minimum. Thus $\mathrm{qW}(\mathcal{P}, \mathcal{P}) = 0$. □

5.7.3 Regularized Quantum Wasserstein Distance

The regularized primal version of the Quantum Wasserstein GAN is constructed from (5.11) by adding the relative entropy between the optimization variable π and the joint distribution of the real and fake states $P \otimes Q$, given by $S(\pi \| P \otimes Q) = \mathrm{Tr}(\pi \log(\pi) - \pi \log(P \otimes Q))$:

$$\begin{aligned} \min_{\pi} \quad & \mathrm{Tr}(\pi C) + \lambda \,\mathrm{Tr}(\pi \log(\pi) - \pi \log(P \otimes Q)) \\ \text{s.t.} \quad & \mathrm{Tr}_{\mathcal{Y}}(\pi) = P, \mathrm{Tr}_{\mathcal{X}}(\pi) = Q, \pi \in \mathcal{D}(\mathcal{X} \otimes \mathcal{Y}). \end{aligned} \tag{5.21}$$

Here λ is a parameter that is chosen during training, and determines the weight given to the regularizer.

To formulate the dual, we use Hermitian Lagrange multipliers ϕ and ψ to construct a saddle point problem:

$$\begin{aligned} \min_{\pi} \max_{\psi,\phi} \quad & \mathrm{Tr}(\pi C) + \lambda \,\mathrm{Tr}(\pi \log(\pi) - \pi \log(P \otimes Q)) \\ & \quad + \mathrm{Tr}(\phi(\mathrm{Tr}_{\mathcal{Y}}(\pi) - P)) - \mathrm{Tr}(\psi(\mathrm{Tr}_{\mathcal{X}}(\pi) - Q)) \\ = \min_{\pi} \max_{\psi,\phi} \quad & \mathrm{Tr}(\pi(C + \phi \otimes \mathbb{I}_{\mathcal{Y}} - \mathbb{I}_{\mathcal{X}} \otimes \psi)) - \mathrm{Tr}(P\phi) \\ & \quad + \mathrm{Tr}(Q\psi) + \lambda \,\mathrm{Tr}(\pi \log(\pi) - \pi \log(P \otimes Q)). \end{aligned} \tag{5.22}$$

Switching the order of the optimizations:

$$\begin{aligned} \max_{\psi,\phi} \min_{\pi} \quad & \mathrm{Tr}(\pi(C + \phi \otimes \mathbb{I}_{\mathcal{Y}} - \mathbb{I}_{\mathcal{X}} \otimes \psi)) - \mathrm{Tr}(P\phi) \\ & \quad + \mathrm{Tr}(Q\psi) + \lambda \,\mathrm{Tr}(\pi \log(\pi) - \pi \log(P \otimes Q)). \end{aligned} \tag{5.23}$$

Solving the inner optimization problem for π and using Lemma 5.7.1, we have that for the optimal

π,

$$(C + \phi \otimes \mathbb{I}_{\mathcal{Y}} - \mathbb{I}_{\mathcal{X}} \otimes \psi) + \lambda \log(\pi) + \lambda \mathbb{I} - \log(P \otimes Q) = 0. \tag{5.24}$$

Thus the dual optimization problem reduces to

$$\begin{aligned} \max_{\phi,\psi} \quad & \mathrm{Tr}(Q\psi) - \mathrm{Tr}(P\phi) - \frac{\lambda}{e} \mathrm{Tr}\left(\exp\left(\frac{\log(P \otimes Q) - C - \phi \otimes \mathbb{I}_{\mathcal{Y}} + \mathbb{I}_{\mathcal{X}} \otimes \psi}{\lambda}\right)\right) \\ \text{s.t.} \quad & \phi \in \mathcal{H}(\mathcal{X}), \psi \in \mathcal{H}(\mathcal{Y}). \end{aligned} \tag{5.25}$$

Note that the additional term in the objective of the dual cannot be directly written as the expected value of measuring a Hermitian operator. However, we can use the Golden-Thompson inequality (Lemma 5.7.2) to upper bound on the objective, which can be written in terms of the expectation as

$$\begin{aligned} \max_{\phi,\psi} \quad & \mathrm{Tr}(Q\psi) - \mathrm{Tr}(P\phi) - \frac{\lambda}{e} \mathrm{Tr}\left((P \otimes Q)\exp\left(\frac{-C - \phi \otimes \mathbb{I}_{\mathcal{Y}} + \mathbb{I}_{\mathcal{X}} \otimes \psi}{\lambda}\right)\right) \\ & = \max_{\phi,\psi} \mathbb{E}_Q[\psi] - \mathbb{E}_P[\phi] - \frac{\lambda}{e} \cdot \mathbb{E}_{P \otimes Q}\left[\exp\left(\frac{-C - \phi \otimes \mathbb{I}_{\mathcal{Y}} + \mathbb{I}_{\mathcal{X}} \otimes \psi}{\lambda}\right)\right] \\ \text{s.t.} \quad & \phi \in \mathcal{H}(\mathcal{X}), \ \psi \in \mathcal{H}(\mathcal{Y}). \end{aligned} \tag{5.26}$$

The regularized optimization problem has the following property:

Lemma 5.7.4. *Let $f \colon \mathcal{D}(\mathcal{X}) \to \mathbb{R}$ be defined as*

$$\mathbb{E}_Q[\psi] - \mathbb{E}_P[\phi] - \frac{\lambda}{e} \cdot \mathbb{E}_{P\otimes Q}\left[\exp\left(\frac{-C - \phi \otimes \mathbb{I}_{\mathcal{Y}} + \mathbb{I}_{\mathcal{X}} \otimes \psi}{\lambda}\right)\right] \tag{5.27}$$
$$\textit{s.t.} \quad \phi \in \mathcal{H}(\mathcal{X}),\ \psi \in \mathcal{H}(\mathcal{Y}).$$

Then $f(P)$ is a differentiable function of P.

Proof. The optimization objective (5.27) is clearly convex with respect to its parameters. Furthermore, the second derivatives are non-zero for all ϕ, ψ, and the optimum hence is reached at a unique point. The objective function can be rewritten as

$$\mathbb{E}_{P\otimes Q}\left(-\phi \otimes \mathbb{I}_{\mathcal{Y}} + \mathbb{I}_{\mathcal{X}} \otimes \psi - \frac{\lambda}{e} \cdot \exp\left(\frac{-C - \phi \otimes \mathbb{I}_{\mathcal{Y}} + \mathbb{I}_{\mathcal{X}} \otimes \psi}{\lambda}\right)\right). \tag{5.28}$$

Since P and Q are density matrices and are constrained to lie within a compact set, there exists a compact region $\mathbb{S}$ that is independent of P (but may depend on λ) such that the maximum lies inside $\mathbb{S}$. $f(P)$ can therefore be written as $f(P) = \max g(P, \phi, \psi)$, where $\phi, \psi \in \mathcal{S}$, g is convex, and attains its maximum at a unique point. By Danskin's theorem [217], the result follows. □

5.8 More Details on Quantum Wasserstein GAN

5.8.1 Parameterization of the Generator

The generator G is a quantum operation that maps a fixed distribution ρ_0 to a quantum state P. Two pure distributions (states with rank 1) are mapped to each other by unitary matrices. ρ_0 is fixed to be the pure state $\bigotimes_{i=1}^{n} e_0$. If the target state is of rank r, G can be parameterized by an ensemble $\{(p_1, U_1), \ldots, (p_r, U_r)\}$ of unitary operations U_i, each of which is applied with probability p_i. Applying a unitary U_i to ρ_0 produces the state $U_i \rho_0 U_i^\dagger$. Applying G to ρ_0 thus produces the fake state $p_i U_i \rho_0 U_i^\dagger$.

Each Unitary U_i is parameterized as a quantum circuit consisting of simple parameterized 1- or 2- qubit Pauli-rotation quantum gates. An n-qubit Pauli-rotation gate $R_\sigma(\theta)$ is given by $\exp\left(\frac{i\theta\sigma}{2}\right)$ where θ is a real parameter, and σ is a tensor product of 1 or 2 Pauli matrices. Pauli-rotation gates can be efficiently implemented on quantum computers. Thus each unitary U_i can be expressed as $U_i = \prod_j e^{\frac{i\theta_{i,j}\sigma_{i,j}}{2}}$.

5.8.2 Parameterization of the Discriminator

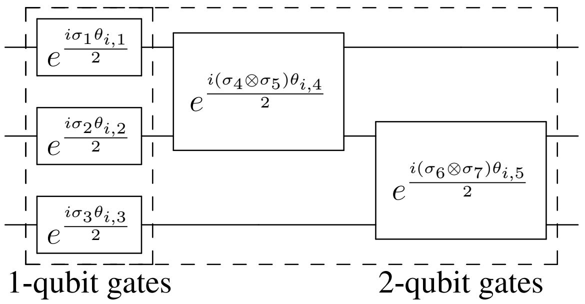
The optimization variables in the discriminator are Hermitian operators, ϕ and ψ. There are two common parameterizations for a Hermitian matrix H:

1. As $U^\dagger H_0 U$, where U is a parameterized unitary operator, and H_0 is a simpler fixed Hermitian matrix that is easy to measure. Measuring H then corresponds to applying the operator U and then measuring H_0.

2. As a linear combination $\sum_{i=0}^{\dim(H)} \alpha_i H_i$, where H_is are fixed Hermitian matrices that are easy to measure. Measuring H corresponds to measuring each H_i to obtain the expectation value m_i, and then returning $\sum_{i=0}^{\dim(H)} \alpha_i m_i$ as the expected value of measuring H.

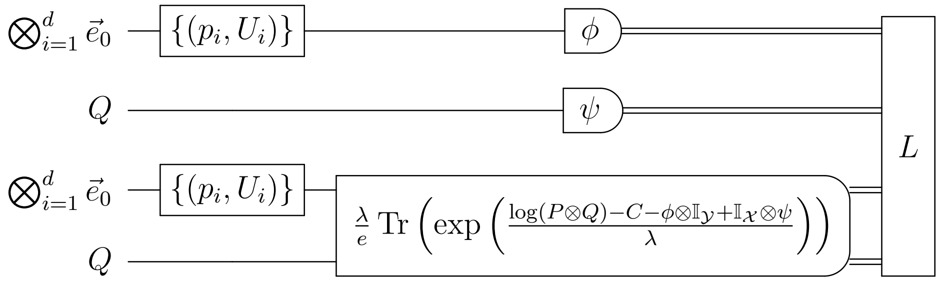
We choose the latter option because it allows ξ_R to be conveniently approximated by a linear combination of simple Hermitian matrices. Thus ϕ and ψ are represented by $\sum_k \alpha_k A_k$ and $\sum_l \beta_l B_l$ where A_k, B_l are tensor products of Pauli matrices. The α_ks, β_ls constitute the parameters of the discriminator.

The overall structure of the Quantum Wasserstein GAN is given in Figure 5.19.



$e^{\frac{i\sigma_1\theta_{i,1}}{2}}$
$e^{\frac{i\sigma_2\theta_{i,2}}{2}}$
$e^{\frac{i\sigma_3\theta_{i,3}}{2}}$
$e^{\frac{i(\sigma_4\otimes\sigma_5)\theta_{i,4}}{2}}$
$e^{\frac{i(\sigma_6\otimes\sigma_7)\theta_{i,5}}{2}}$
1-qubit gates
2-qubit gates

Figure 5.18: Example parameterization of a unitary U_i acting on 3 qubits. There are 12 possible 1-qubit gates and 48 possible 2-qubit gates.



$\bigotimes_{i=1}^{d} \vec{e}_0$
$\{(p_i, U_i)\}$
ϕ
Q
ψ
$\bigotimes_{i=1}^{d} \vec{e}_0$
$\{(p_i, U_i)\}$
$\frac{\lambda}{e}\operatorname{Tr}\left(\exp\left(\frac{\log(P\otimes Q)-C-\phi\otimes\mathbb{I}_{\mathcal{Y}}+\mathbb{I}_{\mathcal{X}}\otimes\psi}{\lambda}\right)\right)$
Q
L

Figure 5.19: The structure of the quantum WGAN. Here Q is the input state and $\vec{e}_0$ is the 0^{th} computational basis vector, meaning that the corresponding system is empty at the beginning. The final gate L combines the outputs of the measurements of ϕ, ψ, ξ_R to produce the final loss function.

5.8.3 Estimating the Loss Function

The loss function is given by $\operatorname{Tr}(Q\psi) - \operatorname{Tr}(P\phi) - \operatorname{Tr}((P \otimes Q)\xi_R) = \mathbb{E}_Q[\psi] - \mathbb{E}_P[\phi] - \mathbb{E}_{P\otimes Q}[\xi_R]$ where ξ_R is the Hermitian corresponding to the regularizer term $\frac{\lambda}{e}\exp\left(\frac{-C-\phi\otimes\mathbb{I}_{\mathcal{Y}}+\mathbb{I}_{\mathcal{X}}\otimes\psi}{\lambda}\right)$.

The fake state P is generated by applying a quantum operation G to a fixed quantum state ρ_0. The quantum operation is represented by applying a set of unitary operations $\{U_1, U_2, \ldots, U_k\}$ with corresponding probabilities $\{p_1, p_2, \ldots, p_k\}$ where k is the rank of the final state that would be generated:

$$P = \sum_{i \in [k]} p_i U_i \rho_0 U_i^\dagger. \tag{5.29}$$

Lemma 5.8.1. *Given a quantum state $\rho = \sum_{i=1}^k \alpha_i \rho_i$ and a Hermitian matrix H then $\mathbb{E}_\rho(H)$ can be estimated given only the ability to generate each ρ_i and to measure H.*

Proof. Since ρ is a quantum state $\{\alpha_1, \dots, \alpha_k\}$ must form a probability distribution. Thus,

$$\mathbb{E}_\rho[H] = \mathrm{Tr}[\rho H] = \mathrm{Tr}\left[\sum_i \alpha_i \rho_i H\right] = \sum_i \alpha_i \,\mathrm{Tr}[\rho_i H] = \sum_i \alpha_i \mathbb{E}_{\rho_i}[H] = \mathbb{E}_\alpha \mathbb{E}_{\rho_i}[H]. \tag{5.30}$$

Thus we can measure the expected value of H measured on ρ, by sampling an i with probability α_i, measuring the expected value of H on ρ_i, and then computing the expectation over i sampled from the distribution α. We can also simply measure the expectation value m_i corresponding to each ρ_i and return $\sum_i \alpha_i m_i$ as the estimate. □

The unitaries U_i are parameterized by a network of gates of the form $e^{i\theta_{i,j}\sigma_{i,j}}$ where $\sigma_{i,j}$ is a tensor product of the matrices $\sigma_x, \sigma_y, \sigma_z, I$ acting on some/all of the registers. With a sufficient number of such gates, any unitary can be represented by an appropriate choice of $\theta_{i,j}$. Since each U_i is expressed as a composition of simple parameterized gates each of them can be implemented on a quantum computer and thus each $U_i \rho_0 U_i^\dagger$ can be generated.

Note that $P = \sum_{i \in [k]} p_i U_i \rho_0 U_i^\dagger$ and $P \otimes Q = \sum_{i \in [k]} p_i (U_i \rho_0 U_i^\dagger \otimes Q)$. From Lemma 5.8.1, if ϕ and ξ_R can be measured, we can estimate the terms $\mathbb{E}_P[\phi]$ and $\mathbb{E}_{P\otimes Q}[\xi_R]$. Next we show how to measure ϕ, ψ, ξ_R where ϕ, ψ are parameterized as a linear combination of tensor products of the Pauli matrices $\sigma_X, \sigma_Y, \sigma_Z, \sigma_I$.

Lemma 5.8.2. *Any Hermitian that is expressed as a linear combination $\sum_i \alpha_i H_i$ of Hermitian matrices H_i that can be measured on a quantum computer, can also be measured on a quantum computer.*

Proof. For any fixed state ρ,

$$\mathbb{E}_\rho[H] = \mathrm{Tr}[\rho H] = \mathrm{Tr}\left[\rho \sum_i \alpha_i H_i\right] = \sum_i \alpha_i \,\mathrm{Tr}[\rho H_i] = \sum_i \alpha_i E_\rho[H_i]. \tag{5.31}$$

Thus each of the Hermitians H_i can be separately measured and the final result is the weighted average of the corresponding expectation values with coefficients α_i.

If the α_i form a probability distribution, the expectation can be estimated by sampling a batch of indices from the distribution of α_i, measuring H_i, and estimating the expectation averaging over the sampled indices. This procedure can be more efficient if some of the α_i are of very small magnitude in comparison to the others. Note that any Hermitian that can be written by as a linear combination $\sum_i \beta_i H_i$ where each H_i is easy to measure can be transformed such that the coefficients form a probability distribution as $(\sum_i |\beta_i|) \sum_i \frac{|\beta_i|}{\sum_i |\beta_i|} \mathrm{sgn}(\beta_i) H_i$. If H_i can be measured on a quantum computer, $-H_i$ can also be measured by measuring H_i and negating the result. $\square$

Tensor products of Pauli matrices can be measured on quantum computers using elementary techniques [207]. As a result, Lemma 5.8.2 implies that ϕ, ψ can be measured on a quantum computer.

Now, we prove the following lemma for expressing the regularizer term ξ_R:

Lemma 5.8.3. *The Hermitian corresponding to the regularizer term ξ_R can be approximated* *via a linear combination of Hermitians from* $\{\Sigma, \mathrm{SWAP}\cdot\Sigma\}$ *where* Σ *is a tensor product of 2-dimensional Hermitian matrices.*

Proof. Since $C = \frac{\mathbb{I}-\mathrm{SWAP}}{2}$,

$$\exp\left(\frac{-C - \phi\otimes\mathbb{I}_{\mathcal{Y}} + \mathbb{I}_{\mathcal{X}}\otimes\psi}{\lambda}\right) = \exp\left(\frac{\mathrm{SWAP} - \mathbb{I} - 2\phi\otimes\mathbb{I}_{\mathcal{Y}} + 2\mathbb{I}_{\mathcal{X}}\otimes\psi}{2\lambda}\right). \tag{5.32}$$

Observe the following two facts:

- if Σ_1 and Σ_2 are both tensor products of 2-dimensional Hermitian matrices, then $\Sigma_1\cdot\Sigma_2$ is also a tensor product of 2-dimensional Hermitian matrices;
- if Σ is a tensor product of 2-dimensional Hermitian matrices, then $\mathrm{SWAP}\cdot\Sigma\cdot\mathrm{SWAP}$ is also a tensor product of 2-dimensional Hermitian matrices.

As a result, any integral power of $\mathrm{SWAP} - \mathbb{I} - 2\phi\otimes\mathbb{I}_{\mathcal{Y}} + 2\mathbb{I}_{\mathcal{X}}\otimes\psi$ can be written as a linear combination of the matrices $\{\Sigma, \mathrm{SWAP}\cdot\Sigma\}$ where Σ is a tensor product of 2-dimensional Hermitian matrices. Thus any Taylor approximation of $\exp(\mathrm{SWAP} - \mathbb{I} - 2\phi\otimes\mathbb{I}_{\mathcal{Y}} + 2\mathbb{I}_{\mathcal{X}}\otimes\psi)$ is a linear combination of the same Hermitian matrices, each of which can be easily measured on a quantum computer. Thus the Taylor series for the exponential can be used to approximately measure the regularizer term.

A representation as a linear combination of the Hermitians $\{\Sigma, \mathrm{SWAP}\cdot\Sigma\}$, where Σ is a tensor product of Pauli matrices, can be obtained more easily for a relaxed regularizer term

$$\xi_R' = \exp\left(\frac{-C}{2\lambda}\right)\exp\left(\frac{-\phi\otimes\mathbb{I}_{\mathcal{Y}} + \mathbb{I}_{\mathcal{X}}\otimes\psi}{\lambda}\right)\exp\left(\frac{-C}{2\lambda}\right); \tag{5.33}$$

this is motivated by the Trotter formula [218] of matrix exponentiation: for any Hermitian matrices A, B such that $\|A\|, \|B\| \leq \delta \leq 1$, $\|e^{A+B} - e^A e^B\| = O(\delta^2)$ but $\|e^{A+B} - e^{A/2} e^B e^{A/2}\| = O(\delta^3)$. Using this regularizer gives us a concrete closed form for ξ'_R as a linear combination of simpler Hermitian matrices. It is less computationally intensive to compute than the original regularizer, since the only operation acting on $2n$ qubits at the same time is SWAP. This relaxation also yields good numerical results in practice.

Since $(-\phi \otimes \mathbb{I}_{\mathcal{Y}})(\mathbb{I}_{\mathcal{X}} \otimes \psi) = (\mathbb{I}_{\mathcal{X}} \otimes \psi)(-\phi \otimes \mathbb{I}_{\mathcal{Y}}) = (-\phi \otimes \psi)$, the central term in the RHS of (5.33) is an exponential of commuting terms. If A and B are commuting matrices, we have $\exp(A+B) = \exp(A)\exp(B)$, and hence

$$\xi'_R = \exp\left(\frac{-C}{2\lambda}\right) \exp\left(\frac{-\phi}{\lambda}\right) \otimes \exp\left(\frac{\psi}{\lambda}\right) \exp\left(\frac{-C}{2\lambda}\right). \tag{5.34}$$

We choose ϕ and ψ to be tensor products of terms of the form $a\sigma_x + b\sigma_y + c\sigma_z + d\mathbb{I}$. It can be verified that $\sigma_i\sigma_i = \mathbb{I}$ and $\sigma_i\sigma_j + \sigma_j\sigma_i = 2\delta_{i,j}\mathbb{I}$ and therefore $(a\sigma_x + b\sigma_y + c\sigma_z)^2 = (a^2 + b^2 + c^2)\mathbb{I}$. Given $r = \bigotimes_{i=1}^n (a_i\sigma_x + b_i\sigma_y + c_i\sigma_z + d_i\mathbb{I})$, we therefore have

$$r^2 = \bigotimes_{i=1}^{n} \left(d_i(a_i\sigma_x + b_i\sigma_y + c_i\sigma_z + d_i\mathbb{I}) + \Pi_{i=1}^n (a_i^2 + b_i^2 + c_i^2 + d_i^2)\mathbb{I}\right) \tag{5.35}$$

and by induction,

$$r^k = \bigotimes_{i=1}^{n} \left(d_i^{k-1}(a_i\sigma_x + b_i\sigma_y + c_i\sigma_z + d_i\mathbb{I}) + \left(\sum_{j=0}^{k-2} d_i^j\right)(a_i^2 + b_i^2 + c_i^2 + d_i^2)\mathbb{I} \right). \tag{5.36}$$

Eq. (5.36) can be used to expand $\exp(-\phi/\lambda) \otimes \exp(\psi/\lambda)$ using the truncated Taylor series for the exponential. Thus $\exp(-\phi/\lambda) \otimes \exp(\psi/\lambda)$ can be approximated by a linear combination of gates in Σ up to any desired accuracy.

In addition, $C = \frac{\mathbb{I}-\mathrm{SWAP}}{2}$ implies that C is a projector, i.e., $C^k = C$ for all $k \in \mathbb{N}^*$ and $C^0 = \mathbb{I}$. This can be used to express $\exp(C)$ in terms of only $\mathbb{I}$ and C:

$$\exp\left(\frac{-C}{2}\right) = \mathbb{I} + \sum_{j=1}^{\infty} \frac{C}{(-2)^j j!} = \mathbb{I} + \left[\exp\left(\frac{-1}{2}\right) - 1\right] C. \tag{5.37}$$

Using (5.36) and (5.37) we can compute an approximate expression (with any desired accuracy) for the relaxed regularizer ξ_R' as a linear combination of the Hermitian $\{\Sigma, \mathrm{SWAP} \cdot \Sigma\}$ where Σ is a tensor product of Hermitian matrices. $\square$

Finally from Lemma 5.8.1,Lemma 5.8.2,Lemma 5.8.3, each of the terms $\mathbb{E}_Q[\psi]$,$\mathbb{E}_P[\phi]$, and $\mathbb{E}_{P\otimes Q}[\xi_R]$ can be computed on a quantum computer.

5.8.4 Direct Estimation of Gradients

In this subsection, we show how the gradients with respect to the parameters of the qWGAN can be directly estimated using quantum circuits. Suppose we have the following parameterization for the optimization variables:

$$\rho_0 = \bigotimes_{i=1}^{d} \vec{e}_0 \vec{e}_0^{\dagger}, \qquad P = \sum_{i=1}^{r} p_i U_i \rho_0 U_i^{\dagger}, \qquad U_i = \prod_j e^{\frac{i\theta_{i,j} H_{i,j}}{2}} \tag{5.38}$$

and

$$\phi = \sum_k \alpha_k A_k, \qquad \psi = \sum_l \beta_l B_l, \tag{5.39}$$

where H_j, A_k, B_l are tensor products of Pauli matrices. The parameters of the generator are given by the variables $p_i, \theta_{i,j}$ and the parameters of the discriminator are given by α_k, β_l. As shown in Lemma 5.8.3, the regularizer term R can be written as $\sum_q r_q R_q$ where each R_q is either a tensor product of Pauli matrices or a product of SWAP with a tensor product of Pauli matrices. Thus the loss function is given by

$$L = \mathrm{Tr}[Q\psi] - \mathrm{Tr}[P\phi] - \mathrm{Tr}\left[(P \otimes Q)R\right], \tag{5.40}$$

and hence

$$\frac{\partial L}{\partial p_i} = -\,\mathrm{Tr}[U_i \vec{e}_0 \vec{e}_0^{\dagger} U_i^{\dagger} \phi] - \mathrm{Tr}\left[(U_i \vec{e}_0 \vec{e}_0^{\dagger} U_i^{\dagger} \otimes Q)R\right]. \tag{5.41}$$

To compute the partial derivative with respect to the parameters p_i, we create a fake state using only the unitary U_i, and compute the regularizer term as shown before:

$$\frac{\partial L}{\partial \alpha_k} = -\,\mathrm{Tr}[PA_k] - \mathrm{Tr}\left[(P \otimes Q)\frac{(A_k \otimes \mathbb{I}_{\mathcal{Y}})R}{\lambda}\right]; \tag{5.42}$$

$$\frac{\partial L}{\partial \beta_l} = \mathrm{Tr}[QB_l] - \mathrm{Tr}\left[(P \otimes Q)\frac{(\mathbb{I}_{\mathcal{X}} \otimes B_l)R}{\lambda}\right]. \tag{5.43}$$

Clearly $(A_k \otimes \mathbb{I}_{\mathcal{Y}})R$ and $(\mathbb{I}_{\mathcal{X}} \otimes B_l)R$ can be written as linear combinations of products of SWAP and tensor products of Pauli matrices, because such form exists for A_k, B_l, R. Thus these gradients can be measured as shown in Lemma 5.8.2.

Regarding the gradients with respect to $\theta_{i,j}$, we have

$$\frac{\partial L}{\partial \theta_{i,j}} = \frac{\partial \operatorname{Tr}[\phi(U_i \rho_0 U_i^\dagger)]}{\partial \theta_{i,j}} - \frac{\partial \operatorname{Tr}[\xi_R(U_i \rho_0 U_i^\dagger \otimes Q)]}{\partial \theta_{i,j}}. \tag{5.44}$$

The terms $\frac{\partial \operatorname{Tr}[\phi(U_i \rho_0 U_i^\dagger)]}{\partial \theta_{i,j}}, \frac{\partial \operatorname{Tr}[\xi_R(U_i \rho_0 U_i^\dagger \otimes Q)]}{\partial \theta_{i,j}}$ can be evaluated by modifying the quantum circuits for U_i using with an ancillary control register, using previously known techniques [214, Section III. B]. This allows us to evaluate the partial derivatives of the loss function w.r.t. the $\theta_{i,j}$ parameters.

5.8.5 Computational Cost of Evaluating the Loss Function

Consider a quantum WGAN designed to learn an n-qubit target state with rank r; the generator hence consists of r unitary matrices. Suppose that each unitary U_i is a composition of at most N fixed unitary gates. Furthermore, assume that ϕ and ψ are parameterized as a linear combination of at most M tensor products of Pauli matrices. The size of the network (the number of parameters) is thus $O(rNM)$.

The loss function consists of 3 terms:

- The expectation value of ϕ measured on the state P.
- The expectation value of ψ measured on the state Q.
- The expectation value of ξ_R measured on the state $P \otimes Q$.

The complexity of a quantum operation is quantified by the number of elementary gates required to be performed on a quantum computer. We show that a single measurement of ϕ on $U_i \rho_0 U_i^\dagger$, ψ on Q, and ξ_R on $U_i \rho_0 U_i^\dagger \otimes Q$ can be carried out using $\operatorname{poly}\left(n, k, N, M, \log\left(\frac{1}{\epsilon}\right)\right)$ gates.

The expectation values can then be estimated by computing the empirical expectation on a batch of measurements. These expectation values are combined as shown earlier in Appendix 5.8.3 to obtain the expected values measured on P and $P \otimes Q$.

First, ξ_R can be approximated to precision ϵ via truncation of a Taylor series consisting of $\log\left(\frac{1}{\epsilon}\right)$ terms. Thus ξ_R is approximated by a linear combination of $\mathrm{poly}\left(M, \frac{1}{\epsilon}\right)$ fixed Hermitian matrices of the form Σ or $\mathrm{SWAP} \cdot \Sigma$ where each Σ is a tensor product of 2-dimensional Hermitian matrices.

Second, by the Solovay-Kitaev theorem [219], any n-qubit unitary operator can be implemented to precision ϵ using $\mathrm{poly}\left(\log\left(n, \frac{1}{\epsilon}\right)\right)$ gates. Similarly, any fixed n-qubit Hermitian matrix can be measured using a circuit with $\mathrm{poly}\left(n, \log\left(\frac{1}{\epsilon}\right)\right)$ gates. Consequently:

- ψ can be measured on Q using M measurements of fixed tensor products of Pauli matrices, therefore using $\mathrm{poly}\left(n, M, \log\left(\frac{1}{\epsilon}\right)\right)$ gates.
- ϕ can be measured on $U_i \rho_0 U_i^\dagger$ for any i using M measurements of fixed tensor products of Pauli matrices, therefore using $\mathrm{poly}\left(n, M, \log\left(\frac{1}{\epsilon}\right)\right)$ gates.
- ξ_R can be measured on $U_i \rho_0 U_i^\dagger \otimes Q$ for any i using $\mathrm{poly}\left(M, \frac{1}{\epsilon}\right)$ measurements of fixed tensor products of Pauli matrices, using $\mathrm{poly}\left(n, M, \log\left(\frac{1}{\epsilon}\right)\right)$ gates.
- Each unitary U_i can be applied by a composition of N fixed unitaries, therefore using $\mathrm{poly}\left(n, N, \log\left(\frac{1}{\epsilon}\right)\right)$ gates.

From Appendix 5.8.4, it can be seen that the partial derivatives with respect to the parameters p, α, β are each computed by the same procedure as the loss function with some of the variables restricted. Furthermore, the partial derivatives with respect to $\theta_{i,j}$ can be evaluated using the circuit for U_i with an ancillary register and a constant number of extra gates [214]. Each partial derivative therefore has the same complexity as the loss function. Since there are $O(rNM)$ parameters, the total gradient can be evaluated with a multiplicative overhead of $O(rNM)$ compared to evaluating the loss function.

5.9 More Details on Experimental Results

Pure states We used the quantum WGAN to learn pure states consisting of $1, 2, 4$, and 8 qubits. In this case, the generator is fixed to be a single unitary. The parameters to be chosen in the training are λ (the weight of the regularizer) and η_g, η_d (the learning rates for the discriminator and generator parameters, respectively). The training parameters for our experiments for learning pure states are listed in Table 5.1.

Parameters	1 qubit	2 qubits	4 qubits	8 qubits
λ	2	2	10	10
$\eta = \eta_g = \eta_d$	10^{-1}	10^{-1}	10^{-1}	10^{-2}

Table 5.1: Parameters for learning pure states.

For 1,2, and 4 qubits, in addition to Figure 5.11, we also plot the average loss function for a number of runs with random initializations in Figure 5.24 which shows the numerical stability of our quantum WGAN.

Mixed states We also demonstrate the learning of mixed quantum states of rank 2 with $1, 2$, and 3 qubits in Figure 5.15. The generator now consists of 2 unitary operators, and 2 real probability parameters p_1, p_2 which are normalized to form a probability distribution using a softmax layer.

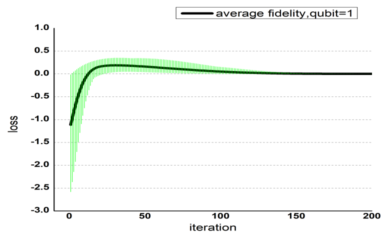


Figure 5.20: 1 qubit

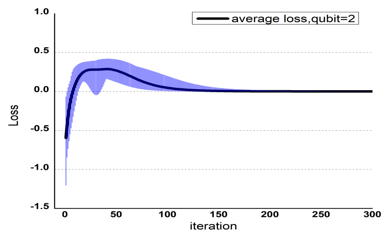


Figure 5.21: 2 qubits

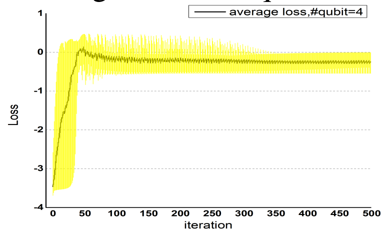


Figure 5.22: 4 qubits

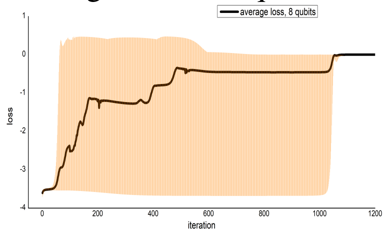


Figure 5.23: 8 qubits

Figure 5.24: Average performance of learning pure states (1, 2, 4 qubits) where the black line is the average loss over multi-runs with random initializations and the shaded area refers to the range of the loss.

The learning rate for the probability parameters is denoted by η_p. The training parameters are listed in Table 5.2.

Parameters	1 qubits	2 qubits	3 qubits
λ	10	10	10
η_d, η_g, η_p	$(10^{-1}, 10^{-1}, 10^{-1})$	$(10^{-1}, 10^{-1}, 10^{-1})$	$(10^{-1}, 10^{-1}, 10^{-1})$

Table 5.2: Parameters for learning mixed states.

Learning pure states with noise In a recent experiment result [2], a quantum-classical hybrid training algorithm using the KL divergence between classical measurement outcomes as the loss function on the canonical Bars-and-Stripes data set was performed on an ion-trap quantum computer. Specifically, they use the generator in Figure 5.25. Even though the goal of [2] is to generate a classical distribution, we still deem it as a good example of practically implementable quantum generator to testify our quantum WGAN.

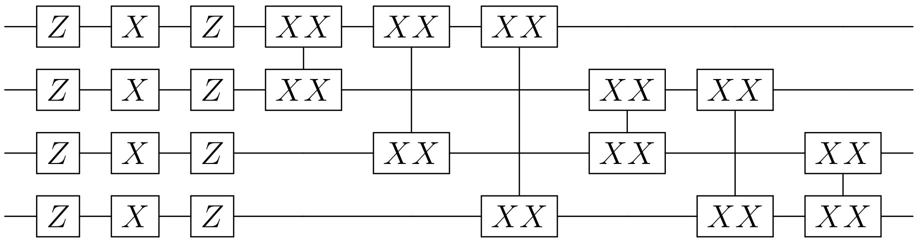


Figure 5.25: The generator circuit used in Ref. [2] where Z stands for the $e^{i\theta\sigma_z}$ gate, X stands for the $e^{i\theta\sigma_x}$ gate, and XX stands for the $e^{i\theta\sigma_x\otimes\sigma_x}$ gate.

We use the same training parameters as in the noiseless case (Table 5.1). Furthermore, we add the sampling noise (modeled as a Gaussian distribution with standard deviation σ) which is a reasonable approximation of the noise for the ion-trap machine [206]. Our results show that the quantum WGAN can still learn a 4-qubit mixed state in the presence of this kind of noise. As is to be expected, noise with higher degrees (i.e., higher σ) increases the number of epochs required before the state is learned successfully. The corresponding results are plotted in Figure 5.16.

Our finding also demonstrates the different outcomes between choosing different metrics as the loss function. In particular, some of the training results reported in [2] demonstrate a KL distance $< 10^{-4}$ but the actual quantum fidelity is only about 0.16. On the other side, our quantum WGAN is guaranteed to achieve close-to-1 fidelity all the time.

Application: Approximating Quantum Circuits The quantum Wasserstein GAN can be used to approximate the behavior of quantum circuits with many gates using fewer quantum gates. Consider a quantum circuit U_0 over n qubits. It is well known [207] that there exists an isomorphism between n qubit quantum circuits U and quantum states Ψ_U such that

$$\Psi_U = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} (U \otimes \mathbb{I})(\vec{e}_i \otimes \vec{e}_i) = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} (U(\vec{e}_i) \otimes \vec{e}_i). \tag{5.45}$$

The quantum Wasserstein GAN can be used to learn a smaller quantum circuit U_1 such that Ψ_{U_1} is close to Ψ_{U_0}. This can be done by setting the real state to Ψ_{U_0}, and using the GAN to learn to generate it using a circuit of the form $(U_1 \otimes \mathbb{I})$ applied to $\frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} (\vec{e}_i \otimes \vec{e}_i)$. The fidelity between Ψ_{U_1} and Ψ_{U_0} is given by the average output fidelity for uniformly chosen inputs to U_1 and U_0.

We apply these techniques to the quantum circuit that simulates the evolution of a quantum system in the 1-dimensional nearest-neighbor Heisenberg model with a random magnetic field in the z-direction (considered in [208]). The time evolution for time t is described by the unitary operator $e^{i\hat{H}t}$ with the Hamiltonian $\hat{H}$ given by

$$\hat{H} = \sum_{j=1}^{n} \left(\sigma_x^{(j)} \sigma_x^{(j+1)} + \sigma_y^{(j)} \sigma_y^{(j+1)} + \sigma_z^{(j)} \sigma_z^{(j+1)} + h^{(j)} \sigma_z^{(j)} \right) \tag{5.46}$$

where $\sigma_i^{(j)}$ denotes the Pauli gate σ_i applied at the j^{th} qubit, and the $h^{(j)} \in [-h, h]$ are uniformly chosen at random.

We study the specific case with $t = n = 3$ and $h = 1$, with a fixed target error of $\epsilon = 10^{-3}$ in the spectral norm. Quantum circuits for simulating Hamiltonians that are represented as the sum of local parts, $e^{iHt} = e^{it \sum_{i=1}^{L} \alpha_j H_j}$, are obtained using k^{th} order Suzuki product formulas S_{2k} defined by

$$S_2(\lambda) = \prod_{j=1}^{L} \exp(\alpha_j H_j \lambda / 2) \prod_{j=L}^{1} \exp(\alpha_j H_j \lambda / 2) \tag{5.47}$$

$$S_{2k}(\lambda) = \left[S_{2k-2}(p_k \lambda) \right]^2 S_{2k-2}\left((1 - 4p_k) \lambda \right)^2 \left[S_{2k-2}(p_k \lambda) \right]^2 \tag{5.48}$$

where $p_k = 1 / \left(4 - 4^{1/(2k-1)} \right)$ for $k \geq 1$.

We then approximate e^{iHt} by $\left[S_{2k}\left(\frac{it}{r}\right)\right]^r$. Obtaining error ϵ in the spectral norm requires $r = \frac{(Lt)^{1+1/2k}}{\epsilon^{1/2k}}$. From (5.47), each evaluation of S_{2k} requires $(2L)5^{k-1}$ gates of the form $e^{iH_j\theta}$ where θ is a real parameter. In the case of the Hamiltonian (5.46), it is the sum of 12 terms each of which is the product of up to 2 Pauli matrices. Thus the k^{th} order formula S_{2k} yields a circuit for simulating (5.46) requiring $(24)5^{k-1}\frac{(36)^{1+1/2k}}{0.001^{1/2k}}$ gates of the form $e^{i\theta\sigma}$ where σ is a product of up to 2 Pauli matrices. These are the gates used in the parameterization of our quantum Wasserstein GAN, and can be implemented easily on ion trap quantum computers. The smallest circuit is obtained using S_2 and requires ~ 11900 gates.

Using the quantum Wasserstein GAN for 6-qubit pure states, we discovered a circuit for the above task with 52 gates, an average output fidelity of 0.9999, and a worst case error 0.15. The worst case input is not realistic, and thus the 52 gate circuit provides a very reasonable approximation in practice.

Chapter 6: Separations in Expressivity between Quantum Neural Networks and Feed-Forward ReLU Networks

6.1 Introduction

The recent establishment of quantum supremacy ([220, 221]) has spurred a new era of interest in the practical applications of quantum computers. Variational quantum algorithms based on *Quantum Neural Networks* [12, 29] have emerged as a candidate for demonstrating quantum advantage for *machine learning*, on modern Noisy Intermediate-Scale Quantum (NISQ [135]) quantum computers.

Quantum Neural Networks (QNNs) consist of a series of *parameterized* quantum operations, applied on a quantum state that encodes the input. Any classical input encoded by an n-qubit state corresponds to a 2^n dimensional vector in the Hilbert space, which has the potential to represent functions on high-dimensional features, and at the same time hard for classical computers to simulate [222]. Indeed, it has been conjectured that the power of QNNs could come from their *expressive* power to kernel methods in high dimensions [223, 224, 225].

A learning model can be advantageous in three main ways: the model can be more *expressive* than others of equivalent complexity, it can be easier to *optimize*, or can *generalize* better given fewer training examples. Despite the common belief in the quantum machine learning community, there is not yet any rigorous evidence that either shows a concrete advantage of QNNs over classical learning models [139] or separates the expressive power between quantum and classical. Any progress on such a fundamental question about QNNs could provide principled guideline on the seek of useful applications of QNNs.

The major conjectured advantage of QNNs lies in their ability in efficient computing of certain functions, in particular those functions involving high-dimensional spaces (e.g., quantities from quantum physics). As a result, it is conjectured that QNNs would be more *expressive* than classical models. Establishing such separations is however subtle: classical neural networks (even those with just one hidden layer) have been proven to be universal approximators [226] and the same is conjectured to be true for sufficiently general QNNs [227]. A valid comparison must therefore consider models of the same complexity. Furthermore, the scale of empirical comparison between quantum and classical models is also limited by the relatively small size of existing quantum machines as well as the exponential cost associated with simulating quantum models via classical means. As a result, theoretical analysis is likely the only approach to a scalable comparison between quantum and classical models. On the other side, such a theoretical separation between even classical models of different depths has proven notoriously difficult [228]. Establishing such a separation between quantum and classical would be conceivably more difficult.

Contributions. We provide a rigorous comparison between the expressivity of QNNs to feedforward ReLU networks (ReNNs), which have been a very successful model for classical problems in machine learning. Specifically, we compare the minimum complexity of QNNs that approximate various classes of functions to that of an ReNN for an equally good approximation. Our findings lead to a *2-way separations* between quantum and classical , i.e. each model may be superior for certain function classes. Such separations already indicate that simply replacing ReNNs with QNNs for achieving quantum advantages is unlikely successful, as there are specific cases that favor the use of ReNNs. The particular cases where advantages are obtained for QNNs can be used to identify potential quantum advantages in classical learning problems such as classification.

We have two major findings: Firstly, we surprisingly discover that there are classes of *highly oscillatory* functions that ReNNs approximate exponentially more efficiently than QNNs. It has been conjectured that the inherent periodicity of functions represented by QNNs results in an expressive advantage for periodic, oscillatory functions [227]. Our theoretical results indicate that this is false when restricted to a finite domain and may in fact be exponentially disadvantageous. Thus the choice of learning tasks for which to deploy QNNs may be subtler than previously imagined. This result relies on the representation of QNNs as truncated Fourier Series [227, 229].

Conversely, we show that univariate *sinusoidal* functions are more efficiently expressed by QNNs than ReNNs with only one hidden layer. This result leverages the fact that ReNNs with width$-w$ and depth$-t$ approximate t-wise compositions of functions with w-linear regions [230, 231]. We show that there exist *complexity theoretic barriers* to proving an exponential advantage (in terms of the input dimension) for QNNs against ReNNs with depth≥ 4. Theoretically, the situation for ReNNs with lower depth is less clear and is left as an open question. However, a naive construction of an ReNN with depth≤ 4 to approximate a general QNN with complexity linear in the input dimension (based on ReNNs for multivariate polynomials [232]) requires exponential width. We conjecture that such an exponential dependence is *necessary*, and perform an empirical evaluation to verify this hypothesis. To this end we introduce a notion of *empirical* *separation*, where instead of analytically determining the complexities necessary for approximation, we compare the minimum complexities of QNNs and ReNNs that can be *trained* to represent a function accurately. Due to the non-convexity of the optimization problems, and the infeasibility of training (or testing) a function over a continuous data set, the empirical expressivity of a QNN does not necessarily capture its theoretical expressive power. Nevertheless, since models need to be trained to be deployed, the empirical expressivity is a good proxy for a models effective expressivity in practice. Our empirical results are obtained for functions represented by *random* QNNs. This indicates that QNNs may have exponential advantages over ReNNs for the majority of functions represented by quantum circuits. Specifically for cases where the task is to learn an unknown quantum circuit or process [31, 233], the experiments are in line with the hypothesis of a practical advantage for QNNs. We therefore anticipate that the advantage of QNNs will be primarily for functions originating in quantum phenomena, while our first result proves that they are not universally advantageous for general functions.

ReNN depth **Input dimension**	2 $d = 1$	$\omega(1)$ $d = 1$	2 $d > 1$	$\omega(1)$ $d > 1$
Classical Advantage	*Analytical* $O(2^{\text{poly}(\log(1/\epsilon))})$ (Corollary 6.3.1)	*Analytical* $O(2^{\text{poly}(\log(1/\epsilon))})$ (Corollary 6.3.1)	*Analytical* $O(2^{\text{poly}(\log(1/\epsilon))})$ (Corollary 6.3.1)	*Analytical* $O(2^{\text{poly}(\log(1/\epsilon),d)})$ (Corollary 6.3.1)
Quantum Advantage	*Analytical* $O(2^{\text{poly}(\log(1/\epsilon))})$ (Corollary 6.3.2)	*Analytical* $O(\text{poly}(\log(1/\epsilon))$ (Corollary 6.3.2)	*Empirical* $O(2^{\text{poly}(\log(1/\epsilon),d)})$ (Figures 6.4,6.5)	*Empirical* $O(2^{\text{poly}(d)})$ (Figure 6.6)

Table 6.1: Expressive power separations between QNNs and depth t ReNNs on d-dimension inputs. For each type of separation we indicate the ratio of minimum complexity required to ϵ-approximate a certain class of function

Related Works. The memory capacity (related to the VC-dimension) of QNNs has been found to have limited advantages over classical networks [234]. As we show in this paper, QNNs and classical NNs each have expressive advantages over the other, so separations are likely to be found for specific classes of functions rather than in the total capacity. The functional form of QNNs has been investigated in [227, 229] and shown to be expressed by truncated Fourier series. These works do not establish explicit separations from classical networks based on this observation. Finally, [235] investigates the ”effective dimension” (based on the Fisher information) and provides evidence that QNNs may have a larger effective capacity than classical networks when taking into account trainability and generalization.

Explicit separations between classes of neural networks have been a hot topic of study in classical machine learning. There have been demonstrations of exponential separations (in d) between ReLU networks with depth-2 and depth-3 [236, 237]. The advantages of depth have also been investigated in the dimension-free univariate setting, with exponential separations shown in [230, 238]. A width-based phase transition in expressivity was shown in [239]. Related work [231] studies the functional form of ReLU networks based on their width and depth, and characterizes affine regions in the landscape.

6.2 Background

6.2.1 Expressive Power of Parameterized Models.

A *parameterized* function model (including classical and quantum neural networks) is a mapping from a set of real parameters to a function from the input domain to the output range. Specifically, a model with p parameters, domain D, and range R can be described by a function $\mathcal{F}\colon \Theta \to \mathcal{D}^{\mathcal{R}}$ where $\Theta \subseteq \mathbb{R}^p$ is the parameter space. A parameterized model is often identified by the set of functions that can be represented by using some setting of the parameters. A model $\mathcal{M}$ approximates a function f if some setting of its parameters results in a function f_ϵ such that the L_∞-distance $\sup_{x\in\mathcal{D}} |f(x) - f_\epsilon(x)|$ is less than some predetermined error parameter ϵ, referred to as an ϵ-approximation. The ϵ-expressive power of a class of a models $\mathcal{M}$ is the set of functions that it can ϵ-approximate and is denoted $\mathrm{expr}_\epsilon(\mathcal{M})$.

6.2.2 ReLU Neural Networks (ReNNs).

Classical neural networks have been an extremely succesful parameterized function model in machine learning. Typical feedforward networks are parameterized by a sequence of matrices $\{W_i\}_{i=1}^t$ and bias vectors $\{b_i\}_{i=1}^t$, where $W_i \in \mathbb{R}^{w_i \times w_{i-1}}, b_i \in \mathbb{R}^{w_i}$, with w_0, w_t set to the dimension of the domain and co-domain respectively (for a real valued network $w_t = 1$). Each W_i, b_i defines an affine transformation $\mathbf{W}_i \colon \mathbb{R}^{w_{i-1}} \to \mathbb{R}^{w_i}$ such that $\mathbf{W}_i(\mathbf{x}) = W_i\mathbf{x} + b_i$. For input vector $\mathbf{x}$, the output y of the neural network is

$$y = \mathbf{W}_t\sigma(\mathbf{W}_{t-1}\sigma(\ldots\sigma(\mathbf{W}_1 x)\ldots)), \tag{6.1}$$

where $\sigma(\cdot)$ denotes an element-wise activation on the output of each layer. Different choices of activation functions have been studied in the literature including identity maps ($\sigma(x) = x$) [240], quadratic maps ($\sigma(x) = x^2$) [241] and sigmoid functions $\sigma(x) = (1 + e^{-x})^{-1}$. In this paper we focus on real valued ReLU networks (ReNNs) that use the ReLU activation function $\sigma(x) = \max(0, x)$.

Definition 6.2.1. *A feedforward ReLU network (ReNN) is a neural network that uses the activation function $\sigma(x) = \max(0, x)$ on each layer. The* width*(w) of an ReNN is the maximum of the layer* *sizes ie.* $\max w_1, \ldots, w_t$ *and the depth is the number of layers* t*. We use ReNN*$_{w,t,d}$ *to denote the family of ReLU neural networks with width* w *and depth* t*, acting on an input space with dimension* d*.*

6.2.3 Quantum Neural Networks (QNNs)

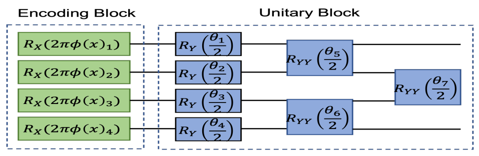

Encoding Block
Unitary Block
$R_X(2\pi\phi(x)_1)$
$R_X(2\pi\phi(x)_2)$
$R_X(2\pi\phi(x)_3)$
$R_X(2\pi\phi(x)_4)$
$R_Y\left(\frac{\theta_1}{2}\right)$
$R_Y\left(\frac{\theta_2}{2}\right)$
$R_Y\left(\frac{\theta_3}{2}\right)$
$R_Y\left(\frac{\theta_4}{2}\right)$
$R_{YY}\left(\frac{\theta_5}{2}\right)$
$R_{YY}\left(\frac{\theta_6}{2}\right)$
$R_{YY}\left(\frac{\theta_7}{2}\right)$

Figure 6.1: An example layer of a 1-parallel QNN. Such blocks are repeated r times in parallel, and L times in succession for a general QNN.

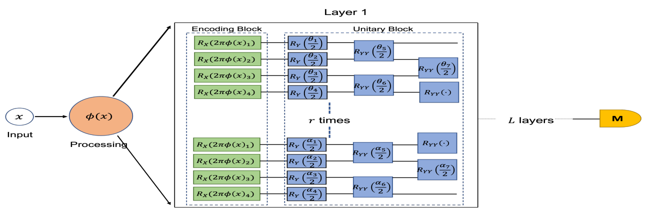


Figure 6.2: An r-parallel QNN with L layers, with a pre-processing function ϕ and final measurement M

Figure 6.3: Architecture of a Quantum Neural Network

Quantum Neural Networks are parameterized models where the input and the parameterized computations are replaced by networks of unitary operations (quantum gates) operating on a set of n registers initialized to $\mathbf{e}_{0^n}$. They share the layered structure of a classical neural network where operations are sequentially applied to the output of the previous layer, with the following differences:

Input. The inputs to ReNNs are simply feature vectors. For QNNs however, the input must be encoded into quantum states that are then fed to quantum circuits. There are a variety of encoding schemes used for quantum neural networks [2, 13, 233] that have been used in quantum neural networks. A common scheme that has been applied in many experiments based on QNNs is the rotation-encoding scheme. Here a vector $\mathbf{x} = (x_i, \ldots, x_d)$ is represented by Pauli-rotation gates with the individual components of the input encoded in their parameters. Specifically, r of the n registers of the circuit are chosen to correspond to each component of the d-dimensional input (rd total). An r-parallel *encoding block* then corresponds to applying the gate $R_\sigma(2\pi x_j)$ (where $\sigma \in \{X, Y, Z\}$) on every register that corresponds to component j (a common choice is $R_\sigma(2\pi x_j)$ on any register q such that $q \cong j(\mod d)$).

Classical Parameterization of Quantum Operations. The parameterized transformations applied in ReNNs are affine transforms $\{\mathbf{W_i}\}_{i=1}^t$ acting on vectors. In a QNN the parameterized transformations must be unitary operators acting on quantum states. For generality, the gates chosen must allow for the approximation of any unitary operation using arbitrarily many gates. The parameterized single qubit gates chosen are of the form $R_\sigma(\theta/2)$ (where $\sigma \in \{X, Y, Z\}$) where θ is a real parameter. The two qubit *entangling* gates are chosen from $e^{i\theta\sigma_a \otimes \sigma_b}$ (where $a, b \in \{X, Y, Z\}$) with real parameter θ, or the unparameterized CNOT. These parameterized gates are structured into *unitary blocks*, with single qubit gates applied to every register and double qubit gates to every adjacent pair of registers.

Output. In contrast to ReNNs, where we can simply read out the output after the appropriate transformations, in QNNs the output must be obtained by measuring the state obtained by applying the encoding and parameterized operations. This measurement is represented by a Hermitian matrix M and is applied on some subset of the registers. There is no direct analog of the non-linear activation functions used in classical NNs.

Preprocessing Input. The input to a QNN can be pre-processed by applying a bijective processing function $\phi\colon \mathbb{R}^d \to \mathbb{R}^d$ whose output is subsequently fed to the encoding gates.

There have been several different architectures proposed for QNNs. The version we define below is general enough to accommodate most proposals in the literature, while leading to an at most polynomial increase in complexity. A Quantum Neural Network interleaves blocks of encoding blocks and unitary blocks as follows (Figure 6.3):

Definition 6.2.2 (Quantum Neural Network (QNNs))**.** *Given an input $x \in [0,1]^d$ a d-dimensional, L-layer, r-parallel Quantum Neural Network is defined by a sequence of L layers of quantum gates acting on $n = rd + O(1)$ registers. Each layer consists of (1) (Optionally) An* encoding block *that encodes each component x_j of x on each register m such that $m \mod d \equiv (j-1)$ (2) A* unitary block *of parameterized gates. The output is measured as the expectation value of an observable M applied to the set of registers (can be taken to be applied to the first register wlog). The* input redundancy *R of a QNN is the total number of encoding gates corresponding to each component of the input ($R \leq rL$), and the total number of gates is called the* size *of the network. The corresponding parameterized model is denoted QNN$_{r,L,d}$.*
The model with a pre-processing function ϕ is denoted PQNN$_{r,L,d,\phi}$ and represents QNN$_{r,L,d}$ with input $\phi(x)$.

Truncated Fourier Representation of QNNs. Recent works have shown that the functions represented by QNNs where the input encoding is via gates of the form $e^{ix_jH_j}$ (for Hermitian matrices H_j) have representations as truncated Fourier series [227]. Furthermore, the maximal frequencies of this encoding are closely connected to the input redundancy R of the QNN. We state this observation in a proposition tailored to our model of QNNs with proof deferred to the supplementary material.

Proposition 6.2.1. *Given a L-layer, r-parallel PQNN Q with measurement M and preprocessing function ϕ, the corresponding function $F_Q : \mathbb{R}^d \to \mathbb{R}$ can be expressed as a truncated Fourier Series $\sum_\omega a_\omega e^{i2\pi\omega\cdot\phi(x)}$, where $\omega \in \mathbb{R}^d$ such that $\omega_j \in \{-2R, -2R+1, \ldots, 2R-1, 2R\}, \forall j \in [1,d]$ where $R = Lr$ is the redundancy of the circuit Q. Equivalently, $F_Q = \sum_\omega a_\omega \cos(2\pi\omega \cdot$*

$\phi(x)) + b_\omega \sin(2\pi\omega \cdot \phi(x))$.

6.3 Analytical Separations in Expressive Power

Cost of parameterized models The computational complexity associated with a parameterized model comes from the cost to *evaluate* the function for a particular input and parameters, and the cost to *optimize* the parameters for a specific learning task. Due to the heuristic nature of non-convex optimization, the number of parameters can be used as a proxy for the optimization cost. ReNNs with width-w, depth-t have w^2t parameters, and are evaluated using t w-dimensional matrix-vector products resulting in a cost of $\Theta(w^2t)$. The evaluation cost and number of parameters of a QNN are both equal to the number of quantum gates $\Theta(rdL)$. We define cost as measure of model complexity.

$$\mathrm{cost}(\text{ReNN}_{w,t,d}) = w^2t, \quad \mathrm{cost}(\text{QNN}_{r,L,d}) = rdL \tag{6.2}$$

$\text{PQNN}_{r,L,d,\phi}$ additionally incurs the classical cost of evaluating ϕ. We therefore restrict ϕ to functions that can be computed by an ReNN with cost $O(rLd)$ so as to not simply replace the quantum computation by a more expensive classical processing step. $\text{PQNN}_{r,L,d,\phi}$ with ϕ satisfying this condition is simply denoted $\text{PQNN}_{r,L,d}$

Defining expressive advantage. A class of models C_1 has an analytical advantage over C_2 if there provably exists a class of functions F such that $\mathcal{M}_1 \in C_1$ ϵ-approximates every function F, while any equally good approximation $\mathcal{M}_2 \in C_2$ satisfies $\mathrm{cost}(\mathcal{M}_1) = o(\mathrm{cost}(\mathcal{M}_2))$ ie. $\mathrm{expr}_\epsilon(\mathcal{M}_1) \subseteq \mathrm{expr}_\epsilon(\mathcal{M}_2)$ only if $\mathrm{cost}(\mathcal{M}_1) = o(\mathrm{cost}(\mathcal{M}_2))$. When $C_1 = \text{ReNN}, C_2 =$

QNN (and vice versa), we denote the corresponding results as *classical* and *quantum* advantage respectively.

We compare several sub-classes of ReNNs to QNNs and summarize our results in Table 6.1. The worst case error of approximation can be constantly increased by simply multiplying all functions by a constant. To eliminate such trivial scaling considerations, we only consider functions with bounded range ($[0, 1]$ w.l.o.g.). The functions approximated by QNNs are inherently periodic in contrast to ReNNs, and so these classes trivially cannot approximate each other over $\mathbb{R}^d$. We therefore consider approximation only on the bounded domain $[0, 1]^d$. We also consider preprocessed networks from $\mathrm{PQNN}_{r,L,d}$ where the pre-processing does not change the asymptotics of the model complexity.

Classical Advantage Functions encoded by QNNs as truncated Fourier series with frequencies bounded by the input redundancy R. This representation places two main restrictions on the properties of the function represented by a QNN: (1) A Truncated Fourier Series approximation must retain all frequencies with large coefficients (Lemma 6.3.1). (2) The number of *oscillations* (or times crossing a fixed value) of a truncated Fourier series is bounded by the largest frequency (Lemma 6.3.2).

Lemma 6.3.1 (Proof in appendix Lemma 6.5.1)**.** *Let* $f\colon [0, 1] \to [0, 1]$ *be a continuous function represented by a Fourier series given as* $\sum_{k=-\infty}^{\infty} c(k)e^{i2\pi kx}$. *If* $\hat{f}\colon [0, 1] \to [0, 1]$ *is a continuous function that* ϵ*-approximates* f *and has a truncated Fourier series* $\sum_{k=-K}^{K} \hat{c}(k)e^{i2\pi kx}$, *then* $c(k) \leq \epsilon$ *for all* $|k| > K$.

Lemma 6.3.2 ([242])**.** *Let* $f\colon [0, 1] \to [0, 1]$ *be a continuous function represented by a truncated Fourier Series* $f(x) = \sum_{k=-K}^{K} c(k)e^{i2\pi kx}$. *The equation* $f(x) = c$ *for any* $c \in [0, 1]$ *has a* *maximum of $2K$ roots in $[0,1]$ unless $f(x)$ is identically c for $x \in [0,1]$.*

To obtain a classical advantage we show that ReNNs can efficiently approximate several functions with slowly decaying Fourier coefficients. Furthermore, based on the observations of [230] the number of oscillations in the output of an ReNN can increase exponentially with its depth.

Our results employ the following result based on the work of [232] on the approximation of univariate piecewise-polynomial functions by ReNNs.

Lemma 6.3.3 (Lemma 6.5.2)**.** *Let $\epsilon > 0$. Let $f : [0,1] \to [0,1]$ be a continuous function such that $[0,1]$ can be divided into p intervals on each of which f is some polynomial with degree at most D. Then f can be ϵ-approximated on $[0,1]$ by a ReNN with width and depth bounded by $O(pD\log(D/\epsilon)^2)$.*

We define the following class of functions to demonstrate our results on classical advantage.

Definition 6.3.1. *Let $\mathcal{F}_{p,G,D}$ be the class of G-Lipschitz continuous functions $f\colon [0,1] \to [0,1]$ such that $[0,1]$ can be divided into p intervals on each of which f is given by a polynomial with degree at most D, and $f(0) = f(1) = 0$. $\mathcal{F}_{p,G,D,k}$ is the class of functions $f\colon [0,1]^d \to [0,1]$ such that $f = f_1 \circ f_2 \circ \cdots \circ f_{k-1} \circ f_k$ where each $f_j \in \mathcal{F}_{p,G,d}$. $\mathcal{F}_{p,G,D,d,k}$ is the class of functions $f : [0,1]^d \to [0,1]$ such that $f(x = (x_1, \ldots, x_d)) = \frac{1}{d}\sum_{j=1}^{d} f_j(x_j)$ where each $f_j \in \mathcal{F}_{p,G,D,k}$.*

We next show that $\mathcal{F}_{p,L,d}$ is highly oscillatory

Lemma 6.3.4 (Lemma 6.5.3)**.** *Let $f \in \mathcal{F}_{p,L,D}$ such that $f(x) = 1$ for some $x \in [0,1]$. Then the value of f^k oscillates between 0 and 1 at least 2^k times on $[0,1]$, and $f(x) = 1/2$ at $\geq 2^k$ points.*

Our main result is expressed in the following theorem and corollary.

Theorem 6.3.1. *Define the class of functions $\mathcal{F}_{d,k} = \mathcal{F}_{p,G,D,d,k}$ with $p, G, D = O(1)$ and $\mathcal{F}_{p,G,D,d,k}$ as in Definition 6.3.1. Every function in $\mathcal{F}_{d,k}$ can be ϵ-approximated by a ReNN with width $O(dk^2 \log(d/\epsilon)^2)$ and depth $O(k^2 \log(d/\epsilon)^2)$. However, QNN$_{r,L,d}$ can ϵ-approximate every function in $\mathcal{F}_{d,k}$ only if $rL = \Omega(\max(2^{k-2}, 1/\epsilon^{1/2}))$.*

Proof. We first upper bound the complexity of approximation by ReNNs. By Lemma 6.3.3, any function in $\mathcal{F}_{1,1}$ can be δ-approximated using at most $O(\log(1/\delta)^2)$ width and depth. To compute functions $f \in \mathcal{F}_{d,1}$ to error δ, we must make d parallel computations of functions in $\mathcal{F}_{1,1}$ to error δ/d, each using width and depth bounded by $O(\log(d/\delta)^2)$. The final computation of f uses a total width of $O(d \log(d/\delta)^2)$ and a depth of $O(\log(d/\delta)^2)$. Finally the composition $f_1 \circ f_2$ can be computed by appending the ReNN computing f_1 to that computing f_2. Functions in $\mathcal{F}_{d,k}$ are computed using k layers of ReNNs corresponding to functions in $\mathcal{F}_{d,1}$. If each of these layers has error δ, the error at the next layer will be at most $(G+1)\delta$ (δ from the new layer, $G\delta$ due to the input error). The final error in $f = f_1 \circ \cdots \circ f_k$ is guaranteed to be $\leq \epsilon$, if the error for each f_j is $\leq \epsilon/(G+1)^k$. Since each $f_j \in \mathcal{F}_{1,k}$, any function $f \in \mathcal{F}_{d,k}$ can be ϵ-approximated by a function in ReNN$_{O(dk^2 \log(d/\epsilon)^2), O(k^2 \log(d/\epsilon)^2), d}$.

Next, we lower bound the complexity of approximation by QNNs. Consider first the case with $d = 1$. There is an infinitude of functions $f \in \mathcal{F}_{1,1}$ such that $f(0) = f(1) = 0$ and $f(p) = 1$ for some $p \in [0, 1]$ (eg. $f(x) = 1 - 4(x - 1/2)^2$). Consider any such function f and let f^k be approximated by some $Q \in$ QNN$_{r,L,d}$. By Proposition 6.2.1, the corresponding function f_Q is a truncated Fourier series with integer frequencies ranging from $-2R$ to $2R$, where the input redundancy $R \leq rL$. Therefore by Lemma 6.3.2, $f_Q = 1/2$ at $\leq 2R$ points in $[0, 1]$. On the other hand by Lemma 6.3.4, $f^k = 1/2$ at $\geq 2^k$ points. If $2(2R) \leq 2^k$, there must exist an interval where $f_Q - 1/2$ has constant sign, but f^k ranges from 0 to 1, therefore $|f_Q - f^k| > 1/2$ at some point in the interval. Therefore, for any $\epsilon \geq 1/2$, a QNN Q that ϵ-approximates f^k must satisfy $rL \geq R \geq 2^{k-2}$.

Now consider the piecewise affine triangle function $g(x) = \min(2x, 2-2x)$. By definition $g \in \mathcal{F}_{1,1}$. Let f_Q be the function corresponding to any QNN Q with redundancy R that ϵ-approximates $g(x)$ (on $\mathbb{R}$, f_Q will approximate a 1-periodic function that matches f on $[0,1]$). By Proposition 6.2.1 f_Q is a truncated Fourier series with frequencies in $[-2R, 2R]$. Let $c(n)$ be the Fourier co-efficients of f. By Lemma 6.3.1, $c(n) \leq \epsilon$ for all $n \geq 2R$. It can be verified by direct computation that $c(n) = \Theta(1/n^2)$. Therefore, $rL \geq R = \Omega(1/\epsilon^{1/2})$.

Now let $d > 1$. Let $f \in \mathcal{F}_{1,k}$ be such that any QNN approximating Q must have $R \geq \Omega(\max(2^{k-2}, 1/\epsilon^{1/2}))$. Consider $f_d : [0,1]^d \to [0,1]$, $f_d(x) = (1/d)\sum_{j=1}^{d} f(x_j)$. Let R_j be the redundancy of component x_j. If $R_j = o(\max(2^{k-2}, 1/\epsilon^{1/2}))$, then setting $x_i = 0, \forall i \neq j$ yields $Q \in \text{QNN}_{r,L,1}$ with $rL = o(\max(2^{k-2}, 1/\epsilon^{1/2}))$ that ϵ-approximates f_j, contradicting the already established case for $d = 1$. Thus the total redundancy $R = \sum_{j=1}^{d} R_j = \Omega(\max(d2^{k-2}, d/\epsilon^{1/2}))$.

□

Corollary 6.3.1 (Corollary 6.5.1)**.** *1.* $\text{expr}_\epsilon(\mathit{ReNN}_{2,2,1}) \subseteq \text{expr}_\epsilon(\mathit{PQNN}_{r,L,1}) \implies Lr = \Omega(1/\epsilon^{1/2})$

2. $\text{expr}_\epsilon(\mathit{ReNN}_{O(\log(1/\epsilon)),O(k\log(1/\epsilon)),1}) \subseteq \text{expr}_\epsilon(\mathit{PQNN}_{r,L,1}) \implies Lr = \Omega(\max(2^{k-2}, 1/\epsilon^{1/2}))$

3. $\text{expr}_\epsilon(\mathit{ReNN}_{2d,2,d}) \not\subseteq \text{expr}_\epsilon(\mathit{PQNN}_{r,L,d}) \implies Lr = \Omega(d/\epsilon^{1/2})$

4. $\text{expr}_\epsilon(\mathit{ReNN}_{O(d^3\log(d/\epsilon)^2),O(d^2\log(d/\epsilon)^2),d}) \not\subseteq \text{expr}_\epsilon(\mathit{PQNN}_{r,L,d}) \implies Lr = \Omega(d2^{d-2})$ *and* $Lr = \Omega(d/\epsilon^{1/2})$.

Corollary 6.3.1 shows that the cost of QNNs that can ϵ-approximate every function that is exactly represented by a ReNN of depth $2d$ must be exponential in d and polynomial in $1/\epsilon^{1/2}$ indicating exponential classical advantage.

Quantum Advantage for univariate functions Our second main result shows a quantum advantage for approximating univariate sinusoidal functions. We employ the following results (folklore, but shown in supplementary material for completeness).

Lemma 6.3.5 (Lemma 6.5.4)**.** *Any $f\colon [0,1] \to \mathbb{R}$ in ReNN$_{w,t,1}$ has $O((2w+2)^{t-1})$ affine regions.*

Lemma 6.3.6 (Lemma 6.5.5)**.** *Any quadratic function of $(\cos(2\pi kx), \sin(2\pi kx))$ can be exactly represented by a function in QNN$_{1,k+3,1}$.*

Theorem 6.3.2 is the main result, demonstrating that simple sinusoidal functions require $\text{poly}(1/\epsilon)$-piecewise affine functions for an ϵ-approximation.

Theorem 6.3.2. *For any integer k, the class of quadratic functions of $(\cos(2\pi kx), \sin(2\pi kx))$ cannot be ϵ-approximated by any class of piecewise affine functions with $o(k/\epsilon^{1/3})$ affine regions.*

Proof. Consider $f_k(x) = (1 + \cos(4\pi kx))/2 = \cos^2(2\pi kx)$. Let $f_k(x)$ be ϵ-approximated by some piecewise-affine function. We now determine the maximum possible size of each affine region of the approximating function. The affine regions will be largest at the points $x = n/4k$ for odd n where the second derivatives are 0, thereby minimizing the deviation from linearity. Consider any such point $x = a$, and let it be approximated in some interval $[a - \delta_1, a + \delta_2]$ with $\delta_1, \delta_2 > 0$. Since the number of affine regions required is independent of a function shift or scaling, we can equivalently consider the problem of approximating the function $\sin(4\pi kx)$ in the region $[-\delta_1, \delta_2]$. Let $\delta = \min(\delta_1, \delta_2)$ and the affine ϵ-approximation in the region be

$gx + h$. We have $-\epsilon \leq h \leq \epsilon$. For sufficiently small ϵ, it must hold that $4\pi k\delta < 1$. For $x < 1, x^3/12 < |\sin(x) - x| < x^3/6$. For ϵ-approximation to hold $g(\delta/2) + h \geq sin(4\pi k\delta) - \epsilon$ and $g\delta + h \leq \sin(4\pi k\delta) + \epsilon$. Via simple algebra, we obtain $(4\pi k\delta)^3 \leq 24(4\epsilon)$ Therefore, $\delta = O(\epsilon^{1/3}/k)$ and the total number of affine windows covering $[0,1]$ must be $\Omega(k/\epsilon^{1/3})$ □

Theorem 6.3.2 and Lemma 6.3.5 directly yield the claimed advantage of QNNs over ReNNs for univariate functions.

Corollary 6.3.2. *For constant k, $\operatorname{expr}_\epsilon(QNN_{1,k+3,1}) \not\subseteq \operatorname{expr}_\epsilon(ReNN_{w,t,1})$ unless $(2w+2)^{t-1} = \Omega(k/\epsilon^{1/3})$ (eg. w constant and $t = \Omega(\log(1/\epsilon))$, or t constant and $w = \Omega(1/\epsilon^{1/3})$).*

Barriers to Analytical Super-polynomial Quantum Advantage in d. Following the work of [228], we observe that there exist complexity theoretic barriers to proving a superpolynomial quantum advantage for L_p-approximation over ReNNs with depth ≥ 4 for any $p < \infty$.

Theorem 6.3.3 ([228])**.** *Let $f : [0,1]^d \to [0,1]$ be a $\operatorname{poly}(d)$-Lipschitz continuous that is computable using $\operatorname{poly}(d)$ space. If f cannot be L_p-approximated to error $1/\operatorname{poly}(d)$ (for $p < \infty$) by a ReLU network of size $\operatorname{poly}(d)$ and constant depth $k \geq 4$, then $\mathrm{PSPACE} \not\subseteq \mathrm{TC}^0_{k-2}$.*

The function f_Q corresponding to an efficient QNN satisfies the conditions of Theorem 6.3.3

Lemma 6.3.7 (Lemma 6.5.6)**.** *Let Q be a QNN with input redundancy $R \leq \operatorname{cost}(Q) = \operatorname{poly}(d)$. The corresponding function f_Q is $\operatorname{poly}(d)$-Lipschitz continuous and computable in $\operatorname{poly}(d)$ space.*

Therefore, if an ReNN with depth≥ 4 requires superpolynomial width to approximate f_Q, then $\mathrm{PSPACE} \not\subseteq \mathrm{TC}^0_2$. Since $\mathrm{PSPACE} \subseteq \mathrm{EXP}$, proving a super-polynomial quantum advantage also shows $\mathrm{EXP} \not\subseteq \mathrm{TC}_2^0$ thereby proving a decades-long open conjecture in complexity theory [243, 244].

6.4 Empirical Separations in Expressive Power

The *empirical* expressive power of a model is the minimum complexity of the model that can be *trained* to ϵ-approximate a class of functions. Due to the heuristic nature of non-convex optimization, and the limitations imposed by a finite data-set, the expressivity cannot be exactly determined in this manner. However, several well known gradient based optimization methods have proven quite successful for ReNNs and QNNs and thus the empirical expressivity serves as a good proxy in the absence of theoretical proof, or as a tool to verify theoretical results. Learning models need to be trained to be deployed, so empirical expressivity may be the more important measure in practical settings.

6.4.1 Empirical quantum advantage for multi-variate functions

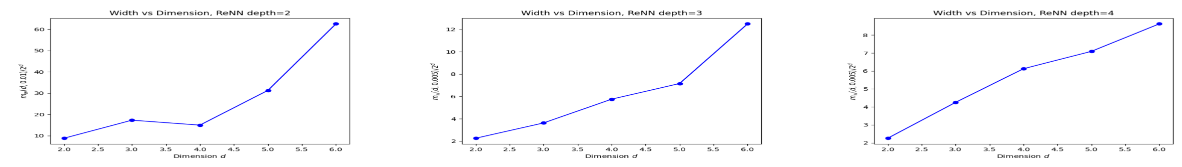


Figure 6.4: $m_w(d, \epsilon)/2^d$ vs d for ReNNs with depth 2 (ϵ = 0.01) and $3, 4$ (ϵ = 0.005)

Based on a naive construction similar to those used classically for approximating multivariate polynomials by ReNNs, a general QNN Q with $\mathrm{poly}(d)$ redundancy can be ϵ approximated by a ReNN with width $O(2^{\mathrm{poly}(d)}/\epsilon)$ and constant depth=4, or with total size bounded by $O(d2^{\mathrm{poly}(d)} \log(d/\epsilon)^4)$ respectively (Lemma 6.5.7). We hypothesize that these exponential dependences are

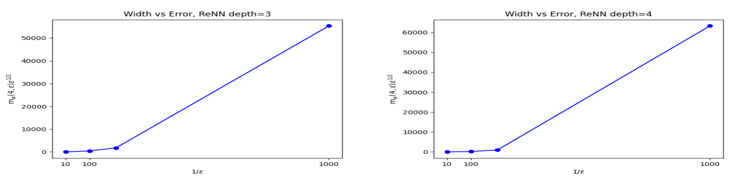


Figure 6.5: $m_w(4, \epsilon)\epsilon^{1/2}$ vs $1/\epsilon$ for ReNNs with depth $3, 4$, $d = 4$

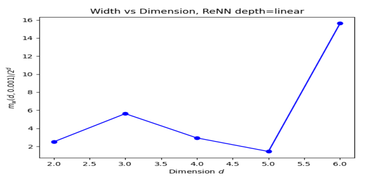


Figure 6.6: $m_w(d, 0.001)/2^d$ for ReNNs with depth-d

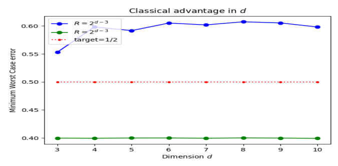


Figure 6.7: Worst case error for different redundancies

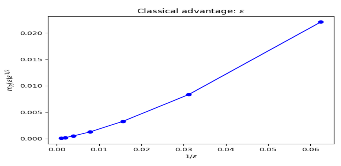


Figure 6.8: $m_R(\epsilon)\epsilon^{1/2}$ vs $1/\epsilon$

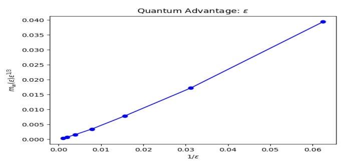


Figure 6.9: $m_w(\epsilon)\epsilon^{1/3}$ vs $1/\epsilon$

necessary: ie. QNNs have an exponential advantage in $d, \log(1/\epsilon)$ over ReNNs with depth ≤ 4, and an exponential advantage in d over variable depth ReNNs. To verify our hypothesis we perform experiments with the following setup: a target function with d-dimensional input is fixed, and ReNNs with a particular chosen depth are trained to ϵ-approximate the function. The width of the ReNN is varied and binary search is used to determine the minimum width at which the ReNN achieves the desired approximation. By varying d, ϵ we *extrapolate* the dependence of the minimum width on these quantities. If this dependence is exponential, we can establish an exponential empirical separation.

Experimental Details We fix a QNN architecture consisting of two 1-parallel R_X encoding layers, and two unitary blocks. Each parameterized unitary block consists of two sub-blocks, each with a layer of R_Z gates followed by a layer of R_Y gates followed by a CNOT entangling layer. Corresponding to any setting of d, we select a target QNN Q_d by randomly fixing the gates of this architecture, with corresponding function f_{Q_d}. Any instance Q_d of this architecture has $\mathrm{cost}(Q_d) = O(d)$. Given d, ϵ we search for lower bounds on the minimum width $m_w(d, \epsilon)$

of an ReNN that ϵ-approximates f_{Q_d} at a fixed depth t. A subset $P \in [0,1]^d$ of random points (usually 4^d) are selected. We fix an interval $[0, B]$ for our search. For any width $w \in [0, B]$ an ReNN is trained to minimize the mean-squared error on the set P for 10000-15000 epochs or until convergence. The training is *successful* if the worst-case loss is ever $\leq \epsilon$, and corresponds to finding a good approximation. We perform binary search in $[0, B]$ to find the minimum width $m_w(d, \epsilon)$ for successful training. If none of the widths allow for successful training, we report $m_w(d, \epsilon) = B$. Each ReNN is trained using the Adam optimizer [245] with hyper-parameters manually tuned for effective training, and 3 different restarts with random initializations. (In the graphs below, we plot $m_w(d, \epsilon)/g(d, \epsilon)$ where $g(d, \epsilon)$ is a function representing the separation we are trying to demonstrate. If the separation is valid, the obtained ratio should be greater than 1 and monotonically increasing.) Due to the limitations of simulating quantum circuits on classical hardware, we are only able to vary d up to 6. While our experiments show a clear trend, improvements in simulation capacity may eventually allow us to improve our results.

Implementation Details The experiments are implemented using the Pytorch library [246] and offered under the MIT license. The experiments were run on an AWS cluster with an 8-core Intel Xeon E5-2686 v4 (Broadwell) processor, and a single NVIDIA Tesla V100 GPU.

Exponential quantum advantage in d over ReNNs with depth ≤ 4. The minimum cost for approximation at dimension d is empirically greater than 2^d : Fixing $\epsilon = 0.01$ for ReNNs of depth 2 and $\epsilon = 0.005$ for depth $3, 4$ we observe (Figure 6.4) that $m_w(d, \epsilon)/2^d \geq 1$ and increases in d.

Exponential quantum advantage in $\log(1/\epsilon)$ over ReNNs with depth ≤ 4. The minimum approximation cost is empirically greater than $1/\epsilon^{1/2} = O(2^{\text{poly}(\log(1/\epsilon))})$: Fixing $d = 4$, we observe (Figure 6.5) for ReNNs of depth $3, 4$ that $m_w(4, \epsilon)\epsilon^{1/2}$ is an increasing function of $1/\epsilon$.

Exponential quantum advantage in d over variable depth ReNNs. For dimension d we train ReNNs of depth d to ϵ-approximate f_{Q_d} on a random set of $2^{d\sqrt{d}}$ points [1] for fixed $\epsilon = 0.001$. $m_w(d, 0.001)/2^d$ is observed (Figure 6.6) to be greater than 1 indicating a large empirical advantage for small d, but is not clearly monotonically increasing as in the constant depth cases. There is hence weaker evidence for an overall exponential separation.

6.4.2 Empirical Verification of Analytical Separations

Classical Advantage (Corollary 6.3.1). We train truncated Fourier Series (corresponding to QNNs with redundancy R) to ϵ-approximate g^d for $g(x) = 1+2\max(x-1/2, 0)-2\max(1/2-x, 0)$ ($g^d \in \text{ReNN}_{2,2d,1}$). The models are trained to convergence, from 3 random initializations.

1. With $R = 2^{d-3}$ the minimum worst-case error observed on the training set $\{i/2^k \mid i \in \mathbb{N}, 0 \leq i \leq 2^k\}$ is strictly greater than $1/2$ for $3 \leq d \leq 10$. (Figure 6.7). Models with $R = 2^{d-2}$ are able to attain a worst-case error less than $1/2$.
2. Fixing $d = 1$, we search for the minimum redundancy $m_R(\epsilon)$ required for ϵ-approximation of g on integer multiples of ϵ, as in Section 6.4.1. We observe (Figure 6.8) that $m_R(\epsilon)\epsilon^{1/2}$ is greater than 1 and increasing in $1/\epsilon$, indicating $R = \Omega(1/\epsilon^{1/2})$

Quantum Advantage (Corollary 6.3.2). We train ReNNs with depth=2 to ϵ-approximate $\cos(4\pi x)$ on integer multiples of ϵ, and determine a lower bound on the minimum required width $m_w(\epsilon)$ (as in Section 6.4.1). The minimum width is empirically $O(1/\epsilon^{1/3})$ since we observe (Figure 6.9) that $m_w(\epsilon)\epsilon^{1/3}$ is greater than 1 and increasing in $1/\epsilon$.

[1] A network with constant width and linear depth can approximate functions with k^d affine regions thereby exactly fitting k^d points for any constant d, therefore $2^{\omega(d)}$ training points must be used to observe an advantage.

6.5 Deferred technical details

Proposition 6.5.1. *Given a L-layer, r-parallel PQNN Q with final measurement M and pre-processing function ϕ, the corresponding function F_Q has the following property:*

$F_Q(x)$ can be expressed as a truncated Fourier Series $\sum_\omega a_\omega e^{i2\pi\omega\cdot\phi(x)}$, where $\omega \in \mathbb{R}^d$ such that $\omega_j \in \{-2R, -2R+1, \ldots, 2R-1, 2R\}, \forall j \in [1, d]$ where $R = Lr$ is the redundancy of the circuit Q. Equivalently, $F_Q = \sum_\omega a_\omega \cos(2\pi\omega \cdot \phi(x)) + b_\omega \sin(2\pi\omega \cdot \phi(x))$.

Proof. Let U be the parameterized unitary operator represented by the quantum circuit. The output of the circuit is then given by $F_Q = \mathbf{e}_{0^d}^\dagger U^\dagger M U \mathbf{e}_{0^d}$, where $\mathbf{e}_\mathbf{b} = \otimes_{i=1}^d \mathbf{e}_{b_i}$ for all $b \in \{0,1\}^d$. Notice that in each layer l of the circuit, we apply at most one encoding unitary of the form

$$E_l = \bigotimes_{j=r}^{d} \left(\bigotimes_{k=1}^{d} \exp(i2\pi\sigma_k \phi(x_k)) \right) \tag{6.3}$$

where σ_k is a Pauli matrix. Since each Pauli matrix has eigenvalues $\{1, -1\}$, we have

$$\begin{aligned} E_l = \bigotimes_{j=r}^{d} \Bigg(\bigotimes_{k=1}^{d} \big(& \exp(i2\pi\phi(x_k))\mathbf{u}_k\mathbf{u}_k^\dagger + \\ & \exp(-i2\pi\phi(x_k))\mathbf{v}_k\mathbf{v}_k^\dagger \Bigg) \end{aligned} \tag{6.4}$$

where $\mathbf{u}_k, \mathbf{v}_k$ are the eigenvectors of σ_k. Denoting the remaining gates in each layer (that are independent of the input) as U_l, we have $U = \prod_{j=i}^{l} E_j U_j$. Making the corresponding substitutions, we have that $F_Q = \sum_\omega a_\omega e^{i2\pi\omega\cdot\phi(x)}$ where each component of ω is the sum of $2R$ terms in $[1, -1]$. A complex exponential $\exp(iy)$ can be written as the linear combination of $\sin(|y|), \cos(|y|)$. Since F_Q is real by construction, $F_Q = \sum_\omega a_\omega \cos(2\pi\omega \cdot \phi(x)) + b_\omega \sin(2\pi\omega \cdot$

$\phi(x))$ with real coefficients a_ω, b_ω □

Lemma 6.5.1. *Let $f\colon [0,1] \to [0,1]$ be a continuous function represented by a Fourier series given as $\sum_{k=-\infty}^{\infty} c(k)e^{i2\pi kx}$. If $\hat{f}\colon [0,1] \to [0,1]$ is a continuous function that ϵ-approximates f and has a truncated Fourier series representation $\sum_{k=-K}^{K} \hat{c}(k)e^{i2\pi kx}$, then $c(k) \leq$ for all $|k| > K$.*

Proof. We can extend the Fourier series of $\hat{f}$ by

$$\hat{f}(x) = \sum_{k=-K}^{\infty} \hat{c}(k)e^{i2\pi kx}, \qquad \text{s.t. } \hat{c}(k) = 0, \forall |k| > K \tag{6.5}$$

Consider the integral $\mathcal{I}(k) = \int_0^1 \hat{f}(k)e^{-i2\pi kx}\,dx$ that is well defined because $\hat{f}$ is continuous by definition. Then, from (6.5) $\mathcal{I}(k) = \hat{c}(k)$. Therefore,

$$|c(k) - \hat{c}(k)| = |\int_0^1 f(x)e^{-i2\pi kx}\,dx - \mathcal{I}(k)| \tag{6.6}$$

$$= |\int_0^1 (f(x) - \hat{f}(x))e^{-i2\pi kx}\,dx| \tag{6.7}$$

$$\leq \int_0^1 |(f(x) - \hat{f}(x))e^{-i2\pi kx}|\,dx \tag{6.8}$$

$$\leq \int_0^1 \left(|f(x) - \hat{f}(x)|\right) |e^{-i2\pi kx}|\,dx \tag{6.9}$$

$$\leq \epsilon \tag{6.10}$$

From (6.5), $\hat{c}(k) = 0$ for $k > K$, and therefore $c(k) \leq \epsilon$. □

Lemma 6.5.2 (based on [232])**.** *Let $\epsilon > 0$. Let $f : [0,1] \to [0,1]$ be a continuous function such that $[0,1]$ can be divided into p intervals on each of which, f is given by a polynomial with degree at most D. Then f can be ϵ-approximated on $[0,1]$ by a ReNN with width and depth bounded by*

$O(pD\log(D/\epsilon)^2)$.

Proof. We proceed by induction on p. For $p = 1$, f is simply a polynomial of degree D. By [232][Lemma 3.3], the function x^y can be ϵ-approximated by an ReNN with width and depth bounded by $O(\log(y/\epsilon)^2)$. A polynomial of degree D is the linear combination of D such terms with $y \leq D$, and can be approximated by an ReNN with width and depth bounded by $O(D\log(D/\epsilon)^2)$.

Now suppose the lemma is true for all $p-1$. Let a be the leftmost point of separation between two intervals on which f is polynomial. Specifically,

$$f(x) = \begin{cases} f_1(x), & x < a \\ f_2(x), & x \geq a \end{cases} \tag{6.11}$$

where f_1 is a polynomial and f_2 is $p-1$ piecewise polynomial. By the continuity of f, $f_1(a) = f_2(a) = f(a)$. Consider the function,

$$\begin{aligned} \tilde{f}(x) = \tilde{f}_1(a-&\max(a-x,0))+ \\ &\tilde{f}_2(\max(x-a,0)+a) - f(a) \end{aligned} \tag{6.12}$$

where $\tilde{f}_1, \tilde{f}_2$ are $\epsilon/2$-approximations of f_1, f_2 respectively. By inspection,

$$\tilde{f}(x) \begin{cases} \tilde{f}_1(x) + \tilde{f}_2(a) - f(a), & x < a, \\ \tilde{f}_1(a) + \tilde{f}_2(x) - f(a), & x \geq a \end{cases} \tag{6.13}$$

$\tilde{f}$ is therefore an ϵ-approximation of f, that can be approximated by a ReNN with width/depth bounded by $O(pD\log(D/\epsilon)^2)$. □

Lemma 6.5.3. *Let $f \in \mathcal{F}_{p,L,D}$ such that $f(0) = f(1) = 0$ and $f(x) = 1$ for some $x \in [0,1]$. Then the value of f^k oscillates between 0 to 1 at least 2^k times on the domain $[0,1]$, and $f^k(x) = 1/2$ at $\geq 2^k$ points.*

Proof. We proceed by induction. Clearly the lemma is true for $k = 1$. Now let f^{k-1} oscillate from 0 to 1 at least 2^{k-1} times. On each interval $[a,b]$ where f^{k-1} ranges from 0 to 1 or vice versa, $f^k = f \circ f^{k-1}$ oscillates from 0 to 1 and back again, by the defining property of f. Furthermore by the continuity of f, f^k attains the value $1/2$ at at least 2 points in this interval. Since, there are 2^{k-1} intervals $[a,b]$, the result follows by induction. □

Corollary 6.5.1. *1.* $\mathrm{expr}_\epsilon(\mathit{ReNN}_{2,2,1}) \subseteq \mathrm{expr}_\epsilon(\mathit{PQNN}_{r,L,1}) \implies Lr = \Omega(1/\epsilon^{1/2})$

2. $\mathrm{expr}_\epsilon(\mathit{ReNN}_{O(\log(1/\epsilon)),O(k\log(1/\epsilon)),1}) \subseteq \mathrm{expr}_\epsilon(\mathit{PQNN}_{r,L,1}) \implies Lr = \Omega(\max(2^{k-2}, 1/\epsilon^{1/2}))$

3. $\mathrm{expr}_\epsilon(\mathit{ReNN}_{2d,2,d}) \not\subseteq \mathrm{expr}_\epsilon(\mathit{PQNN}_{r,L,d}) \implies Lr = \Omega(d/\epsilon^{1/2})$

4. $\mathrm{expr}_\epsilon(\mathit{ReNN}_{O(d^3\log(d/\epsilon)^2),O(d^2\log(d/\epsilon)^2),d}) \not\subseteq \mathrm{expr}_\epsilon(\mathit{PQNN}_{r,L,d}) \implies Lr = \Omega(d2^{d-2})$ *and* $Lr = \Omega(d/\epsilon^{1/2})$.

Proof. Observe that if a function f can be approximated only by QNNs Q with $\mathrm{cost}(Q) = \Omega(q(d,\epsilon))$, the function $f \circ \phi$ can be approximated only by PQNNs Q with $\mathrm{cost}(Q) = \Omega(q(d,\epsilon))$ if Q uses ϕ as the pre-processing function. Let the classical cost of approximating f be $\Omega(c(d,\epsilon))$, by our condition on pre-processing functions the classical cost of approximating $f \circ \phi$ is $\Omega(c(d,\epsilon) + q(d,\epsilon))$. Therefore, any exponential classical advantages over QNNs also hold over PQNNs.

1. The triangle function $g(x) = \min(2x, 2-2x) = 1 + 2\max(x - 1/2, 0) - 2\max(1/2 - x, 0)$ and can thus be exactly represented by a ReNN with width-2 and depth-2. By the proof of

Theorem 6.3.1, g can only be expressed by QNNs with redundancy $\Omega(1/\epsilon^{1/2})$.

2. Follows by setting $d = 1$ in Theorem 6.3.1.

3. Follows by the proof on Theorem 6.3.1, specifically for $(1/d)\sum_{j=1}^{d} g(x_j)$ where g is the triangle function.

4. Follows by setting $k = d$ in Theorem 6.3.1.

□

Lemma 6.5.4. *A function $f\colon [0,1] \to \mathbb{R}$ in ReNN$_{w,t,1}$ has $O((2w+2)^{t-1})$ affine regions.*

Proof. We proceed by induction. A univariate ReNN with depth-2 can be written as $\gamma + \sum_{i=1}^{w} \max(x\alpha_i - \beta_i)$, which is clearly affine everywhere except the w points α_i/β_i resulting in $w+1$ affine regions. We now observe the following: the linear combination of two piecewise affine functions with p_1 and p_2 affine regions has $\leq p_1 + p_2$ affine regions. Furthermore, the function $\max(0, a + bf(x))$ where $f(x)$ is piecewise affine with p affine regions, has $2p$ affine regions (possible points of non-differentiability arise where $f(x)$ is non-differentiable, or when $f(x) = -a/b$ which occurs at $\leq p$ points).

Consider a ReNN with depth-t and width-w. The output is the sum of w functions, each of which is of the form $\max(0, f(x))$ where $f(x)$ is the output of a ReNN with depth-$t-1$ and width w. By the inductive hypothesis, each such $f(x)$ has at most $O(2^{t-2}(w+1)^{t-2})$ affine regions. Therefore each $\max(0, f(x))$ has at most $O(2^{t-1}(w+1)^{t-1})$ affine regions and the whole network has $O(w2^{t-1}(w+1)^{t-2}) \leq O(2^{t-1}(w+1)^{t-1})$ affine regions. □

Lemma 6.5.5. *Any quadratic function of $(\cos(2\pi kx), \sin(2\pi kx))$ is exactly represented by a function in QNN$_{1,k+3,1}$.*

Proof. It can be verified that $R_X(\theta)\mathbf{e}_0 = \cos(\theta)\mathbf{e}_0 + \sin(\theta)\mathbf{e}_1$. Consider a univariate QNN with input x, where the first k gates are encoding gates given by $R_X(2\pi x)$. By definition, $R_X(\theta)^k = R_X(k\theta)$. Therefore, the state of the system after applying these encoding gates is $\psi = \cos(2\pi k\theta)\mathbf{e}_0 + \sin(2\pi k\theta)\mathbf{e}_1$. Let the following three gates be $R_X(\alpha_1), R_Y(\alpha_2), R_X(\alpha_3)$ with parameters $\alpha_1, \alpha_2, \alpha_3$. An appropriate setting of the parameters allows this sequence of gates to represent an arbitrary unitary operator U. Given some fixed final measurement M, $U^\dagger M U$ can represent any Hermitian operator with the same eigenvalues as M. The circuit can therefore effectively compute the expectation value of an arbitrary measurement operator on ψ. Thus a 1-parallel circuit with $k+3$ layers can compute a function

$$\begin{aligned} f_Q(x) = &\begin{pmatrix} \cos(2\pi kx) & -i\sin(2\pi kx) \end{pmatrix} \begin{pmatrix} a & c+id \\ c-id & b \end{pmatrix} \\ &\begin{pmatrix} \cos(2\pi kx) \\ i\sin(2\pi kx) \end{pmatrix} \end{aligned} \tag{6.14}$$

$$\begin{aligned} &= a\cos^2(2\pi kx) + 2d\sin(2\pi kx)\cos(2\pi kx) \\ &\quad + b\sin^2(2\pi kx) \end{aligned} \tag{6.15}$$

Since a, b, d are arbitrary real numbers, any quadratic function can be realized. □

Lemma 6.5.6. *Let Q be a QNN with input redundancy $R \le \mathrm{cost}(Q) = \mathrm{poly}(d)$. The corresponding function f_Q is $\mathrm{poly}(d)$-Lipschitz continuous and computable in $\mathrm{poly}(d)$ space.*

Proof. 1. From definition $\partial_j R_X(x_j) = 2\pi i \sigma_X R_X(x_j)$. Consider the function $f_Q(x = (x_1, \dots, x_d))$ corresponding to a quantum neural network Q and consider its partial derivative with respect to each component x_j. As a function of x_j, f_Q is expressed as

$$f_Q(x_j) = \left(\left(\prod_{i=1}^{d} U_i R_X(x_j)\right) \mathbf{e}_0\right)^{\dagger} M \left(\left(\prod_{i=1}^{d} U_i R_X(x_j)\right) \mathbf{e}_0\right) \tag{6.16}$$

Applying the chain rule $\partial_j f_Q$ is the sum of $2R$ terms of the form

$$2\pi i \left(\left(\prod_{i=j+1}^{d} U_i R_X(x_j)\right) U_j \sigma_X R_X(x_j) \left(\prod_{i=1}^{j-1} U_i R_X(x_j)\right) \mathbf{e}_0\right)^{\dagger} M \left(\left(\prod_{i=1}^{d} U_i R_X(x_j)\right) \mathbf{e}_0\right) \tag{6.17}$$

Since $f_Q \in [0,1]$, $\partial_j f_Q = O(R)$. The function is thereby $R\sqrt{d} = \mathrm{poly}(d)$-Lipschitz.

2. The output of a QNN Q with $\mathrm{cost}(Q) = \mathrm{poly}(d)$ is the result of simulating a quantum circuit with $\mathrm{poly}(d)$ gates. It is well known that such a simulation can be carried out using polynomial space [247] to any $\mathrm{poly}(d)$ number of bits.

□

Lemma 6.5.7. *Any QNN with* $\mathrm{poly}(d)$ *redundancy can be* ϵ*-approximated by a ReNN with width* $O(2^{\mathrm{poly}(d)}/\epsilon)$ *and constant depth-4, or width* $O(2^{\mathrm{poly}(d)})$ *and depth* $O(d\log(d/\epsilon)^2)$.

Proof. By Proposition 6.2.1, the function f_Q corresponding to any QNN Q is represented as the linear combination of at most $2(4R+1)^d$ terms of the form $\cos(2\pi\omega \cdot x), \sin(2\pi\omega \cdot x)$ with $\omega \in \{-2R, -2R+1, \ldots, 2R-1, 2R\}$. Clearly each term $\omega \cdot x$ can be exactly computed using a ReNN of depth=2, width=d with $R = \mathrm{poly}(d)$ bounded weights. All the terms $\omega \cdot x$ can be exactly computed by a hidden layer with $(4R+1)^d$ nodes.

The linear combination corresponding to f_Q can be computed to accuracy ϵ if every $\sin(\cdot)$, $\cos(\cdot)$ term is computed to accuracy $\epsilon/2(4R+1)^d$. By [236][Lemma 6] each term can be computed using a depth=2 ReNN with width $2(4R+1)^d/\epsilon$. f_Q can therefore be computed using an ReNN of depth=4 and width $O(4(4R+1)^{2d}/\epsilon) = O(2^{\text{poly}(d)}/\epsilon)$.

Since $x \in [0,1]^d$, $|2\pi\omega \cdot x| \leq 4\pi Rd$. By [248][Theorem 4.1], $\cos(2\pi\omega \cdot x), \sin(2\pi\omega \cdot x)$ can be computed to error δ using constant width and depth $O(\log(1/\delta) + \log(4\pi Rd))$. Thus f_Q can be computed using depth $O(d\log(d/\epsilon)^2)$ and width $O(2^{poly(d)})$. $\square$

Chapter 7: A convergence theory for over-parameterized variational quantum algorithms

7.1 Introduction

Quantum Variational Methods (QVMs) (see for eg. [12, 28, 29] have become a leading candidate for quantum applications on Near-Term Intermediate Scale Quantum Computers. The last chapter covered Quantum Neural Networks (QNNs) compared to common classical counterparts such as feed-forward ReLU networks (ReNNs). In this chapter, we focus on a different popular quantum variational method, the so called *Variational Quantum Eigensolver* or VQE. VQE bear a similarity to QNNs in that they both involve the optimization of quantum circuits that are parameterized by classical real parameters that are optimized using classical optimization routines. This moves the bulk of the *control* operations, that may be hard to implement on near-term quantum machines to the classical controller, leaving the quantum computer to repeatedly implement the corresponding parameterized circuit. The parameterization of these circuits is often chosen so as to be efficiently implementable on a quantum computer.

In contrast to QNNs however, VQEs are a form of quantum *generative* learning, where the goal is to learn a circuit that generates a quantum state satisfying certain properties. In [31], the goal is to learn a circuit that exactly reproduces some unknown quantum state. For a VQE, the goal is to generate a quantum state that is the *ground state*; or eigenvector with the smallest eigenvalue of a given Hermitian matrix. Due to the role played by Hermitian operators as observables in quantum mechanics; the learning loop of a VQE measures the expectation value of the target observable on the output of the parameterized circuit; and this value is then treated as a loss function to optimize the parameters. Despite the apparent inflexibility of this approach, choosing the Hermitian and ansatz carefully can lead to a great number of interesting problems being expressed this way: VQE has been employed in physical problems where obtaining properties of ground states can be of central importance, and several physically inspired ansatz have been designed for such problems including the Transverse Field Ising model, XXZ Heisenberg model, and the Kitaev Honeycomb model. On the other hand, there are approaches to embedding combinatorial optimization problems in the target Hamiltonians in the search for quantum speedups through VQE like systems, the primary example here is the Quantum Approximate Optimization Algorithm (QAOA) [29].

VQEs suffer from some of the same problems as QNNs in that their expressive advantages over classical systems have not been theoretically established. However, another central problem arises from the non-convexity of the associated optimization problems: therefore even if it could be shown that expressive advantages exist, there is no guarantee that the parameters can be optimized to have small training loss. There have been empirical observations of difficulties in optimizing VQEs due to the presence of suboptimal local minima; the presence of exponentially many local minima has been theoretically confirmed in the case of under-parameterized QNNs [249]. The problem of non-convexity is further excacerbated by the observation that randomly initialized deep VQEs are likely to suffer from the *barren plateau problem* [250], wherein it is likely that for any given setting of the parameters the measured gradients will be vanishingly small. This leads to two further problems: small gradients are more difficult to measure accurately on noisy systems, and even in a noiseless setting small gradients can lead to optimization algorithms stalling at objective values far from the minimum. Several empirical studies [251, 252, 253, 254, 255] have empirically explored the impact of architecture and initialization choices on the convergence of these systems.

A very similar issue occurs in the case of classical deep neural networks. The associated optimization problems in deep learning are manifestly non-convex, and the empirical observation that deep neural networks tend to converge to very small training error, has been the basis of a long time puzzle in classical machine learning. In recent years there have been explanations of this phenomenon [256, 257, 258, 259] based on *over-parameterization*. Deep Neural Networks often have a very large number of parameters, compared to the input dimension as well as the size of the data set. It has been shown that coupled with suitably chosen random initializations, this leads to a *regularizing* effect where the dynamics of training concentrates around some convergent dynamics. This is often referred to as *lazy training* because some important quantity connected to the dynamics is shown to vary very slowly through the training process. In the specific case of deep neural networks, this quantity is the *Neural Tangent Kernel* [258].

Contributions. In this chapter, we attempt to construct a theory based on over-parameterization to give the first known *sufficient conditions* for a particular VQE ansatz to converge. The theoretical results are obtained for a specially formulated ansatz which we call the *Partially Trainable Ansatz*; we argue empirically and theoretically that this ansatz is a good stand in for more common architectures of VQEs. Our specific contributions are as follows:

1. We provide the first rigorous proof of convergence for over-parameterized VQEs for the

partially trainable ansatz, and establish sufficient conditions on the degree of over-parameterization required to ensure this convergence.

2. We show that certain restrictions made in the choice of ansatz can reduce the degree of over-parameterization required for convergence.

3. We perform an empirical analysis based on our results: first we confirm our theoretical hypothesis in the practical setting. Importantly we show that the conditions on convergence established hold also for commonly used practical ansatz as well as the partially trainable ansatz.

4. We use the notion of reduced over-parameterization requirements to study and evaluate the design of commonly used physical ansatz choices. We investigate whether the design choices are justified according to our theory. This may establish a framework for future heuristic ansatz designs to be studied and evaluated in a principled manner.

Related Work

Existing Studies of VQA dynamics Exploring the role of overparameterization in the convergence of large classical variational systems such as deep neural networks has been a very active area of research in theoretical machine learning in recent years. Jacot et. al [258] introduced the notion of a neural tangent kernel, identifying the dynamics of training highly overparameterized neural networks with kernel training with a fixed kernel. Arora et. al [257] and Allen-Zhu et. al [259] make this notion exact by showing sufficient overparameter ization conditions for the convergence of various architectures of deep neural networks based on this observation.

The theoretical study of training variational quantum algorithms is even more recent, leading to a number of papers over the previous couple of years. Anschuetz [260] studies the role of overparameterization in the optimization landscape of quantum generative models, and show that there exists a critical point in the number of parameters above which all local minima are close to the global minimum in function value, and below which local minima are far from the global minimum in general. This provides strong evidence that overparameterized neural networks are more likely to converge, but [260] does not rigorously show convergence to the desired quantum state since non-convex optimization problems may not converge to local minima due to getting stuck in saddle points or barren plateaus. Larocca et. al [261] study overparameterization from an information theoretic perspective, defining the overparameterization threshold as the point beyond which adding new parameters does not increase the rank of a Quantum Fischer Information Matrix. To the best of our knowledge our work is the first to show that sufficient overparameterization guarantees the successful convergence of a VQE system.

There has also been some work on the study of tangent kernels in the quantum setting. Liu et. al [262] and Shirai et. al [263] hypothesize that the the training of Quantum Neural Networks (QNNs) can be identified with kernel training with the corresponding tangent kernel, and empirically study the training of such kernels as a stand in for directly training QNNs. Abadi et. al [264] show that the tangent kernel can indeed be shown to be slow varying when the system dimension is arbitrarily large. There has not yet been a rigorous analysis of Quantum Neural Networks in the overparameterized setting, and this remains an interesting problem for future study.

A final paper that may be of interest to our setting is [265], where the authors explore Riemannian Gradient Flow directly over the unitary group. We instead analyze the optimization of VQEs using Riemannian Gradient Flow over the sphere, the convergence of which has already been established classically [266], allowing us to establish convergence results in the quantum setting.

Connection to the *Barren Plateau* phenomenon A phenomenon that is anticipated to present difficulties in the training of varaitonal quantum algorithms is the so-called *barren plateau* phenomenon (first observed by McClean et. al [250]). The phenomenon shows that the gradients of sufficiently large randomly initialized parameterized quantum systems are likely to be exponentially decaying (with the number of qubits in the system). Specifically, [250] considers a n-qubit parameterized quantum circuit with an ansatz $\mathbf{U} : \mathbb{R}^p \to SU(2^n)$. When the parameters are randomly initialized to $\boldsymbol{\theta} \in \mathbb{R}^p$, the loss function $L(\boldsymbol{\theta}) = \langle 0|\mathbf{U}(\boldsymbol{\theta})\mathbf{M}\mathbf{U}(\boldsymbol{\theta})|0\rangle$ and its partial derivatives are random variables. If $\mathbf{U}$ is deep enough that $\mathbf{U}(\boldsymbol{\theta})$ is approximately *Haar* distributed on $SU(d)$,

$$\mathbb{E}[L] = 0 \text{ and } \mathrm{Variance}[\partial_k L] = O\left(\frac{1}{2^{2n}}\right), \quad \forall k \in [p] \tag{7.1}$$

With probability at least $1-\delta$ therefore, $|\partial_k L|^2 \leq O\left(\frac{1}{2^{2n}} \log\left(\frac{1}{\delta}\right)\right)$ and for systems with a large number of qubits n, the gradient components can be vanishingly small, leading to the eponymous barren plateaus in the landscape. The main possible difficulties arising from this phenomenon are two-fold

- Firstly, the components of the gradient of variational quantum sytems are measured in practice by estimating the expectation value of some Hermitian operator through repeated measurements of *shots*. If the components are exponentially decaying in n, the estimates of the expectations need to be correspondingly precise, leading to the number of shots necessary growing exponentially with n. This represents an exponential overhead in the training cost of the circuit.

- Secondly, the existence of vanishingly small gradients may indicate that the training landscape is infeasible to optimization by gradient based methods. Even if the landscape is free of spurious local minima, an optimization algorithm can in principle require a long time to find any minimum at all. Alongside the existence of *saddle points* that trap gradient based algorithms, barren plateaus constitute one of the main difficulties in non-convex optimization.

Our results show that, for variational quantum eigensolvers with sufficient over-parameterization, the latter issues does *not* arise and the deviation of the output from the target space decays exponentially over time as $\exp(\Delta t/n)$ where the target Hamiltonian has spectral gap Δ (see Theorem 7.3.1). This convergence can exist even with vanishing gradients because the gradients along the trajectory are spatially correlated along the training trajectory leading to significant progress towards the global minimum despite the small gradient components. Intuitively, this situation is similar to that in unstable equilibria in dynamical systems, where small forces can combine to cause significant deviations from equilibrium positions. We also show that the convergence is robust to a certain threshold of noise in the gradients (Corollary 7.4.1). The tolerable noise threshold is however $O(1/2^{2n})$ in n-qubit systems and therefore cannot resolve the first issue observed above. We note however, that the degree of parameterization required to ensure convergence is already exponentially large in n, and we have the noise tolerance improves along with the parameterization requirements. This is because the noise tolerance as well as the parameterization threshold are dictated by a quantity called the *effective dimension* (Corollary 7.5.1) which is equal to 2^n in general but may be significantly smaller for certain structured ansatz (see Section 7.5). Finally, we mention that the vanishing gradient problem occurs also in classical neural networks where the gradients decay exponentially with the network *depth*. Over-parameterization has been shown to still enable convergence in such systems ([259]), our results effectively establish the same phenomenon for VQEs.

Organization We first give an introduction to the framework used to obtain convergence results for overparameterized classical systems in Section 7.2.2. In Section 7.2.1 we introduce the background behind VQEs and formally introduce and motivate the Partially Trainable Ansatz. The main theory of converge- nce for VQE is introduced in Section 7.3. We then present a noise robustness threshold for convergence in Section 7.4. Finally, we present a more specific form of the theory that depends on the properties of the specific ansatze used, and discuss how this can be used to understand the properties of these ansatze. We empirically confirm our theory in Section 7.6 and use the results of Section 7.4 to predict the empirical performance of various ansatz in Section 7.7.

7.2 Preliminaries

7.2.1 Variational Quantum Algorithms

Variational Quantum Algorithms (VQA) is a paradigm of quantum algorithms for search over a family of functions modeled as quantum ansatze. A quantum ansatz is a parameterized model that maps real parameters to unitary operators. A p-parameter ansatz on a d-dimensional

Hilbert space $\mathcal{H}$ can be denoted $\mathbf{U}\colon \mathbb{R}^p \to \mathbb{C}^{d\times d}$ and maps real parameters $\boldsymbol{\theta}$ to an operator $\mathbf{U}(\boldsymbol{\theta})$. A prominent example of VQA is the Variational Quantum Eigensolvers (VQE). A VQE intance is specified by a triplet $(\mathbf{M}, |\Phi\rangle, \mathbf{U})$, and the goal is to approximate the ground state of a $d \times d$ problem Hamiltonian $\mathbf{M}$ as $\mathbf{U}(\boldsymbol{\theta})|\Phi\rangle$ using the p-parameter ansatz $\mathbf{U}\colon \mathbb{R}^p \to \mathbb{C}^{d\times d}$ and the input state $|\Phi\rangle \in \mathbb{C}^d$, by solving the following optimization problem

$$\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) := \langle\Phi|\mathbf{U}^\dagger(\boldsymbol{\theta})\mathbf{M}\mathbf{U}(\boldsymbol{\theta})|\Phi\rangle \tag{7.2}$$

The search for optimal parameter $\boldsymbol{\theta}^\star$ are usually performed by gradient descent $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$. For sufficiently small learning rate η, the dynamics of gradient descent reduces to that of gradient flow

$$d\boldsymbol{\theta}/dt = -\eta\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) \tag{7.3}$$

Popular choices of parameterizations include the Hardware-Efficient Ansatz (HEA) and Hamiltonian Variational Ansatz (HVA). Hardware-efficient ansatz (e.g. [267]) is a family of parameterized circuits that makes use of native gates of quantum computers and is mostly composed of interleaving single-/two-qubit Pauli rotations and entanglement unitaries implemented with CZ / CNOT gates. The main motivation behind the design is to facilitate the implementation on real quantum machines. Hamiltonian variational ansatz (e.g. [3, 268]) is a family of problem-specific ansatz design that utilize the structure of the problem Hamiltonian.

Fully- and partially-trainable Ansatz. In this work, we consider a general family of ansatze that includes HEA and HVA as special cases, specified by the number of layers L and a set of K

$d \times d$ Hermitians $\{\mathbf{H}_1, \mathbf{H}_2, \cdots, \mathbf{H}_K\}$:

Definition 7.2.1 (Fully-trainable ansatz)**.** *A fully-trainable L-layer ansatz with a set of Hermitians $\mathcal{A} = \{\mathbf{H}_1, \mathbf{H}_2, \cdots, \mathbf{H}_K\}$ has $K \cdot L$ trainable parameters and is defined as*

$$\mathbf{U}^{(L)}(\boldsymbol{\theta}) = \prod_{l=1}^{L} \prod_{k=1}^{K} \exp(-i\theta_{l,k}\mathbf{H}_k) \tag{7.4}$$

The superscript L will be omitted when there is no ambiguity.

For a fixed set $\mathcal{A} = \{\mathbf{H}_1, \cdots, \mathbf{H}_K\}$ and the parameter domain $\Theta \subseteq \mathbb{R}^K$, the set of all achievable matrices $\cup_{L=0}^{\infty}\{\mathbf{U}^{(L)}(\boldsymbol{\theta}) : \boldsymbol{\theta} \in \Theta^L \subseteq \mathbb{R}^{K \cdot L}\}$ forms a subgroup of $SU(d)$ which we will refer to as $G_{\mathcal{A},\Theta}$. For many choices of ansatz with a limited set of $\mathcal{A}$, $G_{\mathcal{A},\Theta}$ is a strict subgroup of $SU(d)$. The subscript Θ will be dropped when $\Theta = \mathbb{R}^K$ for a more concise notation.

Using the group $G_{\mathcal{A}}$, we define the partially-trainable ansatze associated with $\mathcal{A}$ as:

Definition 7.2.2 (Partially-Trainable Anastz for $\mathcal{A}$)**.** *Let the subgroup $G_{\mathcal{A}}$ be a subgroup of $SU(d)$ associated with fully-trainable ansatze with a set of Hermitians $\mathcal{A} = \{\mathbf{H}_1, \mathbf{H}_2, \cdots \mathbf{H}_K\}$. The corresponding p-parameter partially-trainable ansatz is defined as:*

$$\mathbf{U}(\boldsymbol{\theta}) = \big(\prod_{l=1}^{p} \mathbf{U}_l \exp(-i\boldsymbol{\theta}_l\mathbf{H}_k)\big)\mathbf{U}_0 \tag{7.5}$$

where k is arbitrarily chosen in $[K]$ and $\mathbf{U}_l$ are $i.i.d.$ sampled from the Haar measure over $G_{\mathcal{A}}$. (Due to the Haar distribution of the matrices $\mathbf{U}_l$ the distribution of $\mathbf{U}(\boldsymbol{\theta})$ is independent of the choice of k.)

The partially-trainable ansatz can be viewed as a fully-trainable ansatz trained on a subset of the parameters starting from a random initialization by identically and independently sample

$(\theta_{l,1}, \cdots, \theta_{l,k})$ from distribution $\mathcal{D}$ over $\Theta \subseteq \mathbb{R}^K$ for all $l \in [L]$, hence the name "partially-trainable": Let S be a subset of $SU(d)$ defined as $S := \{\prod_{k=1}^{K} \exp(-i\theta_k \mathbf{H}_k) : \boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^K\}$, the distribution of the fully-trainable unitary $\mathbf{U}^{(L)}$ is equivalent to a random walk over the group $G_{\mathcal{A},\Theta}$ with each step sampled from S according to $\mathcal{D}$. Under mild regularity conditions, the random walk converges to the Haar measure over $G_{\mathcal{A},\Theta}$ (See [269, Section 3]). In the case where the ansatz has a set of parameters spanning the entire group $SU(d)$ it has been shown ([270]) $O(t^{10}(\log d)^2)$ random nearest-neighbor 2-qubit gates suffice to realize an *approximate t-design*, ie. a distribution that matches the first t-moments of the Haar distribution. Therefore training a fully-trainable ansatz in Eq (7.4) on $\{\theta_{l,k}\}$ for l's that are multiples of a large constant and $k = O(t^{10}(\log d)^2)$ is approximately equivalent to optimizing the partially-trainable ansatz defined in Eq (7.5) upto the first t-moments.

The Haar measure constitutes the most natural distribution over unitary matrices, and integration over these measures is well studied [271]. In this work, we theoretically analyze the convergence of gradient flow in VQE instances with partially-trainable ansatze for analytic convenience. We argue above that the behavior of partially-trainable ansatze may closely mimic that of fully-trainable ansatze that are only logarithmically larger in the Hilbert dimension (d). Furthermore, we will see in our empirical study (Section 7.6.1), our theory for partially-trainable ansatze makes precise predictions also for fully-trainable ansatze.

7.2.2 Convergence in overparameterized classical systems

Overparameterization has been proposed as an explanation for the convergence of the highly nonconvex training of parameterized classical models such as artificial neural networks [256, 257, 258]. The convergence of the models arises from two main phenomenon:

1. **Convergence of expected dynamics:** When the parameters are randomly initialized, the expected dynamics of the training are shown to exhibit convergence to a global minima. The expected dynamics is therefore a smoothed version of the actual dynamics that removes some of the irregularities that can lead to a failure in convergence.

2. **Convergence under perturbation:** Despite the convergence of the training dynamics in expectation, the actual training corresponds to a particular setting of initial parameters. This leads to the actual training being a perturbed version of the expected dynamics, it is thus necessary to show that the convergence of this dynamics is robust to small perturbations.

3. **Concentration at initialization:** Due to the law of large numbers, with high probability, deviations from the expected dynamics decrease as the number of random parameters increases. Overparameterization thus plays the crucial role of leading to the *concentration* of the dynamics around the expected value, allowing the magnitude of random perturbations to be bounded with high probability.

4. **Lazy training:** It must be shown that the actual training concentrates throughout the training given the convergence at initialization. This phenomenon has been characterized as *lazy training* [256], where the dynamics of a system at initialization remain a good approximation throughout its training. Once again, overparameterization plays an important role in ensuring this phenomenon; as the number of parameters increases the changes in each parameter become smaller with high probability over the course of training.

This method can be illustrated by the example of the *Neural Tangent Kernel* [258], which has been used to show convergence while training several overparameterized classical neural networks including wide feedforward networks [257].

Consider a classical classification problem where the input data is drawn from a distribution p_{in} over $\mathbb{R}^{n_0}$ and an output in $\mathbb{R}^{n_L}$, the space of valid functions is given by $\mathcal{F} = \{\mathbf{f} \colon \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}\}$. The model is specified as a *realization function* mapping p parameters to candidate functions $\mathbf{F}^{(L)} \colon \mathbb{R}^p \to \mathcal{F}$. Denoting the parameters at time t by $\theta(t) = (\theta_1(t), \dots, \theta_p(t))$, the function at time t is given by $\mathbf{F}^{(L)}(\theta(t))$. The data distribtution induces an inner product over $\mathcal{F}$ given by $\langle \mathbf{f}, \mathbf{g} \rangle_{p_{in}} = \mathbb{E}_{x \sim p_{in}}[\mathbf{f}(x)^T \mathbf{g}(x)]$. Given a cost function C, the gradient flow dynamics of the system correspond to *kernel training* with respect to the *neural tangent kernel* (NTK) given by $\tilde{\mathbf{K}} = \sum_{l=1}^{p} \partial_{\theta_l} \mathbf{F}^{(l)}(\theta) \otimes \partial_{\theta_l} \mathbf{F}^{(l)}(\theta)$.

Let $\mathbf{y} \in \mathcal{F}$ be the true function mapping inputs to ouputs resulting in the residual function $\nabla(\theta(t)) = \mathbf{y} - \mathbf{F}^{(L)}(\theta(t))$. If C is the squared loss function, the dynamics of the system is simply given by $\dot{\mathbf{r}} = -\eta \tilde{\mathbf{K}} \mathbf{r}$ where η is the chosen step size. It is known that if $\tilde{\mathbf{K}}$ is a constant positive definite matrix, the system exhibits linear convergence. Following the above recipe, this leads to a framework for showing the convergence of classical neural networks, it is shown that $\mathbf{K} = \mathbb{E}(\tilde{\mathbf{K}}(\theta(0)))$ is a positive definite constant matrix. It is also shown that the dynamics $\dot{\mathbf{r}} = -\eta \tilde{\mathbf{K}} \mathbf{r}$ converges whenever $\|\tilde{\mathbf{K}} - \mathbf{K}\| \leq \epsilon_0$. Further define an overparameterization threshold $P^{(L)}(n_0, n_L)$Convergence can then be established via the following propositions:

1. **Concentration at initialization**: If $p > P^L$, $\|\tilde{\mathbf{K}}(\theta(0)) - \mathbf{K}\| \leq \epsilon_0$ with probability at least $9/10$.

2. **Small perturbations imply convergence**: $\|\tilde{\mathbf{K}}(\theta(t)) - \mathbf{K}\| \leq \epsilon_0$ for all $t < t'$, we have $\|\mathbf{r}(t) - \tilde{\mathbf{r}}(\mathbf{t})\| \leq \epsilon_1$ for all $t \leq t_1$, where $\tilde{\mathbf{r}}$ denotes the residuals when the kernel is frozen

at initialization (in which case the system is known to converge).

3. **Convergence implies small perturbations**: If $p > P^L$, and $\|\mathbf{r}(t) - \tilde{\mathbf{r}}(\mathbf{t})\| \leq \epsilon_1$ for all $t < t'$, we have $\|\tilde{\mathbf{K}}(\theta(t)) - \mathbf{K}\| \leq \epsilon_0$ for all $t \leq t'$ with probability at least $9/10$

These propositions are sufficient to inductively prove the convergence of the training dynamics to a global minimum. Consider the earliest time t_0 where the perturbation in the kernel is too large; by the final proposition this can only occur if the convergence of the system is violated at some time $t'_0 < t_o$. However, by the second proposition, this would imply that for an earlier time t''_0 the kernel perturbation must have been too large, contradicting our initial assumption that t_0 was the earliest such time. This shows that both the small perturbation condition as well as the convergence of the system are maintained throughout the training.

7.3 A convergence theory for VQE

In this section we use ideas from the classical theory of overparameterized variational systems to give sufficient conditions for the convergence of a VQE to zero loss. We also establish the main factors influencing the (linear) rate of convergence. As discussed in Section 7.2.2, the random initialization of the parameters plays an important role in the convergence. For the results of this section, we rely on the *Partially Trainable Ansatz*. To demonstrate the main techniques we restrict ourselves to the case where the set of Hermitians defining the ansatz form a complete basis of the Lie Algebra of $SU(d)$. The corresponding induced subgroup is therefore the entire unitary group $SU(d)$. Under this setting, we recall the definition as follows

Definition 7.3.1 (Partially Trainable Ansatzover $SU(d)$)**.** *Consider a quantum system over n-qubits with a corresponding Hilbert space of dimension $d = 2^n$. We define a (p-parameter)* *random ansatz parameterized by* $\boldsymbol{\theta} \in \mathbb{R}^p$*:*

$$\mathbf{U}(\boldsymbol{\theta}) = \mathbf{U}_p \exp(-i\theta_p \mathbf{H}) \cdot \dots \cdot \mathbf{U}_1 \exp(-i\theta_1 \mathbf{H})\mathbf{U}_0 \tag{7.6}$$

where $\mathbf{U}_l$ *are sampled* $i.i.d.$ *with respect to the Haar measure over* $SU(d)$*, and* $\mathbf{H}$ *is a trace-*0 *Hermitian.*

The Partially Trainable Ansatz has several appealing analytical properties for the analysis of convergence. Firstly, the random initialization of the ansatz is restricted to the unitaries $\mathbf{U}_l$ which are never updated during the training. There is thus a clear separation between the random initialization and dynamic evaluation of the system. Furthermore, since $\mathbf{U}_l$ are sampled from the right-invariant Haar measure over $SU(d)$, the distribution of $\mathbf{U}(\theta)$ is independent of the scheme used to initialize the trainable parameters $\boldsymbol{\theta}$. Finally, the distribution of the ansatz is also invariant under *arbitrarily* changes to the parameters, therefore the distribution of the ansatz does not change as the system evolves. This intuitively indicates that important quantities connected to the ansatz may be *slow-varying* as their expectation value remains the same. The remainder of the section will formalize this notion.

Recall that a VQE instance is specified by a specified by a problem Hamiltonian $\mathbf{M} \in \mathbb{C}^{d\times d}$, an input state $|\Phi\rangle \in \mathbb{C}^d$ and an p-parameter variational ansatz $\mathbf{U} : \mathbb{R}^p \to \mathbb{C}^{d\times d}$, and seeks to solve the following optimization problem

$$\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) := \langle\Phi|\mathbf{U}^\dagger(\boldsymbol{\theta})\mathbf{M}\mathbf{U}(\boldsymbol{\theta})|\Phi\rangle \tag{7.7}$$

We investigate the dynamics of training the system using *gradient flow*, ie. the parameters are updated according to the differential equation $d\boldsymbol{\theta}/dt = -\eta \nabla_{\boldsymbol{\theta}} L(\theta)$, where η is some previously chosen learning rate.

Notation and Definitions . We define here some notations, quantities, and conventions that will play an important role in the forthcoming analysis.

- $\mathbf{W}_{d^2\times d^2}$ denotes the swap operator $\sum_{a,b\in[d]} \mathbf{E}_{ab} \otimes \mathbf{E}_{ba}$.
- $\mathbf{I}_{d^2\times d^2}$ denotes the identity matrix in $\mathbb{C}^{d^2\times d^2}$
- Without loss of generality we assume $\mathrm{Tr}(\mathbf{M}) = \mathrm{Tr}(\mathbf{H}) = 0$.
- We use the short hand $\mathbf{U}_{l+1:p}(\boldsymbol{\theta})$ to represent

$$\mathbf{U}_{l+1:p}(\boldsymbol{\theta}) := \mathbf{U}_p(\theta_p) \cdots \mathbf{U}_{l+1}(\theta_{l+1}) \tag{7.8}$$

  where $\boldsymbol{\theta}$ may be omitted when not ambiguous

- A common normalizing factor appears due to Haar integral: for any $d \times d$-Hermitian $\mathbf{A}$, define $Z(\mathbf{A}, d) := \frac{\mathrm{Tr}(\mathbf{A}^2)}{d^2-1}$.
- The matrix

$$\mathbf{V}_l(\boldsymbol{\theta}) := \mathbf{U}_p(\theta_p) \cdots \mathbf{U}_l(\theta_l) = \mathbf{U}_p \exp(-i\theta_p \mathbf{H}) \cdot \cdots \cdot \mathbf{U}_l \exp(-i\theta_l \mathbf{H}) \tag{7.9}$$

  is defined as the composition of all the layers of the ansatz from p to l. Using this notation, the ansatz $\mathbf{U}(\boldsymbol{\theta})$ can be written as $\mathbf{V}_1(\boldsymbol{\theta})\mathbf{U}_0$.

- We define the matrix $\mathbf{H}_l(\boldsymbol{\theta}) := \mathbf{V}_l(\boldsymbol{\theta})\mathbf{H}\mathbf{V}_l(\boldsymbol{\theta})^\dagger$.

- The central quantity dictating the dynamics of VQE (see Lemma 7.3.1) is given by the matrix

$$\mathbf{Y}(\boldsymbol{\theta}) := \frac{1}{pZ(\mathbf{H}, d)} \sum_{l=1}^{p} \mathbf{H}_l^{\otimes 2}. \tag{7.10}$$

- We note finally that $|\Psi\rangle, \{\mathbf{U}_l\}, \{\mathbf{V}_l\}, \{\mathbf{H}_l\}, \mathbf{Y}$ are also functions of time t through the evolution of the parameters $\boldsymbol{\theta}(t)$.

Main Elements of Theory. Following the structure in Section 7.2.2 we outline the main components of our analysis:

1. **Identifying (idealized) gradient flow dynamics of VQE with classical dynamics that is known to converge.** We show that the dynamics of VQE is equivalent to Riemannian Gradient Flow (RGF) over the unit sphere in d-dimensions, by tracking the evolution of the output state. Specifically, we have the following lemma

   Lemma 7.3.1 (VQE output-state dynamics under gradient flow)**.** *For a VQE instance* $(\mathbf{M}, |\Psi\rangle, \mathbf{U})$*, with* $\mathbf{U}$ *being the ansatz defined in Definition 7.3.1, when optimized with gradient flow with learning rate* η*, the output state* $|\Psi\rangle$ *follow the dynamics*

$$\frac{d}{dt}|\Psi\rangle = -(\eta \cdot p \cdot Z(\mathbf{H}, d)) \operatorname{Tr}_1(\mathbf{Y} \cdot ([\mathbf{M}, |\Psi\rangle\langle\Psi|] \otimes \mathbf{I}_{d\times d}))|\Psi\rangle \tag{7.11}$$

   Here the Hermitian $\mathbf{Y} \in \mathbb{C}^{d^2\times d^2}$ *is a time-dependent matrix defined as*

$$\mathbf{Y}(\boldsymbol{\theta}) := \frac{1}{pZ(\mathbf{H}, d)} \sum_{l=1}^{p} \left(\mathbf{U}_{l:p}(\boldsymbol{\theta}(t))\mathbf{H}\mathbf{U}_{l:p}^{\dagger}(\boldsymbol{\theta}(t))\right)^{\otimes 2} \tag{7.12}$$

Over the randomness of ansatz initialization, for all $\boldsymbol{\theta} \in \mathbb{R}^p$, the expected value of matrix $\mathbf{Y}$ is $\mathbf{Y}^\star = \mathbf{W}_{d^2\times d^2} - \frac{1}{d}\mathbf{I}_{d^2\times d^2}$. If we choose $\eta = \frac{1}{pZ(\mathbf{H})}$, the VQE dynamics allows the following decomposition:

$$\frac{d}{dt}|\Psi\rangle = -[\mathbf{M}, |\Psi\rangle\langle\Psi|]|\Psi\rangle - \mathrm{Tr}_1\left((\mathbf{Y} - \mathbf{Y}^\star)\cdot[\mathbf{M}, |\Psi\rangle\langle\Psi| \otimes \mathbf{I}_{d\times d}]\right)|\Psi\rangle \tag{7.13}$$

The first term is exactly the Riemannian gradient descent over the unit sphere with loss function $\langle\Psi|\mathbf{M}|\Psi\rangle$, which is known to converge linearly to the ground state [266]. Lemma 7.3.1 shows that the main quantity that controls the deviation of the VQE gradient flow from RGF over the sphere is $\mathbf{Y} - \mathbf{Y}^\star$, ie. the deviation of $\mathbf{Y}$ from it's expectation.

2. **Convergence of idealized dynamics under small perturbations.** The deviation of $\mathbf{Y}$ from its expectation cannot be zero in general, so we must establish that a small perturbation to RGF on the sphere maintains the property of linear convergence. The following lemma (analogous to Lemma F.1 in [257]) states that, if $\mathbf{Y}(t)$ is close to $\mathbf{Y}^\star$ through out the optimization, then the gradient flow is guaranteed to find the ground state efficiently

Lemma 7.3.2 (VQE Perturbation Lemma)**.** *Conditioned on the event that the output state at initialization $|\Psi(0)\rangle$ has non-negligible overlap with the target ground state $|\Psi^\star\rangle$: $|\langle\Psi(0)|\Psi^\star\rangle|^2 \geq O(\frac{1}{d})$, if for all $t \geq 0$, $\|\mathbf{Y}(t) - (\mathbf{W} - \frac{1}{d}\mathbf{I}_{d^2\times d^2})\|_{\mathsf{op}} \leq \frac{\lambda_2-\lambda_1}{\lambda_d-\lambda_1}\cdot\frac{1}{2\sqrt{2}d}$, then under the dynamics $\frac{d}{dt}|\Psi\rangle = -\mathrm{Tr}_1(\mathbf{Y}([\mathbf{M}, |\Psi\rangle\langle\Psi|] \otimes \mathbf{I}_{d\times d}))|\Psi\rangle$, the output states converges to the ground state:*

$$1 - |\langle\Psi(t)|\Psi^\star\rangle|^2 \leq \exp(-\frac{\lambda_2 - \lambda_1}{2\log d}t). \tag{7.14}$$

3. **Concentration to idealized dynamics throughout training.** In order to show that the perturbations from RGF over the sphere are small, we leverage the concentration properties of $\mathbf{Y}$ arising from the large number of parameters used in order to bound the deviation from expectation by a quantity decreasing in p. We first show that concentration holds at initialization

   Lemma 7.3.3 (Concentration at initialization for VQE)**.** *Over the randomness of ansatz initialization (i.e. for $\{\mathbf{U}_l\}_{l=1}^p$ sampled $i.i.d.$ with respect to the Haar measure), for any initial $\boldsymbol{\theta}(0)$, with probability $1-\delta$:*

$$\|\mathbf{Y}(\boldsymbol{\theta}(0)) - \mathbf{Y}^\star\|_{\mathsf{op}} \leq \frac{1}{\sqrt{p}} \cdot \frac{2\|\mathbf{H}\|_{\mathsf{op}}^2}{Z}\sqrt{\log\frac{d^2}{\delta}} \tag{7.15}$$

   We further show that the concentration is maintained throughout the evolution of the dynamics as long as exponential convergence holds

   Lemma 7.3.4 (Concentration during training (time dependent))**.** *Suppose that under learning rate $\eta = \frac{1}{pZ(\mathbf{H},d)}$, for all $0 \leq t \leq T$, $1 - |\langle\Psi|\Psi^\star|\rangle|^2 \leq \exp(-\frac{(\lambda_2-\lambda_1)}{2\log d}t)$, then with probability $\geq 1-\delta$, for all $t \geq 0$:*

$$\|\mathbf{Y}(t) - \mathbf{Y}(0)\|_{\mathsf{op}} \leq C_3\left(\frac{T}{\sqrt{p}} \cdot \sqrt{2}(\lambda_d - \lambda_1) \cdot \sqrt{\frac{d^2-1}{Z(\mathbf{H},d)^3}}\left(1 + \sqrt{\log\left(\frac{2d}{\delta}\right)}\right)\right) \tag{7.16}$$

   where C_3 is a constant.

4. **(Main Result) Sufficient conditions for convergence.** The previously established conditions on concentration and convergence under perturbations are combined to yield a sufficient

condition on the degree of overparameterization required to ensure that a VQE converges to its ground state under gradient flow.

Theorem 7.3.1 (Exponential convergence for VQE)**.** *Consider a VQE system in a d-dimensional Hilbert space (with architecture as described in Definition 7.3.1) with a target Hamiltonian $\mathbf{M}$ with eigenvalues $\lambda_1 \leq \lambda_2 \ldots \lambda_d$, generating Hamiltonian $\mathbf{H}$ and number of parameters p be greater than $\Omega\left(\left(\frac{\lambda_d-\lambda_1}{\lambda_2-\lambda_1}\right)^4, \frac{d^4}{Z(\mathbf{H},d)^3}, \log\left(\frac{2d}{\delta}\right)\right)$. Denote the ground state of the system by $|\psi^*\rangle$. Conditioned on the event that the output state at initialization $|\Psi(0)\rangle$ has non-negligible overlap with the target ground state $|\Psi^\star\rangle$: $|\langle\Psi(0)|\Psi^\star\rangle|^2 \geq O(\frac{1}{d})$, and with a learning rate of $\eta = \frac{1}{pZ(\mathbf{H},d)}$, the system converges to the ground state with error $\epsilon = 1 - \langle\psi|\psi^*\rangle^2$ in time $T_\epsilon = \frac{2\log d}{\lambda_2-\lambda_1}\log\left(\frac{1}{\epsilon}\right)$ with failure probability at most δ.*

Technical details and proofs. In the following sections we describe the main technical ideas behind the results outlined previously. The proof of convergence under perturbation, and concentration of initialization follow relatively well known techniques and are postponed to Section 7.8.1 in the appendix. The identification of VQE gradient flow (Lemma 7.3.1) is proved in Section 7.3.1. The proof of concent- ration during training (Lemma 7.3.4) is in Section 7.3.2, and the main theorem is proved in Section 7.3.3.

7.3.1 Identify VQE with Reimannian Gradient Flow (RGF) over unit sphere

Lemma 7.3.1 (VQE output-state dynamics under gradient flow)**.** *For a VQE instance* $(\mathbf{M}, |\Psi\rangle, \mathbf{U})$, *with $\mathbf{U}$ being the ansatz defined in Definition 7.3.1, when optimized with gradient flow with* *learning rate η, the output state $|\Psi\rangle$ follow the dynamics*

$$\frac{d}{dt}|\Psi\rangle = -(\eta \cdot p \cdot Z(\mathbf{H}, d)) \operatorname{Tr}_1(\mathbf{Y} \cdot ([\mathbf{M}, |\Psi\rangle\langle\Psi|] \otimes \mathbf{I}_{d\times d}))|\Psi\rangle \tag{7.11}$$

Here the Hermitian $\mathbf{Y} \in \mathbb{C}^{d^2\times d^2}$ is a time-dependent matrix defined as

$$\mathbf{Y}(\boldsymbol{\theta}) := \frac{1}{pZ(\mathbf{H}, d)} \sum_{l=1}^{p} \left(\mathbf{U}_{l:p}(\boldsymbol{\theta}(t))\mathbf{H}\mathbf{U}_{l:p}^{\dagger}(\boldsymbol{\theta}(t))\right)^{\otimes 2} \tag{7.12}$$

Proof. We start by calculating the gradient of $\mathbf{U}_{r:p}(\boldsymbol{\theta})$ with respect to θ_l: For $r > l$, $\mathbf{U}_{r:p}$ is independent of θ_l ; For $r \leq l$,

$$\frac{\partial \mathbf{U}_{r:p}}{\partial \theta_l} = \mathbf{U}_{l:p}(\boldsymbol{\theta}) \cdot (-i\mathbf{H}) \cdot \mathbf{U}_{r:l-1}(\boldsymbol{\theta}) = -i\mathbf{U}_{l:p}\mathbf{H}\mathbf{U}_{l:p}^{\dagger}\mathbf{U}_{r:p} \tag{7.17}$$

Therefore

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \theta_l} = \langle\Phi|\mathbf{U}_0^{\dagger}\partial_l\mathbf{U}_{1:p}^{\dagger}\mathbf{M}\mathbf{U}_{1:p}\mathbf{U}_0|\Phi\rangle + \langle\Phi|\mathbf{U}_0^{\dagger}\mathbf{U}_{1:p}^{\dagger}\mathbf{M}\partial_l\mathbf{U}_{1:p}\mathbf{U}_0|\Phi\rangle \tag{7.18}$$

$$= \langle\Phi|\mathbf{U}_0^{\dagger}\mathbf{U}_{1:p}^{\dagger}i[\mathbf{U}_{l:p}\mathbf{H}\mathbf{U}_{l:p}^{\dagger}, \mathbf{M}]\mathbf{U}_{1:p}\mathbf{U}_0|\Phi\rangle \tag{7.19}$$

$$= i \operatorname{Tr}([\mathbf{M}, |\Psi\rangle\langle\Psi|]\mathbf{U}_{l:p}\mathbf{H}\mathbf{U}_{l:p}^{\dagger}) \tag{7.20}$$

Following gradient descent with learning rate η:

$$\frac{d\theta_l}{dt} = -\eta\partial_l\frac{1}{pZ}\partial_l L(\boldsymbol{\theta}) == -i\eta \operatorname{Tr}([\mathbf{M}, |\Psi\rangle\langle\Psi|]\mathbf{U}_{l:p}\mathbf{H}\mathbf{U}_{l:p}^{\dagger}) \tag{7.21}$$

The dynamics for $\mathbf{U}_{l:p}$ and $|\Psi\rangle$ are therefore:

$$\frac{d}{dt}\mathbf{U}_{l:p}(t) = \sum_{r=l}^{p} \frac{d\theta_r}{dt}\partial_r \mathbf{U}_{l:p} = -\eta \sum_{r=l}^{p} \mathrm{Tr}([\mathbf{M}, |\Psi\rangle\langle\Psi|]\mathbf{U}_{l:p}\mathbf{H}\mathbf{U}_{l:p)}^{\dagger})\mathbf{U}_{l:p}\mathbf{H}\mathbf{U}_{l:p}^{\dagger}\mathbf{U}_{l:p} \tag{7.22}$$

$$\frac{d}{dt}|\Psi\rangle = \frac{d}{dt}\mathbf{U}_{1:p}\mathbf{U}_0|\Phi\rangle = -(\eta \cdot pZ)\frac{1}{pZ}\Big(\sum_{l=1}^{p} \mathrm{Tr}([\mathbf{M}, |\Psi\rangle\langle\Psi|]\mathbf{U}_{l:p}\mathbf{H}\mathbf{U}_{l:p)}^{\dagger})\mathbf{U}_{l:p}\mathbf{H}\mathbf{U}_{l:p)}^{\dagger}\Big)\mathbf{U}_{1:p}\mathbf{U}_0|\Phi\rangle \tag{7.23}$$

$$= -(\eta \cdot pZ)\,\mathrm{Tr}_1(\mathbf{Y} \cdot [\mathbf{M}, |\Psi\rangle\langle\Psi|]] \otimes \mathbf{I})|\Psi\rangle \tag{7.24}$$

□

7.3.2 Concentration of dynamics from overparameterization

In this section we wish to prove that $\mathbf{Y}$ concentrates to its expected value throughout training upto any point in time until which the linear convergence condition holds on the gradient flow dynamics. The proof will be based on two main ideas:

1. The linear convergence of the gradient flow dynamics allows the deviation of the parameters $\boldsymbol{\theta}$ from their initial values to be bounded in terms of the evolution time (See Lemma 7.3.5).

2. The random variables $\mathbf{Y}(\boldsymbol{\theta}(t))$ for different times t form a *random field*, whose deviations $\mathbf{Y}(\boldsymbol{\theta}(t_1)) - \mathbf{Y}(\boldsymbol{\theta}(t_1))$ we show to be bounded by a quantity proportional to $|t_1 - t_2|/\sqrt{p}$. We then use Dudley's lemma on the concentration of random fields to bound the supremum of the deviation from initialization over time.

We first show a result connecting the evolution time to the corresponding deviation in $\boldsymbol{\theta}$.

Lemma 7.3.5 (Slow-varying θ)**.** *Suppose that under learning rate* $\eta = \frac{1}{pZ(\mathbf{H},d)}$*, for all* $0 \leq t \leq T$*,* $1 - |\langle\Psi|\Psi^\star|\rangle|^2 \leq \exp(-\frac{(\lambda_2-\lambda_1)}{2\log d}t)$*, then for all* $0 \leq t_1, t_2 \leq T$*:*

$$\begin{aligned}
\|\boldsymbol{\theta}(t_2) - \boldsymbol{\theta}(t_1)\|_\infty &\leq \frac{1}{p} \cdot 4\sqrt{2}\frac{\lambda_d - \lambda_1}{\lambda_2 - \lambda_1} \cdot \sqrt{\frac{d^2-1}{Z(\mathbf{H},d)}} \log d \left(e^{-\frac{\lambda_2-\lambda_1}{4\log d}t_1} - e^{-\frac{\lambda_2-\lambda_1}{4\log d}t_2}\right) \\
&\leq \frac{1}{p} \cdot \sqrt{2}(\lambda_d - \lambda_1) \cdot \sqrt{\frac{d^2-1}{Z(\mathbf{H},d)}} \cdot |t_2 - t_1| \qquad (7.25) \\
\|\boldsymbol{\theta}(t_2) - \boldsymbol{\theta}(t_1)\|_2 &\leq \frac{1}{\sqrt{p}} \cdot 4\sqrt{2}\frac{\lambda_d - \lambda_1}{\lambda_2 - \lambda_1} \cdot \sqrt{\frac{d^2-1}{Z(\mathbf{H},d)}} \log d \left(e^{-\frac{\lambda_2-\lambda_1}{4\log d}t_1} - e^{-\frac{\lambda_2-\lambda_1}{4\log d}t_2}\right) \\
&\leq \frac{1}{\sqrt{p}} \cdot \sqrt{2}(\lambda_d - \lambda_1) \cdot \sqrt{\frac{d^2-1}{Z(\mathbf{H},d)}} \cdot |t_2 - t_1| \qquad (7.26)
\end{aligned}$$

Proof. Let $\mathbf{K}(t) := [\mathbf{M}, |\Psi\rangle\langle\Psi|]$ and $\mathbf{H}_l := \mathbf{U}_{l:p}\mathbf{H}\mathbf{U}_{l:p}^\dagger$. Recall that

$$\frac{d\theta_l}{dt} = -\frac{1}{pZ(\mathbf{H},d)} \operatorname{Tr}(i\mathbf{K}(t), \mathbf{H}_l(t)), \qquad (7.27)$$

$$\begin{aligned}
|\theta_l(T) - \theta_l(0)| = |\int_{t_1}^{t_2} \frac{d\theta_l(t)}{dt} dt| &= \frac{1}{pZ}|\int_{t_1}^{t_2} \operatorname{Tr}(\mathbf{H}_l(t)\mathbf{K}(t)dt| \qquad (7.28) \\
&\leq \frac{1}{pZ}\|\mathbf{H}(t)\|_F \int_{t_1}^{t_2} \|\mathbf{K}(t)\|_F dt \qquad (7.29) \\
&\leq \frac{1}{pZ}\sqrt{\operatorname{Tr}(\mathbf{H}^2)} \int_{t_1}^{t_2} \sqrt{2}(\lambda_d - \lambda_1)e^{-\frac{\lambda_2-\lambda_1}{4\log d}t} dt \qquad (7.30) \\
&= 4\sqrt{2}\frac{\lambda_d - \lambda_1}{\lambda_2 - \lambda_1}\sqrt{\frac{d^2-1}{p^2 Z}} \log d \left(e^{-\frac{\lambda_2-\lambda_1}{4\log d}t_1} - e^{-\frac{\lambda_2-\lambda_1}{4\log d}t_2}\right) \qquad (7.31)
\end{aligned}$$

Here we use the fact that $\|\mathbf{K}\|_F \leq \sqrt{2}(\lambda_d - \lambda_1)\sqrt{1 - |\langle\Psi(t)|\Psi^\star|\rangle|^2}$ (from technical Lemma 7.8.3)

∎

We next consider the random variables $\mathbf{Y}(\boldsymbol{\theta}(t))$ for t in some interval $[0, T]$. These variables form a random field and we show a concentration inequality on the expected deviation in $\mathbf{Y}(\boldsymbol{\theta}(t))$ over different intervals. A random variable $\mathbf{X}$ is said to be *sub-gaussian* [272, Proposition 2.5.2] if its tails satisfy $\mathbf{Pr}[\mathbf{X} \geq t] \leq 2\exp\left(-t^2/K_1^2\right)$, here the quantity K_1 is defined to be the second Orlicz norm, or ψ_2 norm of $\mathbf{X}$.

Lemma 7.3.6 (Expectation of deviations in $\mathbf{Y}(\boldsymbol{\theta}(t))$**).**

$$\mathbf{Pr}[\|\mathbf{Y}(\boldsymbol{\theta}(t_2)) - \mathbf{Y}(\boldsymbol{\theta}(t_1))\|_{\mathsf{op}} > t] \leq 2\exp\left(-\frac{-t^2 Z(\mathbf{H}, d)^2}{2C_1\|\boldsymbol{\theta}(t_2) - \boldsymbol{\theta}(t_1)\|_2^2}\right) \tag{7.32}$$

for some constant C_1

Proof. We first observe that due to the Haar distribution of the unitaries U_l, $\mathbf{Y}(\boldsymbol{\theta}(t_2)) - \mathbf{Y}(\boldsymbol{\theta}(t_1))$ is distributed identically to $\mathbf{Y}(\boldsymbol{\theta}(t_2) - \boldsymbol{\theta}(t_1)) - \mathbf{Y}(0)$. For convenience we define $\delta\boldsymbol{\theta} = \boldsymbol{\theta}(t_2) - \boldsymbol{\theta}(t_1)$ in the remainder of the proof.

Define $\mathbf{Y}_l(\boldsymbol{\theta}) = \mathbf{H}_l^{\otimes 2}$; then $\mathbf{Y}(\boldsymbol{\theta}) = \frac{1}{pZ(\mathbf{H},d)}\sum_{l=1}^{p}\mathbf{Y}_l$. We consider a re-parameterization of the random variables $\mathbf{H}_l(\theta)$ by constructing random variables that are identically distributed, but are functions on a different latent probability space. Defining $\mathbf{H}_l$ as $\mathbf{U}_p \cdots \mathbf{U}_l \mathbf{H} \mathbf{U}_l^\dagger \cdots \mathbf{U}_p^\dagger$, $\mathbf{Y}$ can be rewritten as:

$$\mathbf{Y}(\boldsymbol{\theta}) = \frac{1}{pZ}\sum_{l=1}^{p}\left(e^{-i\theta_p\mathbf{H}_p} \cdots e^{-i\theta_{l+1}\mathbf{H}_{l+1}}\mathbf{H}_l e^{i\theta_{l+1}\mathbf{H}_{l+1}} \cdots e^{i\theta_p\mathbf{H}_p}\right)^{\otimes 2} \tag{7.33}$$

By the Haar randomness of $\{\mathbf{U}_l\}_{l=1}^p$, we can view $\{\mathbf{H}_l\}_{l=1}^p$ as random Hermitians generated by $\{\mathbf{V}_l\mathbf{H}\mathbf{V}_l^\dagger\}$ for *i.i.d.* Haar random $\{\mathbf{V}_l\}_{l=1}^p$. This variable is identically distributed to $\mathbf{Y}$ and $\mathbf{Y}_l$ can be defined as each term in the sum.

We will apply the well-known McDiarmid inequality that can be stated as follows: Consider independent random variables $X_1, \ldots, X_k \in \mathcal{X}$. Suppose a random variable $\phi\colon \mathcal{X}^k \to \mathbb{R}$ satisfies the condition that for all $1 \leq j \leq k$ and for all $x_1, \ldots, x_j, \ldots, x_k, x_j' \in \mathcal{X}$,

$$|\phi(x_1, \ldots, x_j, \ldots, x_k) - \phi(x_1, \ldots, x_j', \ldots, x_k)| \leq c_j \tag{7.34}$$

then the tails of the distribution satisfy

$$\mathbf{Pr}[|\phi(X_1, \ldots, X_k) - \mathbb{E}\phi| \geq t] \leq \exp\left(\frac{-2t^2}{\sum_{i=1}^k c_i^2}\right) \tag{7.35}$$

With our earlier re-parameterization we can consider $\mathbf{Y}$ and consequently $\mathbf{Y}_l$ as functions of the randomly sampled Hermitian operators $\mathbf{H}_l$. Define the variable $\mathbf{Y}^{(k)}$ as that obtained by resampling $\mathbf{H}_k$ independently, and $\mathbf{Y}_l^{(k)}$ correspondingly. Finally we define $\Delta^{(k)}\mathbf{Y} = \|(\mathbf{Y}(\delta\boldsymbol{\theta}) - \mathbf{Y}(0)) - (\mathbf{Y}^{(k)}(\delta\boldsymbol{\theta}) - \mathbf{Y}^{(k)}(0))\|_{\mathsf{op}} = \|\mathbf{Y}(\delta\boldsymbol{\theta}) - \mathbf{Y}^{(k)}(\delta\boldsymbol{\theta})\|_{\mathsf{op}}$. Via a helper technical lemma (Lemma 7.8.4 proved in the appendix) we have that for any unitary W be generated by Hermitian H ($W = \exp(-i\theta H)$), we have

$$(WKW^\dagger)^{\otimes 2} - K^{\otimes 2} \preceq 4|\theta|\|H\|_{\mathsf{op}}\|K\|_{\mathsf{op}}^2 I \tag{7.36}$$

Via the triangle inequality,

$$\Delta^{(k)}\mathbf{Y} = \|\mathbf{Y}(\delta\boldsymbol{\theta}) - \mathbf{Y}^{(k)}(\delta\boldsymbol{\theta})\| \tag{7.37}$$

$$= \frac{1}{pZ}\|\sum_{l=1}^{k}\mathbf{Y}_l(\delta\boldsymbol{\theta}) - \mathbf{Y}_l^{(k)}(\delta\boldsymbol{\theta})\| \tag{7.38}$$

$$\leq \frac{1}{pZ}\sum_{l=1}^{k}\|\mathbf{Y}_l(\delta\boldsymbol{\theta}) - \mathbf{Y}_l^{(k)}(\delta\boldsymbol{\theta})\| \tag{7.39}$$

Then by definition,

$$\|\mathbf{Y}_l(\delta\boldsymbol{\theta}) - \mathbf{Y}_l^{(k)}(\delta\boldsymbol{\theta})\| \tag{7.40}$$

$$=(e^{-i\delta\boldsymbol{\theta}_p\mathbf{H}_p}\cdots e^{-i\delta\boldsymbol{\theta}_{k+1}\mathbf{H}_{k+1}})^{\otimes 2}\big((e^{-i\delta\boldsymbol{\theta}_k\mathbf{H}_k}\mathbf{K}e^{i\delta\boldsymbol{\theta}_k\mathbf{H}_k})^{\otimes 2} - (e^{-i\delta\boldsymbol{\theta}_k\mathbf{H}'_k}\mathbf{K}e^{i\delta\boldsymbol{\theta}_k\mathbf{H}'_k})^{\otimes 2}\big)$$

$$(e^{i\delta\boldsymbol{\theta}_{l+1}\mathbf{H}_{l+1}}\cdots e^{i\delta\boldsymbol{\theta}_p\mathbf{H}_p})^{\otimes 2} \tag{7.41}$$

where $\mathbf{K} := e^{-i\delta\boldsymbol{\theta}_{k-1}\mathbf{H}_{k-1}}\cdots e^{-i\delta\boldsymbol{\theta}_{l+1}\mathbf{H}_{l+1}}\mathbf{H}_l e^{i\delta\boldsymbol{\theta}_{l+1}\mathbf{H}_{l+1}}\cdots e^{i\delta\boldsymbol{\theta}_{k-1}\mathbf{H}_{k-1}}$. By Lemma 7.8.4,

$$\|\big(\mathbf{Y}_l(\delta\boldsymbol{\theta}) - \mathbf{Y}_l(\mathbf{0})\big) - \big(\mathbf{Y}_l^{(k)}(\delta\boldsymbol{\theta}) - \mathbf{Y}_l^{(k)}(\mathbf{0})\big)\| \leq 8|\delta\boldsymbol{\theta}_k|\|\mathbf{H}\|_{\mathsf{op}}\|\mathbf{K}\|_{\mathsf{op}}^2 = 8|\delta\boldsymbol{\theta}_k|\|\mathbf{H}\|_{\mathsf{op}}^3 \tag{7.42}$$

We finally have $\Delta^{(k)}(y) \leq \frac{8|\delta\boldsymbol{\theta}_k|\|\mathbf{H}\|_{\mathsf{op}}^3}{Z}$. By the McDiarmid inequality, the result follows. □

To bound the supremum of the deviation over an entire time interval, we employ Dudley's integral inequality (stated below in it's matrix form)

Lemma 7.3.7 (Dudley's integral inequality: subgaussian matrix version (Adapted from Theorem 8.1.6 in [272])**.** *Let $\boldsymbol{\mathcal{R}}$ be a metric space equipped with a metric $\mathbf{d}(\cdot,\cdot)$, and $\mathbf{X} : \boldsymbol{\mathcal{R}} \mapsto \mathbb{R}^{D\times D}$*

with subgaussian increments ie. it satisfies

$$\mathbf{Pr}[\|\mathbf{X}(r_1) - \mathbf{X}(r_2)\|_{\mathsf{op}} > t] \leq 2D \exp\left(-\frac{t^2}{C_\sigma^2 \mathbf{d}(r_1, r_2)^2}\right) \tag{7.43}$$

Then with probability at least $1 - 2D\exp(-u^2)$ *for any subset* $\mathcal{S} \subseteq \boldsymbol{\mathcal{R}}$*:*

$$\sup_{(r_1,r_2)\in\mathcal{S}} \|\mathbf{X}(r_1) - \mathbf{X}(r_2)\|_{\mathsf{op}} \leq C \cdot C_\sigma \left[\int_0^{\operatorname{diam}(\mathcal{S})} \sqrt{\mathcal{N}(\mathcal{S}, \mathbf{d}, \epsilon)}\, d\epsilon + u \cdot \operatorname{diam}(\mathcal{S})\right] \tag{7.44}$$

for some constant C*, where* $\mathcal{N}(\mathcal{S}, \mathbf{d}, \epsilon)$ *is the metric entropy defined as the logarithm of the* ϵ*-covering number of* $\mathcal{S}$ *using metric* d*.*

We then have the following main result:

Lemma 7.3.4 (Concentration during training (time dependent))**.** *Suppose that under learning rate* $\eta = \frac{1}{pZ(\mathbf{H},d)}$*, for all* $0 \leq t \leq T$*,* $1 - |\langle\Psi|\Psi^\star|\rangle^2 \leq \exp(-\frac{(\lambda_2-\lambda_1)}{2\log d}t)$*, then with probability* $\geq 1-\delta$*, for all* $t \geq 0$*:*

$$\|\mathbf{Y}(t) - \mathbf{Y}(0)\|_{\mathsf{op}} \leq C_3 \left(\frac{T}{\sqrt{p}} \cdot \sqrt{2}(\lambda_d - \lambda_1) \cdot \sqrt{\frac{d^2-1}{Z(\mathbf{H},d)^3}} \left(1 + \sqrt{\log\left(\frac{2d}{\delta}\right)}\right)\right) \tag{7.16}$$

where C_3 *is a constant.*

Proof. Via Lemma 7.3.6,

$$\mathbf{Pr}[\|\mathbf{Y}'(\boldsymbol{\theta}(t_2)) - \mathbf{Y}'(\boldsymbol{\theta}(t_1))\|_{\mathsf{op}} > t] \leq 2\exp\left(-\frac{-t^2 Z(\mathbf{H},d)^2}{2C_1\|\boldsymbol{\theta}(t_2) - \boldsymbol{\theta}(t_1)\|^2}\right) \tag{7.45}$$

By Lemma 7.3.5 $\|\boldsymbol{\theta}(t_2) - \boldsymbol{\theta}(t_1)\|_2 \leq \frac{1}{\sqrt{p}} \cdot \sqrt{2}(\lambda_d - \lambda_1) \cdot \sqrt{\frac{d^2-1}{Z(\mathbf{H},d)}} \cdot |t_2 - t_1|$. Thus, $\mathbf{Y}'$ has subgaussian increments if we define the metric $\mathbf{d}(t_2, t_1) = \frac{1}{\sqrt{p}} \cdot \sqrt{2}(\lambda_d - \lambda_1) \cdot \sqrt{\frac{d^2-1}{Z(\mathbf{H},d)^3}} \cdot |t_2 - t_1|$, thereby satisfying the conditions for Lemma 7.3.7. Under this metric, the diameter of the interval $[0, T]$ is $\frac{T}{\sqrt{p}} \cdot \sqrt{2}(\lambda_d - \lambda_1) \cdot \sqrt{\frac{d^2-1}{Z(\mathbf{H},d)^3}}$. Applying Lemma 7.3.7, with failure probability at most δ we have

$$\sup_{t\in[0,T]} \|\mathbf{Y}(\theta(t)) - \mathbf{Y}(\theta(0))\|_{\mathsf{op}} \leq C_2 \left(\operatorname{diam}([0,T]) \left(1 + \sqrt{\log\left(\frac{2d}{\delta}\right)}\right)\right) \tag{7.46}$$

By the previous consideration,

$$\sup_{t\in[0,T]} \|\mathbf{Y}(\theta(t)) - \mathbf{Y}(\theta(0))\|_{\mathsf{op}} \leq C_3 \left(\frac{T}{\sqrt{p}} \cdot \sqrt{2}(\lambda_d - \lambda_1) \cdot \sqrt{\frac{d^2-1}{Z(\mathbf{H},d)^3}} \left(1 + \sqrt{\log\left(\frac{2d}{\delta}\right)}\right)\right) \tag{7.47}$$

where C_2, C_3 are constants. $\square$

7.3.3 Linear convergence to the ground state

Finally, we can combine our previous results to show that with sufficient overparameterization, the VQE dynamics can be made to exponentially converge to the ground state

Theorem 7.3.1 (Exponential convergence for VQE)**.** *Consider a VQE system in a d-dimensional Hilbert space (with architecture as described in Definition 7.3.1) with a target Hamiltonian $\mathbf{M}$ with eigenvalues $\lambda_1 \leq \lambda_2 \ldots \lambda_d$, generating Hamiltonian $\mathbf{H}$ and number of parameters p be greater than $\Omega\left(\left(\frac{\lambda_d-\lambda_1}{\lambda_2-\lambda_1}\right)^4, \frac{d^4}{Z(\mathbf{H},d)^3}, \log\left(\frac{2d}{\delta}\right)\right)$. Denote the ground state of the system by $|\psi^*\rangle$. Conditioned on the event that the output state at initialization $|\Psi(0)\rangle$ has non-negligible overlap with the target ground state $|\Psi^\star\rangle$: $|\langle\Psi(0)|\Psi^\star\rangle|^2 \geq O(\frac{1}{d})$, and with a learning rate of $\eta = \frac{1}{pZ(\mathbf{H},d)}$,* *the system converges to the ground state with error $\epsilon = 1 - \langle\psi|\psi^*\rangle^2$ in time $T_\epsilon = \frac{2\log d}{\lambda_2-\lambda_1}\log\left(\frac{1}{\epsilon}\right)$ with failure probability at most δ.*

Proof. By Lemma 7.3.2, if the closeness condition on $\mathbf{Y}$ is maintained for time $T_\epsilon = \frac{2\log d}{\lambda_2-\lambda_1}\log\left(\frac{1}{\epsilon}\right)$ the obtained error is less than or equal to ϵ. Therefore by Lemma 7.3.4 and Lemma 7.3.3, in order to ensure $\|\mathbf{Y}(t) - \mathbf{Y}(0)\| \leq \frac{\lambda_d-\lambda_1}{\lambda_2-\lambda_1}\cdot\frac{1}{2\sqrt{2}d}$ upto any time $0 < t \leq T_\epsilon$ such that $1 - |\langle\Psi|\Psi^\star|\rangle|^2 \leq \exp(-\frac{(\lambda_2-\lambda_1)}{2\log d}t)$ for all times less than t, it suffices to choose p such that $C_3\left(\frac{T}{\sqrt{p}}\cdot\sqrt{2}(\lambda_d-\lambda_1)\cdot\sqrt{\frac{d^2-1}{Z(\mathbf{H},d)^3}}\left(1+\sqrt{\log\left(\frac{2d}{\delta}\right)}\right)\right) \leq \frac{\lambda_2-\lambda_1}{\lambda_d-\lambda_1}\cdot\frac{1}{2\sqrt{2}d}$. By simple algebra, it can be verified that choosing $p \geq \Omega\left(\left(\frac{\lambda_d-\lambda_1}{\lambda_2-\lambda_1}\right)^4, \frac{d^4}{Z(\mathbf{H},d)^3}, \log\left(\frac{2d}{\delta}\right)\right)$ is sufficient.

Let t_0 be the minimum time such that either $1 - |\langle\Psi_{t_0}|\Psi^\star|\rangle|^2 > \exp(-\frac{(\lambda_2-\lambda_1)}{2\log d}t_0)$ or $\|\mathbf{Y}(t_0) - \mathbf{Y}(0)\| > \frac{\lambda_d-\lambda_1}{\lambda_2-\lambda_1}\cdot\frac{1}{2\sqrt{2}d}$. If $1 - |\langle\Psi|\Psi^\star|\rangle|^2 > \exp(-\frac{(\lambda_2-\lambda_1)}{2\log d}t_0)$, we must have $\|\mathbf{Y}(t_0') - \mathbf{Y}(0)\| > \frac{\lambda_d-\lambda_1}{\lambda_2-\lambda_1}\cdot\frac{1}{2\sqrt{2}d}$ at some earlier time t_0' (Lemma 7.3.2). Similarly, if $\|\mathbf{Y}(t_0) - \mathbf{Y}(0)\| > \frac{\lambda_d-\lambda_1}{\lambda_2-\lambda_1}\cdot\frac{1}{2\sqrt{2}d}$, we must have $1-|\langle\Psi_{t_0'}|\Psi^\star|\rangle|^2 > \exp(-\frac{(\lambda_2-\lambda_1)}{2\log d}t_0')$ at some earlier time t_0' (Lemma 7.3.4). Therefore by contradiction, both conditions must be realized for all times $t \leq T_\epsilon$, yielding the result. $\square$

7.4 Robust convergence under noisy gradient

So far we have assumed perfect access to the exact gradient ∇L. In the practical NISQ setting, the estimation of gradients are usually noise either due to the finite number of measurements, or the noisy implementation of circuits. In this section, we show that our convergence theorem is noise-tolerant. In the continuous time setting we consider the following definitin for noisy gradient descent:

Definition 7.4.1. *Noisy gradient flow. For loss function $L : \mathbb{R}^p \to \mathbb{R}$, the noisy gradient flow on the parameters $\boldsymbol{\theta} \in \mathbb{R}^p$ with learning rate η is defined as*

$$\frac{d\boldsymbol{\theta}}{dt} = -\eta\widehat{\nabla L} = -\eta(\nabla L + \boldsymbol{\varepsilon}) \quad or \quad \frac{d\theta_l}{dt} = -\eta(\frac{\partial L}{\partial \theta_l} + \varepsilon_l(t)) \quad \forall l \in [p] \tag{7.48}$$

where $\boldsymbol{\varepsilon}(t) := (\varepsilon_1(t), \cdots, \varepsilon_p(t))$ is the noise to the gradient estimation.

The robust version of our theorem states that when $\boldsymbol{\varepsilon}(t)$ is bounded in ℓ_∞-norm, the convergence result still holds:

Corollary 7.4.1 (Robust convergence for VQE with noisy gradient)**.** *Consider a VQE system in a d-dimensional Hilbert space (with architecture as described in Definition 7.3.1) with a target Hamiltonian $\mathbf{M}$ with eigenvalues $\lambda_1 < \lambda_2 \leq \cdots \leq \lambda_d$, generating Hamiltonian $\mathbf{H}$ and number of parameters p be greater than $\Omega\left(\left(\frac{\lambda_d-\lambda_1}{\lambda_2-\lambda_1}\right)^4, \frac{d^4}{Z(\mathbf{H},d)}, \log\left(\frac{\delta}{2d}\right)\right)$. Denote the ground state of the system by $|\Psi^\star\rangle$. Conditioned on the event that the output state at initialization $|\Psi(0)\rangle$ has non-negligible overlap with the target ground state $|\Psi^\star\rangle$: $|\langle\Psi(0)|\Psi^\star\rangle|^2 \geq O(\frac{1}{d})$, and with a learning rate of $\eta = \frac{1}{pZ(\mathbf{H},d)^3}$, the system converges under noisy gradient to the ground state with error $\epsilon = 1 - |\langle\Psi(t)|\Psi^\star\rangle|^2$ in time $T_\epsilon = \frac{4\log d}{\lambda_2-\lambda_1}\log\left(\frac{1}{\epsilon}\right)$ with failure probability at most δ, if the gradient estimation error $\|\boldsymbol{\varepsilon}(t)\|_\infty \leq \frac{Z}{2\|\mathbf{H}\|}(\lambda_2 - \lambda_1)\sqrt{1 - |\langle\Psi(t)|\Psi^\star\rangle|^2}|\langle\Psi(t)|\Psi^\star\rangle|$.*

To interpret the upper bound on $\|\boldsymbol{\varepsilon}\|_\infty$, notice that

$$\sqrt{1 - |\langle\Psi(t)|\Psi^\star\rangle|^2}|\langle\Psi(t)|\Psi^\star\rangle| \leq \max\{|\langle\Psi(t)|\Psi^\star\rangle|^2, 1 - |\langle\Psi(t)|\Psi^\star\rangle|^2\}. \tag{7.49}$$

At the initial stage of training, $\|\boldsymbol{\varepsilon}\|_\infty$ need to be $O(|\langle\Psi(t)|\Psi^\star\rangle|^2)$ so that the worst-case perturbation in the gradient does not eliminate the overlap between $|\Psi(t)\rangle$ and $|\Psi^\star\rangle$; at the final stage of training $\|\boldsymbol{\varepsilon}\|_\infty$ need to be $O(1-|\langle\Psi(t)|\Psi^\star\rangle|^2)$ to obtain solutions with high quality. We also highlight that the analysis considers the worst-case (or adversarial) gradient noise $\boldsymbol{\varepsilon}$. It is possible for the more practical scenario, the structure of $\boldsymbol{\varepsilon}$ can relax the requirement on $\|\boldsymbol{\varepsilon}\|_\infty$ (for example when the noise is purely due the finite measurements and therefore stochastic).

The following Lemma 7.4.1 calculates the dynamics at the presence of gradient noise. Corollary 7.4.1 then follows directly from Lemma 7.4.2, which states the convergence of the noisy dynamics. The proof for the lemmas are modified based on Lemma 7.3.1 and 7.3.2 and are postponed to Section 7.8.2.

Lemma 7.4.1 (Output-state dynamics with noisy gradient estimation)**.** *Consider VQE instance* $(\mathbf{M}, |\Phi\rangle, \mathbf{U})$*, with* $\mathbf{U}$ *being the ansatz defined in Definition 7.3.1. Under gradient flow with learning rate* η *and noisy gradient estimation* $\widehat{\nabla L} := \nabla L + \boldsymbol{\varepsilon}(t) = \big(\frac{\partial L}{\partial\theta_l} + \varepsilon_l(t)\big)_{l\in[p]}$*, the output state* $|\Psi\rangle$ *follow the dynamics*

$$\frac{d}{dt}|\Psi\rangle = -(\eta\cdot p\cdot Z(\mathbf{H},d))\,\mathrm{Tr}_1(\mathbf{Y}([\mathbf{M},|\Psi\rangle\langle\Psi|]\otimes\mathbf{I}_{d\times d}))|\Psi\rangle + \eta\sum_{l=1}^{p} i\varepsilon_l\mathbf{H}_l|\Psi\rangle \tag{7.50}$$

Here the Hermitian $\mathbf{Y}\in\mathbb{C}^{d^2\times d^2}$ *is the time-dependent matrix defined as*

$$\mathbf{Y}(\boldsymbol{\theta}) := \frac{1}{pZ(\mathbf{H},d)}\sum_{l=1}^{p}\big(\mathbf{U}_{l:p}(\boldsymbol{\theta}(t))\mathbf{H}\mathbf{U}^\dagger_{l:p}(\boldsymbol{\theta}(t))\big)^{\otimes 2} \tag{7.51}$$

and $\mathbf{H}_l$ *are defined as* $\mathbf{U}_{l:p}(\boldsymbol{\theta}(t))\mathbf{H}\mathbf{U}^\dagger_{l:p}(\boldsymbol{\theta}(t))$.

The following modified version of Lemma 7.3.2 implies that the main theorem holds with noisy gradient estimation:

Lemma 7.4.2 (VQE Perturbation Lemma under noisy gradients)**.** *Conditioned on the event that* *the output state at initialization* $|\Psi(0)\rangle$ *has non-negligible overlap with the target ground state* $|\Psi^\star\rangle$*:* $|\langle\Psi(0)|\Psi^\star\rangle|^2 \geq O(\frac{1}{d})$*, if for all* $t \geq 0$, $\|\mathbf{Y}(t) - \mathbf{Y}^\star(t)\|_{\mathsf{op}} \leq \frac{\lambda_2-\lambda_1}{\lambda_d-\lambda_1} \cdot \frac{1}{4\sqrt{2}d}$ *and* $\|\boldsymbol{\varepsilon}(t)\|_\infty \leq \frac{Z}{2\|\mathbf{H}\|}(\lambda_2 - \lambda_1)\sqrt{1 - |\langle\Psi(t)|\Psi^\star\rangle|^2}|\langle\Psi(t)|\Psi^\star\rangle|$*, Then under the dynamics*

$$\frac{d}{dt}|\Psi\rangle = -\operatorname{Tr}_1(\mathbf{Y}([\mathbf{M}, |\Psi\rangle\langle\Psi|] \otimes \mathbf{I}_{d\times d}))|\Psi\rangle + \frac{1}{pZ}\sum_{l=1}^{p} i\varepsilon_l \mathbf{H}_l|\Psi\rangle \tag{7.52}$$

the output states converges to the ground state:

$$1 - |\langle\Psi(t)|\Psi^\star\rangle|^2 \leq \exp(-\frac{\lambda_2 - \lambda_1}{4\log d}t). \tag{7.53}$$

7.5 Ansatz-dependent results

7.5.1 Ansatz-dependent upperbound

The bounds obtained in Theorem 7.3.1 provide sufficient conditions on the number of classical parameters required to ensure that an instance of the Partially Trainable Ansatz over $SU(d)$ (Definition 7.3.1) converges to its ground state with a linear rate of convergence. These bounds are primarily influenced by the dimension d of the Hilbert Space, and a quantity $\frac{\lambda_d-\lambda_1}{\lambda_2-\lambda_1}$ influenced by the conditioning of the spectrum (which we will call the spectral ratio in the following) of the problem Hamiltonian $\mathbf{M}$. In this section, we provide ansatz-dependent upperbounds on over-parameterization. We show that the choice of parameterized ansatz can result in the sufficient overparameterization threshold being governed by quantities the effective dimension d_{eff} and spectral ratio κ_{eff} which can be significantly smaller than d and κ. In the later section 7.7, we evaluate these two quantities for popular Hamiltonian variational ansatz (HVA) and use them to explain the success of HVA over general ansatz designs such as HEA.

Group Representations To present our result for specifc ansatze, we first state some elementary results from group representation theory (for a full introduction to representations of Lie Groups see [273, Chapter 4]). A finite-dimensional representation (V, Π) of a group G is specified by a vector space V and a group homomorphism $\Pi\colon G \to GL(V)$, such that $\Pi(g_1)\Pi(g_2) = \Pi(g_1 g_2)$ for all $g_1, g_2 \in G$. The representation is *unitary* if $\Pi(g)$ is unitary for all $g \in G$. An important concept in the group representation theory is *irreducibility*. Given a representation (V, Π) of G, a subspace $W \subseteq V$ is said to be *invariant* if $\Pi(g)w \in W$ for all $w \in W$ and $g \in G$. A representation is further said to be *irreducible* if it has no invariant subspaces other than the trivial subspaces consisting of the empty set $\varnothing$ and the whole space V.

For a unitary matrix Lie group G defined on V, the simple identity map furnishes a unitary representation of G on the Hilbert space G. We will refer to this representation as the *natural* representation. In this case we will not distinguish the group element from its representation. The following proposition indicates that a unitary matrix group G induces a decomposition of the ambient space V.

Proposition 7.5.1 (Adapted from [273, Proposition 4.27])**.** *Let G be a group with* unitary *representation Π acting on a vector space V. Then this representation is* completely reducible *ie. V is isomorphic to a direct sum $V_1 \oplus \cdots \oplus V_m$ where each V_i is an invariant subspace which itself has no non-trivial invariant subspaces.*

For ansatz design $\mathcal{A}$, let $G_{\mathcal{A}}$ be the associated subgroup of $SU(d)$ as defined in Section 4.2. $G_{\mathcal{A}}$ is a unitary matrix subgroup defined on the state space $\mathcal{H}$. By Proposition 7.5.1, the natural representation of $G_{\mathcal{A}}$ induces a decomposition of the state space $\mathcal{H} = V_1 \oplus \cdots \oplus V_m$. We now define the ansatz compatibility and the key quantities d_{eff} and κ_{eff} for a VQE instance $(\mathbf{M}, |\Phi\rangle, \mathbf{U})$ using this decomposition.

Definition 7.5.1 (Compatible Ansatz)**.** *Consider an VQE instance* $(\mathbf{M}, |\Phi\rangle, \mathbf{U})$ *with ansatz design* $\mathcal{A}$*. Let* $\mathcal{H} = V_1 \oplus \cdots \oplus V_m$ *be the completely-reduced decomposition induced by the ansatz design* $\mathcal{A}$ *through the natural representation of* $G_{\mathcal{A}}$ *and Let* $|\Psi^{\star}\rangle$ *denote the ground state of* $\mathbf{M}$*. The ansatz design* $\mathcal{A}$ *is said to be compatible with the VQE problem if there exists* $j \in [m]$ *such that both* $|\Phi\rangle$ *and* $|\Psi^{\star}\rangle$ *lie within an invariant subspace* V_j*.*

The effective quantities for compatible ansatz can be defined using the invariant subspace:

Definition 7.5.2 (Effective dimension d_{eff} and Effective ratio κ_{eff})**.** *Consider an VQE instance* $(\mathbf{M}, |\Phi\rangle, \mathbf{U})$ *with compatible ansatz design* $\mathcal{A}$*. And let* W *denote the invariant subspace where the input and the ground state lies with projection* $\mathbf{\Pi} = \mathbf{Q}\mathbf{Q}^{\dagger}$*. The effective dimension* d_{eff} *is defined as the dimension of* W*. The effective spectrum is defined as the ordered eigenvalues* $(\lambda'_1, \cdots, \lambda'_{d_{\text{eff}}})$ *of the Hermitian* $\mathbf{Q}^{\dagger}\mathbf{M}\mathbf{Q}$*. The effective spectral ratio* κ_{eff} *is defined as* $\frac{\lambda'_{d_{\text{eff}}} - \lambda'_1}{\lambda'_2 - \lambda'_1}$*. The effective generating Hamiltonian* $\mathbf{H}_{\text{eff}}$ *is defined as* $\mathbf{Q}^{\dagger}\mathbf{H}\mathbf{Q}$

Given the projection $\mathbf{\Pi}$ onto W, the decompostion $\mathbf{Q}$ is not unique but allow a $d_{\text{eff}} \times d_{\text{eff}}$ unitary transformation. This does not introduce any ambiguity in the definition of the effective spectrum as unitary transformations does not change the eigenvalues of $\mathbf{Q}^{\dagger}\mathbf{M}\mathbf{Q}$.

The Killing-Cartan classification indicates that the subgroup $G_{\mathcal{A}}$ restricted on the invariant subspace Wj must be one of the simple lie groups. Here we focus on the case where the subgroup $G_{\mathcal{A}}$ restricted on the invariant subspace Wj is a special unitary group $SU(d_{\text{eff}})$. Similar results can be proved for special orthogonal, symplectic group by replacing the integral forumla (e.g. See [271]).

By definition W is invariant under the action of any operator represented by ansatz $\mathcal{A}$, indicating the dynamics of the output state is restricted to the subspace V_j spanned by the column space of $\mathbf{Q}$. By transforming all the Hamiltonians and the input state by $\mathbf{Q}$ in the proof of Theorem 7.3.1, we have the following corollary:

Corollary 7.5.1. *Let* $(\mathbf{M}, |\Phi\rangle, \mathbf{U})$ *be a VQE instance using compatible ansatz design* $\mathcal{A}$ *with* d_{eff}*,* κ_{eff}*,* $\mathbf{H}_{\text{eff}}$ *and* $(\lambda'_1, \cdots, \lambda'_{d_{\text{eff}}})$ *as defined in Definition 7.5.1. Let* $|\Psi^\star\rangle$ *denote the ground state of* $\mathbf{M}$ *and* $|\Psi(t)\rangle$ *the output state at time* t*, then conditioned on the event that the output state at initialization* $|\Psi(0)\rangle$ *has non-negligible overlap with the target ground state* $|\Psi^\star\rangle$*:* $|\langle\Psi(0)|\Psi^\star\rangle|^2 \geq O(\frac{1}{d_{\text{eff}}})$*, and with a learning rate of* $\eta = \frac{1}{pZ(\mathbf{H}_{\text{eff}}, d_{\text{eff}})}$*, the system converges to the ground state with error* $\epsilon = 1 - \langle\Psi(t)|\Psi^\star\rangle^2$ *in time* $T_\epsilon = \frac{2\log d_{\text{eff}}}{\lambda'_2 - \lambda'_1}\log\left(\frac{1}{\epsilon}\right)$ *with failure probability at most* δ*, if the number of parameters* p *is greater than* $\Omega\left(\left(\kappa_{\text{eff}}^4, \frac{d_{\text{eff}}^4}{Z(\mathbf{H}_{\text{eff}}, d_{\text{eff}})^3}, \log\left(\frac{2d_{\text{eff}}}{\delta}\right)\right)\right)$.

For general ansatz design $\mathcal{A}$ including HEA with $G_{\mathcal{A}} = SU(d)$, the effective dimension d_{eff} (resp. effective ratio κ_{eff} is the same as the system dimension d (resp. the ratio $\kappa = \frac{\lambda_d - \lambda_1}{\lambda_2 - \lambda_1}$). In fact this holds for fully-trainable ansatze that contain universal gate sets and satisfy the premise of [270]. On the other hand, a problem-specific compatible ansatz design can have much smaller d_{eff} and κ_{eff} and achieve reasonable performance with much fewer number of parameters. As we see in Section 7.7 For physical problems like Transverse field ising model and Heinsenberg model, certain HVA designs can achieved d_{eff} and κ_{eff} orders of magnitudes smaller than d and κ.

7.5.2 Estimating d_{eff} and κ_{eff}

Given VQE problem $(\mathbf{M}, |\Phi\rangle, \mathbf{U})$ with a compatible ansatz design $\mathcal{A}$, we can estimate the column space of $\mathbf{Q}$ of the invariant subspace by estimating the support of the matrix

$$\hat{\mathbf{\Pi}} = \frac{1}{R}\sum_{r=1} \mathbf{U}_r|\Phi\rangle\langle\Phi|\mathbf{U}_r^\dagger \tag{7.54}$$

with $\mathbf{U}_r$ sampled $i.i.d.$ from the Haar measure over $G_{\mathcal{A}}$. Empirically we approximate the Haar measure over $G_{\mathcal{A}}$ by calculating

$$\mathbf{U}(\boldsymbol{\phi}) = \prod_{l'=1}^{L_{\mathsf{sample}}}\prod_{k=1}^{K}\exp(-i\phi_{l',k}\mathbf{H}_k) \tag{7.55}$$

for sufficiently large L_{sample} and randomly initialized $\{\phi_{l',k}\}_{k\in[K],l'\in[L_{\mathsf{sample}}]}$. Any orthonormal basis of the support of $\mathbf{\Pi}$ can be used as $\mathbf{Q}$ to estimate d_{eff} and κ_{eff} using Definition 7.5.2.

Example: Kitaev Model For a concrete example, consider the HVA for the Kitaev model on square-octagon lattice with external field introduced in [3]. We will see that the proper ansatz design leads to an effective dimension much smaller than the system dimension ($d_{\mathsf{eff}} = 76$ v.s. $d = 256$) and that the effective ratio κ_{eff} can be orders of magnitudes smaller than $\frac{\lambda_d-\lambda_1}{\lambda_2-\lambda_1}$ (Figure 7.3).

The problem Hamiltonian for Kitaev models with external field is defined as

$$\mathbf{M}_{\mathsf{Kitaev}}(J_{xy}, h) = \sum_{(u,v)\in S_Z} Z_u Z_v + \frac{J_{xy}}{\sqrt{2}} \cdot \Big(\sum_{(u,v)\in S_X} X_u X_v + \sum_{(u,v)\in S_Y} Y_u Y_v \Big) + h \cdot \sum_{i=0}^{7} X_i + Y_i + Z_i \tag{7.56}$$

with X_i denoting the Pauli-X matrix acting on the i-th qubit. This system has coupling in the X, Y, Z directions on edge sets S_X, S_Y and S_Z respectively. The parameter J_{xy} controls the coupling in the X/Y-direction and h controls the strength of the external field. For 8-qubit Kitaev models on square-octagon lattice, by labeling each qubit with indexes 0 through 7, the edge sets are defined as $S_X = \{(0,1),(2,3)\}$, $S_Y = \{(1,2),(0,3)\}$, and $S_Z = \{(4,0),(1,5),(3,7),(2,6)\}$ (See Figure 7.1 or Figure 1(c) in [3]).

We use the ansatz proposed in [3]: the ansatz design $\mathcal{A} = \{\mathbf{H}_1, \cdots, \mathbf{H}_6\}$ with

$$\begin{aligned} &\mathbf{H}_1 \propto \sum_{(u,v)\in S_X} X_u X_v, \mathbf{H}_2 \propto \sum_{(u,v)\in S_Y} Y_u Y_v, \mathbf{H}_3 \propto \sum_{(u,v)\in S_Z} Z_u Z_v, \\ &\mathbf{H}_4 \propto \sum_{i=0}^{7} X_i, \mathbf{H}_5 \propto \sum_{i=0}^{7} Y_i, \mathbf{H}_6 \propto \sum_{i=0}^{7} Z_i \end{aligned} \tag{7.57}$$

In Figure 7.2, we plot the eigenvalues of $\hat{\mathbf{\Pi}}$ for the Kitaev models for input state $|\Phi\rangle = |0\rangle^{\otimes 8}$ and the ansatz specified in Eq (7.57) using $R = 100$ and $L_{\mathsf{sample}} = 20$. As the number of samples R increases from 0 to 100, $\hat{\mathbf{\Pi}}$ converges to a matrix with uniform eigenvalues. Figure 7.2 indicates that the $|\Phi\rangle$ lies within the 76-dimensional invariant subspace W embedded in a 256-dimensional state space $\mathcal{H}$. It is also verified that the ground state of $\mathbf{M}_{\mathsf{Kitaev}}$ lies within the subspace W as well. We also compare the effective ratio κ_{eff} with $\kappa = \frac{\lambda_d - \lambda_1}{\lambda_2 - \lambda_1}$ (i.e. the effective ratio for generic ansatz designs) for a wide range of parameters (J_{xy}, h) in Figure 7.3. We observe that the HVA

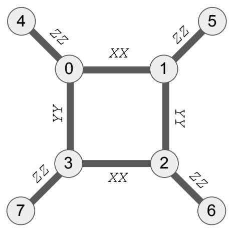


Figure 7.1: Configuration of the 8-qubit Kitaev model on square-octagon lattice defined in [3]. Qubits are labeled by $0, 1, \cdots, 7$, and each edge corresponds to an interation term. The types of interactions XX, YY and ZZ are as specified in texts.

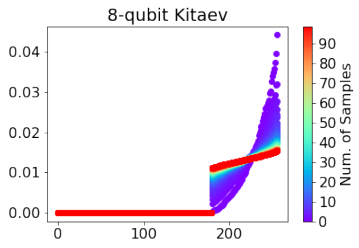
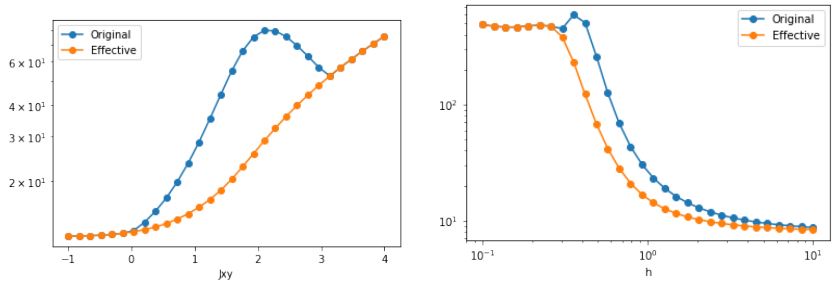


Figure 7.2: Specturm of $\hat{\mathbf{\Pi}}$ for 8-qubit Kitaev model with 8 qubits for number of samples $R = 1, 2, \cdots, 100$. As the number of samples increases, $\hat{\mathbf{\Pi}}$ converges to Hermitians with uniform spectrum, and can thus be good approximation of the normalized projection to the invariant subspace V.



(a) Kitaev Model: N=8, varying J_{xy} (b) Kitaev Model: N=8, varying h

Figure 7.3: The spectral ratio κ_{eff} for 8-qubit Kitaev models by varying J_{xy} while fixing the external field $h = 1$ and varying h while fixing $J_{xy} = 1$. The effective ratio is significantly smaller than the actual ratio for a wide range of (J_{xy}, h).

proposed in [3] decreases κ_{eff} by orders of magnitudes.

7.6 Empirical Study: Verifying the Convergence Theory

In this section we present two sets of numerical simulations to corroborate our theoretical results.

1. In Section 7.6.1 we first calculate the deviation of $\mathbf{Y}$ and $\boldsymbol{\theta}$ for HVAs and HEAs. We show that Lemma 7.3.4 and 7.3.5 correctly predict the maximal deviation of $\mathbf{Y}$ and $\boldsymbol{\theta}$ for both

the partially-trainable and the fully-trainable settings.

2. In Section 7.6.2, we confirm that the over-parameterization threshold is positively correlated to the proposed quantities κ_{eff} and d_{eff} as predicted in Theorem 7.3.1 and Corollary 7.5.1 using synthetic toy VQE examples.

7.6.1 Experiment 1: Deviation of key quantities during training

In this section we optimize Hamiltonian variational ansatz (HVA) and Hardware-efficient ansatz (HEA) in both the partially- and fully-trainable settings and evaluate $\mathbf{Y}$ and $\boldsymbol{\theta}$ during training. Recall that $\mathbf{Y}$ is a function of time step t through its dependency on the parameters $\boldsymbol{\theta}(t)$. Lemma 7.3.5 and 7.3.4 predict that $\boldsymbol{\theta}$ remains in a ℓ_∞-ball centered at $\boldsymbol{\theta}(0)$ with radius $O(1/p)$ throughout training, and that $\|\mathbf{Y}(t) - \mathbf{Y}(0)\|_{\text{op}} = O(\frac{1}{\sqrt{p}})$. Our experiments show it is true for both partially- and fully-trainable HVAs and HEAs.

Definition of Y for fully-trainable ansatz For partially-trainable ansatz defined in Definition 7.2.2, $\mathbf{Y}$ can be equivalently expressed as

$$\mathbf{Y}(\boldsymbol{\theta}) := \frac{1}{p}\sum_{l=1}^{p}\left(\mathbf{U}_{l,+}(\boldsymbol{\theta})\mathbf{H}\mathbf{U}_{l,+}^\dagger(\boldsymbol{\theta})\right)^{\otimes 2} \tag{7.58}$$

using $\mathbf{U}_{l,+}(\boldsymbol{\theta}) = \left(\prod_{l'=l+1}^{p}\mathbf{U}_{l'}\exp(-i\theta_{l'}\mathbf{H})\right)\cdot\mathbf{U}_l$, the matrix applied to the input state after the rotation $\exp(-i\theta_l\mathbf{H})$. Similarly for fully-trainable ansatz, define $\mathbf{U}_{(l,k),+}(\boldsymbol{\theta})$ as $\prod_{l'=l+1}^{L}\prod_{k=1}^{K}\exp(-i\theta_{l',k}\mathbf{H}_k)\cdot \prod_{k'=k+1}^{K}\exp(-i\theta_{l,k'}\mathbf{H}_{k'})$ as the matrix applied to the input state after the rotation $\exp(-i\theta_{l,k}\mathbf{H}_k)$,

$\mathbf{Y}$ can be defined as:

$$\mathbf{Y}(\boldsymbol{\theta}) := \frac{1}{p}\sum_{l=1}^{L}\sum_{k=1}^{K}\left(\mathbf{U}_{(l,k),+}(\boldsymbol{\theta})\mathbf{H}_k\mathbf{U}^{\dagger}_{(l,k),+}(\boldsymbol{\theta})\right)^{\otimes 2} \tag{7.59}$$

Here $p = K \cdot L$ is the total number of trainable parameters.

HVA for Transverse Field Ising Models For HVA, we consider N-site one-dimensional transverse field Ising models (TFI1d). The problem Hamiltonian is defined as

$$\mathbf{M}_{\mathsf{TFI1d}}(g) = \sum_{i=0}^{N-1} X_i X_{i+1} + g\sum_{i=0}^{N-1} Z_i \tag{7.60}$$

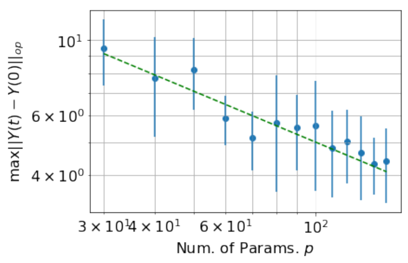
with periodic boundary (i.e the N-th site is identified with the 0-th site). The parameter g is the strength of the transverse field. In this experiment we choose the input state $\frac{1}{\sqrt{2^N}}(1, 1, \cdots, 1)^T$ and the compact HVA for TFI1d model proposed in [268] with $K = 2$ and

$$\mathbf{H}_1 \propto \sum_{i=0}^{N-1} X_i X_{i+1}, \quad \mathbf{H}_2 \propto \sum_{i=0}^{N-1} Z_i \tag{7.61}$$

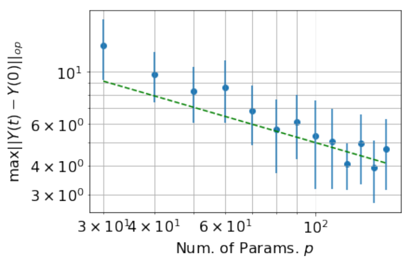
For all the experiments, $\{\mathbf{H}_k\}_{k=1}^{K}$ are normalized such that $Z(\mathbf{H}, d) = \mathrm{Tr}(\mathbf{H}_k^2)/(d^2 - 1) = 1$.

For both partially- and fully-trainable settings, we solve 4-qubit TFI1d model with external field $g = 0.3$ using gradient descent with learning rate $1 \times 10^{-4}/p$, where the numbers of trainable parameters are varied from 30 to 150. For each p the trainings are repeat over 20 random initializations.

During training The deviations as a function of time t are plotted We calculate the deviation of $\mathbf{Y}$ and $\boldsymbol{\theta}$ during trraining for solving 4-qubit TFI1d model with transverse field $g = 0.3$ for both



(a) Partially-trainable HVA



(b) Fully-trainable HVA

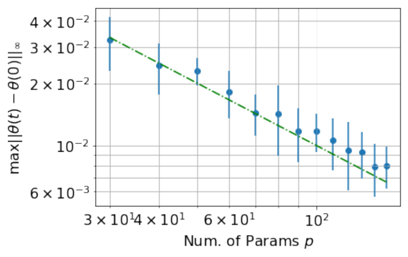
Figure 7.4: Maximal deviation of $\mathbf{Y}$ from initial value as a function of number of trainable parameters during the training of HVA for 4-qubit TFI1d model with transverse field $g = 0.3$. The mean values and the standard deviations are calculated over 20 random initializations for each number of trainable parameters $p = 30, 40, \cdots, 150$. In both figures, the reference lines $50/\sqrt{p}$ are plotted in green, showing that our theory correctly predicts the $O(1/\sqrt{p})$-dependency of $\max_{t\geq 0}\|\mathbf{Y}(t) - \mathbf{Y}(0)\|_{\mathsf{op}}$ for both settings.

partially- and fully-trainable HVAs. For the number of trainable parameters $p = 30, 40, \cdots, 150$, the partially- (fully-)trainable ansatze are optimized using gradient descent with learning rate $1 \times 10^{-4}/p$.

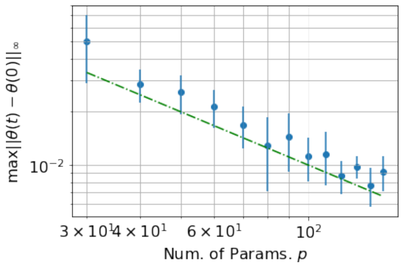
For both settings, the deviations of $\mathbf{Y}$ in operator norm ($\|\mathbf{Y}(t) - \mathbf{Y}(0)\|_{\mathsf{op}}$) saturate after a few iterations (Figure 7.16 in the appendix), and $\max_{t\geq 0}\|\mathbf{Y}(t) - \mathbf{Y}(0)\|_{\mathsf{op}}$ displays an $O(1/\sqrt{p})$ dependency on p 7.4. Moreover, note that in both Figure 7.4(a) and (b) the reference lines plotted in green are $50/\sqrt{p}$. This indicate that the max deviation of $\mathbf{Y}$ in the two settings not only match in the dependency on p but also on constants.

Similarly, the $O(1/p)$ dependencies of $\max_{t\geq 0}\|\boldsymbol{\theta}(t) - \boldsymbol{\theta}(0)\|_\infty$ are demonstrated in Figure 7.5.

HEA with CZ entanglement Similar observations occur in hardware-efficient ansatz (HEA) with layers of single-qubit X/Y-rotations and CZ entanglements. For an N-qubit instance, let CZ_{ij} denote the CZ gate acting on the i-th and j-th qubits, we define the CZ entanglement layer



(a) Partially-trainable HVA



(b) Fully-trainable HVA

Figure 7.5: Maximal deviation of $\boldsymbol{\theta}$ from initial value as a function of number of trainable parameters during the training of HVA for 4-qubit TFI1d model with transverse field $g = 0.3$. The mean values and the standard deviations are calculated over 20 random initializations for each number of trainable parameters $p = 30, 40, \cdots, 150$. In both figures, the reference lines $1/p$ are plotted in green, showing that our theory correctly predicts the $O(1/p)$-dependency of $\max_{t\geq 0}\|\boldsymbol{\theta}(t) - \boldsymbol{\theta}(0)\|_\infty$ for both settings.

$\mathbf{U}_{\mathsf{CZ}}$ as:

$$\mathbf{U}_{\mathsf{CZ}} = \prod_{\mathsf{even}\, i\in[N]} CZ_{i,i+1} \prod_{\mathsf{odd}\, i\in[N]} CZ_{i,i+1} \tag{7.62}$$

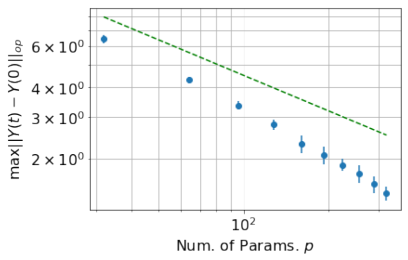
Using that fact that CZ_{ij}^2 is identity for any pair of (i,j), the HEA can be fit into the ansatz defined in Def 7.2.1 with $K = 4N$ and

$$\mathbf{H}_{4i+1} \propto X_i, \quad \mathbf{H}_{4i+2} \propto Y_i, \quad \mathbf{H}_{4i+3} \propto \mathbf{U}_{\mathsf{CZ}} X_i \mathbf{U}_{\mathsf{CZ}}, \quad \mathbf{H}_{4i+4} \propto \mathbf{U}_{\mathsf{CZ}} Y_i \mathbf{U}_{\mathsf{CZ}} \quad \forall i \in [N] \tag{7.63}$$

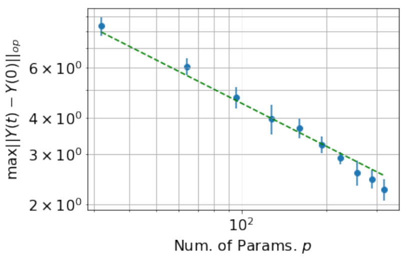
We use the ansatz defined in Eqn (7.63) to solve problem Hamiltonian

$$\mathbf{M}_{\mathsf{HEA}} = \mathsf{diag}(0, 0.5, 1, \cdots, 1) \tag{7.64}$$

with input state $|\Phi\rangle = |0\rangle^{\otimes N} = (1.0, 0, \cdots, 0)^\dagger$ and learning rate $1 \times 10^{-2}/p$. The empirical results are summarized in Figure 7.6 and 7.7.

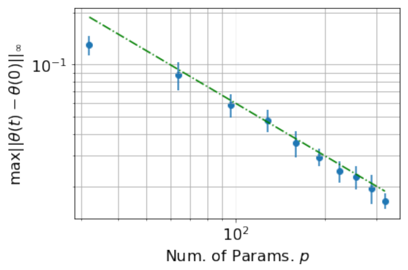


(a) Partially-trainable HEA

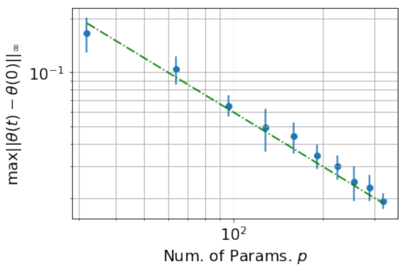


(b) Fully-trainable HEA

Figure 7.6: Maximal deviation of $\mathbf{Y}$ in 4-qubit hardware-efficient ansatz (HEA) with CZ entanglement. The mean values and the standard deviations are calculated over 10 random initializations for each number of trainable parameters $p = 32, 64, \cdots, 320$. In both figures, the reference lines are $45/\sqrt{p}$.

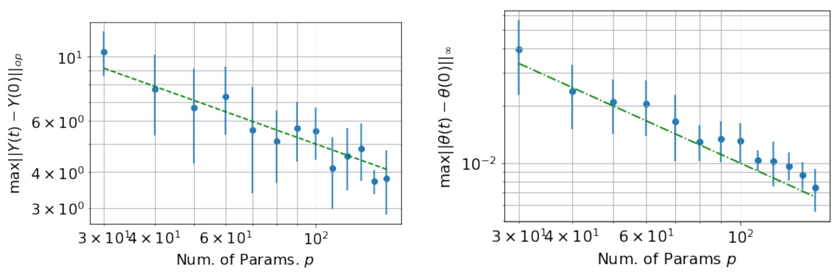


(a) Partially-trainable HEA



(b) Fully-trainable HEA

Figure 7.7: Maximal deviation of $\boldsymbol{\theta}$ in 4-qubit HEA with CZ entanglement. The mean values and standard deviations are calculated over 10 random initialization for each p varying from 32 to 320. The references lines in both figures are $6/p$.



(a) Deviation of $\mathbf{Y}$ with noisy gradient (b) Deviation of $\boldsymbol{\theta}$ with noisy gradient

Figure 7.8: Maximal deviation of $\mathbf{Y}$ and $\boldsymbol{\theta}$ in HVA for 4-qubit TFI1d model with transverse field $g = 0.3$. The references lines for $\mathbf{Y}$ in both figures are $50/\sqrt{p}$. The references lines for $\boldsymbol{\theta}$ in both figures are $1/p$.

We also extend our experiments to the setting when the gradient estimation is noisy. In Figure 7.8, we consider $\varepsilon_l(t)$ sampled $i.i.d.$ from $\mathcal{N}(0, 1\times10^{-5})$ for all t and $l \in [p]$, and have similar observation on the dependency of the maximal deviation of $\mathbf{Y}$ and $\boldsymbol{\theta}$ on p.

7.6.2 Experiment 2: Over-parameterization for toy models

In this section, we simulate gradient descent in toy VQE problems with varying d, d_{eff} and κ_{eff} using ansatze with different number of parameters. we show that the over-parameterization thresholds are positively correlated to the effective dimensions d_{eff} and spectral ratios κ_{eff} as predicted in Corollary 7.5.1.

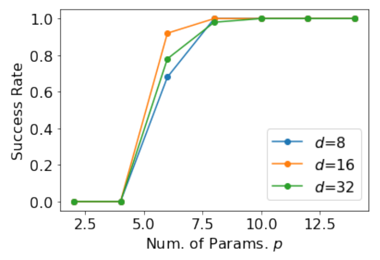
Estimating Over-parameterization Threshold For a concrete criterion of over-parameterization, we estimate the success rate for the training to converge to an output state $|\Psi(t)\rangle$ such that the error $1-|\langle\Psi(t)|\Psi^\star|\rangle^2$ is less than 0.01, where $|\Psi^\star\rangle$ is the ground state. We define the over-parameterization threshold as the smallest p such that $1-|\langle\Psi(t)|\Psi^\star|\rangle^2 > 0.01$ with probability $\geq 98\%$ over random initialization.

For physical problems like TFI1d, the system dimension d, effective dimension d_{eff} and

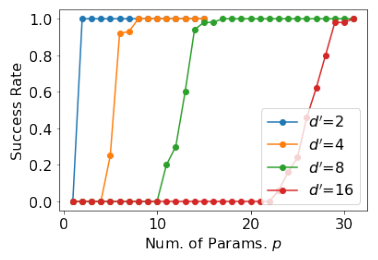
κ_{eff} are jointly defined by the number of qubits and the system parameters, be it external fields or the strengths of coupling. We decouple these parameters by starting with toy problems. For a toy problem with $(d, d_{\text{eff}}, \kappa_{\text{eff}})$, we embed a $d_{\text{eff}} \times d_{\text{eff}}$ Hermitian with eigenvalues $(0, \frac{1}{\kappa_{\text{eff}}}, 1, \cdots, 1)$ into a d-dimensional space and consider ansatze with rotations restricted to the d_{eff}-dimensional space (see Section 7.9 for the concrete definition for the toy problems). For each set of $(d, d_{\text{eff}}, \kappa_{\text{eff}})$ and each number of trainable parameters p, the training is repeated over 100 random initializations with learning rate $1 \times 10^{-2}/p$.

In Figure 7.9 we examine how the convergence of HVAs depends on the number of parameters p for toy instances with varying $(d, d_{\text{eff}}, \kappa_{\text{eff}})$: In Figure 7.9(a) we change the system dimension d with d_{eff} and κ_{eff} fixed. For all $d = 8, 16, 32$, the over-parameterization threshold is around 8, showing that the convergence is almost independent of the system dimension for fixed d_{eff} and κ_{eff}. In Figure 7.9(b), we fix the system dimension $d = 16$, $\kappa_{\text{eff}} = 4.0$ and vary the effective dimension d_{eff}: the over-parameterization threshold increases as the effective dimension increases; For a more quantitative evaluation, we define the over-parameterization threshold as the smallest p to achieve a success rate of at least 98%, and plot the threshold for different d_{eff} in Figure 7.10(a). The dependency of the over-parameterization thresholds on κ_{eff} are displayed in a similar way in Figure 7.9(c) and Figure 7.10(b). It is clearly reflected in Figure 7.10 (a) and (b) that the over-parameterization threshold is positively correlated to d_{eff} and κ_{eff}.

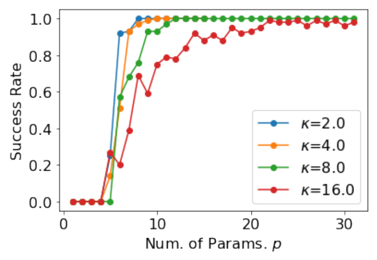
Remark 7.6.1. *Readers may notice that the dependency on d_{eff} is almost linear, seemingly contradicting previous empirical observation in [274]. There are several factors that may have contributed to this discrepancy. (1) The most prominent factor is the statistical error due to the finite number of random initializations. (2) Another plausible reason is that the ratio κ_{eff} is a*



(a) Varying system dimension d



(b) Varying effective dimension d_{eff}



(c) Varying condition number κ_{eff}

Figure 7.9: Dependency of the over-parameterization threshold on system dimension d, effective dimension d_{eff} and the condition number κ_{eff} in toy problems: the x-axes are the numbers of trainable parameters p, and the y-axes are the success rates for finding solutions with error less than 0.01. For each data point, the success rate is estimated over 100 random initializations. (a) Fixing $d_{\mathsf{eff}} = 4, \kappa_{\mathsf{eff}} = 4.0$, the over-parameterization threshold does not depend on the system dimension for $d = 8, 16, 32$. (b) Fixing $d = 16, \kappa_{\mathsf{eff}} = 4.0$ for $d_{\mathsf{eff}} = 2, 4, 6, 8$. The threshold increases as the system dimension increases. (c) Fixing $d = 16, d_{\mathsf{eff}} = 4$ for $\kappa_{\mathsf{eff}} = 2.0, 4.0, 8.0, 16.0$. The threshold is positively correlated to the condition number of the system.

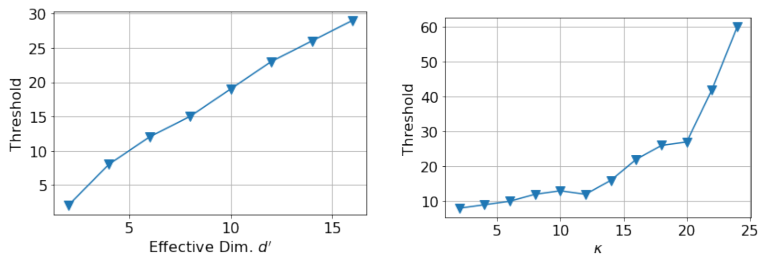
concise but inexact descriptor of all the eigenvalues of the problem Hamiltonian. As the effective dimension varies, the eigenvalues also vary in spite of the controlled κ_{eff}. (3) The third reason is that the over-parameterization threshold are defined differently in [274].

7.7 Empirical Study: Predicting Ansatz Performance

In this section, we use Corollary 7.5.1 to explain the performances of different ansatze for Ising models and Heisenberg models by (1) calculating κ_{eff} and d_{eff} using the procedure described in Subsection 7.5.2 and (2) estimating the over-parameterization thresholds. The results are summarized as follows:

- For Transverse field Ising (TFI) model, we compare ansatze TFI_2 and TFI_3 (defined later). TFI_2 and TFI_3 have identical κ_{eff}, but TFI_2 has smaller d_{eff}. Empirically, we observe TFI_2 reaches over-parameterization with fewer number of parameters.

- For the Heisenberg XXZ model, we compare ansatze XXZ_4 and XXZ_6 (defined later).



(a) Varying effective dimension d_{eff} (b) Varying condition number κ_{eff}

Figure 7.10: Dependency of the over-parameterization threshold on effective dimension d_{eff} and the condition number κ_{eff} in toy problems: the over-parameterization thresholds in the plots are defined as the smallest p achieving a success rate $\geq 98\%$ to find a solution with error less than 0.01. Success rates are estimated over 100 random initializations. (a) Fixing $d = 16, \kappa_{\text{eff}} = 4.0$ for $d_{\text{eff}} = 2, 4, 6, \cdots, 16$. The threshold increases as the system dimension increases. (b) Fixing $d = 16, d_{\text{eff}} = 4$ for $\kappa_{\text{eff}} = 2.0, 4.0, 6.0, \cdots, 26.0$. The threshold is positively correlated to the condition number of the system.

XXZ_4 and XXZ_6 have d_{eff} of same order of magnitude, but the κ_{eff} of XXZ_6 diverges at the critical point while κ_{eff} of XXZ_4 remain bounded. Empirically we observe as the system approaches the level-crossing point, XXZ_6 requires significantly more number of parameters to obtain a reasonable solution.

- For both TFI and XXZ models and all HVA considered, d_{eff} is much smaller than the system dimension d. Also for TFI_2, TFI_3, XXZ_4, the effective ratio κ_{eff} remain bounded near level-crossings where $\kappa = \frac{\lambda_d - \lambda_1}{\lambda_2 - \lambda_1}$ approaches infinity. This explains why problem-specifc HVA can be used to solve VQEs that can not be efficiently solved by generic ansatz like HEA ([268]) (Recall that for typical HEA design, d_{eff} is the system dimension d and κ_{eff} is simply κ).

These observations demonstrate the predicting power of the quantities d_{eff} and κ_{eff} and highlight that good, problem-specific ansatz designs are crucial to the efficient training of VQE in practice.

1-d TFI problems For 1d TFI models, in addition to HVA with 2-alternating Hermitian mentioned in Section 7.6.1 Eq 7.61 (which we will now refer to as TFI_2), we consider variaitonal ansatz TFI_3 containing 3 Hermitians $\mathcal{A} = \{\mathbf{H}_1, \mathbf{H}_2, \mathbf{H}_3\}$ with

$$\mathbf{H}_{xx1} \propto \sum_{\text{even}\,i} X_i X_{i+1}, \quad \mathbf{H}_{xx2} \propto \sum_{\text{odd}\,i} X_i X_{i+1}, \quad \mathbf{H}_z \propto \sum_{i=0}^{N-1} Z_i \tag{7.65}$$

Compared with TFI_2, TFI_3 decouples the odd and even coupling in the X direcion. The effective dimension d_{eff} of TFI_2 and TFI_3 for $N = 4, 6, 8, 10$ are summarized in Table 7.1: both ansatz design achieves small effective dimension compared with the system dimension d, and the effective dimension d_{eff} for TFI_2 is consistently smaller than that of TFI_3 for different N's.

N	4	6	8	10
d	16	64	256	1024
TFI_2	4	8	16	32
TFI_3	5	10	25	50

Table 7.1: System dimension d for N-qubit TFI1d models with $N = 4, 6, 8, 10$ and effective dimension d_{eff} for variational ansatze TFI_2 and TFI_3.

Despite the difference in d_{eff}, TFI_2 and TFI_3 has similar κ_{eff}: in Figure 7.11, we visualize the eigenvalues and κ_{eff} of TFI_2, TFI_3 and the original problem Hamiltonian $\mathbf{M}_{\text{TFI1d}}(g)$ with varying transverse field g for 6-qubit 1d TFI models. In Figure 7.11 (a) and (b), we plot the 4 smallest eigenvalues of the effective Hamiltonian $\mathbf{M}'$ associated with TFI_2 and TFI_3: while TFI_2 and TFI_3 have different effective dimensions, they have similar eigenvalues.

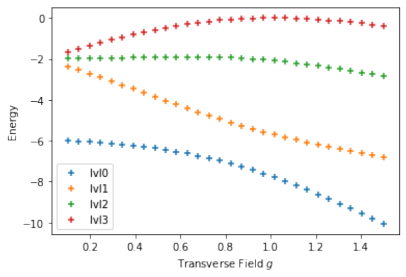
This allows us to demonstrate the dependency of the threshold on the d_{eff} with controlled κ_{eff}. In Figure 7.12 we plot the success rate against the number of parameters p for both ansatze with number of qubits $N = 4, 6, 8, 10$: it is observed that TFI_2 (▼) consistently achieve lower over-parameterization threshold p than TFI_3 (■) due to smaller d_{eff}.

Ground states of TFI1d models are degenerated for $|g| \leq 1$ in the thermodynamic limit $N \to \infty$. Although there are no degeneracy for finite N, the first excitation energy (i.e. the smallest eigen-gaps) decrease quickly as g drops below 1.0. In Figure 7.11(c), we visualize the smallest 4 eigenvalues for $N = 6$. The vanishing eigen-gap for small g leads to drastic increase of κ_{eff} as plotted in blue in Figure 7.11. On the contrary, the effective ratio κ_{eff} for both TFI_2 and TFI_3 remain small as g approaches 0. As a result, the over-parameterization threshold remains almost the same for TFI_2 as the transverse field g decreases from 0.5 to 0.1 (as shown in Figure 7.13). This shows that the usage of HVA instead of general purpose ansatz design allows solving VQE problems efficiently near critical points.

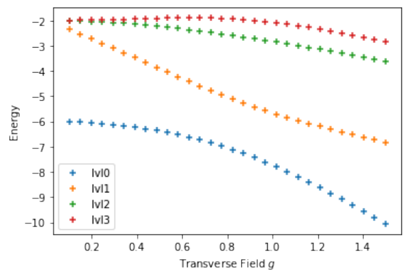
1-d XXZ model XXZ1d model is a special case of Heisenberg model with problem Hamiltonian defined as

$$\mathrm{M}_{\text{XXZ1d}}(J_{zz}) = \sum_{i=0}^{N-1} X_i X_{i+1} + Y_i Y_{i+1} + J_{zz} \sum_{i=0}^{N-1} Z_i Z_{i+1} \tag{7.66}$$

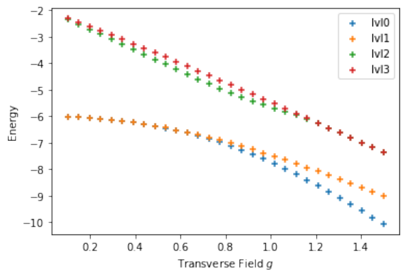
The parameter J_{zz} controls the coupling in the Z-direction. XXZ1d model is essentially different from the TFI1d model in that actual level-crossing happens for finite N at $J_{zz} = -1$.



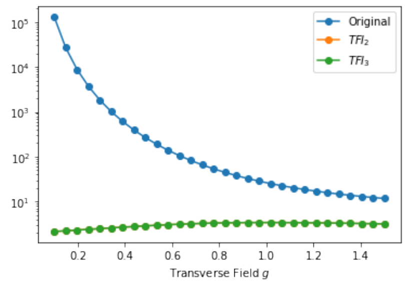
(a) TFI_2



(b) TFI_3

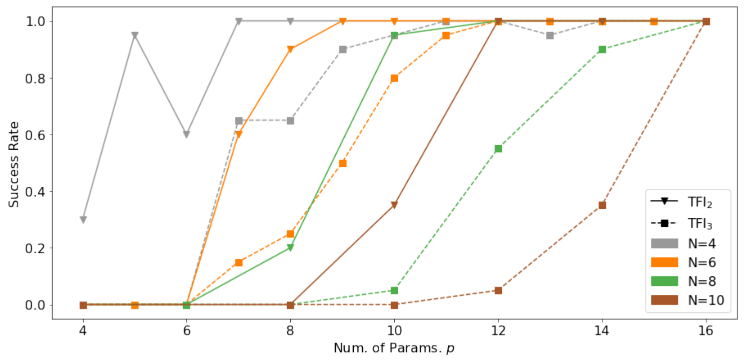


(c) Original

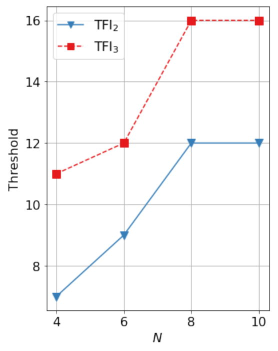


(d) Comparison of κ_{eff}

Figure 7.11: Energy of the ground state and the first 3 excitation states. The smallest 4 eigenvalues for the effective Hamiltonian with TFI_2 (a), TFI_3 (b) and for the original Hamiltonian $H_{\mathsf{TFI1d}}(g)$ (c) for $N = 6$ with transverse field g varying from 0.1 to 1.5. As plotted in (d) κ_{eff} for the original Hamiltonian increases quickly for g close to 0 while κ_{eff} for both TFI_2 and TFI_3 remain small.

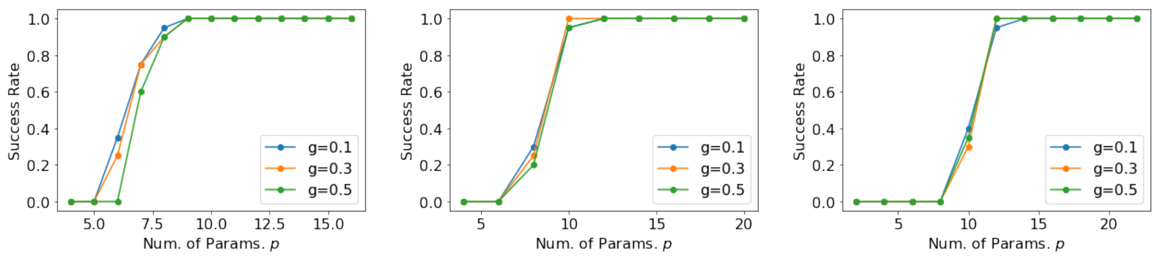


(a)



(b)

Figure 7.12: Comparison of the over-parameterization threshold for TFI_2 and TFI_3 ansatze for $N = 4, 6, 8, 10$. (a) The success rates for finding a solution with error less than 0.01 versus the number of parameters for instances with different ansatz and different sizes. The number of qubits is encoded by different colors and the ansatz design is encoded by ▼ for TFI_2 and ■ for TFI_3. For each data point, the success rate is estimated over 20 random initializations. (b) Plot of the over-parameterization threshold versus number of qubits for different ansatze. The threshold is defined as the the smallest number of parameters to achieve success rate over 98%. For each N, the threshold for TFI_2 is lower than that of TFI_3.



(a) N=6 (b) N=8 (c) N=10

Figure 7.13: Comparison of the over-parameterization threshold for TFI_2 with transverse field $g = 0.1, 0.3, 0.5$ for (a) $N = 6$ (b)$N = 8$ (c) $N = 10$. The x-axis is the number of trainable parameters p, and the y-axis is the success rate for finding a solution with error less than 0.01. For $N = 6, 8, 10$, despite the vanishing eigen-gap of $H_{TFI1d}(g)$ for small g, the ground state can be found with reasonable p with ansatz TFI_2. For each data point, the success rate is estimated over 20 random initializations.

We examine the HVA proposed in [268] (denoted as XXZ_4):

$$\mathbf{H}_1 \propto \sum_{\text{even}i} X_iX_{i+1} + \sum_{\text{even}i} Y_iY_{i+1}, \tag{7.67}$$

$$\mathbf{H}_2 \propto \sum_{\text{odd}i} X_iX_{i+1} + \sum_{\text{odd}i} Y_iY_{i+1}, \tag{7.68}$$

$$\mathbf{H}_3 \propto \sum_{\text{even}i} Z_iZ_{i+1}, \quad \mathbf{H}_4 \propto \sum_{\text{odd}i} Z_iZ_{i+1} \tag{7.69}$$

as well as a similar HVA (denoted as XXZ_6)

$$\mathbf{H}_1 \propto \sum_{\text{even}i} X_iX_{i+1}, \quad \mathbf{H}_2 \propto \sum_{\text{odd}i} X_iX_{i+1}, \quad \mathbf{H}_3 \propto \sum_{\text{even}i} Y_iY_{i+1}, \tag{7.70}$$

$$\mathbf{H}_4 \propto \sum_{\text{odd}i} Y_iY_{i+1}, \quad \mathbf{H}_5 \propto \sum_{\text{even}i} Z_iZ_{i+1}, \quad \mathbf{H}_6 \propto \sum_{\text{odd}i} Z_iZ_{i+1} \tag{7.71}$$

The effective dimensions for XXZ_4 and XXZ_6 are summarized in Table 7.2. While both XXZ_4 and XXZ_6 significantly reduce the effective dimension d_{eff}, XXZ_4 further removes the level-crossing: in Figure 7.14, we see that both XXZ_4 and XXZ_6 reduces the ratio κ_{eff} by orders of magnitude, and the ratio κ_{eff} for XXZ_4 (in orange) remains small as $J_{zz} \to -1$ while the ratio for both XXZ_6 (in green) and the original Hamiltonian (in blue) increases to infinity. In Figure 7.15, we present side by side the success rate of XXZ_4 and XXZ_6 for $N = 4$ with $J_{zz} = -0.9, -0.5, -0.3, 0.1$. It is observed that the over-parameterization threshold XXZ_4 remain similar across different values of J_{zz} and the over-parameterization threshold for XXZ_6 increases significantly as J_{zz} decreases to -0.9 due to the vanishing eigen-gaps.

N	4	6	8	10
d	16	64	256	1024
XXZ_4	3	4	12	21
XXZ_6	4	5	19	34

Table 7.2: System dimension d and effective dimension d_{eff} for XXZ1d model with $N = 4, 6, 8, 10$ for XXZ_4 and XXZ_6 ansatze

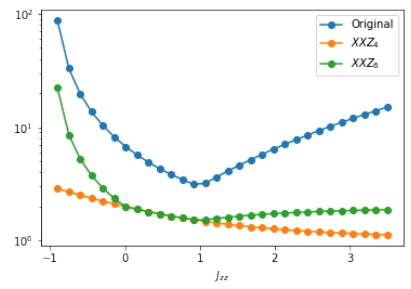
7.8 Deferred Technical Details

7.8.1 Details of Convergence Proof

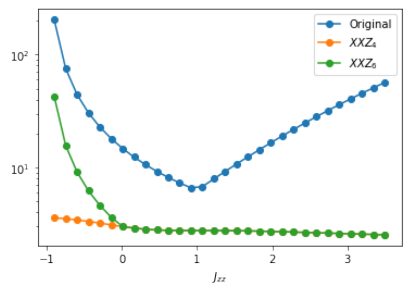
Proof of Lemma 7.3.2.

Lemma 7.3.2 (VQE Perturbation Lemma)**.** *Conditioned on the event that the output state at initialization $|\Psi(0)\rangle$ has non-negligible overlap with the target ground state $|\Psi^\star\rangle$: $|\langle\Psi(0)|\Psi^\star\rangle|^2 \geq O(\frac{1}{d})$, if for all $t \geq 0$, $\|\mathbf{Y}(t) - (\mathbf{W} - \frac{1}{d}\mathbf{I}_{d^2\times d^2})\|_{\text{op}} \leq \frac{\lambda_2-\lambda_1}{\lambda_d-\lambda_1}\cdot\frac{1}{2\sqrt{2}d}$, then under the dynamics $\frac{d}{dt}|\Psi\rangle = -\operatorname{Tr}_1(\mathbf{Y}([\mathbf{M}, |\Psi\rangle\langle\Psi|] \otimes \mathbf{I}_{d\times d}))|\Psi\rangle$, the output states converges to the ground state:*

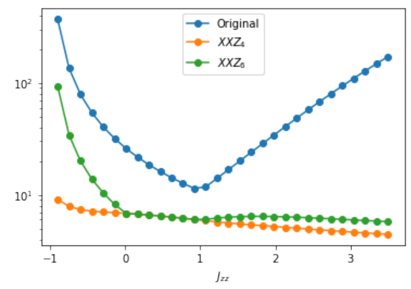
$$1 - |\langle\Psi(t)|\Psi^\star\rangle|^2 \leq \exp(-\frac{\lambda_2-\lambda_1}{2\log d}t). \tag{7.14}$$



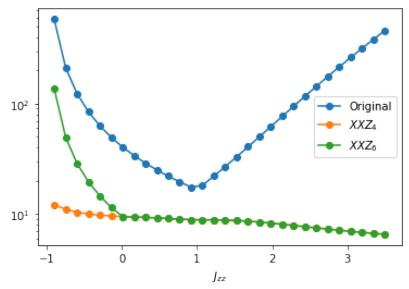
(a) XXZ1d Model: N=4



(b) XXZ1d Model: N=6

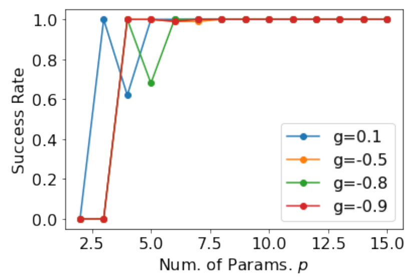


(c) XXZ1d Model: N=8

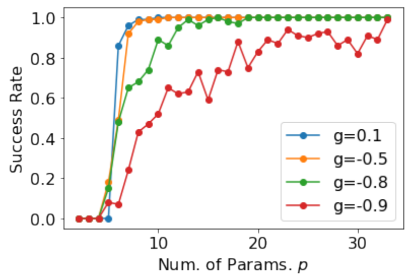


(d) XXZ1d Model: N=10

Figure 7.14: κ and κ_{eff} for XXZ_4 and XXZ_6 for $N = 4, 6, 8, 10$. We plot the κ for 1d XXZ model and κ_{eff} for XXZ_4 and XXZ_6 for different values of J_{zz}. For both XXZ_6 and the original problem Hamiltonian, level crossing happens at $J_{zz} = -1$, making it impossible to solve for the ground state when J_{zz} is close to -1. Note that the level crossing breaks down under XXZ_4.



(a) XXZ_4



(b) XXZ_6

Figure 7.15: Comparison of the over-parameterization threshold for (a) XXZ_4 and (b) XXZ_6 with Z-coupling $J_{zz} = 0.1, -0.3, -0.5, -0.9$. The x-axis is the number of trainable parameters p, and the y-axis is the success rate for finding a solution with error less than 0.01. For XXZ_4 the over-parameterization threshold remain similar for various J_{zz}, while for XXZ_6 the threshold drastically increase as J_{zz} approaches -1 as a result of level-crossing. For each data point, the success rate is estimated over 100 random initializations.

Proof for Lemma 7.3.2. Let $\mathbb{E}(t)$ denote the deviation of $\mathbf{Y}(t)$ from its expected value:

$$\mathbb{E}(t) := \mathbf{Y}(t) - (\mathbf{W} - \frac{1}{d}\mathbf{I}_{d^2\times d^2}) \tag{7.72}$$

The matrix that governs the dynamics can be expressed as

$$\mathrm{Tr}_1(\mathbf{Y}(t)([\mathbf{M}, |\Psi(t)\rangle\langle\Psi(t)|] \otimes \mathbf{I}_{d\times d})) = [\mathbf{M}, |\Psi(t)\rangle\langle\Psi(t)|] + E(t) \tag{7.73}$$

where

$$E(t) := \mathrm{Tr}_1\left(\mathbb{E}(t)([\mathbf{M}, |\Psi(t)\rangle\langle\Psi(t)|] \otimes \mathbf{I}_{d\times d})\right) \tag{7.74}$$

Define h as $|\langle\Psi^\star|\Psi(t)||\rangle^2$

$$\frac{d}{dt}h = (\frac{d}{dt}|\Psi(t)\rangle)^\dagger|\Psi^\star\rangle\langle\Psi^\star|\Psi(t)|+\rangle\langle\Psi(t)|\Psi^\star|\langle\rangle|\Psi^\star\frac{d}{dt}|\Psi(t)\rangle \tag{7.75}$$

$$= 2(\langle\Psi(t)|\mathbf{M}|\Psi(t)\rangle - \lambda_1)|\langle\Psi^\star|\Psi(t)||\rangle^2 + \mathrm{Tr}(E(t)[|\Psi^\star\rangle\langle\Psi^\star|, |\Psi(t)\rangle\langle\Psi(t)|]) \tag{7.76}$$

The first term in Line (7.76) corresponds to the actual Riemannian gradient flow on the sphere:

$$2(\langle\Psi(t)|\mathbf{M}|\Psi(t)\rangle - \lambda_1)|\langle\Psi^\star|\Psi(t)||\rangle^2 = 2(\langle\Psi(t)|\mathbf{M}|\Psi(t)\rangle - \lambda_1)h \tag{7.77}$$

$$\geq 2((1-h)\lambda_2 + h\lambda_1 - \lambda_1)h \tag{7.78}$$

$$= 2(\lambda_2 - \lambda_1)(1-h)h \tag{7.79}$$

The second term in Line (7.76) stems from the deviation of $\mathbf{Y}$ from its expectation:

$$\mathrm{Tr}(E(t)[|\Psi^\star\rangle\langle\Psi^\star|, |\Psi(t)\rangle\langle\Psi(t)|]) \tag{7.80}$$

$$= \mathrm{Tr}\left(\mathrm{Tr}_1 \left(\mathbb{E}(t)([\mathbf{M}, |\Psi(t)\rangle\langle\Psi(t)|] \otimes \mathbf{I}_{d\times d})\right)[|\Psi^\star\rangle\langle\Psi^\star|, |\Psi(t)\rangle\langle\Psi(t)|]\right) \tag{7.81}$$

$$= \mathrm{Tr}\left(\mathbb{E}(t)([\mathbf{M}, |\Psi(t)\rangle\langle\Psi(t)|] \otimes [|\Psi^\star\rangle\langle\Psi^\star|, |\Psi(t)\rangle\langle\Psi(t)|])\right) \tag{7.82}$$

$$\geq -\|\mathbb{E}(t)\|\|[\mathbf{M}, |\Psi(t)\rangle\langle\Psi(t)|] \otimes [|\Psi^\star\rangle\langle\Psi^\star|, |\Psi(t)\rangle\langle\Psi(t)|]\|_{\mathsf{tr}} \tag{7.83}$$

$$= -2\sqrt{h(1-h)}\|\mathbb{E}(t)\|\|[\mathbf{M}, |\Psi(t)\rangle\langle\Psi(t)|]\|_{\mathsf{tr}} \tag{7.84}$$

$$= -2\sqrt{d}\sqrt{h(1-h)}\|\mathbb{E}(t)\|\|[\mathbf{M}, |\Psi(t)\rangle\langle\Psi(t)|]\|_F \tag{7.85}$$

$$= -2\sqrt{2}\sqrt{d}\sqrt{h}(1-h)(\lambda_d - \lambda_1)\|\mathbb{E}(t)\| \tag{7.86}$$

Line (7.84) follows from Lemma 7.8.2; Line (7.86) follows from Lemma 7.8.3

Hence

$$\frac{d}{dt}h \geq 2(\lambda_2 - \lambda_1)(1-h)h(1 - \sqrt{2d}\frac{\lambda_d - \lambda_1}{\lambda_2 - \lambda_1}\|\mathbb{E}(t)\|\frac{1}{\sqrt{h}})) \tag{7.87}$$

Following the technical Lemma 4 in [266]

$$\frac{d}{dt}\log(-\log(h)) \leq -2(\lambda_2 - \lambda_1)\frac{1 - \sqrt{2d}\frac{\lambda_d - \lambda_1}{\lambda_2 - \lambda_1}\|\mathbb{E}(t)\|\frac{1}{\sqrt{h}}}{1 - \log h} \tag{7.88}$$

Apparently for $\|\mathbb{E}(t)\|$: $\sqrt{2d}\frac{\lambda_d-\lambda_1}{\lambda_2-\lambda_1}\frac{1}{\sqrt{h(0)}}\|\mathbb{E}(t)\| < 1$, $h(t)$ is non-decreasing. Conditioned on $h(0) \geq \frac{1}{d}$, if $\sqrt{2d}\frac{\lambda_d-\lambda_1}{\lambda_2-\lambda_1}\|\mathbb{E}(t)\|\frac{1}{\sqrt{1/d}} \leq \frac{1}{2}$, the absolute value of the right-hand side is lower bounded by $\frac{\lambda_2-\lambda_1}{2\log d}$.

Therefore if $\|\mathbb{E}(t)\| \leq \frac{1}{2\sqrt{2d}}\frac{\lambda_2-\lambda_1}{\lambda_d-\lambda_1}$, $1 - h(t) \leq -\log h(t) \leq \exp(-\frac{\lambda_2-\lambda_1}{2\log d}t)$. □

Proof of Lemma 7.3.3.

Lemma 7.3.3 (Concentration at initialization for VQE)**.** *Over the randomness of ansatz initialization (i.e. for $\{\mathrm{U}_l\}_{l=1}^p$ sampled $i.i.d.$ with respect to the Haar measure), for any initial $\boldsymbol{\theta}(0)$, with probability $1-\delta$:*

$$\|\mathbf{Y}(\boldsymbol{\theta}(0)) - \mathbf{Y}^\star\|_{\mathsf{op}} \leq \frac{1}{\sqrt{p}} \cdot \frac{2\|\mathbf{H}\|_{\mathsf{op}}^2}{Z}\sqrt{\log\frac{d^2}{\delta}} \tag{7.15}$$

Proof. Define X_l as the centered random matrix

$$X_l := \frac{1}{Z(\mathbf{H},d)}\left(\mathbf{U}_{l:p}\mathbf{H}\mathbf{U}_{l:p}^\dagger\right)^{\otimes 2} - \mathbf{Y}^\star \tag{7.89}$$

$\{\mathbf{X}_l\}$ can be viewed as independent random matrices as the haar random unitary removes all the correlation. The square of $\mathbf{X}_l$ is bounded in operator norm:

$$\|\mathbf{X}_l^2\| = \|\mathbf{X}_l\|^2 \leq (\frac{\|\mathbf{H}\|^2}{Z} + \frac{d+1}{d})^2 \leq (\frac{2\|\mathbf{H}\|^2}{Z(\mathbf{H},d)})^2 \tag{7.90}$$

The ratio $g_1 = \|\mathbf{H}\|^2/\operatorname{Tr}(\mathbf{H}^2)$ satisfies that $1 \geq g_1 \geq 1/d$. By Hoeffding's inequality([275], Thm 1.3), with probability $\geq 1-\delta$,

$$\|\mathbf{Y} - \mathbf{Y}^\star\| \leq \frac{1}{\sqrt{p}} \cdot \frac{2\|\mathbf{H}\|^2}{Z}\sqrt{\log\frac{d^2}{\delta}} \tag{7.91}$$

□

For comparison, consider the following matrix Berstein inequality (adapted from [276], Thm 6.1.1):

Lemma 7.8.1. *Consider a finite sequence* $\{\mathbf{X}_l\}$ *of identically distributed, independent, random Hermitian with dimension* d'. *Assume* $\mathbb{E}[\mathbf{X}^2] = \sigma^2$ *and* $\|\mathbf{X}\| \leq R$. *Then with probability* $\geq 1-\delta$

$$\|\frac{1}{p}\sum_{l=1}^{p}\mathbf{X}_l\| \leq \frac{\sigma\sqrt{2\log(2d'/\delta)}}{\sqrt{p}} + \frac{2R\log(2d'/\delta)}{3p} \tag{7.92}$$

In our case, $\|\mathbf{X}\| \leq (d^2-1)g_1 + 2$, and $\sqrt{\mathbb{E}[\mathbf{X}^2]} \leq \sqrt{2}d$:

$$\mathbb{E}[\mathbf{X}_l^2] = \mathbb{E}[\frac{1}{Z^2}(\mathbf{V}_l\mathbf{H}^2\mathbf{V}_l^\dagger)^{\otimes 2}] - (\mathbf{Y}^\star)^2 \tag{7.93}$$

$$= \frac{1}{(d^2-1)Z^2}\big[(\mathrm{Tr}^2(\mathbf{H}^2) - \frac{1}{d}\mathrm{Tr}(\mathbf{H}^4))\mathbf{I} + (\mathrm{Tr}(\mathbf{H}^4) - \frac{1}{d}\mathrm{Tr}^2(\mathbf{H}^2))\mathbf{W}\big] - (\frac{d^2+1}{d^2}\mathbf{I} - \frac{2}{d}\mathbf{W}) \tag{7.94}$$

$$= \frac{\mathrm{Tr}^2(\mathbf{H}^2)}{(d^2-1)Z^2}\big[(1-\frac{1}{d}g_2(\mathbf{H}))\mathbf{I} + (g_2(\mathbf{H}) - \frac{1}{d})\mathbf{W}\big] - \frac{d^2+1}{d^2}\mathbf{I} + \frac{2}{d}\mathbf{W} \tag{7.95}$$

$$= (d^2-1)\big[(1-\frac{1}{d}g_2(\mathbf{H}))\mathbf{I} + (g_2(\mathbf{H}) - \frac{1}{d})\mathbf{W}\big] - \frac{d^2+1}{d^2}\mathbf{I} + \frac{2}{d}\mathbf{W} \tag{7.96}$$

$$= (d^2 - 2 - \frac{1}{d^2} - g_2(d-1/d))\mathbf{I} + (g_2(d^2-1) - d + 3/d)\mathbf{W} \tag{7.97}$$

where $g_2(\mathbf{H}) := \mathrm{Tr}(\mathbf{H}^4)/\mathrm{Tr}^2(\mathbf{H}^2)$, and $1 \geq g_2(\mathbf{H}) \geq 1/d$.

Helper Lemmas.

Lemma 7.8.2. *Let* $\mathbf{x}, \mathbf{v}$ *be two vectors in* $\mathbb{C}^d$, *the commutator* $i[\mathbf{x}\mathbf{x}^\dagger, \mathbf{v}\mathbf{v}^\dagger]$ *has eigenvalues* $\pm|\langle\mathbf{x},\mathbf{v}\rangle|\sqrt{1-|\langle\mathbf{x},\mathbf{v}\rangle|^2}$.

Proof. Express $\mathbf{x}$ as $\alpha\mathbf{v} + \beta\mathbf{w}$ with $\mathbf{w}$ orthogonal to $\mathbf{v}$.

$$i[\mathbf{x}\mathbf{x}^\dagger, \mathbf{v}\mathbf{v}^\dagger] = i\alpha^*\beta\mathbf{w}\mathbf{v}^\dagger - i\alpha\beta^*\mathbf{v}\mathbf{w}^\dagger \tag{7.98}$$

This rank-2 Hermitian has two real eigenvalues λ_+ and λ_- such that $\lambda_+ + \lambda_- = 0$ and $\lambda_+\lambda_- = -|\alpha|^2|\beta|^2$. $\square$

Lemma 7.8.3 (Bounding commutator norms)**.** *Let* $\mathbf{M} := \sum_{j=1}^{d} \lambda_j \mathbf{v}_j \mathbf{v}_j^\dagger$ *be a* $d \times d$*-Hermitian matrix with eigenvalues* $\lambda_1 \leq \cdots \leq \lambda_d$*. The frobenius norm of the commutator* $[\mathbf{M}, \mathbf{x}\mathbf{x}^\dagger]$ *can be bounded in terms of* $|\langle \mathbf{x}, \mathbf{v} \rangle|$ *as:*

$$\|[\mathbf{M}, \mathbf{x}\mathbf{x}^\dagger]\|_F \leq \sqrt{2}(\lambda_d - \lambda_1)\sqrt{1 - |\langle \mathbf{x}, \mathbf{v} \rangle|^2} \tag{7.99}$$

Proof. Expand $\mathbf{x}$ as $\alpha\mathbf{v_1} + \beta\mathbf{w}$, where $\mathbf{v}_1$ is the ground state of $\mathbf{M}$ and unit vector $\mathbf{w}$ is orthogonal to $\mathbf{v}_1$:

$$\|[\mathbf{M}, \mathbf{x}\mathbf{x}^\dagger]\|_F^2 = \|[\mathbf{M} - \lambda_1\mathbf{I}, \mathbf{x}\mathbf{x}^\dagger]\|_F^2 \tag{7.100}$$

$$= 2\big(\mathbf{x}^\dagger\mathbf{M}^2\mathbf{x} - (\mathbf{x}^\dagger\mathbf{M}\mathbf{x})^2\big) \tag{7.101}$$

$$= 2\big(|\beta|^2\mathbf{w}^\dagger\mathbf{M}^2\mathbf{w} - (|\beta|^2\mathbf{x}^\dagger\mathbf{M}\mathbf{x})^2\big) \tag{7.102}$$

$$\leq 2|\beta|^2\mathbf{w}^\dagger\mathbf{M}^2\mathbf{w} \tag{7.103}$$

$$\leq 2|\beta|^2(\lambda_d - \lambda_1)^2 \tag{7.104}$$

$\square$

Lemma 7.8.4 (Estimation with taylor expansion)**.** *Let* W *be a unitary generated by Hermitian* H*:* $W = \exp(-i\theta H)$*, we have*

$$(WKW^\dagger)^{\otimes 2} - K^{\otimes 2} \preceq 4|\theta|\|H\|\|K\|^2 I \tag{7.105}$$

For symmetrically distributed random H (i.e. $H \sim -H$)

$$\mathbb{E}_H[(WKW^\dagger)^{\otimes 2} - K^{\otimes 2}] \preceq 8\theta^2 \|H\|^2 \|K\|^2 I \tag{7.106}$$

More generally, consider Hermitian-preserving linear maps $\Phi_1, \Phi_2 \in \mathcal{L}(\mathbb{C}^{d^2 \times d^2})$ such that $\|\Phi_1(X)\| \leq L_1 \|X\|, \|\Phi_2(X)\| \leq L_2 \|X\|$:

$$\mathbb{E}_H[\Phi_1\big((WK_1W^\dagger)^{\otimes 2}\big)\Phi_2\big((WK_2W^\dagger)^{\otimes 2}\big) - \Phi_1\big(K_1^{\otimes 2}\big)\Phi_2\big(K_2^{\otimes 2}\big)] \preceq 64 L_1 L_2 \|H\|^2 \|K_1\|^2 \|K_2\|^2 \mathbf{I} \tag{7.107}$$

Proof. The first- and second-order derivatives of $(WKW^\dagger)^{\otimes 2}$ are:

$$\frac{d}{d\theta}(WKW^\dagger)^{\otimes 2} = W^{\otimes 2}([-iH, K] \otimes K + K \otimes [-iH, K])(W^\dagger)^{\otimes 2} \tag{7.108}$$

$$\frac{d^2}{d\theta^2}(WKW^\dagger)^{\otimes 2} = -W^{\otimes 2}(2[H, K] \otimes [H, K] + [H, [H, K]] \otimes K + K \otimes [H, [H, K]])(W^\dagger)^{\otimes 2} \tag{7.109}$$

Hence

$$(WKW^\dagger)^{\otimes 2} - K^{\otimes 2} \tag{7.110}$$

$$= \int_0^\theta d\theta' (e^{-i(\theta-\theta')H})^{\otimes 2}([-iH, K] \otimes K + K \otimes [-iH, K])(e^{i(\theta-\theta')H})^{\otimes 2} \tag{7.111}$$

$$\preceq 4|\theta| \|H\| \|K\|^2 I \tag{7.112}$$

For symmetrically distributed random H

$$\mathbb{E}(WKW^\dagger)^{\otimes 2} - K^{\otimes 2} \tag{7.113}$$

$$=\mathbb{E}\theta([-iH,K]\otimes K + K\otimes[-iH,K]) \tag{7.114}$$

$$-\mathbb{E}\int_0^\theta \theta' d\theta' \tag{7.115}$$

$$(e^{-i(\theta-\theta')H})^{\otimes 2}(2[H,K]\otimes[H,K] + [H,[H,K]]\otimes K + K\otimes[H,[H,K]])(e^{i(\theta-\theta')H})^{\otimes 2} \tag{7.116}$$

$$\preceq 8\theta^2\|H\|^2\|K\|^2 I \tag{7.117}$$

For the more general case:

$$\frac{d}{d\theta}\Phi_1\big((WKW^\dagger)^{\otimes 2}\big)\Phi_2\big((WKW^\dagger)^{\otimes 2}\big) \tag{7.118}$$

$$=\Phi_1\big(\frac{d}{d\theta}(WKW^\dagger)^{\otimes 2}\big)\Phi_2\big((WKW^\dagger)^{\otimes 2}\big) + \Phi_1\big((WKW^\dagger)^{\otimes 2}\big)\Phi_2\big(\frac{d}{d\theta}(WKW^\dagger)^{\otimes 2}\big) \tag{7.119}$$

$$=\Phi_1\big(W^{\otimes 2}([-iH,K]\otimes K + K\otimes[-iH,K])(W^\dagger)^{\otimes 2}\big)\Phi_2\big((WKW^\dagger)^{\otimes 2}\big) \tag{7.120}$$

$$+\Phi_1\big((WKW^\dagger)^{\otimes 2}\big)\Phi_2\big(W^{\otimes 2}([-iH,K]\otimes K + K\otimes[-iH,K])(W^\dagger)^{\otimes 2}\big) \tag{7.121}$$

When evalutated at $\theta = 0$,

$$\frac{d}{d\theta}\Phi_1\big((WKW^\dagger)^{\otimes 2}\big)\Phi_2\big((WKW^\dagger)^{\otimes 2}\big)|_{\theta=0} \tag{7.122}$$

$$=\Phi_1\big(([-iH,K]\otimes K + K\otimes[-iH,K])\big)\Phi_2\big(K^{\otimes 2}\big) \tag{7.123}$$

$$+\Phi_1\big(K^{\otimes 2}\big)\Phi_2\big(([-iH,K]\otimes K + K\otimes[-iH,K])\big) \tag{7.124}$$

The second order derivative

$$\frac{d^2}{d\theta^2}\Phi_1\big((WKW^\dagger)^{\otimes 2}\big)\Phi_2\big((WKW^\dagger)^{\otimes 2}\big) \tag{7.125}$$

$$=\frac{d}{d\theta}\{\Phi_1\big(\frac{d}{d\theta}(WKW^\dagger)^{\otimes 2}\big)\Phi_2\big((WKW^\dagger)^{\otimes 2}\big)+\Phi_1\big((WKW^\dagger)^{\otimes 2}\big)\Phi_2\big(\frac{d}{d\theta}(WKW^\dagger)^{\otimes 2}\big)\} \tag{7.126}$$

$$=2\Phi_1\big(W^{\otimes 2}([-iH,K]\otimes K+K\otimes[-iH,K])(W^\dagger)^{\otimes 2}\big)$$

$$\Phi_2\big([-iH,K]\otimes K+K\otimes[-iH,K])(W^\dagger)^{\otimes 2}\big) \tag{7.127}$$

$$+\Phi_1\big((WKW^\dagger)^{\otimes 2}\big)\Phi_2\big(-W^{\otimes 2}(2[H,K]\otimes[H,K]+[H,[H,K]]\otimes K$$

$$+K\otimes[H,[H,K]])(W^\dagger)^{\otimes 2}\big) \tag{7.128}$$

$$+\Phi_1\big(-W^{\otimes 2}(2[H,K]\otimes[H,K]+[H,[H,K]]\otimes K+K\otimes[H,[H,K]])(W^\dagger)^{\otimes 2}\big)$$

$$\Phi_2\big((WKW^\dagger)^{\otimes 2}\big) \tag{7.129}$$

For any θ, the second derivative is upperbounded in operator norm by $64L_1L_2\|H\|^2\|K\|^4$. By the Taylor expansion the operator norm of $\mathbb{E}_H[\Phi_1\big((WKW^\dagger)^{\otimes 2}\big)\Phi_2\big((WKW^\dagger)^{\otimes 2}\big)-\Phi_1\big(K^{\otimes 2}\big)\Phi_2\big(K^{\otimes 2}\big)]$ is bounded by $32L_1L_2\|H\|^2\|K\|^4$. □

7.8.2 Proof of Corollary 7.4.1

Lemma 7.4.1 (Output-state dynamics with noisy gradient estimation)**.** *Consider VQE instance* $(\mathbf{M},|\Phi\rangle,\mathbf{U})$*, with* $\mathbf{U}$ *being the ansatz defined in Definition 7.3.1. Under gradient flow with learning rate* η *and noisy gradient estimation* $\widehat{\nabla L}:=\nabla L+\boldsymbol{\varepsilon}(t)=\big(\frac{\partial L}{\partial\theta_l}+\varepsilon_l(t)\big)_{l\in[p]}$*, the output* *state $|\Psi\rangle$ follow the dynamics*

$$\frac{d}{dt}|\Psi\rangle = -(\eta \cdot p \cdot Z(\mathbf{H}, d)) \operatorname{Tr}_1(\mathbf{Y}([\mathbf{M}, |\Psi\rangle\langle\Psi|] \otimes \mathbf{I}_{d\times d}))|\Psi\rangle + \eta \sum_{l=1}^{p} i\varepsilon_l \mathbf{H}_l|\Psi\rangle \tag{7.50}$$

Here the Hermitian $\mathbf{Y} \in \mathbb{C}^{d^2\times d^2}$ is the time-dependent matrix defined as

$$\mathbf{Y}(\boldsymbol{\theta}) := \frac{1}{pZ(\mathbf{H}, d)} \sum_{l=1}^{p} \left(\mathbf{U}_{l:p}(\boldsymbol{\theta}(t))\mathbf{H}\mathbf{U}^{\dagger}_{l:p}(\boldsymbol{\theta}(t))\right)^{\otimes 2} \tag{7.51}$$

and $\mathbf{H}_l$ are defined as $\mathbf{U}_{l:p}(\boldsymbol{\theta}(t))\mathbf{H}\mathbf{U}^{\dagger}_{l:p}(\boldsymbol{\theta}(t))$.

Proof. We start by calculating the gradient of $\mathbf{U}_{r:p}(\boldsymbol{\theta})$ with respect to θ_l: For $r > l$, $\mathbf{U}_{r:p}$ is independent of θ_l ; For $r \leq l$,

$$\frac{\partial \mathbf{U}_{r:p}}{\partial \theta_l} = \mathbf{U}_{l:p}(\boldsymbol{\theta}) \cdot (-i\mathbf{H}) \cdot \mathbf{U}_{r:l-1}(\boldsymbol{\theta}) = -i\mathbf{U}_{l:p}\mathbf{H}\mathbf{U}^{\dagger}_{l:p}\mathbf{U}_{r:p} \tag{7.130}$$

Therefore

$$\begin{aligned}\frac{\partial L(\boldsymbol{\theta})}{\partial \theta_l} &= \langle\Phi|\mathbf{U}_0^{\dagger}\partial_l\mathbf{U}^{\dagger}_{1:p}\mathbf{M}\mathbf{U}_{1:p}\mathbf{U}_0|\Phi\rangle + \langle\Phi|\mathbf{U}_0^{\dagger}\mathbf{U}^{\dagger}_{1:p}\mathbf{M}\partial_l\mathbf{U}_{1:p}\mathbf{U}_0|\Phi\rangle && (7.131)\\ &= \langle\Phi|\mathbf{U}_0^{\dagger}\mathbf{U}^{\dagger}_{1:p}i[\mathbf{U}_{l:p}\mathbf{H}\mathbf{U}^{\dagger}_{l:p}, \mathbf{M}]\mathbf{U}_{1:p}\mathbf{U}_0|\Phi\rangle && (7.132)\\ &= i\operatorname{Tr}([\mathbf{M}, |\Psi\rangle\langle\Psi|]\mathbf{U}_{l:p}\mathbf{H}\mathbf{U}^{\dagger}_{l:p}) && (7.133)\end{aligned}$$

Following gradient descent with learning rate η:

$$\frac{d\theta_l}{dt} = -\eta\big(\partial_l L(\boldsymbol{\theta}) + \varepsilon_l\big) = -i\eta \operatorname{Tr}([\mathbf{M}, |\Psi\rangle\langle\Psi|]\mathbf{U}_{l:p}\mathbf{H}\mathbf{U}^{\dagger}_{l:p}) - \eta\varepsilon_l \tag{7.134}$$

The dynamics for $\mathbf{U}_{l:p}$ and $|\Psi\rangle$ are therefore:

$$\frac{d}{dt}\mathbf{U}_{l:p}(t) \tag{7.135}$$

$$=\sum_{r=l}^{p}\frac{d\theta_r}{dt}\partial_r\mathbf{U}_{l:p} \tag{7.136}$$

$$=-\eta\sum_{r=l}^{p}\operatorname{Tr}([\mathbf{M},|\Psi\rangle\langle\Psi|]\mathbf{U}_{r:p}\mathbf{H}\mathbf{U}_{r:p}^{\dagger})\mathbf{U}_{r:p}\mathbf{H}\mathbf{U}_{r:p}^{\dagger}\mathbf{U}_{l:p}+i\eta\sum_{r=1}^{p}\varepsilon_r\mathbf{U}_{r:p}\mathbf{H}\mathbf{U}_{r:p}^{\dagger}\mathbf{U}_{l:p} \tag{7.137}$$

$$\frac{d}{dt}|\Psi\rangle=\frac{d}{dt}\mathbf{U}_{1:p}\mathbf{U}_0|\Phi\rangle \tag{7.138}$$

$$=-(\eta\cdot pZ)\frac{1}{pZ}\big(\sum_{l=1}^{p}\operatorname{Tr}([\mathbf{M},|\Psi\rangle\langle\Psi|]\mathbf{U}_{l:p}\mathbf{H}\mathbf{U}_{l:p}^{\dagger})\mathbf{U}_{l:p}\mathbf{H}\mathbf{U}_{l:p}^{\dagger}\big)\mathbf{U}_{1:p}\mathbf{U}_0|\Phi\rangle \tag{7.139}$$

$$+i\eta\sum_{l=1}^{p}\varepsilon_l\mathbf{U}_{l:p}\mathbf{H}\mathbf{U}_{l:p}^{\dagger}\mathbf{U}_{1:p}\mathbf{U}_0|\Phi\rangle \tag{7.140}$$

$$=-(\eta\cdot pZ)\operatorname{Tr}_1(\mathbf{Y}\cdot[\mathbf{M},|\Psi\rangle\langle\Psi|]\otimes\mathbf{I})|\Psi\rangle+\eta\sum_{l=1}^{p}i\varepsilon_l\mathbf{H}_l|\Psi\rangle \tag{7.141}$$

□

Lemma 7.4.2 (VQE Perturbation Lemma under noisy gradients)**.** *Conditioned on the event that the output state at initialization $|\Psi(0)\rangle$ has non-negligible overlap with the target ground state $|\Psi^\star\rangle$: $|\langle\Psi(0)|\Psi^\star\rangle|^2\geq O(\frac{1}{d})$, if for all $t\geq 0$, $\|\mathbf{Y}(t)-\mathbf{Y}^\star(t)\|_{\mathsf{op}}\leq\frac{\lambda_2-\lambda_1}{\lambda_d-\lambda_1}\cdot\frac{1}{4\sqrt{2}d}$ and $\|\boldsymbol{\varepsilon}(t)\|_\infty\leq\frac{Z}{2\|\mathbf{H}\|}(\lambda_2-\lambda_1)\sqrt{1-|\langle\Psi(t)|\Psi^\star\rangle|^2}|\langle\Psi(t)|\Psi^\star\rangle|$, Then under the dynamics*

$$\frac{d}{dt}|\Psi\rangle=-\operatorname{Tr}_1(\mathbf{Y}([\mathbf{M},|\Psi\rangle\langle\Psi|]\otimes\mathbf{I}_{d\times d}))|\Psi\rangle+\frac{1}{pZ}\sum_{l=1}^{p}i\varepsilon_l\mathbf{H}_l|\Psi\rangle \tag{7.52}$$

the output states converges to the ground state:

$$1 - |\langle \Psi(t)|\Psi^\star\rangle|^2 \leq \exp(-\frac{\lambda_2 - \lambda_1}{4\log d} t). \tag{7.53}$$

Proof for Lemma 7.4.2. Let $\mathbb{E}(t) := \mathbf{Y}(t) - (\mathbf{W} - \frac{1}{d}\mathbf{I}_{d^2\times d^2})$ denote the deviation of $\mathbf{Y}(t)$ from its expected value. The matrix that governs the dynamics can be expressed as

$$\mathrm{Tr}_1(\mathbf{Y}(t)([\mathbf{M}, |\Psi(t)\rangle\langle\Psi(t)|] \otimes \mathbf{I}_{d\times d})) = [\mathbf{M}, |\Psi(t)\rangle\langle\Psi(t)|] + E(t) \tag{7.142}$$

where

$$E(t) := \mathrm{Tr}_1\left(\mathbb{E}(t)([\mathbf{M}, |\Psi(t)\rangle\langle\Psi(t)|] \otimes \mathbf{I}_{d\times d})\right) \tag{7.143}$$

Define h as $|\langle\Psi^\star|\Psi(t)||\rangle^2$, the time derivative of h

$$\frac{d}{dt}h = (\frac{d}{dt}|\Psi(t)\rangle)^\dagger|\Psi^\star\rangle\langle\Psi^\star|\Psi(t)|+\rangle\langle\Psi(t)|\Psi^\star|\langle\rangle|\Psi^\star\frac{d}{dt}|\Psi(t)\rangle \tag{7.144}$$

$$= 2(\langle\Psi(t)|\mathbf{M}|\Psi(t)\rangle - \lambda_1)|\langle\Psi^\star|\Psi(t)||\rangle^2 \tag{7.145}$$

$$+ \mathrm{Tr}(E(t)[|\Psi^\star\rangle\langle\Psi^\star|, |\Psi(t)\rangle\langle\Psi(t)|]) \tag{7.146}$$

$$+ \mathrm{Tr}(N(t)[|\Psi^\star\rangle\langle\Psi^\star|, |\Psi(t)\rangle\langle\Psi(t)|]) \tag{7.147}$$

with $N(t)$ defined as $-\frac{1}{pZ}\sum i\varepsilon_{lt}\mathbf{H}_l$. The first term corresponds to the actual Riemannian gradient flow on the sphere:

$$2(\langle\Psi(t)|\mathbf{M}|\Psi(t)\rangle - \lambda_1)|\langle\Psi^\star|\Psi(t)|\rangle^2 = 2(\langle\Psi(t)|\mathbf{M}|\Psi(t)\rangle - \lambda_1)h \tag{7.148}$$

$$\geq 2((1-h)\lambda_2 + h\lambda_1 - \lambda_1)h \tag{7.149}$$

$$= 2(\lambda_2 - \lambda_1)(1-h)h \tag{7.150}$$

The second term stems from the deviation of $\mathbf{Y}$ from its expectation:

$$\mathrm{Tr}(E(t)[|\Psi^\star\rangle\langle\Psi^\star|, |\Psi(t)\rangle\langle\Psi(t)|]) \tag{7.151}$$

$$= \mathrm{Tr}\left(\mathrm{Tr}_1\left(\mathbb{E}(t)([\mathbf{M}, |\Psi(t)\rangle\langle\Psi(t)|] \otimes \mathbf{I}_{d\times d})\right)[|\Psi^\star\rangle\langle\Psi^\star|, |\Psi(t)\rangle\langle\Psi(t)|]\right) \tag{7.152}$$

$$= \mathrm{Tr}\left(\mathbb{E}(t)([\mathbf{M}, |\Psi(t)\rangle\langle\Psi(t)|] \otimes [|\Psi^\star\rangle\langle\Psi^\star|, |\Psi(t)\rangle\langle\Psi(t)|])\right) \tag{7.153}$$

$$\geq - \|\mathbb{E}(t)\|\|[\mathbf{M}, |\Psi(t)\rangle\langle\Psi(t)|] \otimes [|\Psi^\star\rangle\langle\Psi^\star|, |\Psi(t)\rangle\langle\Psi(t)|]\|_{\mathsf{tr}} \tag{7.154}$$

$$= - 2\sqrt{h(1-h)}\|\mathbb{E}(t)\|\|[\mathbf{M}, |\Psi(t)\rangle\langle\Psi(t)|]\|_{\mathsf{tr}} \tag{7.155}$$

$$= - 2\sqrt{d}\sqrt{h(1-h)}\|\mathbb{E}(t)\|\|[\mathbf{M}, |\Psi(t)\rangle\langle\Psi(t)|]\|_F \tag{7.156}$$

$$= - 2\sqrt{2}\sqrt{d}\sqrt{h}(1-h)(\lambda_d - \lambda_1)\|\mathbb{E}(t)\| \tag{7.157}$$

$$= - 0.5h(1-h)(\lambda_2 - \lambda_1)\frac{1}{\sqrt{hd}} \tag{7.158}$$

Here we use technical Lemma 7.8.2 and 7.8.3.

The third term is a result of inaccurate estimation of gradients:

$$\mathrm{Tr}(N(t)[|\Psi^\star\rangle\langle\Psi^\star|, |\Psi(t)\rangle\langle\Psi(t)|]) \geq -2\|N(t)\|\sqrt{h(1-h)} \geq -(\lambda_2-\lambda_1)h(1-h) \tag{7.159}$$

Here we use the fact that $\|N(t)\| \leq \frac{1}{2}(\lambda_2 - \lambda_1)\sqrt{h(1-h)}$ if $\|\boldsymbol{\varepsilon}\|_\infty \leq \frac{Z}{2\|\mathbf{H}\|}(\lambda_2 - \lambda_1)\sqrt{h(1-h)}$.

Combining all three terms, we have

$$\frac{d}{dt}h \geq (\lambda_2 - \lambda_1)(1-h)h(1 - \frac{1}{2\sqrt{hd}}) \tag{7.160}$$

Following the technical Lemma 4 in [266]

$$\frac{d}{dt}\log(-\log(h)) \leq -(\lambda_2 - \lambda_1)\frac{1 - \frac{1}{2\sqrt{hd}}}{1 - \log h} \tag{7.161}$$

Conditioned on $h(0) \geq \frac{1}{d}$, the absolute value of the right-hand side is lower bounded by $\frac{\lambda_2 - \lambda_1}{4\log d}$.

□

7.8.3 Proof of Corollary 7.5.1

The proof of Corollary 7.5.1 involves replacing the integration formula in the proof to the main theorem with integration over subgroups. We start by presenting a basic fact about block-diagonal matrices (Lemma 7.8.5) and the integration formula for subgroups of $SU(d)$ (Lemma 7.8.6).

Lemma 7.8.5 (Basic Fact)**.** *Let G be a matrix subgroup of $SU(d)$ inducing a decomposition of invariant subspace $V = \oplus_{j=1}^{m} V_j$ with projections $\{\mathbf{\Pi}_j\}_{j=1}^{m}$. Without loss of generality, assume V_1 to be the subspace of interest: for any Hermitian $\mathbf{A}$ and unitary matrix $\mathbf{U}$ in group G:*

$$\mathbf{\Pi}_1\mathbf{U}\mathbf{A}\mathbf{U}^{\dagger}\mathbf{\Pi}_1 = \mathbf{\Pi}_1\mathbf{U}\mathbf{\Pi}_1 \cdot \mathbf{\Pi}_1\mathbf{A}\mathbf{\Pi}_1 \cdot \mathbf{\Pi}_1\mathbf{U}^{\dagger}\mathbf{\Pi}_1 \tag{7.162}$$

Proof. The decomposition of invariant subspaces dictates that any $\mathbf{U} \in G$ is block-diagonal under $\{\mathbf{\Pi}_j\}_{j=1}^m$, namely $forall \mathbf{U} \in G, \forall j \neq j', \mathbf{\Pi}_{j'}\mathbf{U}\mathbf{\Pi}_j = 0$.

$$\mathbf{\Pi}_1\mathbf{U}\mathbf{A}\mathbf{U}^\dagger\mathbf{\Pi}_1 \tag{7.163}$$
$$=\mathbf{\Pi}_1\mathbf{U}\sum_{j=1}^m \mathbf{\Pi}_j\mathbf{A}\sum_{j'=1}^m \mathbf{\Pi}_{j'}\mathbf{U}^\dagger\mathbf{\Pi}_1 \tag{7.164}$$
$$=\sum_{j,j'\in[m]} (\mathbf{\Pi}_1\mathbf{U}\mathbf{\Pi}_j)\mathbf{A}(\mathbf{\Pi}_{j'}\mathbf{U}^\dagger\mathbf{\Pi}_1) \tag{7.165}$$
$$=\mathbf{\Pi}_1\mathbf{U}\mathbf{\Pi}_1\mathbf{A}\mathbf{\Pi}_1\mathbf{U}^\dagger\mathbf{\Pi}_1 \tag{7.166}$$
$$=\mathbf{\Pi}_1\mathbf{U}\mathbf{\Pi}_1\mathbf{\Pi}_1\mathbf{A}\mathbf{\Pi}_1\mathbf{\Pi}_1\mathbf{U}^\dagger\mathbf{\Pi}_1 \tag{7.167}$$

The last equation uses the property of projections $\mathbf{\Pi}_j^2 = \mathbf{\Pi}_j$. □

As a direct result, we have the following generic integral formula for $\mathbf{U}$ sampled from any $\mathcal{D}$ supported on the subgroup G:

Lemma 7.8.6 (Integration Formula on subgroup restricted to an invariant subspace)**.** *Let G be a matrix subgroup of $SU(d)$ inducing a decomposition of invariant subspace $V = \oplus_{j=1}^m V_j$ with projections $\{\mathbf{\Pi}_j\}_{j=1}^m$. Without loss of generality, assume V_1 to be the subspace of interest. For any Hermitians $\{\mathbf{A}_r\}_{r=1}^R$ and measure $\mathcal{D}$ over G:*

$$(\mathbf{Q}^\dagger)^{\otimes R}\mathbb{E}_{\mathbf{U}\sim\mathcal{D}}[\otimes_{r=1}^R \mathbf{U}\mathbf{A}_r\mathbf{U}^\dagger]\mathbf{Q}^{\otimes R} = \mathbb{E}_{\mathbf{U}^{(1)}\sim\mathcal{D}^{(1)}}[\otimes_{r=1}^R \mathbf{U}^{(1)}\mathbf{A}_r^{(1)}(\mathbf{U}^{(1)})^\dagger] \tag{7.168}$$

where $\mathcal{D}^{(1)}$ is the distribution of $\mathbf{Q}^\dagger\mathbf{U}\mathbf{Q}$ for $\mathbf{U}$ sampled with respect to $\mathcal{D}$, and $\mathbf{A}_r^{(1)} := \mathbf{Q}^\dagger\mathbf{A}_r\mathbf{Q}$ is the Hermitian $\mathbf{A}_r$ restricted to the subspace V_1.

Lemma 7.8.6 allows using the integration formula in [271] when $\mathcal{D}^{(1)}$ is the Haar measure over a special unitary, special orthogonal or symplectic group. We are now ready to present the proof of Corollary 7.5.1.

Proof of Corollary 7.5.1 Without loss of generality, we assume V_1 to be the relevant subspace with projection $\mathbf{\Pi}_1 = \mathbf{Q}\mathbf{Q}^\dagger$. For concise notations, define $\mathbf{U}^{(1)} = \mathbf{Q}^\dagger\mathbf{U}\mathbf{Q}$, $\mathbf{A}^{(1)} = \mathbf{Q}^\dagger\mathbf{A}\mathbf{Q}$ and $|\Psi^{(1)}\rangle = \mathbf{Q}^\dagger|\Psi\rangle$ for any unitary $\mathbf{U}$, Hermitian $\mathbf{A}$ and vector $|\Psi\rangle$.

Note that the potential function we track in the proof of Theorem 7.3.1 $|\langle\Psi^\star|\Psi(t)\rangle|^2 = |\langle\Psi^{(1),\star}|\Psi^{(1)}(t)\rangle|^2$ if both $|\Psi\rangle^\star$ and $|\Psi(t)\rangle \in V_1$. Therefore for the purpose of the proof it suffices to track the dynamics of $|\Psi^{(1)}(t)\rangle$. Below we (1) establish that $|\Psi(t)\rangle \in V_1$ through out the training and (2) show that the dynamics of $|\Psi^{(1)}(t)\rangle$ takes the same form as stated in Lemma 7.3.1 by replacing $\mathbf{M}$ and $\mathbf{H}$ with $\mathbf{M}^{(1)} = \mathbf{Q}^\dagger\mathbf{M}\mathbf{Q}$ and $\mathbf{H}^{(1)} = \mathbf{Q}^\dagger\mathbf{H}\mathbf{Q}$.

By Lemma 7.3.1, the dynamics of $|\Psi\rangle$ takes the form

$$\frac{d}{dt}|\Psi\rangle \propto -\frac{1}{p}\sum_{l=1}^{p}\mathrm{Tr}([\mathbf{M}, |\Psi(t)\rangle\langle\Psi(t)|]\mathbf{U}_{l:p}\mathbf{H}\mathbf{U}_{l:p}^\dagger)\mathbf{U}_{l:p}\mathbf{H}\mathbf{U}_{l:p}^\dagger|\Psi(t)\rangle \tag{7.169}$$

We first show that $|\Psi(t)\rangle$ remains in V_1 for all t (i.e. $|\Psi(t)\rangle = \mathbf{\Pi}_1|\Psi(t)\rangle$). It suffices to show the time derivate $\frac{d|\Psi\rangle}{dt}$ stays in V_1 for $|\Psi\rangle \in V_1$:

$$\mathbf{\Pi}_1\mathbf{U}_{l:p}\mathbf{H}\mathbf{U}_{l:p}^\dagger|\Psi\rangle \tag{7.170}$$

$$=\mathbf{U}_{l:p}\mathbf{\Pi}_1\mathbf{H}\mathbf{U}_{l:p}^\dagger|\Psi\rangle \tag{7.171}$$

$$=\mathbf{U}_{l:p}\mathbf{H}\mathbf{\Pi}_1\mathbf{U}_{l:p}^\dagger|\Psi\rangle \tag{7.172}$$

$$=\mathbf{U}_{l:p}\mathbf{H}\mathbf{U}_{l:p}^\dagger\mathbf{\Pi}_1|\Psi\rangle \tag{7.173}$$

$$=\mathbf{U}_{l:p}\mathbf{H}\mathbf{U}_{l:p}^\dagger|\Psi\rangle \tag{7.174}$$

The first and the third equality is because $\mathbf{U}_{l:p} \in G_{\mathcal{A}}$ for all l and therefore block-diagoanl under $\{\mathbf{\Pi}_j\}_{j=1}^m$; The second equality is because and $\mathbf{H}$ is block-diagoanl under $\{\mathbf{\Pi}_j\}_{j=1}^m$; The last equality follows from $|\Psi\rangle \in V_j$.

We now calculate the dynamics of $|\Psi^{(1)}(t)\rangle$. For the trace operation in each term,

$$\mathrm{Tr}([\mathbf{M}, |\Psi(t)\rangle\langle\Psi(t)|]\mathbf{U}_{l:p}\mathbf{H}\mathbf{U}_{l:p}^\dagger) \tag{7.175}$$

$$= \mathrm{Tr}([|\Psi(t)\rangle\langle\Psi(t)|, \mathbf{U}_{l:p}\mathbf{H}\mathbf{U}_{l:p}^\dagger]\mathbf{M}) \tag{7.176}$$

$$= \mathrm{Tr}([\mathbf{\Pi}_1|\Psi(t)\rangle\langle\Psi(t)|\mathbf{\Pi}_1, \mathbf{U}_{l:p}\mathbf{H}\mathbf{U}_{l:p}^\dagger]\mathbf{M}) \tag{7.177}$$

$$= \mathrm{Tr}([\mathbf{\Pi}_1|\Psi(t)\rangle\langle\Psi(t)|\mathbf{\Pi}_1, \mathbf{\Pi}_1\mathbf{U}_{l:p}\mathbf{\Pi}_1\mathbf{H}\mathbf{\Pi}_1\mathbf{U}_{l:p}^\dagger\mathbf{\Pi}_1]\mathbf{M}) \tag{7.178}$$

$$= \mathrm{Tr}([\mathbf{\Pi}_1|\Psi(t)\rangle\langle\Psi(t)|\mathbf{\Pi}_1, \mathbf{\Pi}_1\mathbf{U}_{l:p}\mathbf{\Pi}_1\mathbf{H}\mathbf{\Pi}_1\mathbf{U}_{l:p}^\dagger\mathbf{\Pi}_1]\mathbf{\Pi}_1\mathbf{M}\mathbf{\Pi}_1) \tag{7.179}$$

$$= \mathrm{Tr}([\mathbf{Q}^\dagger|\Psi(t)\rangle\langle\Psi(t)|\mathbf{Q}, \mathbf{Q}^\dagger\mathbf{M}\mathbf{Q}]\mathbf{Q}^\dagger\mathbf{U}_{l:p}\mathbf{Q}\mathbf{Q}^\dagger\mathbf{H}\mathbf{Q}\mathbf{Q}^\dagger\mathbf{U}_{l:p}^\dagger\mathbf{Q}) \tag{7.180}$$

The first, fourth and the fifth equation follows from basic properties of trace operators; the second equality uses the fact that $|\Psi(t)\rangle$ remains in V_j; the third equality uses the fact that $\mathbf{U}_{l:p}$ and $\mathbf{H}$ are block-diagonal. Therefore we can rewrite Eq (7.169) as

$$\frac{d}{dt}|\Psi^{(1)}(t)\rangle \propto -\frac{1}{p}\sum_{l=1}^{p}\mathrm{Tr}([\mathbf{M}^{(1)}, |\Psi^{(1)}(t)\rangle\langle\Psi^{(1)}(t)|]\mathbf{U}_{l:p}^{(1)}\mathbf{H}^{(1)}(\mathbf{U}_{l:p}^{(1)})^\dagger)\mathbf{U}_{l:p}^{(1)}\mathbf{H}^{(1)}(\mathbf{U}_{l:p}^{(1)})^\dagger|\Psi^{(1)}(t)\rangle \tag{7.181}$$

The dynamics of $|\Psi^{(1)}(t)\rangle$ depends on $\mathbf{Q}^\dagger\mathbf{M}\mathbf{Q}$, $\mathbf{Q}^\dagger\mathbf{H}\mathbf{Q}$ and $\mathbf{Q}^\dagger\mathbf{U}\mathbf{Q}$. The corollary follows trivially by using the integration formula specified in Lemma 7.8.6.

7.9 More on the empirical Study

Implementation of partially-trainable ansatz We implement the partially-trainable ansatz (Definition 7.2.2) by approximate the Haar measure over $G_{\mathcal{A}}$ by calculating

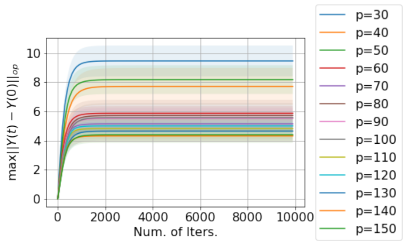
$$\mathrm{U}(\boldsymbol{\phi}) = \prod_{l'=1}^{L_{\mathsf{sample}}} \prod_{k=1}^{K} \exp(-i\phi_{l',k}\mathbf{H}_k) \tag{7.182}$$

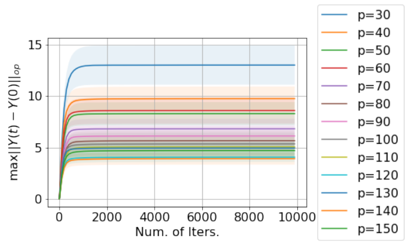
for $L_{\mathsf{sample}} = 20$ and randomly initialized $\{\phi_{l',k}\}_{k\in[K],l'\in[L_{\mathsf{sample}}]}$.

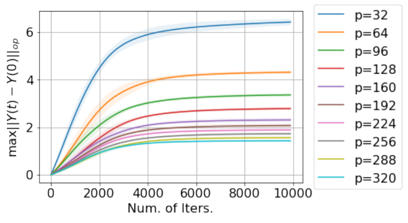
Deviation of $\mathbf{Y}$ and $\boldsymbol{\theta}$ as functions of time t In Figure 7.16 and Figure 7.17, we plot the deviation of $\mathbf{Y}$ and $\boldsymbol{\theta}$ as functions of time steps t for both the partially- and fully-trainable settings. The mean values are plotted in solid lines and the shaded areas represent the standard deviation over random initializations.

The maximum time steps is set to be $10,000$. As observed in Figure 7.16 and 7.17, the deviation of$\mathbf{Y}$ and $\boldsymbol{\theta}$ saturates quickly after a few time steps.

Definition of Toy problems For the toy problem with system dimension d, effective dimension d_{eff} and the effective spectral ratio κ_{eff}, we embed a $d_{\mathsf{eff}} \times d_{\mathsf{eff}}$ problem Hamiltonian $\mathbf{M}^{(1)} = \mathbf{Q}^{\dagger}\mathbf{M}\mathbf{Q}$ with eigenvalues $(0, \frac{1}{\kappa_{\mathsf{eff}}}, 1, \cdots, 1)$, generators $\mathbf{H}^{(1)} = \mathbf{Q}^{\dagger}\mathbf{H}\mathbf{Q}$ and unitaries $\{\mathbf{U}_l^{(1)} = \mathbf{Q}^{\dagger}\mathbf{U}_l\mathbf{Q}\}_{l=1}^{p}$ into a d-dimensional space using arbitrary $d \times d$ unitary $\mathbf{U}_{\mathsf{embed}} = \begin{bmatrix} \mathbf{Q} & \mathbf{Q}^{\perp} \end{bmatrix}$ with


(a) Partially-trainable HVA


(b) Fully-trainable HVA


(c) Partially-trainable HEA

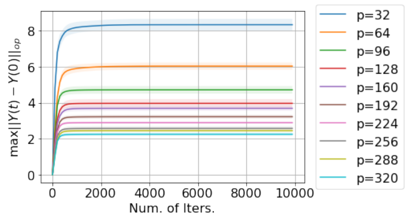
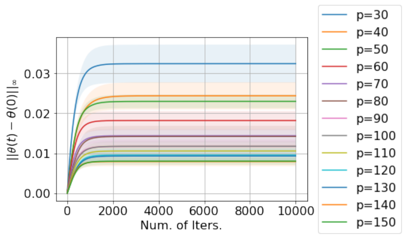
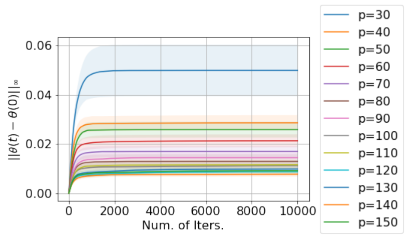
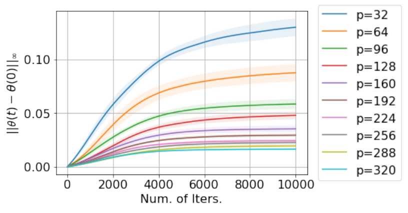

(d) Fully-trainable HEA

Figure 7.16: Deviation of $\mathbf{Y}$ during training for HVA and HEA


(a) Partially-trainable HVA


(b) Fully-trainable HVA


(c) Partially-trainable HEA

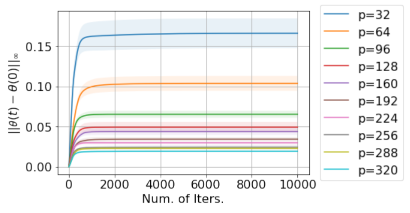

(d) Fully-trainable HEA

Figure 7.17: Deviation of $\boldsymbol{\theta}$ during training for HVA and HEA

$\mathbf{Q}^{\perp}$ being arbitrary complementary columns of $\mathbf{Q}$:

$$\mathbf{M} = \mathbf{U}_{\text{embed}} \begin{bmatrix} \mathbf{M}^{(1)} & 0 \\ 0 & \mathbf{I}_{d-d_{\text{eff}} \times d-d_{\text{eff}}} \end{bmatrix} \mathbf{U}_{\text{embed}}^{\dagger} \tag{7.183}$$

$$\mathbf{H} = \mathbf{U}_{\text{embed}} \begin{bmatrix} \mathbf{H}^{(1)} & 0 \\ 0 & \mathbf{I}_{d-d_{\text{eff}} \times d-d_{\text{eff}}} \end{bmatrix} \mathbf{U}_{\text{embed}}^{\dagger} \tag{7.184}$$

$$\mathbf{U}_l = \mathbf{U}_{\text{embed}} \begin{bmatrix} \mathbf{U}_l^{(1)} & 0 \\ 0 & \mathbf{I}_{d-d_{\text{eff}} \times d-d_{\text{eff}}} \end{bmatrix} \mathbf{U}_{\text{embed}}^{\dagger}, \quad \forall l \in [p] \tag{7.185}$$

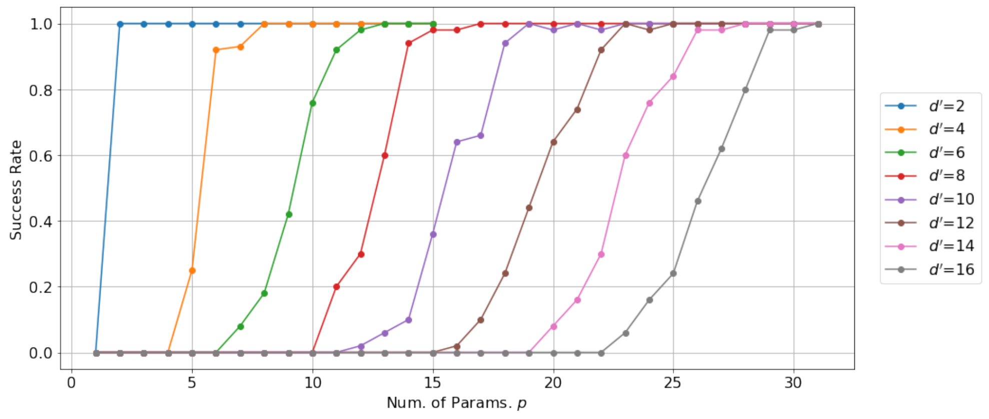
And the ansatz takes the form

$$\mathbf{U}(\boldsymbol{\theta}) = \big(\prod_{l=1}^{p} \mathbf{U}_l \exp(-i\theta_l \mathbf{H}) \big) \cdot \mathbf{U}_0 \tag{7.186}$$

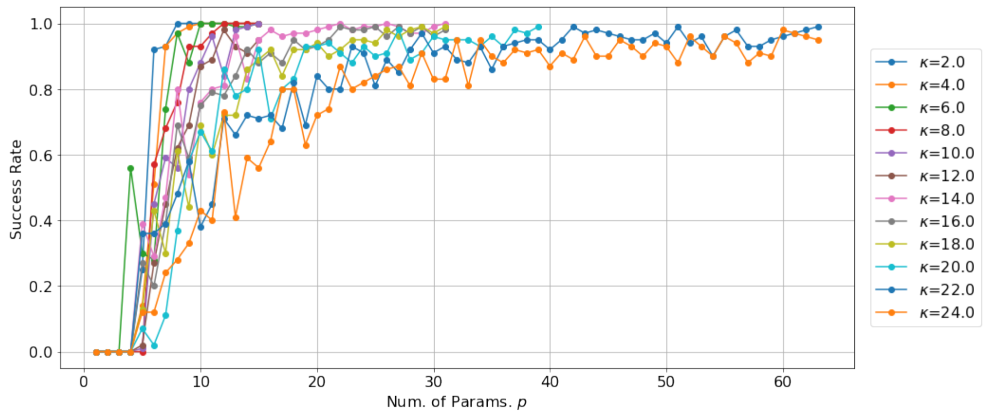
where $d_{\text{eff}} \times d_{\text{eff}}$ unitaries $\{\mathbf{U}_l^{(1)}\}$ are sampled i.i.d from the Haar measure over $SU(d_{\text{eff}})$.

In Figure 7.18, we plot the success rate versus the number of parameters for various d_{eff} and κ_{eff} that are used to generate Figure 7.10.

Estimating the invariant subspace for TFI and XXZ models Similar to the Kitaev model in Section 7.5.2, we numerically confirm that the TFI and XXZ models involved are all compatible. The convergences of the empirical estimatino of projection $\hat{\boldsymbol{\Pi}}$ are summarized in Figure 7.19, Figure 7.20, Figure 7.21 and Figure 7.22

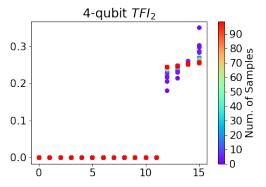


(a) Varying effective dimension d_{eff}

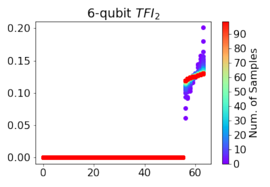


(b) Varying effective ratio κ_{eff}

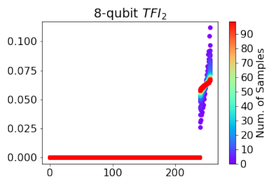
Figure 7.18: The success rate for achieving a 99%-approximation for the ground state as a function of number of parameters. Each curve corresponds to a toy instance with dimension 16 and with varying $(d_{\text{eff}}, \kappa_{\text{eff}})$. Success rates are estimated over 100 random initializations. Top: Fixing $d = 16, \kappa_{\text{eff}} = 4.0$ for $d_{\text{eff}} = 2, 4, 6, \cdots, 16$. The threshold increases as the system dimension increases. Bottom: Fixing $d = 16, d_{\text{eff}} = 4$ for $\kappa_{\text{eff}} = 2.0, 4.0, 6.0, \cdots, 24.0$. The threshold is positively correlated to the condition number of the system.



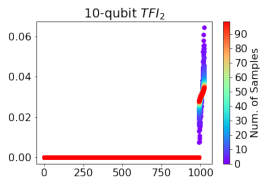
(a) N=4



(b) N=6

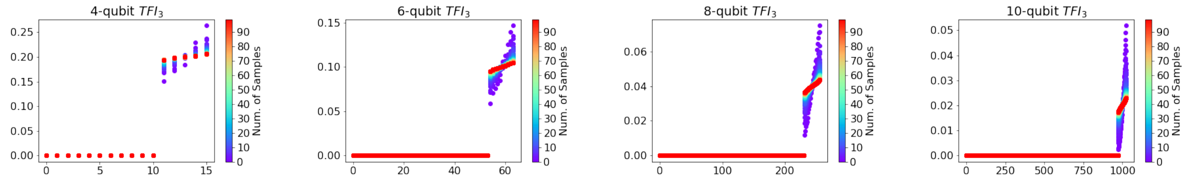


(c) N=8



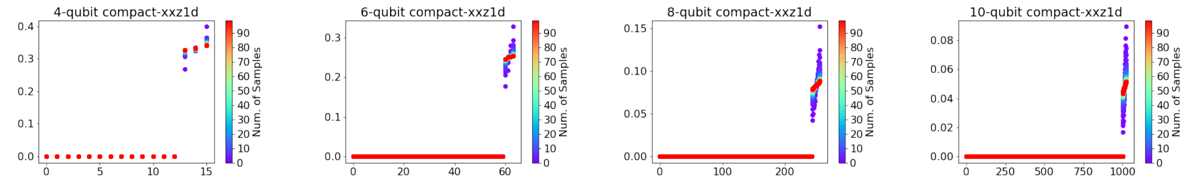
(d) N=10

Figure 7.19: Specturm of $\hat{\Pi}$ for TFI_2 model with $4, 6, 8, 10$ qubits



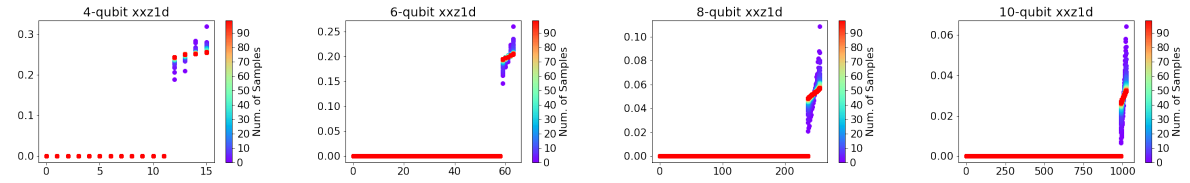
(a) N=4 (b) N=6 (c) N=8 (d) N=10

Figure 7.20: Specturm of $\hat{\Pi}$ for TFI_3 model with $4, 6, 8, 10$ qubits



(a) N=4 (b) N=6 (c) N=8 (d) N=10

Figure 7.21: Specturm of $\hat{\Pi}$ for XXZ_4 model with $4, 6, 8, 10$ qubits



(a) N=4 (b) N=6 (c) N=8 (d) N=10

Figure 7.22: Specturm of $\hat{\Pi}$ for XXZ_6 model with $4, 6, 8, 10$ qubits

Chapter 8: Conclusions

This dissertation analyzes the applications of quantum computing in several areas of machine learning and optimization.

- Chapters 2, 3, and 4 present quantum algorithms with provable speedups for some important tasks including general convex optimization (Chapter 2), the estimation of the volume of convex bodies (Chapter 3), and for matrix games with applications including linear classification and SVMs (Chapter 4).
- Chapters 5, 6, and 7 consider the applications of quantum machine learning methods whose parameters are then optimized by classical methods. We explore the design of generative models (Chapter 5), expressivity of Quantum Neural Networks (Chapter 6), and the optimization of Variational Quantum Eigensolvers (Chapter 7).

These results motivate a search for further applications of quantum computing in machine learning and associated large optimization tasks. The following summarizes some general directions that may be worth exploring in the future as extensions of this work:

Structured Convex Optimization Problems. Our results on convex optimization provide a quantum speedup for the problem in its most general form where the only constraint placed on the objective function and feasible set is convexity. In most important convex optimization problems however, the form of the objective function is *explicit* which allows for more efficient algorithms than those for general convex optimization (see for instance the situation for linear classification or SVMs [26, 126]). This raises the natural question of whether quantum speedups can be obtained for such structured problems. In the common situation where the optimization problem has an efficiently computable gradient, there have recently been some negative results in this direction: [277, 278] have shown that classical *accelerated gradient descent* algorithms are optimal in such settings. The setting where second order or higher derivative information is available remains open. Such settings are interesting both for regression problems [279], and non convex optimization [280]. It is also interesting to study such problems in the non-oracular setting where we seek only a runtime improvement: [281, 282] present some algorithms that may have runtime advantages over classical interior point methods. Making these advantages rigorous or finding a clearer characterization of problems where they may occur can also be very advantageous in the implementation of quantum optimization solvers.

Non-convex Optimization Problems. Non-convex optimization optimizations have become more significant in machine learning as they model the loss functions of deep neural networks. Despite the success of gradient methods in many cases, optimizing networks often presents a fundamental challenge for their deployment and indeed has influenced the design of popular architectures. This raises the natural question of whether quantum algorithms can provide speedups for convex optimization. While the general problem is NP-Hard, several algorithms focus on finding local optima, since there exist landscape results in many settings that indicate their closeness to the global optima [283, 284]. [285] have recently given a quantum speedup for the problem of escaping saddle points. It would be of interest to obtain quantum speedups for some problems where the complete runtime is known (consider for example the case with over-parameterized neural networks.) Such an algorithm could in principle yield a quantum speedup for training classical neural networks.

Quantum Sampling Algorithms. Sampling from log-concave densities is a computational challenge in statistics, optimization and machine learning. It finds applications in estimation, inference, and optimization algorithm. Several problems rely on sampling over bodies specified by explicit constraints. In the setting of sampling from the uniform density over an n-dimensional polytope specified by m constraints, the volume estimation algorithm would simply require $O(mn^5)$ queries to the entries of the constraint matrix ($O(n^4)$ membership queries each costing $O(mn)$). The best known classical algorithm for this task [80] instead achieves an oracle cost of $O(m^2 n^{\omega-1/3})$ where ω is the exponent of matrix multiplication. This is more efficient than the general algorithm whenever $m \leq n$. This raises the question of whether quantum speedups can be obtained in this case, with costs dependent on m. [80] replaces the MCMC algorithm based on the hit-and-run walk algorithm with Hamiltonian Monte-Carlo algorithms where the trajectory of the random walk is given by the stochasticized trajectory of a partial differential equation. A recent classical paper [286] has characterized the spectral gap of Hamiltonian Monte-Carlo for κ-conditioned log-concave functions to be $\Omega(1/\kappa)$. This suggests the investigation of whether $\sqrt{\kappa}$ quantum speedup over the best classical algorithm can be obtained using a quantum analogue of Hamiltonian Monte-Carlo.

Further understanding of Variational Quantum Algorithms. Our results on variational quantum algorithms provide some promising early results towards the ultimate goal of demonstrating quantum advantage on near-term quantum algorithms. Further progress is possible along the lines of each of the projects presented in the dissertation. Our work on generative quantum learning in Chapter 5 [58] demonstrates the utility of variational quantum circuits in approximate circuit compression. The scale of circuits treated is currently limited by simulation constraints, and a careful analysis and engineering of the resource requirements of the system could allow for the system to be run on a real quantum system. This could allow for the pre-computed approximate compression of larger circuits.

A better understanding of the expressivity of Quantum Neural Networks is also essential to choosing proper avenues for their application. Our work indicates that large advantages may exist for the majority of functions expressed by QNNs of a fixed size, but that such advantages may be difficult to establish rigorously. There are two directions that could be pursued for further progress: 1) We search for exponential advantages over neural networks of depth ≤ 3. Our complexity theoretic barriers do not apply to this case. If a rigorous separation could be found, then the particular functions that yield the separation could yield a characterization of *hard* quantum functions. 2) We increase the scale of our empirical study to obtain a larger sample size for interpolation. This may be through running experiments on real quantum systems, improving the simulation performance, or through finding easily simulable QNNs that also yield an empirical expressive advantage.

It may be also valuable to further investiagate the convergence of quantum systems. Our results show that sufficient overparameterization is *sufficient* for the convergence of VQEs, but for a general quantum ansatz the degree of overparameterization required is large enough to remove any possible quantum advantage. We do show that carefully chosen ansatz can reduce this requirement, but all the cases we investigated still have exponential dependence on the number of qubits in the system. Finer analyses of the parameterization threshold is required to give even an exponential algorithm that is faster than a classical optimization algorithm for the same problem. It is also important to study the convergence of other variational algorithms like QNNs and variants such as Convolutional QNNs that have valuable properties such as not suffering from the barren plateau phenomenon.

Finally, one may consider the extension of all the above ideas to other forms of parameterization of quantum systems. Analog quantum systems based on direct manipulation of control pulses have gained interest for near term applications due to the possibility of more resource efficient optimizations [287, 288], and these interfaces are starting to become available on commercial systems (see for eg. [289]). These systems offer the potential of new parameterizations which have different optimization and generalization properties while sharing the potential for large speedups over classical simulation. These properties can be studied by forming hypotheses from the lens of our results on the circuit model, as well as via new theoretical and empirical analyses.

Bibliography

[1] Lov K. Grover. A different kind of quantum search, 2005.

[2] D. Zhu, N. M. Linke, M. Benedetti, K. A. Landsman, N. H. Nguyen, C. H. Alderete, A. Perdomo-Ortiz, N. Korda, A. Garfoot, C. Brecque, L. Egan, O. Perdomo, and C. Monroe. Training of quantum circuits on a hybrid quantum computer. 2018.

[3] Andy CY Li, M Sohaib Alam, Thomas Iadecola, Ammar Jahin, Doga Murat Kurkcuoglu, Richard Li, Peter P Orth, A Barış Özgüler, Gabriel N Perdue, and Norm M Tubman. Benchmarking variational quantum eigensolvers for the square-octagon-lattice kitaev model. *arXiv preprint arXiv:2108.13375*, 2021.

[4] Paul Benioff. The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. *Journal of Statistical Physics*, 22(5):563–591, May 1980.

[5] Richard P. Feynman. Simulating Physics with Computers. *International Journal of Theoretical Physics*, 21(6-7):467–488, June 1982.

[6] David Deutsch and Richard Jozsa. Rapid Solution of Problems by Quantum Computation. *Proceedings of the Royal Society of London Series A*, 439(1907):553–558, December 1992.

[7] Daniel R. Simon. On the power of quantum computation. *SIAM J. Comput.*, 26(5):1474–1483, October 1997.

[8] Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Review*, 41(2):303–332, January 1999.

[9] Lov K. Grover. A framework for fast quantum mechanical algorithms. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, page 53–62, New York, NY, USA, 1998. Association for Computing Machinery.

[10] Andrew M. Childs, Robin Kothari, and Rolando D. Somma. Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. *SIAM Journal on Computing*, 46(6):1920–1950, 2017.

[11] Ashley Montanaro. Quantum speedup of Monte Carlo methods. 471(2181):20150301, 2015.

[12] Edward Farhi and Hartmut Neven. Classification with quantum neural networks on near term processors, 2018.

[13] Seth Lloyd and Christian Weedbrook. Quantum generative adversarial learning. *Phys. Rev. Lett.*, 121:040502, Jul 2018.

[14] Srinivasan Arunachalam and Ronald de Wolf. Optimal quantum sample complexity of learning algorithms. *Journal of Machine Learning Research*, 19(71):1–36, 2018.

[15] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. *Nature Physics*, 10(9):631–633, September 2014.

[16] Patrick Rebentrost, Adrian Steffens, Iman Marvian, and Seth Lloyd. Quantum singular-value decomposition of nonsparse low-rank matrices. *Phys. Rev. A*, 97:012327, Jan 2018.

[17] Fernando G.S.L. Brandão and Krysta Svore. Quantum speed-ups for semidefinite programming. In *Proceedings of the 58th Annual Symposium on Foundations of Computer Science*, pages 415–426, 2017.

[18] Fernando G.S.L. Brandão, Amir Kalev, Tongyang Li, Cedric Yen-Yu Lin, Krysta M. Svore, and Xiaodi Wu. Quantum SDP solvers: Large speed-ups, optimality, and applications to quantum learning. *arXiv:1710.02581*, 2017.

[19] Joran van Apeldoorn and András Gilyén. Improvements in Quantum SDP-Solving with Applications. *arXiv e-prints*, page arXiv:1804.05058, April 2018.

[20] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15):150502, 2009.

[21] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: Exponential improvements for quantum matrix arithmetics. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 193–204, New York, NY, USA, 2019. Association for Computing Machinery.

[22] Iordanis Kerenidis and Anupam Prakash. Quantum recommendation systems. In *Proceedings of the 8th Innovations in Theoretical Computer Science Conference*, pages 49:1–49:21, 2017. arXiv:1603.08675.

[23] Ewin Tang. A quantum-inspired classical algorithm for recommendation systems. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 217–228, New York, NY, USA, 2019. Association for Computing Machinery.

[24] Shouvanik Chakrabarti, Andrew M. Childs, Tongyang Li, and Xiaodi Wu. Quantum algorithms and lower bounds for convex optimization. *Quantum*, 4:221, January 2020.

[25] Shouvanik Chakrabarti, Andrew M. Childs, Shih-Han Hung, Tongyang Li, Chunhao Wang, and Xiaodi Wu. Quantum algorithm for estimating volumes of convex bodies, 2019.

[26] Tongyang Li, Shouvanik Chakrabarti, and Xiaodi Wu. Sublinear quantum algorithms for training linear and kernel-based classifiers. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3815–3824. PMLR, 2019.

[27] Tongyang Li, Chunhao Wang, Shouvanik Chakrabarti, and Xiaodi Wu. Sublinear classical and quantum algorithms for general matrix games. *arXiv e-prints*, page arXiv:2012.06519, December 2020.

[28] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5(1), jul 2014.

[29] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[30] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4):043001, 2019.

[31] Shouvanik Chakrabarti, Huang Yiming, Tongyang Li, Soheil Feizi, and Xiaodi Wu. Quantum wasserstein generative adversarial networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 6781–6792. Curran Associates, Inc., 2019.

[32] Shouvanik Chakrabarti, Andrew M. Childs, Tongyang Li, and Xiaodi Wu. Quantum algorithms and lower bounds for convex optimization. In *Contributed talk at the 22nd Annual Conference on Quantum Information Processing*, 2019.

[33] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[34] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, 2012.

[35] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing*, pages 302–311, 1984.

[36] George B. Dantzig and Mukund N. Thapa. *Linear programming 2: Theory and extensions*. Springer, 2006.

[37] James E. Kelley, Jr. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, 8(4):703–712, 1960.

[38] Pravin M. Vaidya. A new algorithm for minimizing convex functions over convex sets. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 338–343, 1989.

[39] Adam T. Kalai and Santosh Vempala. Simulated annealing for convex optimization. *Mathematics of Operations Research*, 31(2):253–266, 2006.

[40] László Lovász and Santosh Vempala. Fast algorithms for logconcave functions: Sampling, rounding, integration and optimization. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 57–68, 2006.

[41] Yin Tat Lee, Aaron Sidford, and Santosh S. Vempala. Efficient convex optimization with membership oracles. In *Proceedings of the 31st Conference on Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 1292–1294, 2018.

[42] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science*, pages 1049–1065, 2015.

[43] Arkadi S. Nemirovsky and David B. Yudin. *Problem complexity and method efficiency in optimization*. Wiley, 1983.

[44] Arkadi Nemirovski. Information-based complexity of convex programming. lecture notes, 1995.

[45] Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. Quantum SDP-solvers: Better upper and lower bounds. In *Proceedings of the 58th Annual Symposium on Foundations of Computer Science*, 2017.

[46] Fernando G.S.L. Brandão, Amir Kalev, Tongyang Li, Cedric Yen-Yu Lin, Krysta M. Svore, and Xiaodi Wu. Quantum SDP solvers: Large speed-ups, optimality, and applications to quantum learning, 2017.

[47] Joran van Apeldoorn and András Gilyén. Improvements in quantum SDP-solving with applications, 2018.

[48] Stephen P. Jordan. Fast quantum algorithm for numerical gradient estimation. *Physical Review Letters*, 95(5):050501, 2005.

[49] András Gilyén, Srinivasan Arunachalam, and Nathan Wiebe. Optimizing quantum optimization algorithms via faster quantum gradient computation. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1425–1444. Society for Industrial and Applied Mathematics, 2019.

[50] Patrick Rebentrost, Maria Schuld, Leonard Wossnig, Francesco Petruccione, and Seth Lloyd. Quantum gradient descent and Newton's method for constrained polynomial optimization, 2016.

[51] Iordanis Kerenidis and Anupam Prakash. Quantum gradient descent for linear systems and least squares, 2017.

[52] Andris Ambainis and Ashley Montanaro. Quantum algorithms for search with wildcards and combinatorial group testing. *Quantum Information and Computation*, 14(5&6):439–453, 2014.

[53] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87 of *Applied Optimization*. Springer, 2013.

[54] Yin Tat Lee. personal communication, 2018.

[55] Lars Hörmander. *The analysis of linear partial differential operators: Distribution theory and Fourier analysis*. Springer-Verlag, 1990.

[56] Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. Quantum oracles for convex optimization. manuscript, 2018.

[57] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.

[58] Shouvanik Chakrabarti, Yiming Huang, Tongyang Li, Soheil Feizi, and Xiaodi Wu. Quantum wasserstein GANs. In *Advances in Neural Information Processing Systems 32*. 2019. (to appear).

[59] Santosh Vempala. Geometric random walks: a survey. In *Combinatorial and Computational Geometry*, volume 52 of *Mathematical Sciences Research Institute Publications*, pages 573–612. MSRI, 2005.

[60] Imre Bárány and Zoltán Füredi. Computing the volume is difficult. *Discrete & Computational Geometry*, 2(4):319–326, 1987.

[61] György Elekes. A geometric inequality and the complexity of computing volume. *Discrete & Computational Geometry*, 1(4):289–292, 1986.

[62] Martin E. Dyer and Alan M. Frieze. On the complexity of computing the volume of a polyhedron. *SIAM Journal on Computing*, 17(5):967–974, 1988.

[63] Leonid G. Khachiyan. On the complexity of computing the volume of a polytope. *Izvestia Akad. Nauk SSSR, Engineering Cybernetics*, 3:216–217, 1988.

[64] Leonid G. Khachiyan. The problem of computing the volume of polytopes is NP-hard. *Uspekhi Mat. Nauk*, 44(3):199–200, 1989.

[65] Martin Dyer, Alan Frieze, and Ravi Kannan. A random polynomial-time algorithm for approximating the volume of convex bodies. *Journal of the ACM (JACM)*, 38(1):1–17, 1991.

[66] László Lovász and Miklós Simonovits. The mixing rate of Markov chains, an isoperimetric inequality, and computing the volume. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 346–354, 1990.

[67] David Applegate and Ravi Kannan. Sampling and integration of near log-concave functions. In *Proceedings of the 23nd Annual ACM Symposium on Theory of Computing*, pages 156–163, 1991.

[68] Martin Dyer and Alan Frieze. Computing the volume of convex bodies: a case where randomness provably helps. *Probabilistic Combinatorics and its Applications*, 44:123–170, 1991.

[69] László Lovász and Miklós Simonovits. Random walks in a convex body and an improved volume algorithm. *Random Structures & Algorithms*, 4(4):359–412, 1993.

[70] Ravi Kannan, László Lovász, and Miklós Simonovits. Random walks and an $O^*(n^5)$ volume algorithm for convex bodies. *Random Structures & Algorithms*, 11(1):1–50, 1997.

[71] László Lovász. Hit-and-run mixes fast. *Mathematical Programming*, 86(3):443–461, 1999.

[72] László Lovász and Santosh Vempala. Hit-and-run from a corner. *SIAM Journal on Computing*, 35(4):985–1005, 2006.

[73] László Lovász and Santosh Vempala. Simulated annealing in convex bodies and an $O^*(n^4)$ volume algorithm. *Journal of Computer and System Sciences*, 72(2):392–417, 2006. an earlier version appeared at the 44th Annual IEEE Symposium on Foundations of Computer Science, pp. 650-659, 2003.

[74] Pawel Wocjan, Chen-Fu Chiang, Daniel Nagaj, and Anura Abeyesinghe. Quantum algorithm for approximating partition functions. *Physical Review A*, 80(2):022340, 2009.

[75] Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum counting. In *Proceedings of the 25th International Colloquium on Automata, Languages, and Programming*, pages 820–831, 1998.

[76] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.

[77] Peter C. Richter. Quantum speedup of classical mixing processes. *Physical Review A*, 76(4):042306, 2007.

[78] Pawel Wocjan and Anura Abeyesinghe. Speedup via quantum sampling. *Physical Review A*, 78(4):042336, 2008.

[79] Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. Convex optimization using quantum oracles. *Quantum*, 4:220, 2020.

[80] Yin Tat Lee and Santosh S. Vempala. Convergence rate of Riemannian Hamiltonian Monte Carlo and faster polytope volume computation. In *Proceedings of the 50th Annual Symposium on Theory of Computing*, pages 1115–1121. ACM, 2018.

[81] Christopher M. Dawson and Michael A. Nielsen. The Solovay-Kitaev algorithm. *Quantum Information & Computation*, 6(1):81–95, 2006.

[82] Ben Cousins and Santosh Vempala. Bypassing KLS: Gaussian cooling and an $O^*(n^3)$ volume algorithm. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing*, pages 539–548. ACM, 2015.

[83] Yin Tat Lee, Aaron Sidford, and Santosh S. Vempala. Efficient convex optimization with membership oracles. In *Proceedings of the 31st Conference on Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 1292–1294, 2018.

[84] Luis Rademacher and Santosh Vempala. Dispersion of mass and the complexity of randomized geometric algorithms. *Advances in Mathematics*, 219(3):1037–1069, 2008.

[85] Mario Szegedy. Quantum speed-up of Markov chain based algorithms. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 32–41. IEEE, 2004.

[86] Frédéric Magniez, Ashwin Nayak, Jérémie Roland, and Miklos Santha. Search via quantum walk. *SIAM Journal on Computing*, 40(1):142–164, 2011.

[87] Alan Frieze and Ravi Kannan. Log-Sobolev inequalities and sampling from log-concave distributions. *The Annals of Applied Probability*, 9(1):14–26, 1999.

[88] Yassine Hamoudi and Frédéric Magniez. Quantum Chebyshev's inequality and applications. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019.

[89] Aram W. Harrow and Annie Y. Wei. Adaptive quantum simulated annealing for Bayesian inference and estimating partition functions. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 193–212, 2020.

[90] Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 222–227. IEEE, 1977.

[91] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, 1997.

[92] Ashwin Nayak and Felix Wu. The quantum query complexity of approximating the median and related statistics. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 384–393, 1999.

[93] Yin Tat Lee and Santosh S. Vempala. The Kannan-Lovász-Simonovits conjecture. *Current Developments in Mathematics*, 2017(1):1–36, 2017.

[94] Robert L. Smith. Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, 32(6):1296–1308, 1984.

[95] Yin Tat Lee and Santosh S. Vempala. Eldan's stochastic localization and the KLS hyperplane conjecture: An improved lower bound for expansion. In *Proceedings of the 58th Annual Symposium on Foundations of Computer Science*, pages 998–1007, 2017.

[96] Ben Cousins and Santosh Vempala. A cubic algorithm for computing Gaussian volume. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1215–1228. Society for Industrial and Applied Mathematics, 2014.

[97] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov chains and mixing times*. American Mathematical Society, 2017.

[98] Edward Farhi and Sam Gutmann. Quantum computation and decision trees. *Physical Review A*, 58(2):915–928, 1998.

[99] Andris Ambainis, Eric Bach, Ashwin Nayak, Ashvin Vishwanath, and John Watrous. One-dimensional quantum walks. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 37–49, 2001.

[100] Dorit Aharonov, Andris Ambainis, Julia Kempe, and Umesh Vazirani. Quantum walks on graphs. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 50–59, 2001.

[101] Andrew M. Childs, Richard Cleve, Enrico Deotto, Edward Farhi, Sam Gutmann, and Daniel A. Spielman. Exponential algorithmic speedup by quantum walk. In *Proceedings of the 35th ACM Symposium on Theory of Computing*, pages 59–68, 2003.

[102] Gorjan Alagic and Alexander Russell. Decoherence in quantum walks on the hypercube. *Physical Review A*, 72:062304, 2005.

[103] Peter C. Richter. Almost uniform sampling via quantum walks. *New Journal of Physics*, 9(3):72, 2007.

[104] Shantanav Chakraborty, Kyle Luh, and Jérémie Roland. Analog quantum algorithms for the mixing of Markov chains. *Physical Review A*, 102(2):022423, 2020.

[105] Dorit Aharonov and Amnon Ta-Shma. Adiabatic quantum state generation and statistical zero knowledge. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 20–29, 2003.

[106] Rolando D. Somma, Sergio Boixo, and Howard Barnum. Quantum simulated annealing, 2007.

[107] Rolando D. Somma, Sergio Boixo, Howard Barnum, and Emanuel Knill. Quantum simulations of classical annealing processes. *Physical Review Letters*, 101(13):130504, 2008.

[108] Sergio Boixo and Rolando D. Somma. Quantum algorithms for simulated annealing, 2015.

[109] Kristan Temme, Tobias J. Osborne, Karl G. Vollbrecht, David Poulin, and Frank Verstraete. Quantum Metropolis sampling. *Nature*, 471(7336):87–90, 2011.

[110] Man-Hong Yung and Alán Aspuru-Guzik. A quantum–quantum Metropolis algorithm. *Proceedings of the National Academy of Sciences*, 109(3):754–759, 2012.

[111] Davide Orsucci, Hans J. Briegel, and Vedran Dunjko. Faster quantum mixing for slowly evolving sequences of Markov chains. *Quantum*, 2:105, 2018.

[112] Daniel Štefankovič, Santosh Vempala, and Eric Vigoda. Adaptive simulated annealing: A near-optimal connection between sampling and counting. *Journal of the ACM*, 56(3):1–36, 2009.

[113] Lov K. Grover. Quantum mechanics helps in searching for a needle in a haystack. *Physical Review Letters*, 79(2):325, 1997.

[114] Theodore J. Yoder, Guang Hao Low, and Isaac L. Chuang. Fixed-point quantum search with an optimal number of queries. *Physical Review Letters*, 113(21):210501, 2014.

[115] Wang, Chunhao. *Computational Problems Related to Open Quantum Systems*. PhD thesis, University of Waterloo, 2018.

[116] Tongyang Li and Xiaodi Wu. Quantum query complexity of entropy estimation. *IEEE Transactions on Information Theory*, 65(5):2899–2921, 2019.

[117] Jun John Sakurai and Jim Napolitano. *Modern quantum mechanics*. Pearson Harlow, 2014.

[118] László Lovász and Santosh Vempala. The geometry of logconcave functions and sampling algorithms. *Random Structures Algorithms*, 30(3):307–358, 2007.

[119] Mark Rudelson. Random vectors in the isotropic position. *Journal of Functional Analysis*, 164(1):60–72, 1999.

[120] Adam Holmes and A. Y. Matsuura. Efficient quantum circuits for accurate state preparation of smooth, differentiable functions. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 169–179, 2020.

[121] Lov Grover and Terry Rudolph. Creating superpositions that correspond to efficiently integrable probability distributions, 2002.

[122] Mervin E. Muller. A note on a method for generating points uniformly on n-dimensional spheres. *Communications of the ACM*, 2(4):19–20, 1959.

[123] George Marsaglia. Choosing a point from the surface of a sphere. *The Annals of Mathematical Statistics*, 43(2):645–646, 1972.

[124] Albert B. Novikoff. On convergence proofs for perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume 12, pages 615–622, 1963.

[125] Marvin Minsky and Seymour A. Papert. *Perceptrons: An introduction to computational geometry*. MIT Press, 1988.

[126] Kenneth L Clarkson, Elad Hazan, and David P Woodruff. Sublinear optimization for machine learning. *Journal of the ACM (JACM)*, 59(5):23, 2012.

[127] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195, 2017.

[128] Srinivasan Arunachalam and Ronald de Wolf. Guest column: a survey of quantum learning theory. *ACM SIGACT News*, 48(2):41–67, 2017. arXiv:1701.06806.

[129] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. An introduction to quantum machine learning. *Contemporary Physics*, 56(2):172–185, 2015.

[130] Ashish Kapoor, Nathan Wiebe, and Krysta Svore. Quantum perceptron models. In *Advances in Neural Information Processing Systems*, pages 3999–4007, 2016.

[131] Ewin Tang. A quantum-inspired classical algorithm for recommendation systems. *arXiv:1807.04271*, 2018.

[132] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, pages 212–219. ACM, 1996. arXiv:quant-ph/9605043.

[133] Dan Garber and Elad Hazan. Approximating semidefinite programs in sublinear time. In *Advances in Neural Information Processing Systems 24*, pages 1080–1088. Curran Associates, Inc., 2011.

[134] Fernando Brandão. Quantum speed-up for sdps and kernel learning. QuICS quantum machine learning workshop, 2018. Also available at http://qml2018.umiacs.io, 2018.

[135] John Preskill. Quantum computing in the NISQ era and beyond. *Quantum*, 2:79, 2018.

[136] Nathan Wiebe, Ashish Kapoor, Christopher Granade, and Krysta M Svore. Quantum inspired training for boltzmann machines, 2015.

[137] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum algorithms for supervised and unsupervised machine learning. *arXiv:1307.0411*, 2013.

[138] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Physical Review Letters*, 113(13):130503, 2014. arXiv:1307.0471.

[139] Scott Aaronson. Read the fine print. *Nature Physics*, 11(4):291, 2015.

[140] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. 2013.

[141] Ewin Tang. Quantum-inspired classical algorithms for principal component analysis and supervised clustering. *arXiv:1811.00414*, 2018.

[142] Nai-Hui Chia, Han-Hsuan Lin, and Chunhao Wang. Quantum-inspired sublinear classical algorithms for solving low-rank linear systems. *arXiv:1811.04852*, 2018.

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

[144] Nai-Hui Chia, Tongyang Li, Han-Hsuan Lin, and Chunhao Wang. Quantum-inspired classical sublinear-time algorithm for solving low-rank semidefinite programming via sampling approaches. *arXiv:1901.03254*, 2019.

[145] Alan Frieze, Ravi Kannan, and Santosh Vempala. Fast Monte-Carlo algorithms for finding low-rank approximations. *Journal of the ACM (JACM)*, 51(6):1025–1041, 2004.

[146] Vojtech Havlicek, Antonio D. Córcoles, Kristan Temme, Aram W. Harrow, Jerry M. Chow, and Jay M. Gambetta. Supervised learning with quantum enhanced feature spaces. *arXiv:1804.11326*, 2018.

[147] Iordanis Kerenidis and Alessandro Luongo. Quantum classification of the mnist dataset via slow feature analysis. arxiv:1805.08837, 2018.

[148] Yoav Freund and Robert E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103, 1999.

[149] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning*, pages 928–936, 2003.

[150] Lov K. Grover. Synthesis of quantum superpositions by quantum computation. *Physical Review Letters*, 85(6):1334, 2000.

[151] Christoph Dürr and Peter Høyer. A quantum algorithm for finding the minimum. 1996.

[152] Bernhard Schölkopf and Alexander J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond.* MIT Press, 2002.

[153] Ankan Saha, S.V.N. Vishwanathan, and Xinhua Zhang. New approximation algorithms for minimum enclosing convex shapes. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1146–1160. Society for Industrial and Applied Mathematics, 2011. arXiv:0909.1062.

[154] Johan A.K. Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

[155] Suvrit Sra, Sebastian Nowozin, and Stephen J. Wright. *Optimization for machine learning*. MIT Press, 2012.

[156] Michael D. Grigoriadis and Leonid G. Khachiyan. A sublinear-time randomized approximation algorithm for matrix games. *Operations Research Letters*, 18(2):53–58, 1995.

[157] George B. Dantzig. *Linear programming and extensions*, volume 48. Princeton University Press, 1998.

[158] Marvin Minsky and Seymour A. Papert. *Perceptrons: An introduction to computational geometry*. MIT Press, 1988.

[159] Siddharth Barman. Approximating Nash equilibria and dense bipartite subgraphs via an approximate version of Carathéodory theorem. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing*, pages 361–369, 2015. arXiv:1406.2296

[160] Vahab Mirrokni, Renato Paes Leme, Adrian Vladu, and Sam Chiu-wai Wong. Tight bounds for approximate Carathéodory and beyond. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2440–2448, 2017. arXiv:1512.08602

[161] Cyrille W. Combettes and Sebastian Pokutta. Revisiting the approximate Carathéodory problem via the Frank-Wolfe algorithm, 2019. arXiv:1911.04415

[162] Naiyang Deng, Yingjie Tian, and Chunhua Zhang. *Support vector machines: optimization based theory, algorithms, and extensions*. CRC press, 2012.

[163] Frank Arute et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019. arXiv:1910.11333

[164] Tongyang Li, Shouvanik Chakrabarti, and Xiaodi Wu. Sublinear quantum algorithms for training linear and kernel-based classifiers. In *Proceedings of the 36th International Conference on Machine Learning*, pages 3815–3824, 2019. arXiv:1904.02276

[165] Joran van Apeldoorn and András Gilyén. Quantum algorithms for zero-sum games, 2019. arXiv:1904.03180

[166] Shai Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

[167] John von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320, 1928.

[168] Arkadi Nemirovski. Prox-method with rate of convergence $O(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1):229–251, 2004.

[169] Yurii Nesterov. Dual extrapolation and its applications to solving variational inequalities and related problems. *Mathematical Programming*, 109(2-3):319–344, 2007.

[170] Dan Garber and Elad Hazan. Approximating semidefinite programs in sublinear time. In *Advances in Neural Information Processing Systems*, pages 1080–1088, 2011.

[171] Elad Hazan, Tomer Koren, and Nati Srebro. Beating SGD: Learning SVMs in sublinear time. In *Advances in Neural Information Processing Systems*, pages 1233–1241, 2011.

[172] Balamurugan Palaniappan and Francis Bach. Stochastic variance reduction methods for saddle-point problems. In *Advances in Neural Information Processing Systems*, pages 1416–1424, 2016. `arXiv:1605.06398`

[173] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013.

[174] Zhan Shi, Xinhua Zhang, and Yaoliang Yu. Bregman divergence for stochastic variance reduction: saddle-point and adversarial prediction. In *Advances in Neural Information Processing Systems*, pages 6031–6041, 2017.

[175] Yair Carmon, Yujia Jin, Aaron Sidford, and Kevin Tian. Variance reduction for matrix games. In *Advances in Neural Information Processing Systems*, pages 11377–11388, 2019. `arXiv:1907.02056`

[176] Maurice Sion. On general minimax theorems. *Pacific Journal of Mathematics*, 8(1):171–176, 1958.

[177] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[178] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. 2014.

[179] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O?brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5:4213, 2014.

[180] Nikolaj Moll, Panagiotis Barkoutsos, Lev S. Bishop, Jerry M. Chow, Andrew Cross, Daniel J. Egger, Stefan Filipp, Andreas Fuhrer, Jay M. Gambetta, Marc Ganzhorn, Abhinav Kandala, Antonio Mezzacapo, Peter Muller, Walter Riess, Gian Salis, John Smolin, Ivano Tavernelli, and Kristan Temme. Quantum optimization using variational algorithms on near-term quantum devices. *Quantum Science and Technology*, 3(3):030503, 2018.

[181] Pierre-Luc Dallaire-Demers and Nathan Killoran. Quantum generative adversarial networks. *Physical Review A*, 98:012324, Jul 2018.

[182] Marcello Benedetti, Edward Grant, Leonard Wossnig, and Simone Severini. Adversarial quantum circuit learning for pure state approximation. *New Journal of Physics*, 21(4):043023, 2019.

[183] Jinfeng Zeng, Yufeng Wu, Jin-Guo Liu, Lei Wang, and Jiangping Hu. Learning and inference on generative adversarial quantum circuits. 2018.

[184] Haozhen Situ, Zhimin He, Lvzhou Li, and Shenggen Zheng. Quantum generative adversarial network for generating discrete data. 2018.

[185] Ling Hu, Shu-Hao Wu, Weizhou Cai, Yuwei Ma, Xianghao Mu, Yuan Xu, Haiyan Wang, Yipu Song, Dong-Ling Deng, Chang-Ling Zou, and Luyan Sun. Quantum generative adversarial learning in a superconducting quantum circuit. *Science advances*, 5(1):eaav2761, 2019.

[186] Jonathan Romero and Alan Aspuru-Guzik. Variational quantum generators: Generative adversarial quantum machine learning for continuous distributions. 2019.

[187] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. Quantum generative adversarial networks for learning and loading random distributions. 2019.

[188] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 2017.

[189] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[190] Maziar Sanjabi, Jimmy Ba, Meisam Razaviyayn, and Jason D. Lee. On the convergence and robustness of training GANs with regularized optimal transport. In *Advances in Neural Information Processing Systems 31*, pages 7091–7101. Curran Associates, Inc., 2018.

[191] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, pages 2292–2300, 2013.

[192] Vivien Seguy, Bharath Bhushan Damodaran, Rémi Flamary, Nicolas Courty, Antoine Rolet, and Mathieu Blondel. Large-scale optimal transport and mapping estimation. 2017.

[193] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[194] Henning Petzka, Asja Fischer, and Denis Lukovnicov. On the regularization of Wasserstein GANs. 2017.

[195] Xin Guo, Johnny Hong, Tianyi Lin, and Nan Yang. Relaxed Wasserstein with applications to GANs. 2017.

[196] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for GANs do actually converge? In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3481–3490. PMLR, 10–15 Jul 2018.

[197] Lipeng Ning, Tryphon T. Georgiou, and Allen R. Tannenbaum. On matrix-valued monge-kantorovich optimal mass transport. *IEEE Trans. Autom. Control.*, 60(2):373–382, 2015.

[198] François Golse, Clément Mouhot, and Thierry Paul. On the mean field and classical limits of quantum mechanics. *Communications in Mathematical Physics*, 343(1):165–205, Jan 2016.

[199] Gabriel Peyré, Lenaïc Chizat, François-Xavier Vialard, and Justin Solomon. Quantum optimal transport for tensor field processing. *CoRR*, abs/1612.08731, 2016.

[200] Eric A. Carlen and Jan Maas. Gradient flow and entropy inequalities for quantum markov semigroups with detailed balance. *Journal of Functional Analysis*, 273(5):1810–1869, 2017.

[201] Yongxin Chen, Tryphon T. Georgiou, and Allen Tannenbaum. Matrix optimal mass transport: a quantum mechanical approach. *IEEE Transactions on Automatic Control*, 63(8):2612–2619, 2018.

[202] Yongxin Chen, Tryphon T. Georgiou, and Allen Tannenbaum. Wasserstein geometry of quantum states and optimal transport of matrix-valued measures. In *Emerging Applications of Control and Systems Theory*, pages 139–150. Springer, 2018.

[203] Nengkun Yu, Li Zhou, Shenggang Ying, and Mingsheng Ying. Quantum Earth mover's distance, No-go Quantum Kantorovich-Rubinstein theorem, and Quantum Marginal Problem. *arXiv e-prints*, page arXiv:1803.02673, February 2018.

[204] Emanuele Caglioti, François Golse, and Thierry Paul. Quantum optimal transport is cheaper. *Journal of Statistical Physics*, 181((1),):149–162, May 2020.

[205] Yogesh Balaji, Rama Chellappa, and Soheil Feizi. Normalized Wasserstein distance for mixture distributions with applications in adversarial learning and domain adaptation. 2019.

[206] Daiwei Zhu. Personal Communication, Feb, 2019.

[207] Michael A. Nielsen and Isaac Chuang. *Quantum computation and quantum information*. AAPT, 2002.

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

[209] Aram W. Harrow. The church of the symmetric subspace. 2013.

[210] Eric A. Carlen and Jan Maas. An Analog of the 2-Wasserstein Metric in Non-commutative Probability under which the Fermionic Fokker-Planck Equation is Gradient Flow for the Entropy. *arXiv e-prints*, page arXiv:1203.5377, March 2012.

[211] Jean-David Benamou and Yann Brenier. A computational fluid mechanics solution to the monge-kantorovich mass transfer problem. *Numerische Mathematik*, 84:375–393, 2000.

[212] Sidney Golden. Lower bounds for the Helmholtz function. *Physical Review*, 137(4B):B1127, 1965.

[213] Colin J. Thompson. Inequality with applications in statistical mechanics. *Journal of Mathematical Physics*, 6(11):1812–1813, 1965.

[214] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3):032331, 2019.

[215] Koji Tsuda, Gunnar Rätsch, and Manfred K. Warmuth. Matrix exponentiated gradient updates for on-line learning and Bregman projection. *Journal of Machine Learning Research*, 6(Jun):995–1018, 2005.

[216] John Watrous. Simpler semidefinite programs for completely bounded norms. *arXiv e-prints*, page arXiv:1207.5726, July 2012.

[217] John M. Danskin. The theory of max-min, with applications. *SIAM Journal on Applied Mathematics*, 14(4):641–664, 1966.

[218] Hale F. Trotter. On the product of semi-groups of operators. *Proceedings of the American Mathematical Society*, 10(4):545–551, 1959.

[219] Christopher M. Dawson and Michael A. Nielsen. The Solovay-Kitaev algorithm. 2005.

[220] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

[221] Han-Sen Zhong, Hui Wang, Yu-Hao Deng, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Jian Qin, Dian Wu, Xing Ding, Yi Hu, Peng Hu, Xiao-Yan Yang, Wei-Jun Zhang, Hao Li, Yuxuan Li, Xiao Jiang, Lin Gan, Guangwen Yang, Lixing You, Zhen Wang, Li Li, Nai-Le Liu, Chao-Yang Lu, and Jian-Wei Pan. Quantum computational advantage using photons. *Science*, 370(6523):1460–1463, 2020.

[222] Aram W Harrow and Ashley Montanaro. Quantum computational supremacy. *Nature*, 549(7671):203–209, 2017.

[223] Maria Schuld. Supervised quantum machine learning models are kernel methods. *arXiv e-prints*, page arXiv:2101.11020, January 2021.

[224] Hsin-Yuan Huang, Michael Broughton, Masoud Mohseni, Ryan Babbush, Sergio Boixo, Hartmut Neven, and Jarrod R. McClean. Power of data in quantum machine learning. *Nature Communications*, 12(1):2631, May 2021.

[225] Francesco Tacchino, Chiara Macchiavello, Dario Gerace, and Daniele Bajoni. An artificial neuron implemented on an actual quantum processor. *npj Quantum Information*, 5(1):26, March 2019.

[226] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[227] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. The effect of data encoding on the expressive power of variational quantum machine learning models. *arXiv e-prints*, page arXiv:2008.08605, August 2020.

[228] Gal Vardi, Daniel Reichman, Toniann Pitassi, and Ohad Shamir. Size and Depth Separation in Approximating Natural Functions with Neural Networks. *arXiv e-prints*, page arXiv:2102.00314, January 2021.

[229] Francisco Javier Gil Vidal and Dirk Oliver Theis. Input redundancy for parameterized quantum circuits. *Frontiers in Physics*, 8:297, 2020.

[230] Matus Telgarsky. benefits of depth in neural networks. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir, editors, *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 1517–1539, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR.

[231] Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2924–2932, Cambridge, MA, USA, 2014. MIT Press.

[232] Matus Telgarsky. Neural networks and rational functions. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3387–3393. PMLR, 06–11 Aug 2017.

[233] Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa, and Keisuke Fujii. Quantum circuit learning. *Physical Review A*, 98(3):032309, 2018.

[234] Logan G. Wright and Peter L. McMahon. The capacity of quantum neural networks. In *Conference on Lasers and Electro-Optics*, page JM4G.5. Optical Society of America, 2020.

[235] Amira Abbas, David Sutter, Christa Zoufal, Aurélien Lucchi, Alessio Figalli, and Stefan Woerner. The power of quantum neural networks. *arXiv e-prints*, page arXiv:2011.00027, October 2020.

[236] Amit Daniely. Depth separation for neural networks. In Satyen Kale and Ohad Shamir, editors, *Proceedings of the 2017 Conference on Learning Theory*, volume 65 of *Proceedings of Machine Learning Research*, pages 690–696, Amsterdam, Netherlands, 07–10 Jul 2017. PMLR.

[237] Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. volume 49 of *Proceedings of Machine Learning Research*, pages 907–940, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR.

[238] Shiyu Liang and R. Srikant. Why Deep Neural Networks for Function Approximation? *arXiv e-prints*, page arXiv:1610.04161, October 2016.

[239] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6232–6240, Red Hook, NY, USA, 2017. Curran Associates Inc.

[240] Kenji Kawaguchi. Deep learning without poor local minima. In *Advances in neural information processing systems*, pages 586–594, 2016.

[241] Simon S Du and Jason D Lee. On the power of over-parametrization in neural networks with quadratic activation. *arXiv preprint arXiv:1803.01206*, 2018.

[242] Michael J.D. Powell. *Approximation Theory and Methods*, page 150. Cambridge University Press, 1981.

[243] Alexander A. Razborov. On small depth threshold circuits. In Otto Nurmi and Esko Ukkonen, editors, *Algorithm Theory — SWAT '92*, pages 42–52, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

[244] Lijie Chen. Toward Super-Polynomial Size Lower Bounds for Depth-Two Threshold Circuits. *arXiv e-prints*, page arXiv:1805.10698, May 2018.

[245] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[246] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

[247] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[248] Dmytro Perekrestenko, Philipp Grohs, Dennis Elbrächter, and Helmut Bölcskei. The universal approximation power of finite-width deep ReLU networks. *arXiv e-prints*, page arXiv:1806.01528, June 2018.

[249] Xuchen You and Xiaodi Wu. Exponentially many local minima in quantum neural networks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12144–12155. PMLR, 18–24 Jul 2021.

[250] Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature communications*, 9(1):1–6, 2018.

[251] Edward Grant, Leonard Wossnig, Mateusz Ostaszewski, and Marcello Benedetti. An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *Quantum*, 3:214, 2019.

[252] Taylor L Patti, Khadijeh Najafi, Xun Gao, and Susanne F Yelin. Entanglement Devised Barren Plateau Mitigation. 2020.

[253] Leo Zhou, Sheng Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D Lukin. Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices, 2018.

[254] Tyler Volkoff and Patrick J Coles. Large gradients via correlation in random parameterized quantum circuits. Technical report, 2021.

[255] M Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R. McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, and Patrick J Coles. Variational Quantum Algorithms. 2020.

[256] Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[257] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *arXiv preprint arXiv:1904.11955*, 2019.

[258] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*, 2018.

[259] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization, 2019.

[260] Eric R. Anschuetz. Critical points in quantum generative models, 2022.

[261] Martin Larocca, Nathan Ju, Diego García-Martín, Patrick J. Coles, and M. Cerezo. Theory of overparametrization in quantum neural networks, 2021.

[262] Junyu Liu, Francesco Tacchino, Jennifer R. Glick, Liang Jiang, and Antonio Mezzacapo. Representation learning via quantum neural tangent kernels, 2021.

[263] Norihito Shirai, Kenji Kubo, Kosuke Mitarai, and Keisuke Fujii. Quantum tangent kernel, 2021.

[264] Erfan Abedi, Salman Beigi, and Leila Taghavi. Quantum lazy training, 2022.

[265] Roeland Wiersema and Nathan Killoran. Optimizing quantum circuits with riemannian gradient-flow, 2022.

[266] Zhiqiang Xu, Xin Cao, and Xin Gao. Convergence analysis of gradient descent for eigenvector computation. International Joint Conferences on Artificial Intelligence, 2018.

[267] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M Chow, and Jay M Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549(7671):242, 2017.

[268] Roeland Wiersema, Cunlu Zhou, Yvette de Sereville, Juan Felipe Carrasquilla, Yong Baek Kim, and Henry Yuen. Exploring entanglement and optimization within the Hamiltonian variational Ansatz, aug 2020.

[269] Péter Pál Varjú. Random walks in compact groups. *arXiv preprint arXiv:1209.1745*, 2012.

[270] Fernando GSL Brandao, Aram W Harrow, and Michał Horodecki. Local random quantum circuits are approximate polynomial-designs. *Communications in Mathematical Physics*, 346(2):397–434, 2016.

[271] Benoît Collins and Piotr Śniady. Integration with respect to the haar measure on unitary, orthogonal and symplectic group. *Communications in Mathematical Physics*, 264(3):773–795, 2006.

[272] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

[273] B. Hall and B.C. Hall. *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction*. Graduate Texts in Mathematics. Springer, 2003.

[274] Bobak Toussi Kiani, Seth Lloyd, and Reevu Maity. Learning unitaries by gradient descent, 2020.

[275] Joel A Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of computational mathematics*, 12(4):389–434, 2012.

[276] Joel A Tropp. An introduction to matrix concentration inequalities. *arXiv preprint arXiv:1501.01571*, 2015.

[277] Ankit Garg, Robin Kothari, Praneeth Netrapalli, and Suhail Sherif. Near-optimal lower bounds for convex optimization for all orders of smoothness, 2021.

[278] Ankit Garg, Robin Kothari, Praneeth Netrapalli, and Suhail Sherif. No quantum speedup over gradient descent for non-smooth convex optimization, 2020.

[279] Naman Agarwal, Brian Bullins, and Elad Hazan. Second-order stochastic optimization for machine learning in linear time. 2016.

[280] Naman Agarwal, Zeyuan Allen-Zhu, Brian Bullins, Elad Hazan, and Tengyu Ma. Finding approximate local minima faster than gradient descent, 2017.

[281] Iordanis Kerenidis and Anupam Prakash. A quantum interior point method for lps and sdps, 2018.

[282] Iordanis Kerenidis, Anupam Prakash, and Dá niel Szilágyi. Quantum algorithms for second-order cone programming and support vector machines. *Quantum*, 5:427, apr 2021.

[283] Rong Ge, Jason D Lee, and Tengyu Ma. Matrix completion has no spurious local minimum. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[284] Rong Ge and Tengyu Ma. On the optimization landscape of tensor decompositions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 3656–3666, Red Hook, NY, USA, 2017. Curran Associates Inc.

[285] Chenyi Zhang, Jiaqi Leng, and Tongyang Li. Quantum algorithms for escaping from saddle points. *Quantum*, 5:529, aug 2021.

[286] Zongchen Chen and Santosh S. Vempala. Optimal Convergence Rate of Hamiltonian Monte Carlo for Strongly Logconcave Distributions. *arXiv e-prints*, page arXiv:1905.02313, May 2019.

[287] Pranav Gokhale, Ali Javadi-Abhari, Nathan Earnest, Yunong Shi, and Frederic T. Chong. Optimized quantum compilation for near-term algorithms with openpulse, 2020.

[288] Pranav Gokhale, Yongshan Ding, Thomas Propson, Christopher Winkler, Nelson Leung, Yunong Shi, David I. Schuster, Henry Hoffmann, and Frederic T. Chong. Partial compilation of variational algorithms for noisy intermediate-scale quantum machines. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, oct 2019.

[289] Thomas Alexander, Naoki Kanazawa, Daniel J Egger, Lauren Capelluto, Christopher J Wood, Ali Javadi-Abhari, and David C McKay. Qiskit pulse: programming quantum computers through the cloud with pulses. *Quantum Science and Technology*, 5(4):044006, aug 2020.