An Introduction to Quantum Computing for Statisticians

Anna Lopatnikova* Minh-Ngoc Tran*

1st version: December 2021

Abstract

Quantum computing has the potential to revolutionise and change the way we live and understand the world. This review aims to provide an accessible introduction to quantum computing with a focus on applications in statistics and data analysis. We start with an introduction to the basic concepts necessary to understand quantum computing and the differences between quantum and classical computing. We describe the core quantum subroutines that serve as the building blocks of quantum algorithms. We then review a range of quantum algorithms expected to deliver a computational advantage in statistics and machine learning. We highlight the challenges and opportunities in applying quantum computing to problems in statistics and discuss potential future research directions.

Keywords. Quantum machine learning, quantum Monte Carlo, quantum Markov chain, quantum descriptive statistics, quantum linear algebra.

Contents

1 Introduction **3**

2 Basic Concepts of Quantum Computation **4**
2.1 The simplest quantum system: a qubit 4
2.2 Quantum States and Quantum Operators 6
2.3 Quantum Gates and Other Primitives 11
2.4 Properties of Quantum Computers 15

3 An Overview of Quantum Algorithm Design **17**
3.1 Data Encoding in a Quantum State 17
3.2 Result Postprocessing and Readout 21
3.3 Computational Complexity 23
3.4 A Brief Overview of Quantum Algorithms 24

*Discipline of Business Analytics, the University of Sydney Business School. *The research was partially supported by the Australian Research Council's Discovery Project DP200103015.* *Corresponding to minh-ngoc.tran@sydney.edu.au*

4 Grover's Search and Descriptive Statistics on a Quantum Computer 26
4.1 Grover's Search Algorithm . . . 26
4.2 Quantum Amplitude Amplification (QAA) . . . 28
4.3 Quantum Amplitude Estimation (QAE) . . . 29
4.4 Estimating the Mean of a Bounded Function . . . 29
4.5 Minimum (or Maximum) of a Function over a Discrete Domain . . . 31
4.6 Median and kth Smallest Value . . . 31
4.7 Counting . . . 32
4.8 Quantum Monte Carlo . . . 32

5 Quantum Markov Chains 34
5.1 Coin Walks . . . 34
5.2 Szegedy Walks . . . 35
5.3 Quantum Markov Chain Monte Carlo . . . 36

6 Quantum Linear Systems, Matrix Inversion, and PCA 37
6.1 Quantum Fourier Transform (QFT) . . . 38
6.2 Quantum Phase Estimation (QPE) . . . 39
6.3 Applying a Hermitian Operator . . . 40
6.4 The HHL Linear Systems Algorithm . . . 42
6.5 Fast Gradient Computation . . . 44
6.6 Quantum Principal Component Analysis (QPCA) . . . 45

7 Hamiltonian Simulation 47
7.1 Overview and Preliminaries . . . 47
7.2 Product Formula . . . 50
7.3 Hamiltonian Simulation by Quantum Walk . . . 52
7.4 Linear Combination of Unitaries . . . 54
7.5 Hamiltonian Simulation by Quantum Signal Processing (QSP) . . . 55

8 Quantum Optimization 56
8.1 Adiabatic Quantum Computing (AQC) . . . 56
8.2 Quantum Approximate Optimization Algorithm (QAOA) . . . 57
8.3 Hybrid Quantum-Classical Variational Algorithms . . . 59

9 Quantum Eigenvalue and Singular Value Transformations 59
9.1 Quantum Signal Processing . . . 60
9.2 Quantum Eigenvalue Transformation . . . 61
9.3 Quantum Singular Value Transformation . . . 63
9.4 QSVT and the "Grand Unification" of Quantum Algorithms . . . 64

10 Quantum Machine Learning 65
10.1 Quantum Gradient Descent . . . 67

11 Conclusion 68

1 Introduction

Quantum computing is emerging as a promising way to gain significant efficiency in computationally demanding tasks. It holds the promise to revolutionize statistics and related fields. Quantum computers can speed up both simple statistical tasks such as estimating sample mean and median, and more complex tasks, such as calculating the output of a neural network (Abbas et al., 2020; Rebentrost et al., 2018a) or Bayesian computation with Variational Bayes (Lopatnikova and Tran, 2021), delivering polynomial or, in some cases, exponential gains in computational complexity.

This review article provides a pedagogical introduction to quantum computing for statisticians. Its purpose is to demonstrate that quantum computers could play an important role in statistics and machine learning. There are many excellent review articles on quantum computation, but we have found that none of them speak to the needs of the statistics readership. Our article aims to fill this gap by focusing on the quantum algorithms that support statistical applications and highlighting the algorithms' aspects of greatest interest to statisticians and data scientists. For each quantum algorithm outlined in the review, we distill its core idea, provide relevant details, and supply key references to the interested reader for further follow up.

Quantum computing is a rapidly evolving field and, every year, new more efficient methods overtake the last year's cutting edge. To account for this rapid obsolescence, we seek to provide a balanced overview of timeless foundations and the relevant state of the art. The review is organized as follows:

Section 2 lays out the basic foundations of quantum computing. It starts with an accessible introduction to the relevant concepts of quantum physics, describes quantum computing primitives such as gates, and outlines the surprising properties of quantum computers that make them both powerful and counterintuitive.

Section 3 provides a statistician, viewed as a technical user of quantum algorithms, with a clear framework for quantum algorithm design. It outlines the three basic steps common to all quantum algorithms - data loading, computation, and readout. It also describes methodologies for efficient encoding of data from a classical computer into a quantum state and for the readout of results of quantum computation from a quantum state onto a classical computer. The section concludes with a summary overview of the quantum algorithms presented in detail in Sections 4 to 9.

Section 4 starts with Grover's search algorithm – a seminal algorithm, whose core idea gave rise to many widely-used quantum routines including Quantum Amplitude Amplification and Quantum Amplitude Estimation. We describe how these routines can be used to speed up calculation of statistical descriptive quantities such as sample mean and sample median. We then describe a Quantum Monte Carlo method for estimating the probability expectation of a function - this method can be viewed as an application of Quantum Amplitude Estimation, and offers a provable quadratic speedup over the classical Monte Carlo method.

Section 5 presents quantum Markov chains (known in the literature as quantum walks). Similar to classical Markov chains, they power many statistical applications, including quantum Markov chain Monte Carlo.

Section 6 reviews quantum algorithms for linear algebra computations including solving

a system of linear equations and Principle Component Analysis, central to statistics and machine learning.

Section 7 presents Hamiltonian simulation. While Hamiltonian simulation might be of less interest to statisticians, it is a critical subroutine for quantum linear algebra (Section 6) and is at the heart of potentially transformational applications of quantum computing in vital fields such as chemistry, agriculture, and energy.

Section 8 discusses quantum algorithms to speed up optimization, which plays an important role in quantum machine learning. Section 9 describes Quantum Singular Value Transformation (QSVT) – a cutting-edge algorithm at the time of writing, which enables polynomial transformations of singular values. QSVT is a promising recent framework that encompasses other popular quantum algorithms – such as search, Hamiltonian simulation, or systems of linear equations – as specific instances. Because of its flexibility and expressivity, we expect QSVT to give rise to other influential algorithms of interest to statisticians and data scientists.

We complete the article with Section 10, a brief overview of recent developments in quantum machine learning – a rapidly changing field. We outline major developments, provide references, and discuss research challenges as well as opportunities in quantum machine learning. Section 11 concludes.

2 Basic Concepts of Quantum Computation

This section presents the basic foundations of quantum mechanics required to understand how and why quantum computing works. The mathematical foundation of quantum physics is linear algebra on a complex vector space, with specific mathematical rules (sometimes called *postulates*) added to take into account quantum states' physical properties. We describe the connection between quantum theory and familiar linear algebra concepts, outline the rules specific to quantum theory, and highlight a few consequences of these rules that make quantum algorithms work in ways that are strange and unfamiliar compared with classical algorithms.

2.1 The simplest quantum system: a qubit

The basic unit of information in a classical computer is a bit, taking either 0 or 1 values. The basic unit of a quantum computer that stores information is a *qubit*, which is a two-dimensional system

$$|q\rangle = a\,|0\rangle + b\,|1\rangle\,, \tag{1}$$

where $|0\rangle$ and $|1\rangle$ form the basis of this two dimensional space.[1] The symbol $|\bullet\rangle$ is called a *ket*; the coefficients a and b are called *amplitudes* and can be complex numbers, $a, b \in \mathbb{C}$,

[1]The qubit state $|q\rangle$ in Eq. (1) is a shorthand for $\begin{pmatrix} a \\ b \end{pmatrix}$, a normalized vector in the two-dimensional Hilbert space that describes the state of the single-qubit quantum system. The basis state $|0\rangle$ is a shorthand for $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle$ is a shorthand for $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$. The orthonormality is understood in the usual sense of linear algebra.

normalized so that $|a|^2 + |b|^2 = 1$. The state $|q\rangle$ is a linear *superposition* of the states $|0\rangle$ and $|1\rangle$. A quantum measurement of $|q\rangle$ in the basis of $\{|0\rangle, |1\rangle\}$ yields 0 with probability $|a|^2$ and 1 with probability $|b|^2$; we will discuss quantum measurement in detail in Section 2.2. The state $|q\rangle$ is analogous to a random variable taking the classical bit values 0 and 1 with probabilities $|a|^2$ and $|b|^2$ respectively, but there is a crucial difference: the coefficients a and b can be negative, and more generally, complex. As we discuss below, this property is crucial to the success and also the strangeness of quantum computers.

The superposition property extends to collections of multiple qubits, sometimes referred to as *quantum registers*. For example, a register of three qubits can support quantum mechanical states of the form

$$\begin{aligned} |\psi\rangle = &\psi_{000}\,|000\rangle + \psi_{001}\,|001\rangle + \psi_{010}\,|010\rangle + \psi_{011}\,|011\rangle \\ &+ \psi_{100}\,|100\rangle + \psi_{101}\,|101\rangle + \psi_{110}\,|110\rangle + \psi_{111}\,|111\rangle\,. \end{aligned} \tag{2}$$

The states $|b_1 b_2 b_3\rangle$, where $b_k = \{1, 0\}$ for $k = 1, 2, 3$, form an orthogonal basis, and the 3-qubit register supports $2^3 = 8$ dimensional quantum states – superpositions of 8 basis states. The writing $|b_1 b_2 b_3\rangle$ is a shorthand for $|b_1\rangle \otimes |b_2\rangle \otimes |b_3\rangle$, where $\otimes$ represents the tensor product, which is a 8-dimensional vector. This form shows that the state $|b_1 b_2 b_3\rangle$ is *separable*, which means each qubit in the state can be manipulated independently of the other qubits. It is important to note that not all quantum states formed by multiple qubits can be expressed as a tensor product. For example, the state $|\psi\rangle$ in Eq. (2) is, in general, not separable. States that are not separable are called *entangled*; entanglement is one of the strangest phenomena in quantum mechanics.

A quantum algorithm is a series of linear operations and quantum measurements performed on a set of quantum registers. Like their classical counterparts, quantum algorithms are implemented via a series of simple gates, acting on one, two, or three qubits at a time. A quantum algorithm takes a quantum state as an input and transforms it into an output state that encodes the desired result.

The superposition property suggests that quantum algorithms might offer exponential speedup over classical algorithms. Consider an algorithm that checks 100-bit strings, i.e. $2^{100} \approx 10^{30}$ possibilities. A classical computer would have to check each of the 2^{100} possibilities; a quantum computer could potentially perform the task in 100 operations, checking the 2^{100} possibilities in massive parallel. The trouble is, the output of the quantum computation will be a superposition of all the 2^{100} results. If all results are approximately equally probable, then at least 10^{30} measurements would be required to extract the correct answer. A well-designed quantum algorithm uses the fact that amplitudes can be complex so that the computation suppresses the amplitudes of the wrong answers and amplifies the amplitudes of the correct answer(s). Then, just a few measurements might be enough to extract the correct answers. Finding quantum algorithms able to perform parallel computation while providing an efficient way to extract results has proven challenging, but the list of such algorithms is expanding. The building blocks of these algorithms can be combined to solve a variety of practical problems.

2.2 Quantum States and Quantum Operators

Multi-Qubit Quantum States

A quantum computer stores information using collections of qubits in quantum registers. Equation (2) provided the general form of a quantum state created on a three-qubit register. A convenient shorthand for the expansion of $|\psi\rangle$ in Eq. (2) interprets the "bit strings" of individual qubit basis states as integer numbers, so that as a shorthand for, e.g., $|101\rangle$ we write $|5\rangle$. In this notation, a n-qubit quantum state is written as

$$|\psi\rangle = \sum_{i=0}^{N-1} \psi_i \, |i\rangle \, , \tag{3}$$

where $N = 2^n$, and i in $|i\rangle$ represents the bit string representation of the integer i. The 3-qubit quantum state $|\psi\rangle$ in (2) is fully described by a normalized 8-dimensional vector over complex numbers, $(\psi_0, \psi_1, .., \psi_7)^\top$. The orthonormal basis formed by states $|i\rangle \equiv |b_1 b_2 .. b_n\rangle$, where $b_j = \{0, 1\}$ $\forall j = 1, .., n$, is called the *computational basis*.

The Vector Space of Quantum States

A quantum register of n qubits supports a $N = 2^n$-dimensional vector space, called a *Hilbert space*. A Hilbert space is a vector space with a defined inner product, a generalization of Euclidean space over complex numbers in any (finite or infinite) dimensions.

The inner product between two vectors $|\psi\rangle$ and $|\phi\rangle$ in an N-dimensional Hilbert space is analogous to the one in Euclidean space and can be similarly expressed in terms of vector coordinates $\langle \phi, \psi \rangle = \sum_{i=0}^{N-1} \phi_i^* \psi_i$, where the asterisk $*$ indicates complex conjugation. In quantum notation, the conjugate-transpose of a ket $|\phi\rangle$ is denoted as a *bra* $\langle\phi|$ and decomposed as $\langle\phi| = \sum_{j=1}^{N-1} \phi_j^* \, \langle j|$. The inner product is expressed as $\langle \bullet | \bullet \rangle$. For example, the inner product between two basis states $|i\rangle$ and $|j\rangle$ is $\langle i|j\rangle$. Because the basis states are orthonormal:

$$\langle i|j\rangle = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases} . \tag{4}$$

For two general N-dimensional states, the inner product takes the form:

$$\langle\phi|\psi\rangle = \Big(\sum_{j=0}^{N-1} \phi_j^* \, \langle j| \Big) \Big(\sum_{i=0}^{N-1} \psi_i \, |i\rangle \Big) = \sum_{j=0}^{N-1} \sum_{i=0}^{N-1} \phi_j^* \psi_i \, \langle i|j\rangle = \sum_{i=0}^{N-1} \phi_i^* \psi_i, \tag{5}$$

where Eq. (4) helped simplify the expression.

Operators

The operators of quantum mechanics are linear.[2] They can be expressed as matrices of complex numbers acting on vectors in the N-dimensional Hilbert space. Consider the operator A represented by a matrix with elements $\{a_{ij}\}_{i,j=0}^{N-1}$ with respect to the computational basis. In the *bra-ket* notation it takes the form

$$A = \sum_{i,j=0}^{N-1} a_{ij} \, |i\rangle\langle j| \, , \tag{6}$$

so that, when it acts on state $|\psi\rangle = \sum_{k=0}^{N-1} \phi_k \, |k\rangle$, the operation yields the expected result

$$A \, |\psi\rangle = \Big(\sum_{i,j=0}^{N-1} a_{ij} \, |i\rangle\langle j| \Big) \Big(\sum_{k=0}^{N-1} \phi_k \, |k\rangle \Big) = \sum_{i,j=0}^{N-1} a_{ij} \psi_j \, |i\rangle = \sum_{i=0}^{N-1} \Big(\sum_{j=0}^{N-1} a_{ij} \psi_j \Big) \, |i\rangle \, . \tag{7}$$

In effect, quantum operators take the form of linear combinations of *outer products* $|i\rangle\langle j|$ of basis states in quantum-mechanical notation.

Because of the constraints in quantum mechanics, not all linear operators are quantum mechanical operators. There are two types of quantum mechanical operators: *unitary transformations* and *observables*. Unitary transformations transform one quantum state into another; observables are related to quantum measurement and will be discussed shortly.

A linear operator U is *unitary* if $U^{-1} = U^{\dagger}$, where the dagger $\dagger$ denotes the conjugate transpose. Unitary operators preserve the unit norm of quantum states.[3]

The identity operator I is a unitary operator. It is commonly used in quantum algorithm development to re-express a quantum state in an alternative basis. Let $\{|a_i\rangle\}_{i=0}^{N-1}$ be an orthonormal set of basis states of an N-dimensional Hilbert space and let $\{|b_j\rangle\}_{j=0}^{N-1}$ be an alternative set of orthonormal basis states of the space. A state $|\psi\rangle$ expressed in terms of basis states $\{|a_i\rangle\}$ can be expressed in terms of states $\{|b_j\rangle\}$ by using the identity operator I written in terms of states $\{|b_j\rangle\}$,

$$I = \sum_{j=0}^{N-1} |b_j\rangle\langle b_j| \, , \tag{8}$$

as follows

$$\begin{aligned} |\psi\rangle = \sum_{i=0}^{N-1} \psi_i \, |a_i\rangle &= \Big(\sum_{j=0}^{N-1} |b_j\rangle\langle b_j| \Big) \Big(\sum_{i=0}^{N-1} \psi_i \, |a_i\rangle \Big) \\ &= \sum_{j=0}^{N-1} \Big(\sum_{i=0}^{N-1} \psi_i \, \langle b_j|a_i\rangle \Big) \, |b_j\rangle = \sum_{j=0}^{N-1} \tilde{\psi}_j \, |b_j\rangle \, , \end{aligned}$$

[2]Abrams and Lloyd (1998) demonstrate that, if it were possible to construct non-linear quantum mechanical operators then the computational complexity class NP of problems exponentially hard for classical computers would be equal to the complexity class P – the class of problems classical computers can solve efficiently (i.e. in time that scales polynomially with the size of the input to the problem). While no proof exists that $NP \neq P$, it is considered highly unlikely that $NP = P$.

[3]Let $|\psi\rangle$ be the initial state, and $|\phi\rangle = U \, |\psi\rangle$. Then, $\langle\phi|\phi\rangle = \langle\psi| \, U^{\dagger} U \, |\psi\rangle = \langle\psi|\psi\rangle$=1.

where $\tilde{\psi}_j = \sum_{i=0}^{N-1} \psi_i \langle b_j | a_i \rangle$ are the coordinates of $|\psi\rangle$ in the basis $\{|b_j\rangle\}$.

Note that, even though all transformations of *closed* quantum systems are unitary, transformations of subsystems can be non-unitary and, more generally, non-linear. Quantum algorithms exploit this property and perform non-linear transformations by embedding the transformed quantum state in a larger system using auxiliary qubits, discussed in more detail in Section 2.3. For example, the quantum singular value transformation algorithm (Section 9) is based on the ability to implement non-linear transformations on subsystems embedded in larger quantum systems.

Quantum Measurement

Quantum measurement is a physical way to characterize a quantum mechanical state. Quantum measurement is famously probabilistic. Formally, it is a collection of operators $\{M_m\}$ that correspond to outcomes m (where m can be an outcome or the index of an outcome) that occur with probability p_m. The measurement operators M_m are defined so that, when a measurement is applied to a quantum state $|\psi\rangle$, it yields the outcome m with probability $p_m = \langle \psi | M_m^\dagger M_m | \psi \rangle$. The state of the system after the measurement is

$$\frac{M_m |\psi\rangle}{\sqrt{\langle \psi | M_m^\dagger M_m |\psi\rangle}}.$$

Because the probability of all possible outcomes adds up to 1, $\sum_m p_m = 1$ for all quantum states, the measurement operators satisfy the completion relation $\sum_m M_m^\dagger M_m = I$.

For example, the measurement of the qubit $|q\rangle$ in Eq. (1) in the computational basis is a collection of two measurement operators, called *projection measurement operators* in this case, $P_0 = |0\rangle\langle 0|$ and $P_1 = |1\rangle\langle 1|$, corresponding to outcomes 0 and 1 respectively. These operators satisfy the completeness relation $P_0^\dagger P_0 + P_1^\dagger P_1 = |0\rangle \langle 0|0\rangle \langle 0| + |1\rangle \langle 1|1\rangle \langle 1| = |0\rangle\langle 0| + |1\rangle\langle 1| = I$. The probability that the measurement yields 0 is $p_0 = \langle q | P_0^\dagger P_0 | q \rangle = \langle q|0\rangle \langle 0|q\rangle = |\langle 0|q\rangle|^2 = |a|^2$ and, similarly, $p_1 = |\langle 1|q\rangle|^2 = |b|^2$. Because $|a|^2 + |b|^2 = 1$, the relation $p_0 + p_1 = 1$ is satisfied.

More generally, the measurement of an N-dimensional quantum state $|\psi\rangle$ in the orthonormal basis $\{|u_i\rangle\}_{i=1}^N$ is the set of projection operators $P_i = |u_i\rangle\langle u_i|$. The probability p_i that the measurement of state $|\psi\rangle$ yields basis state $|u_i\rangle$ is

$$p_i = |\langle u_i|\psi\rangle|^2. \tag{9}$$

This property is called *the Born rule*.

One way to interpret a quantum algorithm is to see it as a series of quantum operators applied to an initial quantum state in such a way that the final state has a high probability of being measured in a desired basis state.

Observables

Observables are linear operators that do not preserve the norm, but have the property that all their eigenvalues are real. These operators are self-adjoint, $K^\dagger = K$, and are called

Hermitian.[4] Let $|\kappa_j\rangle$ be eigenvectors of the observable K with (real) eigenvalues κ_j, so that $K|\kappa_j\rangle = \kappa_j|\kappa_j\rangle$. The observable K then can take the form

$$K = \sum_j \kappa_j |\kappa_j\rangle\langle\kappa_j|, \tag{10}$$

where the states $|\kappa_j\rangle$ form an orthonormal basis.[5] Denoting by $P_j = |\kappa_j\rangle\langle\kappa_j|$ the projection operator onto the subspace spanned by $|\kappa_j\rangle$, we can connect the observable K to the quantum measurement defined by the complete set of projection operators $\{P_j\}$ with outcomes $\{\kappa_j\}$. This quantum measurement is often called the measurement of the observable K.

For a quantum state $|\psi\rangle$, the expectation value of the result of the measurement of K is

$$\mathbb{E}_\psi[K] \equiv \langle K\rangle = \sum_j p_j\kappa_j = \sum_j |\langle\kappa_j|\psi\rangle|^2\kappa_j, \tag{11}$$

where $\langle\bullet\rangle$ denotes the expectation value of an observable and $p_j = |\langle\kappa_j|\psi\rangle|^2$ is the probability that the measurement of K in the state $|\psi\rangle$ yields the value κ_j. Since $|\langle\kappa_j|\psi\rangle|^2 = \langle\psi|\kappa_j\rangle\langle\kappa_j|\psi\rangle$, the expectation value is commonly written as

$$\langle K\rangle = \sum_j \langle\psi|\kappa_j\rangle\langle\kappa_j|\psi\rangle\kappa_j = \langle\psi|\left(\sum_j \kappa_j|\kappa_j\rangle\langle\kappa_j|\right)|\psi\rangle = \langle\psi|K|\psi\rangle, \tag{12}$$

as a consequence of linearity of inner and outer products.

Similar logic applies to the expectation value of higher moments of K, e.g., the variance $\mathrm{Var}(K)$. Given a Hermitian K and an integer a, the operator K^a is also Hermitian[6] and can serve as a quantum mechanical observable. The eigenvectors $|\kappa_j\rangle$ of the operator K are also eigenvectors of K^a, with eigenvalues κ_j^a, so that higher moments of K are expressed as:

$$\langle K^a\rangle = \sum_j \kappa_j^a p_j = \sum_j \kappa_j^a |\langle\kappa_j|\psi\rangle|^2 = \langle\psi|K^a|\psi\rangle, \tag{13}$$

with the variance of K equal to $\mathrm{Var}(K) = \langle K^2\rangle - \langle K\rangle^2$.

Unitary and Hermitian operators

For any unitary operator U there exists a Hermitian operator K_U, such that $U = e^{\mathrm{i}K_U}$, where i is the imaginary unit. The operator K_U has a set of eigenstates $\{|u_j\rangle\}$ with eigenvalues $u_j \in \mathbb{R}$, i.e. for any $|u_j\rangle$: $K_U|u_j\rangle = u_j|u_j\rangle$. The eigenstates $\{|u_j\rangle\}$ are also eigenstates of the operator U, with eigenvalues $e^{\mathrm{i}u_j}$. Conversely, for any Hermitian K, the operator $U_K = e^{\mathrm{i}K}$ is unitary. This connection between unitary and Hermitian operators is widely used in quantum algorithms.

[4] Historically, observables corresponded to physical properties, such as energy or momentum, of quantum states that could be observed in physics experiments.

[5] If there are degenerate subspaces of K, we can choose an orthonomal basis.

[6] The product of two Hermitian operators AB is not Hermitian in general; it is Hermitian only if A and B commute. For an integer a, the operator K commutes with itself and K^{a-1}; therefore, K^a is Hermitian.

Time-evolution of quantum states and the Hamiltonian of a system

Because all reversible transformations of quantum states are unitary, the evolution of a quantum state between time t_1 and time t_2 is a unitary operator $U(t_2, t_1)$:

$$|\psi(t_2)\rangle = U(t_2, t_1)\,|\psi(t_1)\rangle\,. \tag{14}$$

For the time-evolution operator $U(t_2, t_1)$, there is a Hermitian operator $K_U(t_2, t_1)$, such that $U(t_2, t_1) = e^{-\mathrm{i}K_U(t_2,t_1)}$. If the system is *stationary* – its fundamentals do not change over time – then we can write $K(t_2, t_1) = \mathcal{H} \times (t_2 - t_1)$, where $\mathcal{H}$ is a Hermitian operator called the *Hamiltonian* of the system.[7] For a stationary system we have[8]

$$|\psi(t)\rangle = e^{-\mathrm{i}\mathcal{H}t}\,|\psi(0)\rangle\,. \tag{15}$$

Time evolution of a quantum state plays a central role in quantum algorithms. The first proposed use for quantum computers was the simulation of quantum mechanical systems (Feynman, 1981). Classical simulations of quantum systems are exponentially hard, quickly running into limitations of current technology. But quantum computers may be able to simulate the evolution of quantum systems in polynomial time. The ability to simulate complex physical systems efficiently would take our engineering to the next level, allowing us to design new materials, fertilizers, superconductors, or pharmaceuticals at the molecular level. Furthermore, the ability to simulate Hamiltonians can help us solve problems beyond direct simulations of quantum systems. Later in this review, we discuss extensions of Hamiltonian simulation techniques to solve combinatorial optimization problems or problems that can be cast as linear systems of equations.

Density matrix formulation of quantum states

So far in this review, we have described quantum states using kets $|\psi\rangle$ – vectors in a Hilbert space. The states that can be represented as vectors in a Hilbert space are called *pure* quantum states. In this section, we briefly introduce an alternative way to describe quantum states using *density operators* aka *density matrices*. Density matrices are more general than kets because enable us to describe not only the pure quantum states but also *mixed* quantum states – classical probabilistic ensembles of pure quantum states. For example, Alice prepares for Bob the pure state $|0\rangle$ with probability 1/3 and pure state $|1\rangle$ with probability 2/3, then the resulting state is a mixed state $\{(1/3, |0\rangle), (2/3, |1\rangle)\}$. Note that this state is different from $1/\sqrt{3}\,|0\rangle + \sqrt{2}/\sqrt{3}\,|1\rangle$, which is a pure state, or from $1/3\,|0\rangle + 2/3\,|1\rangle$ which is not a valid quantum state as the amplitudes are not normalized. Density matrices, in effect, exist on the continuum between quantum states and classical probability distributions. Because of this, density matrices most commonly appear in the literature concerning noise and decoherence – the loss of "quantumness" over time – in physical quantum computers. However,

[7] Sometimes the inverse of Planck's constant $1/\hbar$ pre-multiplies $\mathcal{H}$ in $K_U(t_2, t_1) = \frac{1}{\hbar}\mathcal{H} \times (t_2 - t_1)$, so that eigenvalues of H have the units of energy. In quantum computing and much of quantum physics literature an assumption is made that $\hbar = 1$, corrected when it becomes necessary to consider relative energy scales.

[8] With a time-dependent Hamiltonian, the time evolution operator is more complex $U(t) = \mathcal{T}\exp\Big(-\mathrm{i}\int_0^t \mathcal{H}(t_1)d(t_1)\Big)$, where $\mathcal{T}$ is the time ordering operator which orders operators in the Taylor expansion of the exponent.

some important quantum algorithms, such as the quantum Principal Component Analysis algorithm (Section 6.6), also rely on the density matrix formalism.

The density operator of a pure state $|\psi\rangle$ is defined as $\rho_\psi = |\psi\rangle\langle\psi|$. Consider a system that is in a pure state $|\psi_i\rangle$ with probability p_i. The density operator for the system is

$$\rho = \sum_i p_i \, |\psi_i\rangle\langle\psi_i| \, . \tag{16}$$

All the quantum postulates can be equivalently reformulated using density operators. For example, given quantum measurement operators $\{M_m\}$, the probability of getting outcome m is

$$p(m) = \langle\psi| \, M_m^\dagger M_m \, |\psi\rangle = \mathrm{tr}(M_m^\dagger M_m \rho_\psi).$$

When a unitary operator U is applied to a quantum state ρ comprising pure states $|\psi_i\rangle$, the operator acts on each of the states $|\psi_i\rangle$: $U : |\psi_i\rangle \to U \, |\psi_i\rangle$. The state ρ becomes

$$U : \rho = \sum_i p_i \, |\psi_i\rangle\langle\psi_i| \to \sum_i p_i U \, |\psi_i\rangle\langle\psi_i| \, U^\dagger = U \rho U^\dagger. \tag{17}$$

The expectation value of an observable in a mixed quantum state equals to the trace of the product of the observable with the density matrix:

$$\langle K \rangle = \mathrm{Tr}(K\rho) = \sum_i p_i \, \mathrm{Tr}(K \, |\psi_i\rangle\langle\psi_i|) = \sum_i p_i \, \langle\psi_i| \, K \, |\psi_i\rangle \, , \tag{18}$$

where we used the linearity and the cyclic property of the trace. For a pure state $\rho = |\psi\rangle\langle\psi|$, the expectation value $\langle K \rangle$ equals $\mathrm{Tr}(K\rho) = \langle\psi| \, K \, |\psi\rangle$, just as we have seen in Eq. (12).

An important concept often used in the density matrix formulation of quantum computing is *partial trace*. If the quantum system, described by a density operator ρ, is defined over a Hilbert space that is a tensor product of two Hilbert spaces $H^A \otimes H^B$, we can define a partial trace $\mathrm{Tr}_B(\rho)$ to obtain a density operator ρ^A of the subsystem H^A. This concept is similar to marginalizing a joint probability distribution to obtain a marginal distribution. Let the set of states $\{|u_B\rangle\}$ comprise an orthonormal basis of subspace H^B. Taking a partial trace Tr_B of the density matrix ρ over the subspace H^B results in a reduced density matrix ρ^A on H^A:

$$\rho^A = \mathrm{Tr}_B \, \rho = \sum_{u_B} \langle u_B| \, \rho \, |u_B\rangle \, . \tag{19}$$

The density matrix is a Hermitian operator, and we can transform density matrices using the quantum algorithmic building blocks that apply to general Hermitian operators. For example, because ρ is Hermitian, $e^{-i\rho t}$ is unitary – a property used in the quantum principal component analysis algorithm (Section 6.6).

2.3 Quantum Gates and Other Primitives

Quantum computation is a transformation of a quantum input state into a quantum output state via a series of unitary operations and measurements. These operations and measurements are implemented physically by *quantum circuits*, analogous to classical circuits, which are series of basic *quantum gates*.

Quantum gates are building blocks of quantum circuits. They are unitary transformations applied to one, two, or three qubits at a time.[9] This section provides an overview of a few most common gates. For a more extended discussion of quantum circuits, see, e.g. Nielsen and Chuang (2002) and Kitaev et al. (2002); for a general theory of quantum circuits, see, e.g. Aharonov et al. (1998).

Hadamard Gate

The most widely used quantum gate is the Hadamard gate. In the computational basis $\{|0\rangle, |1\rangle\}$, it is represented by the matrix H:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{20}$$

When applied to a qubit in the basis state $|0\rangle$ it creates a uniform superposition of states $|0\rangle$ and $|1\rangle$, often denoted as $|+\rangle$: $|+\rangle = H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Similarly, $|-\rangle = H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Application of the Hadamard gate to each qubit of an n-qubit register creates a uniform superposition of the $N = 2^n$ possible computational basis states

$$H^{\otimes n}|0\ldots0\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle, \tag{21}$$

where $|i\rangle$ is a shorthand for the computational basis state of n-qubits that can be interpreted as an n-bit integer i; the notation $H^{\otimes n}$ means that the gate H is applied to each of n qubits once (as compared with H^n, which means n sequential applications of H to the same qubit). The operator $H^{\otimes n}$ that creates a uniform superposition on n qubits is sometimes called the *Walsh-Hadamard transform*.

Single-Qubit Rotations

To understand qubit rotations, we reparametrize the qubit in Eq. (1) in terms of angular coordinates of a unit sphere

$$|q\rangle = \cos(\theta/2)|0\rangle + e^{\mathrm{i}\phi}\sin(\theta/2)|1\rangle. \tag{22}$$

The unit sphere representing a qubit is called the *Bloch sphere*. It is a two-dimensional object embedded in a three dimensional space. A single-qubit unitary transformation can be decomposed into a sequence of basic rotations around x, y, and z axes

$$R_x(\varphi) = e^{-\mathrm{i}X\varphi/2} = \begin{pmatrix} \cos\varphi/2 & -\mathrm{i}\sin\varphi/2 \\ -\mathrm{i}\sin\varphi/2 & \cos\varphi/2 \end{pmatrix} \tag{23}$$

$$R_y(\varphi) = e^{-\mathrm{i}Y\varphi/2} = \begin{pmatrix} \cos\varphi/2 & -\sin\varphi/2 \\ \sin\varphi/2 & \cos\varphi/2 \end{pmatrix}, \tag{24}$$

$$R_z(\varphi) = e^{-\mathrm{i}Z\varphi/2} = \begin{pmatrix} e^{-\mathrm{i}\varphi/2} & 0 \\ 0 & e^{\mathrm{i}\varphi/2} \end{pmatrix}, \tag{25}$$

[9] A general quantum transformation requires exponentially many quantum gates (Knill, 1995). The art of writing quantum algorithms is in finding ways to perform useful transformations efficiently with respect to all resources – time, qubits, and gates.

where $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, $Y = \begin{pmatrix} 0 & -\mathrm{i} \\ \mathrm{i} & 0 \end{pmatrix}$, and $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ are *Pauli matrices* (sometimes also written as σ_x, σ_y, and σ_z).

The NOT Gate

The Pauli matrix X has the effect of a NOT gate: $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$.

Controlled-NOT Gate

The Controlled-NOT gate is an example of a two-qubit gate. It applies the NOT gate to the second qubit only if the first qubit is in the state $|1\rangle$. As a matrix in the computational basis of two qubits, $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, Controlled-NOT ($C$-$X$) gate is

$$C\text{-}X = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

so that $C\text{-}X|00\rangle = |00\rangle$, $C\text{-}X|01\rangle = |01\rangle$, $C\text{-}X|10\rangle = |11\rangle$ and $C\text{-}X|11\rangle = |10\rangle$. An alternative way to write down the Controlled-NOT gate is using the bra-ket notation:

$$C\text{-}X = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X. \tag{26}$$

When used as a part of an operator, tensor product $\otimes$ shows that different parts of the operator apply to different qubits (or, more generally, registers). Consider a two-register state $|\psi\rangle|\phi\rangle \equiv |\psi\rangle \otimes |\phi\rangle$, then the operator $A \otimes B$ performs the operation A on the state $|\psi\rangle$ in the first register and the operation B on the state $|\phi\rangle$ in the second register:

$$(A \otimes B)|\psi\rangle|\phi\rangle = (A|\psi\rangle) \otimes (B|\phi\rangle). \tag{27}$$

It is straightforward to verify that the operator C-X in Eq. (26) applied to two-qubit states performs the desired controlled-NOT operation.

The three-qubit extension of the Controlled-NOT gate is the Toffoli gate, also known as the $CCNOT$ gate. It applies NOT to the third qubit conditionally on the state of the first two qubits being $|11\rangle$.

Basic gates such as rotations, the NOT gate, CNOT gate, or the Hadamard gate can be implemented on existing quantum computers.

Auxiliary Qubits

Many algorithms require supplementary qubits in addition to the qubit registers encoding data. These qubits are called *auxiliary* or *ancilla* qubits. The addition of a single auxiliary qubit effectively doubles the Hilbert space: An n-qubit register spans a 2^n-dimensional Hilbert space; the addition of an auxiliary qubit expands this Hilbert space to 2^{n+1} dimensions. Therefore, the addition of auxiliary qubits embeds the data registers in a larger space enabling, for example, non-linear transformation of the data registers.

Controlled Rotation

Controlled-rotation is the application of a series of gates that acts on an auxiliary qubit conditionally on the state of one or more other qubits. For example, consider a state $|x\rangle$, which encodes an n-bit binary string x on a register of n qubits. Append an auxiliary qubit. A popular form of controlled rotation is

$$C\text{-}R_y(f(x)) = |x\rangle\langle x| \otimes e^{-iYf(x)}, \tag{28}$$

where R_y is a single-qubit rotation operator introduced in (24), and $f(x)$ is a function of x reasonably simple to compute. In matrix notation, the operator $e^{-iY\phi}$ has the effect of $e^{-iY\phi} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}$, so that the effect of the operator on the state $|x\rangle$ and the auxiliary qubit is

$$C\text{-}R_y(f(x))\,|x\rangle\,|0\rangle = |x\rangle\,\big(\cos f(x)\,|0\rangle + \sin f(x)\,|1\rangle\,\big). \tag{29}$$

A common example of $f(x)$ is arcsine of x/C, where C is a constant selected so that $|x/C| \leq 0.5$. Arcsine is efficient to compute using an expansion based on the inverse square root (see, e.g. Häner et al., 2018, and references therein). Quantum algorithms exist to approximate the arcsine function, inspired by classical reversible algorithms for the inverse square root – used, e.g., in gaming (Lomont, 2003).

Controlled rotation supports important quantum algorithms such as the application of a Hermitian operator (Section 6.3) or the HHL quantum linear systems algorithm (Section 6.4).

Controlled Unitary

Controlled-unitary $C\text{-}U$ is an operation applied to a multi-qubit register conditional on the state of an auxiliary qubit:

$$C\text{-}U = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U. \tag{30}$$

The controlled-unitary operation exists for some, but not all unitaries Lloyd et al. (2014).

When the operator U applies to a single-qubit register, $C\text{-}U$ is implemented using a decomposition of the operator U such that $U = AXBXC$ and $ABC = I$. Then, substituting CNOT ($C\text{-}X$) for NOT (X), we get the desired controlled-unitary.

Oracles

Oracles are not gates, but they too are important algorithmic building blocks – in both classical and quantum computation. Oracles are "black box" parts of algorithms that solve certain problems in a single operation. They are calls of a function that do not take into account the structure of the function.

Quantum algorithms employ two types of oracles – classical oracles, which provide a classical solution to a given problem, and quantum oracles, which make the solution available to the quantum computer as a quantum state.

Postselection

Another important algorithmic building block is *postselection*, where a quantum state is kept or discarded conditionally on the result of a measurement of a part of the state (usually an auxiliary qubit). Postselection enables nonlinear quantum transformations at the cost of having to discard quantum states where the measurement did not yield the desired result.

For example, let $|\psi\rangle$ be a quantum state such that $|\psi\rangle = \sum_x \psi_x |x\rangle$, where the states $\{|x\rangle\}$ form an orthonormal basis. We can apply a controlled rotation in Eq. (29) to the state $|\psi\rangle$ and an auxiliary qubit. The result of the controlled rotation is

$$C\text{-}R_y |\psi\rangle |0\rangle = \sum_x \psi_x |x\rangle \left(\cos f(x) |0\rangle + \sin f(x) |1\rangle \right). \tag{31}$$

We now measure the auxiliary qubit. If the measurement yields 1, we keep the state; if 0, we discard it and then repeat the preparation of the state $|\psi\rangle$, the controlled rotation, and the measurement until the measurement yields 1. The state we keep will be proportional to $\sum_x \psi_x \sin f(x) |x\rangle$ – a nonlinear transformation of the original state $|\psi\rangle$.

2.4 Properties of Quantum Computers

Quantum computers exploit the laws of quantum physics as a computational resource, resulting in a mode of computation different from that of classical computers. But algorithms that deliver efficiency gains may look unfamiliar and, at times, counter-intuitive, to classical algorithm designers. Quantum computers behave sufficiently differently that even simple operations like copying, erasing, or addition proceed very differently on a quantum v.s. on a classical computer (Draper, 2000; Häner et al., 2018). This section reviews a few properties of quantum computers that highlight these differences.

No Cloning Theorem

An important property of quantum computers that sets them apart from classical computers is the *No Cloning Theorem*. The theorem states that for a general unknown quantum state $|\psi\rangle$ there is no unitary operator O_C that makes an exact copy of $|\psi\rangle$:

$$O_C : |\psi\rangle \otimes |0\rangle \to e^{\alpha(\psi)} |\psi\rangle \otimes |\psi\rangle \text{ does not exist.}$$

Copying is possible if $|\psi\rangle$ is known, by repeating the process of creating the state in a different register. However, the No Cloning Theorem states that it is not possible to copy the unknown result of a computation, for example to conduct repeated measurements. When repeated measurements are required – as is most often the case when a quantum result needs to be interpreted classically – the result has to be re-computed after each measurement.

Reversible computation

Another property of quantum computers is that all computation with unitary gates is reversible — it neither creates nor destroys information. Classical algorithms are often irreversible. Common Boolean gates such as AND or OR are irreversible (Vedral et al., 1996).

For example, given the result of a AND b, it is not possible to recover the values of a and b. An example of a reversible classical gate is the NOT gate: given NOT a, we can recover a.

Any classical algorithm can be represented in terms of reversible gates (Bennett, 1989), e.g. the Fredkin gate, which has three input bits and three output bits. One of the bits is the control bit; if the control bit is 1, then the values of the order two bits are swapped. This logic gate is not only universal – i.e. can be cascaded to simulate any classical circuit – it is also self-inverse and conservative – i.e. it conserves the number of 0 and 1 bits.

Any classical algorithm can therefore be simulated on a quantum computer. First, the classical algorithm is rewritten using reversible gates, then these gates are translated into unitary gates. Such direct translations, however, may be inefficient, because they do not leverage quantum properties and simply replicate classical ideas on the more expensive and noisy hardware of a quantum computer. Efficient quantum algorithms, such as the ones described in the rest of this review, often have a structure fundamentally different to that of the classical algorithms designed for similar tasks.

Uncomputing

Computation often results in temporary "garbage" data produced in auxiliary registers. During classical computation such data can be discarded, but during a quantum computation these "garbage" data may be entangled with the main result of the computation, affecting the ability to extract the result accurately. *Uncomputing* – running parts of a quantum algorithm "in reverse" – is a way to remove the "garbage" data and clear the auxiliary register. Let algorithm $\mathcal{A}$ be such that $\mathcal{A}|0\rangle = |\psi\rangle$. The inverse $\mathcal{A}^{-1}$ uncomputes the register containing the state $|\psi\rangle$: $\mathcal{A}^{-1}|\psi\rangle = \mathcal{A}^{-1}\mathcal{A}|0\rangle = |0\rangle$.[10]

Physical Implementation on NISQs and Beyond

Today's quantum computers are small-scale (i.e., having a small number of qubits) and noisy, similar to classical computers in the 1950s. The largest quantum computers today comprise up to 100 qubits, with error rates at best around 1% and *coherence time* – the duration of time qubits can represent quantum states with sufficient accuracy – up to 100 microseconds. In the 1950s, classical computers suffered a similar problem. Made of vacuum tubes or mechanical relays, the bits in the early classical computers tended to flip at random, introducing errors. To perform computation on these computers, error correction code, based on redundant bits, was required. Modern classical computing technology is so advanced, that bits are extremely stable without error correction. But quantum computers still require redundant qubits to compensate for the decoherence errors. As a result, a system of 100 qubits may have an order of magnitude fewer *effective* qubits for computation. The connectivity between qubits – i.e. our ability to apply two or three-qubit gates with high accuracy – and also how long it takes to apply a gate also play an important role.[11]

[10] Aharonov and Ta-Shma (2007) demonstrate that if quantum computers were able to "forget" information, they could solve the NP-complete graph isomorphism problem efficiently. The uncomputing requirement is consequential – it limits the quantum computers' power.

[11] For a popular account of the role of noise in quantum computing and the importance of quantum error correction, see e.g. Cho (2020).

Most quantum algorithms that offer provable speedups over classical counterparts require much larger, fault tolerant quantum computers. Two metrics can help identify the quantum resources an algorithm might require: the size of the qubit registers and the *circuit depth* of the algorithm. *Circuit depth* refers to the number of gates required to implement a quantum algorithm on a quantum computer sequentially. Higher circuit depth algorithms require higher coherence times.

But even though today's quantum computers are relatively small and noisy, we may be crossing over to the era when these computers are able to solve some classes of problems better than classical computers. In 2018, Preskill (2018) called these noisy computers with 50+ qubits NISQs – noisy intermediate-scale quantum systems. Some algorithms such as the Quantum Approximate Optimization Algorithm, described in Section 8.2, or the Variational Quantum Eigensolver – an algorithm important in simulating quantum physical systems – can be implemented on NISQs. For a discussion of quantum algorithms on NISQs, see e.g. Bharti et al. (2021) and references therein.

The most popular quantum computing technologies today are based on superconductors and cold ions. Superconductor-based quantum computers have been able to achieve highest qubit counts and fast gate times, but the gates and qubits on these computers are noisy. Ion-based quantum computers offer slower gate times, but much higher qubit fidelities and greater connectivity. Other qubit types include silicon qubits, nitrogen-vacancy qubits, and optical qubits.

3 An Overview of Quantum Algorithm Design

This section presents the general structure of quantum algorithms, and highlights the considerations required for the development of successful quantum algorithms.

A general quantum algorithm often proceeds in three steps:

1. Quantum state preparation.
2. Quantum computation.
3. Postprocessing and readout of resulting quantum state

If a quantum algorithm is embedded in another quantum algorithm, then steps 1 and 3 may be omitted; however, a quantum algorithm that classical data as its input and delivers a result for use by classical computers or for human interpretation requires all three steps. We focus the discussion on steps 1 and 3 in this section, while step 2 is the subject of the subsequent sections.

3.1 Data Encoding in a Quantum State

The first thing that a statistician may like to know when using quantum computing in statistics is how to import classical data into a quantum computer. Data can be imported into a quantum computer using *quantum state preparation* - the process of encoding data into a quantum state supported by one or more qubit registers. The qubit registers typically start out initially in the *ground state* – all qubits are in the 0 state. Quantum state preparation is

a quantum routine that transforms this initial state into a state that encodes the necessary data.

Efficient loading of data onto a quantum computer is an open area of research (Ciliberto et al., 2018). The data loading step can require significant computational resources and, if not carefully thought out, can offset the computational efficiency attained via quantum computation. Similarly, extracting the result from a quantum state can be a resource-consuming task requiring careful planning as part of the algorithm design. The creation of a general quantum state on n qubits can be computationally taxing and requires, at a minimum, $O(\frac{2^n}{n})$ quantum operations (see, e.g. Prakash, 2014; Schuld and Petruccione, 2018). The computational complexity of data preparation is reduced considerably when it is possible to exploit the structure of the data, such as if the data has a functional form. For example, if the amplitudes represent probability densities of a discretized integrable probability distribution function, loading can be achieved more efficiently (Grover and Rudolph, 2002). Additional proposals include loading pre-compressed data for analysis, see e.g. Harrow (2020).

We now describe a few methods for encoding data in a quantum state.

Amplitude Encoding

A quantum state provides several natural ways to encode data. Consider a general n-qubit quantum state in Eq. (3). One of the most common ways to encode data in this state is *amplitude encoding* where data are encoded in the amplitudes ψ_i and the basis vectors $|i\rangle$ serve as indices. For example, a vector $x \in \mathbb{C}^{2^n}$, normalized so that $\|x\| = 1$, can be encoded as

$$|x\rangle = \sum_{i=0}^{2^n-1} x_i \, |i\rangle \, . \tag{32}$$

The basis vectors $|i\rangle$ serve as indices and the amplitudes x_i encode the data. This type of encoding is widely used in quantum linear systems of equations (Section 6) and related algorithms. The benefit of this encoding is that it is *qubit efficient*, i.e. a vector of length N needs only $O(\log N)$ qubits (Prakash, 2014; Adcock et al., 2015). The downside is that it may be difficult to initialize and to read out. Initialization may require an intermediate step, such as quantum Random Access Memory (see below). Readout within error ϵ generally requires $O(N/\epsilon^2)$ measurements. Even though, in some cases more efficient readout is possible, for example through compressive sensing methods (see Section 3.2), sometimes alternative ways to transfer information to the classical computer are used, such as distilling the results of a classification algorithms to few-bit summaries that are more efficient to read out.

Computational Basis Encoding

An alternative way to encode the data in an n-qubit quantum state is to encode information in the basis vectors $|i\rangle$ (as opposed to the amplitudes as in (32)). Consider a data set of M vectors $D = \{x^m = (x_1^m, ..., x_N^m)^\top, m = 1, ..., M\}$, each of a dimension of N. Suppose each vector x^m has been already represented by a binary string with $N\tau$ bits

$$x^m = b_{x_1^m} ... b_{x_N^m}$$

where $b_{x_j^m}$ is the binary representation of x_j^m (a string of τ bits with τ the precision). There exists a procedure to prepare data D in the superposition

$$|D\rangle = \frac{1}{\sqrt{M}} \sum_{m=1}^{M} |x^m\rangle$$

with $|x^m\rangle$ the basis quantum state corresponding to the binary representation of x^m; see, Ventura and Martinez (2000) or Schuld and Petruccione (2018), Ch. 5. Technically, $|D\rangle$ can be understood as a superposition state with respect to the computational basis $\{|0\rangle, ..., |2^{N\tau} - 1\rangle\}$, where the basis states corresponding to the $|x^m\rangle$ have the amplitude of $1/\sqrt{M}$ and other states have zero amplitude. This data encoding is known as *basis* or *computational basis* encoding, requires $O(N\tau)$ qubits and takes $O(MN)$ operations to initialize.

The benefit of computational basis encoding is that it enables quantum algorithms to directly leverage quantum parallelism. For example, let U be an operator that implements a function $f(x^m)$:

$$U : |x^m\rangle |0\rangle \mapsto |x^m\rangle |f(x^m)\rangle ,$$

then

$$U : |D\rangle |0\rangle \mapsto \frac{1}{\sqrt{M}} \sum_{m=1}^{M} |x^m\rangle |f(x^m)\rangle .$$

That is, a single application of U gives us M values $f(x^1), ..., f(x^M)$ encoded in a superposition quantum state. Other uses of computational basis encoding include applications in optimization where the quantum optimization algorithms aim to amplify the optimal entry x^m (Section 8).

Qsample Encoding and Quantum Sample State

Another encoding, often called *qsample* encoding (Aharonov and Ta-Shma, 2003), can be used to encode a probability distribution P on a finite set $\{x^m, m = 1, ..., M\}$ with probabilities $p(x^m)$

$$|P\rangle = \frac{1}{\sqrt{M}} \sum_{m=1}^{M} \sqrt{p(x^m)} |x^m\rangle . \tag{33}$$

The quantum state $|P\rangle$ in (33) is also known as *quantum sample state*. This encoding uses the amplitudes $\sqrt{p(x^m)}$ to encode the probabilities $p(x^m)$ and the basis vectors to encode the data points x^m. The qsample encoding is appropriate for use in statistics, especially in Monte Carlo methods. For example, as a measurement in the computational basis yields x^m with probability $p(x^m)$, the measurement serves as a sampling technique: it generates samples from the distribution P. Also, it is computationally efficient, compared to classical methods, to estimate the expectation of a function with respect to the probability distribution P if it is encoded in $|P\rangle$; see Section 4.8. As we will see later in Section 5.3, the output of quantum Markov chain Monte Carlo is a quantum sample state in the form of (33).

Data Encoding Using Multiple Qubit Registers

Splitting the collection of n qubits into multiple registers makes further data structures possible. Consider a collection of r registers of n_k, $k = 1, .., r$ qubits each, such that the total number of qubits is n: $\sum_{k=1}^{r} n_k = n$. Each basis vector $|i\rangle$ of the 2^n-dimensional Hilbert space of n-qubits can be expressed as a tensor product of basis states of the 2^{n_k}-dimensional subspaces spanned by each n_k-qubit register: $|i\rangle = |i_1\rangle \otimes |i_2\rangle \otimes \ldots \otimes |i_r\rangle \equiv |i_1\rangle |i_2\rangle \ldots |i_r\rangle$. In this notation, we can re-express the quantum state in Eq. (3) as a multi-register state:

$$|\psi\rangle = \sum_{i_1=0}^{2^{n_1}-1} \sum_{i_2=0}^{2^{n_2}-1} \cdots \sum_{i_r=0}^{2^{n_r}-1} \psi_{i_1,i_2,..,i_r} |i_1\rangle |i_2\rangle \ldots |i_r\rangle . \tag{34}$$

This general form provides a rich set of possibilities for encoding data including QRAM and QROM that we describe next.

QRAM

A structure particularly popular in quantum machine learning is the quantum Random Access Memory (QRAM). Classical RAM is a scheme which, given an index i, outputs the data element x_i stored at the address indexed with the unique binary address i. QRAM is a scheme that, given a superposition of states corresponding to index values in an input register and an empty output data register, outputs the data elements into the data register

$$|\psi_{in}\rangle = \sum_{i=0}^{N-1} a_i |i\rangle^{in} |0\rangle^{out} \mapsto |\psi_{out}\rangle = \sum_{i=0}^{N-1} a_i |i\rangle^{in} |x_i\rangle^{out} , \tag{35}$$

where both i and x_i are recorded in the computational basis; N represents the size of the memory; and the coefficients a_i provides the (optional) weights of the various addressed data elements. For an overview of the method, see Hann et al. (2021) and also Giovannetti et al. (2008b,a).

The QRAM data structure is suitable for use in some algorithms directly; in others, QRAM is a stepping stone to amplitude encoding, where it is possible to use a controlled rotation to turn the QRAM encoding into amplitude encoding efficiently.

Query complexity in QRAM encoding is $O(\log(N))$; however, the method needs $O(N)$ auxiliary qubits.[12] Critics of QRAM point out that QRAM requires unphysically high qubit fidelity to work (Arunachalam et al., 2015), although Hann et al. (2021) have recently demonstrated that QRAM is more robust than had been previously thought. Additionally, the approach requires a parallel gate architecture; if a classical computer leveraged a similar parallel architecture, it would be able to achieve similar speedups over sequential classical architecture as quantum computers do (Aaronson, 2015; Steiger and Troyer, 2016; Csanky, 1975).

[12] Alternatively, it is possible to reformulate QRAM so that its query complexity of $O(N)$ using $O(\log(N))$ auxiliary qubits.

QROM

Instead of storing data in a quantum state, it is possible to create a classical data structure that provides efficient quantum access to the data for use in some algorithms, such as those based on quantum singular transformation (Section 9). One such structure, proposed by Kerenidis and Prakash (2016) and named *quantum read-only memory (QROM)* by Chakraborty et al. (2018), can store a matrix $A \in \mathbb{R}^{M\times N}$ in $O(w \log^2 MN)$ *classical* operations, where w is the number of non-zero elements of A. Once the structure is in place, it is possible to perform the following quantum initializations with ϵ-precision in $O(polylog(MN)/\epsilon)$ time (requiring $O(N)$ gates accessed in parallel):

$$U : |i\rangle |0\rangle \mapsto |i\rangle \frac{1}{\|A_i\|} \sum_{j=1}^{N} A_{i,j} |j\rangle = |i\rangle |A_i\rangle \tag{36}$$

$$V : |0\rangle |j\rangle \mapsto |1\rangle \|A\|_F \sum_{i=1}^{M} \|A_i\| |i\rangle |j\rangle = |\tilde{A}\rangle |j\rangle , \tag{37}$$

where $|A_i\rangle$ is the quantum state encoding the normalized ith row of A; $|\tilde{A}\rangle$ is the quantum state such that its inner product with the quantum state corresponding to the row index $|i\rangle$ yields the normalization factor $\|A_i\|$: $\langle i|\tilde{A}\rangle = \|A_i\|$.

3.2 Result Postprocessing and Readout

Reading out the results of a quantum algorithm can be a challenging, resource-consuming step. The result of a quantum algorithm is a quantum state that may be handed off for processing either to a classical or a quantum algorithm. Post-processing by quantum algorithm is usually more efficient because the quantum-classical readout requires a number of operations that often overwhelms the number of operations required to run the algorithm itself (Zhang et al., 2021). Consider a state $|x\rangle = \sum_{i=0}^{N-1} x_i |i\rangle$ encoding an N-dimensional vector $x = (x_0, ...x_{N-1})$. Reading out the elements of vector x within precision ϵ requires a minimum of $O(N/\epsilon^2)$ measurements (O'Donnell and Wright, 2016). When the vector x is a result of a quantum computation that has a polylogarithmic complexity dependence on N, the readout complexity of $O(N/\epsilon^2)$ overwhelms the complexity of the quantum computation to obtain x. Full readout of a quantum state is called quantum *tomography*.

Improvements to readout complexity are only possible assuming prior knowledge of the structure of the quantum state to be read out. Efficient readout methodologies that exploit the quantum state's structure often involve reparametrization in order to reduce the effective dimensionality of the state. Methods include compressed sensing (Gross et al., 2010; Kyrillidis et al., 2018)[13], permutationally invariant tomography (Tóth et al., 2010; Moroder et al., 2012), schemes based on tensor networks (Cramer et al., 2010; Baumgratz et al., 2013; Lanyon et al., 2017), or mapping target states onto highly entangled but structured lower dimensional models, such as restricted Boltzmann machines (Torlai and Melko, 2018; Torlai et al., 2018).

[13]Classical compressed sensing is a method of recovering a sparse vector from a small number of measurements. Quantum measurement techniques leveraging compressed sensing aim to recover pure or mostly pure quantum state efficiently.

For many applications, full tomography of the quantum state may not be required. Aaronson (2019) proposed *shadow tomography*, a method to predict specific properties of the state, called target functions, without fully characterizing it. In order to predict with high probability an exponential number of target functions, it is often sufficient to have only a polynomial number of copies of the quantum state. Huang et al. (2020) improved the efficiency of shadow tomography to reduce its exponential circuit depth requirements. The method involves repeated application of random unitaries drawn from a purposefully constructed ensemble followed by a measurement of the resulting state in the computational basis. The expectation value of the repeated unitary transformations and measurements is, in itself, a transformation of the quantum state. The inversion of the transformation applied to the classical measurement provides a snapshot of the quantum state – the classical shadow of the state. Classical shadows are expressive enough to yield many efficient predictions of the quantum state (Huang et al., 2020): A shadow based on M measurements is sufficient to predict L linear functions O_i, with $i = 1, .., L$, of the quantum state up to an error ϵ, provided M exceeds $O(\log L \max_i \|O_i\|^2_{shadow}/\epsilon^2)$, where the norm $\|O_i\|^2_{shadow}$ depends on the distribution of random unitaries used in the construction of the shadow (see Huang et al., 2020, for further details); it has the property $\|O_i\|^2_{shadow} < 4^n \|O_i\|_\infty$, where n is the number of qubits and $\|\cdot\|_\infty$ denotes the operator norm.

Swap Test and Sample Mean Estimation

One of the methods used in post-processing a quantum result state is the *swap test* (see, e.g. Buhrman et al., 2001), used to estimate the inter product $a^\top b$ of two normalized vectors a and b, encoded in two states $|a\rangle$ and $|b\rangle$

$$|a\rangle = \sum_i a_i |i\rangle , \quad |b\rangle = \sum_i b_i |i\rangle .$$

For example, Schuld et al. (2016) use a swap test to perform a prediction by linear regression using a quantum algorithm. Swap test also provides an efficient way for computing a sample mean (see below).

The swap test applies a series of three-qubit *controlled swap gates* (also known as *Fredkin gates*), which swap two qubit states conditional on the state of the auxiliary qubit so that

$$c\text{-}SWAP\Big[\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |a\rangle \otimes |b\rangle \Big] = \frac{1}{\sqrt{2}}(|0\rangle |a\rangle |b\rangle + |1\rangle |b\rangle |a\rangle). \tag{38}$$

Applying a Hadamard gate to the auxiliary qubit and rearranging the terms results in

$$\begin{aligned} &\frac{1}{2}\big(|0\rangle (|a\rangle |b\rangle + |b\rangle |a\rangle) + |1\rangle (|a\rangle |b\rangle - |b\rangle |a\rangle)\big) \\ &= \frac{\alpha_1}{2} |0\rangle \frac{(|a\rangle |b\rangle + |b\rangle |a\rangle)}{\alpha_1} + \frac{\alpha_2}{2} |1\rangle \frac{(|a\rangle |b\rangle - |b\rangle |a\rangle)}{\alpha_2} \end{aligned}$$

with α_1 and α_2 the norm of $(|a\rangle|b\rangle+|b\rangle|a\rangle)$ and $(|a\rangle|b\rangle-|b\rangle|a\rangle)$, respectively. It is easy to see that

$$\alpha_1 = \sqrt{2(1+|\langle a|b\rangle|^2)}.$$

Hence, the probability of measuring state $|0\rangle$ in the first qubit is

$$p=\frac{\alpha_1^2}{4}=\frac{1}{2}\big(1+|\langle a|b\rangle|^2\big), \quad \text{hence,} \quad |\langle a|b\rangle|=\sqrt{2p-1}.$$

Estimating p by repeated measurement gives us an estimate of the absolute value $|\langle a|b\rangle| = |a^\top b|$.

Its sign can also be determined. Consider two states $\tilde{a}$ and $\tilde{b}$ that encode the vectors $\frac{1}{\sqrt{2}}(a_1,...,a_N,1)^\top$ and $\frac{1}{\sqrt{2}}(b_1,...,b_N,1)^\top$ respectively. Applying the swap test to these two states results in $|\langle\tilde{a}|\tilde{b}\rangle|=\sqrt{2p-1}$. By noting that $|\langle\tilde{a}|\tilde{b}\rangle|=a^\top b/2+1/2$, we have

$$a^\top b=2\sqrt{2p-1}-1.$$

Now, suppose that a data vector $(x_0,..,x_{N-1})$ has been encoded into a quantum state $|x\rangle = \sum_i x_i|i\rangle$. Applying the swap test to $|x\rangle$ and the uniform superposition state $|u\rangle=\frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}|i\rangle$ gives us an estimate of $\sqrt{N}\bar{x}$. The algorithm requires $O(1/\epsilon^2)$ measurements to achieve error tolerance of ϵ.

3.3 Computational Complexity

Computational complexity measures the resources, particularly time and memory, an algorithm requires to complete a computational task. For quantum algorithms, the most relevant dimensions of computational complexity are time, the number of qubits (qubit complexity) and the number of gates (gate complexity) required to complete a computation. Time complexity is the most commonly cited metric.

Computational complexity is often expressed as a function of the size of the input, N, using *Big O* notation (Nielsen and Chuang, 2002). The most popular measure of computational complexity is the *upper bound* $O(g(\cdot))$, which indicates that the resources required by the algorithm are bounded from above by a function $g(\cdot)$ of the relevant parameters, such as the size N. For example, naive matrix multiplication of two $N\times N$ matrices has time complexity of $O(N^3)$. Sometimes we use $\tilde{O}$, pronounced as "soft-O", to indicate that $\tilde{O}(g(\cdot))=O(g(\cdot)\log^k g(\cdot))$, for some finite k. Other measures of complexity that sometimes appear in the quantum computing literature are *lower bound* complexity $\Omega(\cdot)$ (pronounced as "Big Omega") and asymptotically tight complexity $\Theta(\cdot)$ (pronounced as "Big Theta") where lower and upper bound coincide.

The set of all problems that a quantum computer can solve in polynomial time – i.e. time polynomial in the size of the input N – with with an error probability of at most $1/3$ comprises the *complexity class BQP*. The complexity class BQP contains some problems that a classical computer cannot resolve in less than exponential time (time exponential in N) – NP-hard problems. However, the class BQP does not contain NP-complete problems – the NP-hard problems that, if solved a polynomial time, lead to a polynomial-time solution of all the other NP-hard problems.

For an informal (although mathematical) and engaging discussion of topics in computational complexity of quantum algorithms and the complexity classes relevant to quantum computation, see Aaronson (2013).

3.4 A Brief Overview of Quantum Algorithms

Having so far outlined the building blocks of quantum computing, we now give a brief overview of the popular families of quantum algorithms that comprise the current state of the art. These core algorithms will be reviewed in some detail in the remaining chapters. This list is not exhaustive - we only select the algorithms that may be most of interest to statisticians and data scientists.

We start with the Grover search family of algorithms (Section 4). Grover's search algorithm finds a labeled item in an unstructured database of size N using $O(\sqrt{N})$ queries, quadratically faster than the best classical approach that requires $O(N)$ queries. Grover proved that his algorithm is optimal – no quantum algorithm can perform unstructured search faster. The structure of Grover's algorithm turns out to be general enough to give rise to a family of algorithms. The most famous and widely used of these algorithms are Quantum Amplitude Amplification (QAA) and the closely related Quantum Amplitude Estimation (QAE) algorithms, which work by amplifying the amplitude of the desired quantum state. The Grover family of algorithms enable quantum Monte-Carlo integration and the estimation of the statistical properties of a function on an unstructured set, such as its mean, median, minimum and maximum. Additionally, Grover's search algorithm is closely related to quantum walks.

Quantum walks (Section 5) are quantum analogues of classical random walks or Markov chains, widely used in randomized classical algorithms. Quantum walks mix quadratically faster than classical random walks. This quadratic speedup can offer efficiencies in applications such as quantum Markov Chain Monte Carlo.

The next important family of quantum algorithms, presented in Section 6, leverages on the quantum Fourier transform (QFT) – the quantum analogue of the classical discrete Fourier transform. QFT leverages quantum parallelism – the ability to apply a function to multiple elements of a vector in parallel. For a vector of size N, QFT requires only $O(\text{polylog}(N))$ calls to the function, provided the vector is encoded into a quantum state using amplitude encoding. Classical discrete Fourier transform requires $O(N\text{log}N)$ calls to transform the classically encoded vector.

Arguably the most famous application of QFT is Shor's factoring algorithm, one of the first quantum algorithms with direct potential real-world consequences – the breaking of the encryption system based on prime factorization. Most encryption systems today that keep financial transactions and other sensitive information secure rely on the fact that factoring a large number N is exponentially hard for classical computers. Shor demonstrated that, using QFT, it is possible to perform the task in time polylogarithmic in N. Since Shor's algorithm does not have a direct application in statistics and many clear reviews of the algorithms are available (see, e.g., Nielsen and Chuang, 2002), we omit this algorithm in this review. We focus instead on another application of QFT, quantum phase estimation (QPE) — an algorithm to record an estimate of a phase, such as the phase θ in an eigenvalue of the form $e^{i\theta}$ of a unitary operator, in a quantum state in computational basis.

QPE is a building block of many algorithms of interest to statisticians. One example is the quantum linear systems algorithm first proposed by Harrow, Hassidim, and Lloyd (HHL). The critical part of the HHL algorithm is matrix inversion, which has a wide range of statistical applications, such as regression analysis. Other applications of QPE include

Quantum Principal Component Analysis (QPCA) and fast gradient estimation. QFT is at the heart of QPE, HHL and QPCA, and powers the exponential speedup that these algorithms deliver over their classical counterparts. Given vector inputs of size N and under certain constraints discussed in detail in Section 6, the QFT-based quantum algorithms deliver results using a number of queries polylogarithmic in N in contrast to classical analogues that require a number of queries at least linear in N.

The next family of core quantum algorithms are Hamiltonian simulation algorithms (Section 7) - algorithms to help simulate the evolution of complex quantum systems, including those of direct practical importance such as biologically active molecules. Advances in Hamiltonian simulation could change the face of agriculture, materials, and energy; they are potentially even more consequential than Shor's factoring algorithm. Many quantum algorithms rely on Hamiltonian simulation as a building block. For example, the HHL algorithm for quantum linear systems uses Hamiltonian simulation together with QPE to perform matrix inversion. The computational complexity of quantum linear systems algorithms has dramatically improved since HHL's first proposal, and most of these advances occurred because of more efficient Hamiltonian simulation techniques.

Another family of quantum algorithms of interest to statisticians is the family of quantum optimization algorithms (Section 8). These algorithms find a quantum state that minimizes a cost function. Adiabatic quantum computation (AQC) algorithms are optimization algorithms that leverage the adiabatic theorem. The adiabatic theorem states that a system in the eigenstate corresponding to the smallest eigenvalue of its Hamiltonian[14] will stay in this state if the system changes slowly enough in such a way that, throughout this evolution, there is a finite eigenvalue gap between the lowest-eigenvalue eigenstate and the next eigenstate. Adiabatic quantum computation is, in effect, analog quantum computing; nevertheless, on a noiseless quantum computer, adiabatic quantum computing is equivalent to circuit-based quantum computing (Aharonov et al., 2008). The Quantum Approximate Optimization Algorithm (QAOA) is a hybrid quantum-classical variational algorithm inspired by AQC. In QAOA, classically-controlled parametrized gates prepare a variational quantum state; the cost function takes the form of a quantum mechanical observable. The parameters are adjusted based on the measurements of the cost functions in a classical outer loop. QAOA can be a powerful method to obtain approximate solutions to combinatorial optimization problems, such as MaxCut.

Section 9 presents the algorithms based on Quantum Singular Value Transformation, a generalization of Quantum Signal Processing. Quantum Signal Processing embeds a quantum system in a larger system to perform a non-linear/non-unitary transformation of the subsystem. Quantum Singular Value Transformation extends the method to general rectangular matrices. By embedding a rectangular matrix inside a larger square unitary matrix, it is possible to perform polynomial transformations of singular values of the matrix. The versatile method acts as a unifying framework (Martyn et al., 2021), providing a consistent way to develop efficient versions of many existing algorithms such Grover search, Hamiltonian simulation, and matrix inversion and enables the systematic development of new algorithms. The algorithm represents the cutting edge of quantum algorithm development. For example,

[14]Physicists refer to the state corresponding to the smallest eigenvalue of its Hamiltonian as the *lowest-energy state* or *ground state* of the system.

it supports the most efficient quantum algorithm for matrix inversion, central to regression analysis. The flexible paradigm may result in novel algorithms of interest to statisticians being developed in the near future.

4 Grover's Search and Descriptive Statistics on a Quantum Computer

One of the most influential quantum algorithms is the search algorithm by Lov Grover (Grover, 1996). Grover considers the problem of searching for a binary string x_0 among N strings, provided that there is an oracle binary function such that $f(x)=0$ if $x \neq x_0$ and $f(x_0)=1$. The classical search algorithm over an unstructured space requires $O(N)$ oracle calls; Grover's quantum algorithm requires $O(\sqrt{N})$ offering a quadratic improvement.

Grover's algorithm has been influential because it led to the development of a class of practically important applications of quantum computers, including those for efficient estimation of statistical quantities such as the sample mean, median, or minimum (maximum) of a function over a discrete domain. Quantum Amplitude Amplification, based on Grover's algorithm, is widely used as a subroutine to many other quantum algorithms. It works by amplifying the amplitude of the correct result in a quantum state holding the result in a superposition with byproducts of computation.

4.1 Grover's Search Algorithm

Assume, without loss of generality, that $N=2^n$, where n is an integer. The starting state is a uniform superposition of all possible n-bit strings x created on n qubits:

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_x |x\rangle. \tag{39}$$

One of these bit strings is x_0 – the target. Grover's algorithm proceeds in a series of iterative steps, each amplifying the amplitude of state holding x_0 by $2/\sqrt{N}$. After $O(\sqrt{N})$ iterations this amplitude – and the corresponding measurement probability – becomes of order unity.

To understand Grover's algorithm, it is helpful to think of the state $|s\rangle$ as a superposition of the target state $|x_0\rangle$ and its orthogonal complement $|s'\rangle$:

$$|s'\rangle = \frac{1}{\sqrt{N-1}} \sum_{x \neq x_0} |x\rangle, \quad \langle s'|x_0\rangle = 0, \tag{40}$$

so that

$$|s\rangle = \frac{\sqrt{N-1}}{\sqrt{N}} |s'\rangle + \frac{1}{\sqrt{N}} |x_0\rangle. \tag{41}$$

When viewed as vectors in a Hilbert space, the starting state $|s\rangle$ is very close to $|s'\rangle$. Let $\theta/2$ be the angle between $|s\rangle$ and $|s'\rangle$. Then, by Eq. (41), $\sin(\theta/2) = 1/\sqrt{N}$.

At each iteration of Grover's algorithm, the quantum state of the system is pushed slightly toward $|x_0\rangle$ and away from $|s'\rangle$ by an angle θ. The state stays in the two-dimensional plane

spanned by $|x_0\rangle$ and $|s'\rangle$, and the result of each iteration is a superposition of $|x_0\rangle$ and $|s'\rangle$. After t iterations such that $\sin^2((t+\frac{1}{2})\theta)\approx 1$, the algorithm results in a quantum state which, when measured in the computational basis, yields x_0 with a probability close to 1. The number of iterations is therefore $t\approx\frac{\pi}{4}\sqrt{N}$ or $O(\sqrt{N})$.

Let U_G represent one Grover iteration. Each iteration consists of two unitary operators U_s and U_f: $U_G = U_s U_f$. The first operator is $U_f = I - 2|x_0\rangle\langle x_0|$, a reflection with respect to $|s'\rangle$ in the plane spanned by $|x_0\rangle$ and $|s'\rangle$ as it maps $|x_0\rangle$ into $-|x_0\rangle$. Application to the initial state $|s\rangle$ yields

$$U_f|s\rangle = (I - 2|x_0\rangle\langle x_0|)|s\rangle = |s\rangle - \frac{2}{\sqrt{N}}|x_0\rangle = |s'\rangle - \frac{1}{\sqrt{N}}|x_0\rangle. \tag{42}$$

The second operator is $U_s = 2|s\rangle\langle s| - I$, a reflection around the state $|s\rangle$ which, applied after the operator U_f, results in an increased amplitude of $|x_0\rangle$:

$$U_G|s\rangle = U_s U_f|s\rangle = (2|s\rangle\langle s| - I)(|s\rangle - \frac{2}{\sqrt{N}}|x_0\rangle) = (1 - \frac{4}{N})|s\rangle + \frac{2}{\sqrt{N}}|x_0\rangle. \tag{43}$$

So, the application of U_G to the state $|s\rangle$ has amplified the amplitude of the target state $|x_0\rangle$. Both operators U_f and U_s can be implemented on a quantum computer.

The implementation of operator U_f requires an auxiliary qubit and quantum oracle access O_f to a function $f(x)$. Given a value X encoded in a quantum state, the oracle O_f records the value $f(x)$ in a quantum state in the following way. Let $|x\rangle|y\rangle$ be a quantum system that represents the value x and, in the auxiliary qubit, some value y. The quantum oracle O_f applied to $|x\rangle|y\rangle$ mod-adds the value $f(x)$ to the value of the auxiliary qubit: $O_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$, where $\oplus$ indicates mod-addition. It is easy to show that, if we start with the auxiliary qubit in the Hadamard $|-\rangle$ state, we have $O_f|x\rangle|-\rangle = (U_f|x\rangle)|-\rangle$.

The operator U_s can be implemented using a decomposition into three parts: an uncomputation of $|s\rangle$ to recover the ground state $|0\rangle$, a sign-flip on the ground state, and a re-computation of $|s\rangle$. Let $\mathcal{A}$ be the operator that creates the state $|s\rangle$ when applied to the ground state: $|s\rangle = \mathcal{A}|0\rangle$. In the case when $|s\rangle$ is a uniform superposition of all n-bit strings as in (39), we know that $\mathcal{A} = H^{\otimes n}$, but we will use the more general form here because it will help derive the Quantum Amplitude Amplification algorithm in Section 4.2. Let $U_0 = I - 2|0\rangle\langle 0|$, an operator that changes the sign of the $|0\rangle$ state. Then,

$$-\mathcal{A}U_0\mathcal{A}^\dagger = -\mathcal{A}(I - 2|0\rangle\langle 0|)\mathcal{A}^\dagger = 2\mathcal{A}|0\rangle\langle 0|\mathcal{A}^\dagger - \mathcal{A}I\mathcal{A}^\dagger = 2|s\rangle\langle s| - I = U_s, \tag{44}$$

where we used the fact that the operator $\mathcal{A}$ is unitary, and therefore, $\mathcal{A}^\dagger = \mathcal{A}^{-1}$. This shows that U_s can be implemented via the three parts mentioned above.

With minor modifications, it is possible to apply Grover's algorithm to the situation with more than one desired states (Boyer et al., 1998).

Brassard et al. (2002) provide an intuitive explanation for how Grover's algorithm delivers the quadratic improvement in computational efficiency. Consider a classical randomized algorithm that succeeds with probability p, where $p \ll 1$. After j repetitions of the algorithm, if j is small enough, the cumulative probability of success is approximately jp. In this classical case, the probability of success increases by a constant increment p at each

iteration. Grover's algorithm increases the *amplitude* of the desired state $|x_0\rangle$ by an approximately constant increment at each iteration. The probability that the measurement of the resulting quantum state yields the outcome x_0 is proportional to the squared amplitude of $|x_0\rangle$, resulting in a quadratically faster increase in probability at every iteration of Grover's algorithm.

4.2 Quantum Amplitude Amplification (QAA)

Many quantum algorithms deliver the result of computation in a quantum superposition state $|\psi\rangle$ of "good" and "bad" results. The extraction of the "good" result often requires repeated measurements of auxiliary qubits. For example, the quantum algorithm in Section 6.3 (application of a Hermitian operator to a quantum state) produces the desired result only when the measurement of the auxiliary qubit yields 1. If the probability of measuring 1 is p, then it takes roughly $1/p$ measurements on average to extract the desired result. In some cases the probability p may be quite small, making the measurement step a substantial computational overhead.

Brassard et al. (2002) generalize Grover's algorithm to mitigate this problem and reduce the computational burden of repeated measurement from $O(1/p)$ to $O(1/\sqrt{p})$. Let $\mathcal{A}$ be the algorithm that creates the state $|\psi\rangle$: $|\psi\rangle = \mathcal{A}|0\rangle = \sum_x a_x|x\rangle$, a superposition of "good" and "bad" results. Consider a validation function χ that returns 1 if x is a "good" result, and 0 if a "bad" result. The aim of the QAA algorithm is to amplify the amplitudes of the subspace of "good" results in order to increase the probability that a measurement yields those results.

The algorithm leverages the fact that the quantum state $|\psi\rangle$ can be decomposed as

$$|\psi\rangle = a_1|\psi_1\rangle + \sqrt{1-a_1^2}|\psi_2\rangle, \tag{45}$$

where $|\psi_1\rangle$ is a projection of $|\psi\rangle$ onto the "good" subspace spanned by the states representing the "good" results and $|\psi_2\rangle$ is the projection onto its complement – the "bad" subspace. The probability p of a measurement of $|\psi\rangle$ yielding a "good" state is $p = |a_1|^2$.

An iteration of QAA, represented as an operator Q, increases the amplitude of the "good" state $|\psi_1\rangle$. By analogy with Grover's search algorithm, the operator Q is

$$Q = -\mathcal{A}U_0\mathcal{A}^\dagger U_\chi, \tag{46}$$

where

$$U_0 = I - 2|0\rangle\langle 0|, \;\; U_\chi = I - 2|\psi_1\rangle\langle\psi_1|, \tag{47}$$

and the algorithm $\mathcal{A}$ is assumed to be reversible, i.e. containing no measurements.

Repeated applications of Q gradually increase the amplitude of the "good" $|\psi_1\rangle$ component. After t measurements, where $t \approx \frac{\pi}{4|a_1|} = \frac{\pi}{4\sqrt{p}}$, the probability that a measurement of $Q^t|\psi\rangle$ yields a "good" component is of order unity.

Both QAA and Grover's algorithms are periodic. Once the minimum error is reached at $t \approx \frac{\pi}{4|a_1|}$, repeated applications of the search operator Q start pushing the state of the system away from the target state, and the error starts to increase until $t \approx \frac{3\pi}{4|a_1|}$. After maximum

error is reached, repeated applications of Q start pushing the system closer to the target state again. Literature refers to this problem as the "soufflé problem," referring to the way the dessert rises during baking, but starts to deflate if baked too long (the analogy would have been more apt if the soufflé inflated and deflated periodically with baking time). Solutions to the problem include fixed-point quantum search (Yoder et al., 2014) and variable time amplitude amplification (Ambainis, 2012).

4.3 Quantum Amplitude Estimation (QAE)

As discussed in the previous section, the QAA algorithm is periodic with respect to repeated application of the amplification step Q. When the amplification step Q is applied t times, Q^t, the amplification error cycles between its minimum and maximum with a period of $t=\frac{\pi}{|a_1|}$, where a_1 is the amplitude of the "good" subspace in (45). Brassard et al. (2002) leverage the fact that the period of Q^t is a function of the absolute value of the amplitude $|a_1|$ to estimate this amplitude using Quantum Phase Estimation, an influential algorithm we describe in Section 6.2.

The algorithm starts with a register of n qubits containing $|\psi\rangle=\mathcal{A}|0\rangle$ and an auxiliary register of n qubits, initialized to a uniform superposition $\frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}|j\rangle$, where $N=2^n$. Let $\Lambda_N(U)$ be a controlled unitary operator that applies multiple copies of a unitary U conditional on the state of the auxiliary qubit register:

$$\Lambda_N(U)|j\rangle|\psi\rangle=|j\rangle(U^j|\psi\rangle), \tag{48}$$

where the state $|j\rangle$ acts as a reference. The operator $\Lambda_N(Q)$ applied to the state $\frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}|j\rangle|\psi\rangle$ creates a superposition of states with a range of repeated quantum amplitude amplification steps Q^j. Quantum Phase Estimation enables the extraction of the phase of Q^j. Quantum Phase Estimation records $\hat{y}$, an n bit approximation of y such that $|a_1|^2=\sin^2(\pi\frac{y}{N})$, in the computational basis in the auxiliary register. Measurement of the auxiliary register yields the outcome $|\hat{y}\rangle$ with probability of at least $8/\pi^2$. With $\hat{a}_1=\sin^2(\pi\frac{\hat{y}}{N})$, the estimation error bound after t iterations of the algorithm is:

$$|\hat{a}_1-a_1|\leq 2\pi\frac{a_1(1-a_1)}{t}+\frac{\pi^2}{t^2}. \tag{49}$$

4.4 Estimating the Mean of a Bounded Function

The QAE algorithm provides an efficient way to estimate the mean of a bounded function (see Section 3.2 for a method to estimate the mean amplitude of a state that leverages the Swap Test).

Let $F:\{0,...,N-1\}\rightarrow X$, where $X=[0,1]$, be a black-box function. Brassard et al. (2011) propose a method to approximate the mean value of F, $\mu=\frac{1}{N}\sum_x F(x)$; see also Heinrich (2002). The idea is to create a state of the form:

$$|\psi\rangle=\alpha|\psi_0\rangle+\beta|\psi_1\rangle, \tag{50}$$

such that $|\psi_0\rangle$ and $|\psi_1\rangle$ are orthogonal and $|\beta|^2 = \frac{1}{N}\sum_{i=0}^{N-1} F(i) = \mu$. If the creation of state $|\psi\rangle$ requires $O(1)$ oracle calls to F, then QAA can yield an estimate of $|\beta|^2$ with precision ϵ in $O(1/\epsilon)$ oracle calls regardless of the size N.

Creation of state $|\psi\rangle$ in (50) proceeds in a few steps and requires three registers. The first, n-qubit register $|\rangle_n$ encodes index values i in the computational basis; the second, m-qubit register $|\rangle_m$ is an auxiliary register that temporarily holds an m-bit approximation of $F(i)$; the third, single-qubit register $|\rangle_1$ is an auxiliary register that helps create orthogonal states $|\psi_0\rangle$ and $|\psi_1\rangle$.

Let $\mathcal{A}$ be an algorithm that encodes an m-bit approximation of $F(i)$ in an m-qubit auxiliary register initialized to $|0\rangle_m$:

$$\mathcal{A}|i\rangle_n|0\rangle_m = |i\rangle_n|F(i)\rangle_m. \tag{51}$$

Using a control rotation operator C-R (see Section 2.3), we transfer the values $F(i)$ into the amplitudes of the quantum state

$$C\text{-}R|F(i)\rangle_m|0\rangle_1 = |F(i)\rangle_m\big(\sqrt{1-F(i)}|0\rangle_1 + \sqrt{F(i)}|1\rangle_1\big). \tag{52}$$

The operator $A = (\mathcal{A}^{-1}\otimes I_1)(I_m \otimes I_n \otimes C\text{-}R)(\mathcal{A}\otimes I_1)$, where I_m, I_n, and I_1 are identity operators acting on the m- and n-qubit registers and the auxiliary qubit, respectively, applies an extension of $\mathcal{A}$ to all the three registers and then uncomputes the stored values of $F(i)$. When applied to a state where the n-qubit register holds a uniform superposition of n-bit strings $|i\rangle$, the operator A produces the desired state $|\psi\rangle$:

$$|\psi\rangle = A\big(\frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}|i\rangle_n\big)|0\rangle_m|0\rangle_1 = \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}|i\rangle_n|0\rangle_m\big(\sqrt{1-F(i)}|0\rangle_1 + \sqrt{F(i)}|1\rangle_1\big). \tag{53}$$

Discarding the m-qubit auxiliary register and rearranging, we have:

$$\begin{aligned}
|\psi\rangle &= \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}|i\rangle_n\big(\sqrt{1-F(i)}|0\rangle_1 + \sqrt{F(i)}|1\rangle_1\big) \\
&= \frac{\sqrt{N-\sum_j F(j)}}{\sqrt{N}}\Big[\frac{1}{\sqrt{N-\sum_j F(j)}}\sum_{i=0}^{N-1}\sqrt{1-F(i)}|i\rangle_n|0\rangle_1\Big] \\
&\quad + \frac{\sqrt{\sum_j F(j)}}{\sqrt{N}}\Big[\frac{1}{\sqrt{\sum_j F(j)}}\sum_{i=0}^{N-1}\sqrt{F(i)}|i\rangle_n|1\rangle_1\Big] \\
&= \alpha|\psi_0\rangle + \beta|\psi_1\rangle,
\end{aligned} \tag{54}$$

where the expressions in the square brackets are the properly normalized and orthogonal states $|\psi_0\rangle$ and $|\psi_1\rangle$. The coefficients in front of the brackets are α and β respectively, so that $|\beta|^2 = \frac{1}{N}\sum_{i=0}^{N-1} F(i) = \mu$ can be efficiently approximated by QAE, described in Section 4.3. The creation of the quantum state $|\psi\rangle$ requires only two calls to the quantum oracle $\mathcal{A}$ (one to initialize the m-qubit auxiliary register containing the values $|F(i)\rangle$ and the other to uncompute it).

Montanaro (2015) extends the Brassard et al. method to functions F with non-negative output. The method decomposes the function F into k components $F_{(x_{i-1},x_i]}$, for $i=1,..,k$, whose output falls within disjoint intervals $(x_{i-1},x_i] \subset X$ of the codomain of F. The mean is then estimated for each function $F_{(x_{i-1},x_i]}$, and the overall mean of the function of F is constructed from the means of $F_{(x_{i-1},x_i]}$. Montanaro (2015) extends the method further to functions with output that does not have to be non-negative but has a bounded variance.

4.5 Minimum (or Maximum) of a Function over a Discrete Domain

The fastest classical algorithm for finding the minimum (or maximum) value of in an unsorted table of N items requires $O(N)$ oracle calls; with the help of a quantum computer this requirement decreases to $O(\sqrt{N})$ through repeated application of the QAA (Durr and Hoyer, 1996).

Let $F:\{0,...,N-1\} \rightarrow X$, $X \subset \mathbb{R}$ be a black-box function. The algorithm outputs y^*, the index corresponding to the minimum value of F. The algorithm starts by selecting uniformly at random an index value y such that $0 \leq y \leq N-1$. Assuming without loss of generality that $N=2^n$ for an integer n, two registers of n qubits each can support a quantum state of the form $\frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}|j\rangle|y\rangle$. The first register encodes a uniform superposition of all indices j, the second register encodes the value y, both in the computational basis (i.e. in the form of binary strings).

Mark every item with $F(j)<F(y)$ using an efficient oracle able to perform the operation in $O(\log N)$ calls to the function F. The marked items represent the "good" state in QAA (Section 4.2). Run the algorithm to amplify this "good" state. Next, select y' until $F(y')<F(y)$ and re-run the algorithm with y' instead of y. Durr and Hoyer (1996) show that, after $22.5\sqrt{N}+1.4\log_2^2 N$ calls to F, the desired state is reached with probability at least $1/2$.

4.6 Median and kth Smallest Value

Nayak and Wu (1999) generalized the algorithm of Durr and Hoyer (1996) to find not just the smallest value, but the kth smallest value of a discrete function F, including the $N/2$th smallest value which is the median. Denote by $\text{rank}(F(i))$ the rank of $F(i)$ in $\{F(0),...,F(N-1)\}$ ordered in non-decreasing order. Let $\Delta \geq 1/2$ be the accuracy parameter. Given k, $1 \leq k \leq N$, the problem is to find the value $F(i)$ such that $\text{rank}(F(i))$ is the smallest value in $(k-\Delta,k+\Delta)$. This is referred to as the Δ-approximation of the kth smallest element of F.

The algorithm relies on two subroutines. The first subroutine implements a function $K(l)$ that returns 'yes' if $F(l)$ is the Δ-approximation of the kth smallest element; '$<$' if the rank of $F(l)$ is at most $k-\Delta$; and '$>$' if the rank of $F(l)$ is at least $k+\Delta$. This subroutine can be implemented based on the counting algorithm of Brassard et al. (1998) (see Section 4.7). The second subroutine implements the sampler $S(i,j)$ that chooses an index l at random such that $F(i)<F(l)<F(j)$. The sampler can be implemented based on the generalized search algorithm of Boyer et al. (1998).

The kth smallest value algorithm works as follows: For convenience, define $F(-1)=-\infty$ and $F(N)=\infty$,

1. $i \leftarrow -1$; $j \leftarrow N$
2. $l \leftarrow S(i,j)$
3. If $K(l)$ returns 'yes', output $F(l)$ (and/or the index l) and stop. Else, if $K(l)$ returns '$<$', $i \leftarrow l$, go to step 2. Else, if K(l) returns '$>$', $j \leftarrow l$, go to step 2.

Nayak and Wu (1999) prove that the expected number of iterations before termination is $O(\log N)$.[15]

4.7 Counting

In some applications, the problem of interest is not to *find* the solution but to *count* how many solutions exist. Consider the function $F:\{0,...,N-1\} \rightarrow X = \{0,1\}$. We are interested in counting the number of indices x such that $F(x)=1$. Then, the algorithm for estimating the mean of F is effectively an algorithm for counting how many indices x so that $F(x)=1$.

4.8 Quantum Monte Carlo

The techniques used in the previous sections provide a way to estimate the mean of a function with respect to a probability distribution. Classically, the most popular method for estimating an expectation is Monte Carlo simulation. In this method, samples are drawn from the probability distribution and the function is evaluated for each sample; the average of these outputs, $\tilde{\mu}$, is the Monte Carlo estimate of the true mean μ. Chebyshev's inequality guarantees that, for k independent draws from the probability distribution, the probability that the estimate is far from the real mean μ is bounded

$$\Pr[|\hat{\mu} - \mu] \geq \epsilon] \leq \frac{\sigma^2}{k\epsilon^2}, \tag{55}$$

where σ^2 is the variance of the function with respect to the probability distribution. In other words, in order to estimate μ up to an additive error ϵ, $k = O(\sigma^2/\epsilon^2)$ samples – and function evaluations – are required.

Quantum Amplitude Estimation holds the promise of reducing the required number of function evaluations to $O(\sigma/\epsilon)$ to achieve the same error bound. That is, Quantum Monte Carlo provides a quadratic speedup over classical Monte Carlo. The quantum method requires two efficient quantum oracles. The first oracle $\mathcal{P}$ helps to prepare a state that encodes the probability distribution in a quantum state – called a *quantum sample state*. The second oracle $\mathcal{F}$ applies the function whose mean is to be estimated with respect to the probability distribution.

To define the quantum sample state, let X be a finite N-dimensional set, and $p(x)$ a probability distribution over X, such that $\sum_{x \in X} p(x) = 1$. Let the basis set of an N-dimensional Hilbert space, $\{|x\rangle\}$, represent the elements x of X. The quantum state $|p\rangle$ of the form

$$|p\rangle = \sum_{x \in X} \sqrt{p(x)} |x\rangle \tag{56}$$

[15] More precisely, let $n = \sqrt{N/\Delta} + \sqrt{k(N-k)}/\Delta$. The expected number of iterations is $O(\log n)$.

is the quantum sample with respect to the distribution p. The state $|p\rangle$ has the property that, by the Born rule, the probability that a measurement of all qubits supporting the state yields x is $p(x)$.

Using these two oracles $\mathcal{P}$ and $\mathcal{F}$ and an auxiliary qubit we have:

$$\mathcal{P}:|0\rangle \mapsto |p\rangle = \sum_{x\in X} \sqrt{p(x)}|x\rangle, \tag{57}$$

$$\mathcal{F}:|p\rangle|0\rangle = \sum_{x\in X} \sqrt{p(x)}|x\rangle(\sqrt{1-f(x)}|0\rangle + \sqrt{f(x)}|1\rangle). \tag{58}$$

The quantum state in Eq. (58) has a structure parallel to that of the quantum state in Eq. (54), and we can apply to it a similar technique to isolate the expectation value of f with respect to the distribution p, $\mathbb{E}_p[f] = \sum_{x\in X} p(x)f(x)$. If $\mathcal{F}$ is *efficiently computable* (i.e., with a sub-polynomial number of operations), then the algorithm yields $\hat{f}$, an estimate of $\mathbb{E}_p[f]$ within the error bound ϵ with $O(\sigma/\epsilon)$ function evaluations.

Further Discussion on Quantum Sample Preparation

The quantum Monte Carlo method described above delivers a quadratic speedup over classical methods provided the quantum oracles $\mathcal{P}$ and $\mathcal{F}$ can be implemented efficiently. If the function f is easily computable, then $\mathcal{F}$ has an efficient implementation (Low and Chuang, 2017; Rebentrost and Lloyd, 2018). But implementing the oracle $\mathcal{P}$ to create a quantum sample can be challenging. For an arbitrary probability distribution, preparing a quantum sample state is computationally equivalent to solving the graph isomorphism problem and is exponentially hard (Plesch and Brukner, 2011; Chakrabarti et al., 2019). For efficiently integrable distributions, such as the normal distribution, the method proposed by Grover and Rudolph (2002) has been popular. However, Herbert (2021) recently demonstrated that the Grover and Rudolph (2002) method is limited to situations simple enough to be solved without the use of the Monte Carlo method – classical or quantum. Because quantum Monte Carlo delivers quadratic rather than exponential speedup over classical methods, the modest computation overhead of the Grover and Rudolph (2002) method negates the quantum gains.

Efficient preparation of quantum samples for distributions of practical interest is, at the time of writing this review, an open problem in quantum algorithm design. A machine-learning approach based on empirical data has been proposed by Zoufal et al. (2019). Vazquez and Woerner (2021) propose to view the probability distribution function p as a function, implemented similarly to function f, using a controlled rotation of an auxiliary qubit. This method works for simpler distributions, with an efficiently-computable $p(x)$. For more complex, high-dimensional distributions Kaneko et al. (2021) propose to create quantum samples using pseudorandom numbers. An et al. (2021) quantize the classical method of multilevel Monte Carlo to find approximate solutions to stochastic differential equations (SDEs), particularly for applications in finance.

It may also be possible to achieve relatively efficient quantum sampling using a sequence of slowly varying quantum walks (see Section 5) – the quantum equivalents of Markov chains (Wocjan and Abeyesinghe, 2008; Wocjan et al., 2009). The method, called Quantum Markov

Chain Monte Carlo and discussed in more detail in Section 5.3, is analogous to classical Markov chain Monte Carlo (MCMC), and can be used to generate a Markov chain with a given equilibrium distribution π.

5 Quantum Markov Chains

This section introduces Quantum Markov chains, often called *quantum walks* in the quantum computing literature, the quantum equivalents of classical Markov chains which are widely used in probability and statistics. Quantum walks have been shown to provide polynomial speed-ups for a wide variety of problems from estimating the volume of convex bodies (Chakrabarti et al., 2019) to option pricing (An et al., 2021), search for marked items (Magniez et al., 2011) and active learning in artificial intelligence (Paparo et al., 2014).[16] However, because of the quantum interference, quantum Markov chains behave substantially differently from their classical counterparts. For example, quantum Markov chains do not admit any equilibrium distribution, but the average of the states of the chain does (see below for a formal definition). This makes applications of quantum Markov chains in statistics different from those of classical chains, and can lead to new interesting and important applications.

Questions of interest to statisticians are: how to quantize a Markov chain, i.e. how to implement a Markov chain in a quantum computer? What are the properties of the resulting quantum Markov chain? How can this quantum Markov chain be used in statistical applications such as MCMC sampling? In this section, we review the two most popular approaches for quantizing a Markov chain: *coin walks* and *Szegedy walks* (Szegedy, 2004; Watrous, 2001). Coin walks quantize Markov chains on unweighted graphs. Szegedy walks work on weighted directional graphs. We focus on Markov chains with a discrete state space, but it is also theoretically possible to quantize a continuous-space Markov chain.[17] For a detailed and thorough review of quantum walks, see Venegas-Andraca (2012).

5.1 Coin Walks

Consider a Markov chain with a discrete state space. It can be represented on a graph $G=(V,E)$: the vertices V represent the states; after each time step, the chain stays at the current vertex or jumps to one of its adjacent vertices according to a transition probability. This creates a random walk on the graph.

Let $\mathcal{H}_V$ and $\mathcal{H}_E$ be the quantum systems whose basis states encode the vertices in V and the edges in E, respectively. Define a shift operator S on $\mathcal{H}_V \otimes \mathcal{H}_E$ that determines the next vertex u given the current vertex v and the edge e, i.e., $S|e\rangle|v\rangle = |e\rangle|u\rangle$. Define a *coin*

[16] For special classes of problems, such as a subclass of black-box graph traversal problems, quantum walks deliver exponential speeds up over any classical algorithm (Childs et al., 2003). Generally, quantum walks are universal for quantum computation (Childs, 2009) (i.e. any sequence of gates can be expressed as a quantum walk).

[17] In practice, quantum computers have finite precision and can only encode finite (if high-dimensional) sets, making discrete-space quantum samples most relevant for quantum algorithm applications (Chakrabarti et al., 2019).

operator C to be a unitary transformation on $\mathcal{H}_E$. Then, $U=S(C\otimes I)$ implements one step of the random walk on graph G. If the initial state is $|\psi_0\rangle$, the state after t steps is

$$|\psi_t\rangle = U^t|\psi_0\rangle.$$

The dynamic of this quantum random walk is governed by the coin operator C. Because of the quantum interference and superposition effect, the distribution of $|\psi_t\rangle$ behaves very differently from the classical Markov chain. Denote by $P_t(v|\psi_0)$ the probability of finding $|\psi_t\rangle$ at a node $v\in V$. The probability distribution $P_t(\cdot|\psi_0)$ does not converge (Venegas-Andraca, 2012), but its average does. More precisely, let

$$\bar{P}_t(v|\psi_0) = \frac{1}{t}\sum_{s=1}^{t} P_s(v|\psi_0),$$

then $\bar{P}_t(\cdot|\psi_0)$ converges and this stationary distribution can be determined. With a suitable definition of mixing time, it is shown that the mixing time of a quantum walk is quadratically faster than a classical random walk - a property that attracted attention of researchers seeking to speed up algorithms based on Markov chains.

5.2 Szegedy Walks

Szegedy (2004), based on the earlier work of Watrous (2001), proposed another approach to quantize Markov chains. Consider a Markov chain operating on a bipartite graph. Let X and Y be two finite sets, and matrices P and Q describe the probabilities of jumps from elements of X to elements of Y and Y to X, respectively. The elements of P and Q, $p_{x,y}$ and $q_{y,x}$, are transition probabilities and, as such, are non-negative and normalized so that $\sum_{y\in Y} p_{x,y}=1$ and $\sum_{x\in X} q_{y,x}=1$. A Markov chain that maps X to X with a transition matrix P is equivalent to a bipartite walk where $q_{y,x}=p_{y,x}$ for every $x,y\in X$ or, equivalently, $Q=P$.

To quantize the bipartite random walk, define a two-register quantum system spanned by $|x\rangle|y\rangle$ with $x\in X$, $y\in Y$. We start with two unitary operators,

$$U_P : |x\rangle|0\rangle \mapsto \sum_{y\in Y} \sqrt{p_{x,y}}|x\rangle|y\rangle, \;\; V_Q : |0\rangle|y\rangle \mapsto \sum_{x\in X} \sqrt{q_{y,x}}|x\rangle|y\rangle, \tag{59}$$

which are quantum equivalents of the transition matrices P and Q. The quantization is based on the observation that the Grover "diffusion" operator U_f (from Section 4.1) is similar to a step of a random walk over a graph – a transition from each state to all the other N states. In matrix form, the operator U_f (up to an overall negative sign) is such that its off-diagonal elements equal $\frac{2}{N}$ and the diagonal elements are $-1+\frac{2}{N}$. This unitary operator effectively distributes quantum probability mass from each node to all the other nodes – a property that led to naming this operator a "diffusion" operator.

Using the operators U_P and V_Q we define operators similar to Grover's diffusion operators:

$$\mathcal{R}_1 = 2U_P U_P^\dagger - I, \;\; \mathcal{R}_2 = 2V_Q V_Q^\dagger - I, \tag{60}$$

where the identity operator I acts on both registers. The quantum walk operator W is defined as the product of the two diffusion operators

$$W = \mathcal{R}_2 \mathcal{R}_1. \tag{61}$$

For a Markov chain from X to X, the expression can be simplified by replacing R_2 with SR_1S, where S is the swap operator, which swaps the two registers:

$$S = \sum_{x,y} |y,x\rangle\langle x,y|. \tag{62}$$

This operator is self-inverse $S^2 = I$ and $SR_1S^{-1} = SR_1S$. For a Markov chain we can write

$$W = S(2U_P U_P^\dagger - I)S(2U_P U_P^\dagger - I), \tag{63}$$

so that some researchers (see e.g. Chakrabarti et al., 2019) define a step of a quantized Markov chain as

$$W_{MC} = S(2U_P U_P^\dagger - I). \tag{64}$$

A Szegedy quantum walk, similar to a coin walk, is a unitary process and, as such, does not converge to a stationary distribution. Instead the quantum walk "cycles through" the stationary distribution π of P, similarly to the way Grover's search (Section 4.1) or QAA (Section 4.2) pass through the desired state with a certain period (QAE, described in Section 4.3, exploits this periodicity). The quantum state analogous to the stationary distribution π, $|\pi\rangle = \sum_x \sqrt{\pi(x)}|x\rangle$, is the highest-eigenvalue eigenstate of the Szegedy quantum walk operator W (Orsucci et al., 2018); the eigenvalue of eigenstate $|\pi\rangle$ equals 1, i.e. $W|\pi\rangle = |\pi\rangle$. This important property of the Szegedy quantum walk can be exploited to develop quantum Markov chain Monte Carlo to sample from π; see Section 5.3.

5.3 Quantum Markov Chain Monte Carlo

Can a quantum walk be used to derive a quantum Markov chain Monte Carlo algorithm for sampling from a target probability distribution? The answer is yes. Consider a distribution π over a discrete space X; the problem is to prepare a quantum sample $|\pi\rangle = \sum_{x\in X} \sqrt{\pi(x)}|x\rangle$.

Let P be the transaction matrix of a classical ergodic Markov chain with the stationary distribution π; P can be derived based on, e.g., the Metropolis algorithm. Let $W(P)$ be the Szegedy quantum walk with respect to P. Then $|\pi\rangle$ is the unique eigenstate of $W(P)$ with the eigenvalue 1. All other eigenstates have an eigenphase which is at least quadratically larger than the spectral gap δ – the difference between the top and the second highest eigenvalues. This property allows one to use phase estimation (or phase detection) to distinguish $|\pi\rangle$ from the other eigenstates of $W(P)$.

However, the mixing time of the Szegedy quantum walk is $O(1/\sqrt{\delta\pi_{min}})$ steps in general with $\pi_{min} = \min_{x\in X}\pi(x)$ (Aharonov and Ta-Shma, 2007; Montanaro, 2015). This can be problematic when the size N of X is large. One way to reduce the dependence of mixing time on π_{min} is to employ a slowly varying series of quantum walks to reach the desired quantum sample state (Wocjan and Abeyesinghe, 2008; Wocjan et al., 2009). This is similar

to the idea of annealed sampling. Let P_0,...,P_r be classical reversible Markov chains with stationary distributions π_0,...,$\pi_r = \pi$ such that each chain has a relaxation time at most τ (Montanaro, 2015). Then given an easy-to-prepare state $|\pi_0\rangle$, e.g. the uniform state $\frac{1}{\sqrt{N}}\sum_{x\in X}|x\rangle$, and the condition that $\langle\pi_i|\pi_{i+1}\rangle \geq p$ for some $p > 0$ and $\forall i = 1,..,r-1$, for any $\epsilon > 0$, there is a quantum algorithm which results in a quantum sample $|\tilde{\pi}_r\rangle$ such that $\||\tilde{\pi}_r\rangle - |\pi_r\rangle\| \leq \epsilon$. The algorithm uses $O(r\sqrt{\tau}\log^2(r/\epsilon)(1/p)\log(1/p))$ quantum walk steps. Chakrabarti et al. (2019) generalized this approach and used it to design a quantum MCMC algorithm to speed up evaluation of volume of convex bodies. Further speedup has been proposed by Magniez et al. (2011) and Orsucci et al. (2018). Additionally, an even more efficient methods exist to reflect about the states $|\pi_i\rangle$, with a runtime that does not depend on r (Yoder et al., 2014).

Applications of quantum Monte Carlo and quantum Markov chain Monte Carlo are rich and varied. They include, for example, speeding up classical annealing approaches to combinatorial optimization problems (Somma et al., 2008), search (Magniez et al., 2011), speeding up learning agents (Paparo et al., 2014), derivative pricing (Rebentrost and Lloyd, 2018), and risk analysis (Woerner and Egger, 2019).

6 Quantum Linear Systems, Matrix Inversion, and PCA

Consider a system of linear equations $Ax = b$, where A is an $N \times M$ matrix, b is an $M \times 1$ input vector, and x is an $N \times 1$ solution vector, provided it exists. For a square $N \times N$ well-conditioned matrix A, the solution to the classical system of linear equations is $x = A^{-1}b$. The system of linear equations powers many applications in statistics and machine learning, especially in high and ultra-high dimensional settings such as deep learning. The quantum analog of the system of linear equations takes the form $A|x\rangle = |b\rangle$, where the quantum states $|x\rangle$ and $|b\rangle$ represent vectors x and b in amplitude encoding (Section 3.1).The solution is the quantum state $|x\rangle = \frac{1}{C}A^{-1}|b\rangle$, where C is a constant to ensure normalization of the state $|x\rangle$. Harrow et al. (2009) discovered a quantum algorithm, known as the HHL algorithm, to solve the quantum system of linear equations in time that scales with $\log N$, provided that the matrix A is sparse. This offers an exponential speedup over the best classical algorithms, which require a runtime of at least $O(N)$. The exponential advantage of the HHL algorithm stems from its use of the Quantum Fourier Transform (QFT) – one of the foundational quantum routines and the engine at the core of many quantum algorithms including Shor's famous factoring algorithm (Shor, 1994).

The QFT directly exploits quantum parallelism – the ability to apply a function to all elements of a vector simultaneously if this vector is encoded in a quantum state. The QFT powers the HHL algorithm through another influential quantum subroutine – Quantum Phase Estimation (QPE), an algorithm that enables recording of a quantum phase θ, for example in an eigenvalue $e^{i\theta}$ of a unitary matrix, into a computational basis state within an error ϵ with a high probability (Section 6.2).

This section presents the QFT (Section 6.1) and its many uses in other quantum algorithms, such as the application of a Hermitian (rather than an unitary) operator to a quantum state (Section 6.3), finding the solution of a system of linear equations (Section 6.4), fast gradient computation (Section 6.5), and quantum principal component analysis (Section 6.6).

Even though more efficient ways to perform some of these computations have been discovered recently (see, e.g. Section 9), the QFT remains an influential and pedagogical quantum subroutine.

6.1 Quantum Fourier Transform (QFT)

The QFT transforms a state $|x\rangle = \sum_{m=0}^{N-1} x_m |m\rangle$, where $|m\rangle$ are binary-encoded basis vectors in the computational basis, so that

$$\text{QFT: } |x\rangle \mapsto |y\rangle = \sum_{k=0}^{N-1} y_k |k\rangle, \tag{65}$$

$$y_k = \frac{1}{\sqrt{N}} \sum_{m=0}^{N-1} x_m e^{i2\pi k \frac{m}{N}}. \tag{66}$$

QFT is the quantum equivalent of the classical discrete Fourier transform where a vector $x = (x_0,..,x_{N-1})$ is transformed into a vector $y = (y_0,..,y_{N-1})$. That is, QFT transforms a superposition state $|x\rangle$ into a new superposition state $|y\rangle$ whose amplitudes y_k are the classical discrete Fourier transforms of the amplitudes x_m.

QFT exploits quantum computers' ability to encode N-dimensional states using $d = \lceil \log N \rceil$ qubits. The structure of the Fourier transform allows the operation to be performed as a series of $\mathcal{O}(\log^2 N)$ Hadamard gates, controlled rotations, and swap gates – an exponential improvement in efficiency compared with $\mathcal{O}(N \log N)$ operations required by classical fast Fourier transform. Because of state preparation and readoff, it is difficult to benefit from the quantum speedup of QFT for estimating the Fourier coefficients. However, QFT serves as a powerful module in other algorithms, such as Shor's factoring algorithm, the HHL linear systems algorithm, and many others.

Consider $|m\rangle$, a basis state of the Hilbert space containing $|x\rangle$. Assume without loss of generality that the dimension of the Hilbert space $N = 2^d$. As shown below, it turns out that the QFT of state $|m\rangle$ is a tensor product of single-qubit states, possible to create in a quantum computer using a series of one- and two-qubit gates.

The QFT of the state $|m\rangle$ is

$$QFT|m\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{i2\pi m \frac{k}{2^d}} |k\rangle, \tag{67}$$

where $|k\rangle$, $k = 0,..,N-1$ represent basis states of an $N = 2^d$-dimensional Hilbert space. These states can be chosen to be binary numbers from 0 to $N-1$ expressed in the computational basis such that, if k is expressed as a binary string $k_1 k_2 ... k_d$, where $k_j = \{0,1\}$, each state $|k\rangle$ is a tensor product state of d qubits in state $|0\rangle$ or $|1\rangle$:

$$|k\rangle = |k_1\rangle \otimes |k_2\rangle \otimes ... \otimes |k_d\rangle, \tag{68}$$

and $k=\sum_{j=1}^{d} k_j 2^{(d-j)}$. Using this notation, QFT$|m\rangle$ becomes

$$QFT|m\rangle = \frac{1}{2^{d/2}} \sum_{k_1,k_2,..,k_d} e^{i2\pi m \frac{\sum_{j=1}^{d} k_j 2^{(d-j)}}{2^d}} |k_1\rangle|k_2\rangle...|k_d\rangle = \frac{1}{2^{d/2}} \sum_{k_1,k_2,..,k_d} \bigotimes_{j=1}^{d} e^{i2\pi m \frac{k_j}{2^j}} |k_j\rangle$$
$$= \frac{1}{\sqrt{2}^d}(|0\rangle + e^{i2\pi \frac{m}{2}}|1\rangle) \otimes (|0\rangle + e^{i2\pi \frac{m}{2^2}}|1\rangle) \otimes ... \otimes (|0\rangle + e^{i2\pi \frac{m}{2^d}}|1\rangle), \quad (69)$$

a separable state of d qubits. The exponent $e^{i2\pi \frac{m}{2^j}}$ effectively extracts the binary "decimal" of m represented as a binary string $m = m_1 m_2 .. m_d$ so that $e^{i2\pi \frac{m}{2^j}} = e^{i2\pi m_1 .. m_{d-j} . m_{d-j+1} .. m_d} = e^{i2\pi 0.m_j .. m_d}$, since $e^{i2\pi k} = 1$ for any integer k. QFT of $|m\rangle$ then simplifies to

$$QFT|m\rangle = \frac{1}{\sqrt{2}^d}(|0\rangle + e^{i2\pi 0.m_1 m_2 .. m_{d-1} m_d}|1\rangle) \otimes (|0\rangle + e^{i2\pi 0.m_2 .. m_{d-1} m_d}|1\rangle) \otimes ...$$
$$...\otimes (|0\rangle + e^{i2\pi 0.m_d}|1\rangle), \quad (70)$$

a state that can be created via a series of relatively simple single-qubit and two-qubit gates. It is easy to see that the QFT operator is linear and applies similarly to a linear superposition of states $|m\rangle$, i.e. any state $|x\rangle$.

6.2 Quantum Phase Estimation (QPE)

Let U be a unitary operator with eigenstates $|u\rangle$. Because the operator U is unitary, its eigenvalues take the form $e^{i2\pi\theta}$, where $\theta \in [0,1)$, and $U|u\rangle = e^{i2\pi\theta}|u\rangle$, as we discuss in Section 2.2.

Quantum Phase Estimation (QPE) is an algorithm to estimate, within a finite precision, the phase θ of the operator U and record its binary approximation in a quantum state in the computational basis. QPE is a building block in many algorithms, particularly those requiring the application of a Hermitian (rather than unitary) operator to a quantum state (Section 6.3).

The linchpin of QPE is the control-unitary gate C-U, which applies the unitary U conditional on the state of an auxiliary qubit (Section 2.2). Consider the state $|0\rangle \otimes |u\rangle$, where $|0\rangle$ represents the auxiliary qubit. Applying a Hadamard gate to the auxiliary qubit yields the state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |u\rangle$. The controlled-unitary operator C-U acting on the state applies the operator U to state $|u\rangle$ if the auxiliary qubit is in state $|1\rangle$ and does nothing if the auxiliary qubit is in the state $|0\rangle$:

$$C\text{-}U\big[\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |u\rangle\big] = \frac{1}{\sqrt{2}}(|0\rangle + e^{i2\pi\theta}|1\rangle) \otimes |u\rangle. \quad (71)$$

Even though the operator U acts on the state $|u\rangle$, it is the auxiliary qubit state that ends up being modified because the operator is controlled on the state of this qubit. This effect is called *phase kickback*.

To capture the n-bit approximation of θ, $\tilde{\theta} = 0.\theta_1\theta_2...\theta_n$ with $\theta_j \in \{0,1\}$, QPE requires n auxiliary qubits. Each auxiliary qubit is initialized to $|0\rangle$ and then a Hadamard gate is

applied to each qubit to yield the state

$$\frac{1}{2^{n/2}}(|0\rangle+|1\rangle)\otimes(|0\rangle+|1\rangle)\otimes...\otimes(|0\rangle+|1\rangle)\otimes|u\rangle. \tag{72}$$

Next, we apply a series of controlled-unitary gates, C_j-$U^{2^{j-1}}$, which apply the unitary operator $U^{2^{j-1}}$ to state $|u\rangle$ conditional to the state of the auxiliary qubit j. This results in the state

$$\begin{aligned}
&\frac{1}{2^{n/2}}(|0\rangle+e^{i2\pi\theta 2^0}|1\rangle)\otimes(|0\rangle+e^{i2\pi\theta 2^1}|1\rangle)\otimes...\otimes(|0\rangle+e^{i2\pi\theta 2^{n-1}}|1\rangle)\otimes|u\rangle \\
&=\frac{1}{2^{n/2}}(|0\rangle+e^{i2\pi\frac{2^n\theta}{2^n}}|1\rangle)\otimes(|0\rangle+e^{i2\pi\frac{2^n\theta}{2^{n-1}}}|1\rangle)\otimes...\otimes(|0\rangle+e^{i2\pi\frac{2^n\theta}{2}}|1\rangle)\otimes|u\rangle \\
&=\Big(\frac{1}{2^{n/2}}\sum_{k=0}^{2^n-1}e^{i2\pi(2^n\theta)\frac{k}{2^n}}|k\rangle\Big)\otimes|u\rangle,
\end{aligned} \tag{73}$$

where ks are integers represented as n-bit strings $k_1...k_n$ by the qubits in the auxiliary register as $|k\rangle=|k_1\rangle\otimes..\otimes|k_n\rangle$.

If θ is an n-bit number, so that $\theta=\tilde{\theta}$, then $\theta 2^n$ is an integer. In this case, the state in the auxiliary register of (73) is the QFT of the n-qubit state $|\theta 2^n\rangle$ (c.f. Eq. 69 with $m=\theta 2^n$). This state represents $\theta 2^n$ as a binary integer encoded in the computational basis. However, in general, θ is a number with greater than n bits, so that $\theta\neq\tilde{\theta}$. In this case, $\theta 2^n$ is not an integer. Splitting θ into its n-bit approximation $\tilde{\theta}$ and a residual δ, $\delta=\theta-\tilde{\theta}$, we can write $\theta 2^n=\tilde{\theta}2^n+\delta 2^n$, where $\tilde{\theta}2^n$ is the integer part of $\theta 2^n$.

In the last step of QPE, we recover $\tilde{\theta}2^n$ using the inverse QFT applied to the auxiliary register:

$$\begin{aligned}
QFT^\dagger\Big(\frac{1}{2^{n/2}}\sum_{k=0}^{2^n-1}e^{i2\pi(2^n\theta)\frac{k}{2^n}}|k\rangle\Big)&=\frac{1}{2^n}\sum_{y=0}^{2^n-1}\sum_{k=0}^{2^n-1}e^{-i2\pi k\frac{y}{2^n}}e^{i2\pi(2^n\theta)\frac{k}{2^n}}|y\rangle \\
&=\frac{1}{2^n}\sum_{y=0}^{2^n-1}\sum_{k=0}^{2^n-1}e^{i2\pi(2^n\tilde{\theta}-y)\frac{k}{2^n}}e^{i2\pi\delta k}|y\rangle
\end{aligned} \tag{74}$$

Probability in the superposition state in (74) is peaked around the state $|\tilde{\theta}2^n\rangle$, which encodes $\tilde{\theta}2^n$ in the computational basis.

The last step is to measure the auxiliary register in the computational basis. If $\delta=0$, i.e. if θ is an n-bit number, then the measurement yields $\theta 2^n$ with probability 1. If $0<|\delta|\leq\frac{|\theta|}{2^n}$, then the measurement yields $\tilde{\theta}2^n$ with probability $4/\pi^2$ or greater (Cleve et al., 1998).[18]

6.3 Applying a Hermitian Operator

Quantum gates are unitary, but it is often useful to apply a Hermitian operator, rather than a unitary operator, to a quantum state. This can be done in two steps: QPE (Section 6.2) and a controlled rotation (Section 2.3).

[18]By using $O(\log(1/\epsilon))$ qubits and discarding the qubits above n, it is possible to increase the probability of measuring $\tilde{\theta}2^n$ to $1-\epsilon$.

Let $\mathcal{H}$ be a Hermitian operator on an N-dimensional Hilbert space, such that $N=2^n$ for an integer n. The goal is to apply the operator $\mathcal{H}$ to an N-dimensional state $|\psi\rangle$. Because the operator $\mathcal{H}$ is Hermitian, there exists an orthonormal basis $\{|u_i\rangle\}_{i=1}^N$ such that $\mathcal{H}|u_i\rangle=\lambda_i|u_i\rangle$. The scalars λ_i are (real) eigenvalues of $\mathcal{H}$. In quantum mechanical notation, the expression $\mathcal{H}=\sum_{i=1}^N\lambda_i|u_i\rangle\langle u_i|$ reflects the fact that $\mathcal{H}$ is diagonal in the basis of its eigenvectors. The target state $\mathcal{H}|\psi\rangle$ then takes the form

$$\mathcal{H}|\psi\rangle=\sum_{i=1}^{N}\lambda_i|u_i\rangle\langle u_i|\psi\rangle=\sum_{i=1}^{N}\lambda_i\beta_i|u_i\rangle, \tag{75}$$

where the scalar coefficients $\beta_i=\langle u_i|\psi\rangle$ equal the inner products of the state $|\psi\rangle$ with the eigenstates $|u_i\rangle$.

For any Hermitian operator $\mathcal{H}$, there exists a unitary operator $U=e^{-i\mathcal{H}t}$, where t is a scalar constant. Because the operator U is unitary, it can be constructed as a series of quantum gates and applied to a quantum state prepared on n qubits. Alternatively, it is often more efficient to interpret the operator U as an evolution operator (Section 2.2) by Hamiltonian $\mathcal{H}$ over time t and to apply it using a suitable Hamiltonian simulation algorithm (Section 7). We will use the operator U to create the target state $\mathcal{H}|\psi\rangle$ on a quantum computer.

The first step is to express the state $|\psi\rangle$ as a linear combination of eigenstates of $\mathcal{H}$ using the identity operator $I=\sum_{i=1}^N|u_i\rangle\langle u_i|$

$$|\psi\rangle=\sum_{i=1}^{N}|u_i\rangle\langle u_i|\psi\rangle=\sum_{i=1}^{N}\beta_i|u_i\rangle, \tag{76}$$

Note that the state $|\psi\rangle$ in (76) does not undergo a transformation; instead, it is simply re-written in the eigenbasis $\{|u_i\rangle\}_{i=1}^N$. The eigenvectors $\{|u_i\rangle\}_{i=1}^N$ are also eigenvectors of U, with corresponding eigenvalues $e^{i\lambda_i t}$.

The algorithm exploits this property to extract the m-bit approximations of eigenvalues λ_i using QPE. QPE requires a m-qubit auxiliary register, where m is the desired binary precision of λ_i. QPE takes the two-register state $|\psi\rangle|0\rangle$ as an input and, for each eigenstate $|u_i\rangle$ records $\tilde{\lambda}_i$, the m-bit approximation of λ_i, in the auxiliary register:

$$QPE|\psi\rangle|0\rangle=\sum_{i=1}^{N}\beta_i QPE|u_i\rangle|0\rangle=\sum_{i=1}^{N}\beta_i|u_i\rangle|\tilde{\lambda}_i\rangle, \tag{77}$$

exploiting the linearity of QPE.

After applying QPE, the next step is to perform a controlled rotation, as described in Section 2.3. The controlled rotation requires an additional auxiliary qubit and uses the m-qubit register holding $\tilde{\lambda}_i$ as the reference register. The $\arcsin\left(\frac{\tilde{\lambda}_i}{C}\right)$ function acts as $f(x)$ in the definition of controlled rotation(Equation 28 in Section 2.3), where C is chosen so that $\frac{|\tilde{\lambda}_i|}{C}\leq 1$ for all $\tilde{\lambda}_i$ (Häner et al. (2018) demonstrated that arcsine is efficiently computable). We obtain:

$$C\text{-}R_y\sum_{i=1}^{N}\beta_i|u_i\rangle|\tilde{\lambda}_i\rangle|0\rangle=\sum_{i=1}^{N}\beta_i|u_i\rangle|\tilde{\lambda}_i\rangle\Big(\sqrt{1-\Big(\frac{\tilde{\lambda}_i}{C}\Big)^2}|0\rangle+\frac{\tilde{\lambda}_i}{C}|1\rangle\Big). \tag{78}$$

The next step is to measure the auxiliary qubit in the computational basis. If the measurement yields 0, the quantum state on all registers is discarded and the computation is performed again. If the measurement yields 1 then the resulting quantum state is:

$$\frac{1}{C_1}\sum_{i=1}^{N}\tilde{\lambda}_i\beta_i|u_i\rangle|\tilde{\lambda}_i\rangle|1\rangle, \tag{79}$$

where $C_1 = C\sqrt{\sum_{i=1}^{N}|\tilde{\lambda}_i\beta_i|^2}$ is the resulting normalization constant. The number of measurements (and recomputations) required to achieve the desired state can be reduced using amplitude amplification (Brassard et al., 2002), described in Section 4.2.

The last step is to uncompute the register $|\tilde{\lambda}_i\rangle$ in order to return it to the ground state $|0\rangle$. Discarding this register and the auxiliary qubit yields

$$\frac{1}{C_1}\sum_{i=1}^{N}\tilde{\lambda}_i\beta_i|u_i\rangle = \frac{1}{C_1}\mathcal{H}|\psi\rangle, \tag{80}$$

the desired result up to a normalization constant.

Similar techniques can be used to implement smooth functions of sparse Hermitian operators (Subramanian et al., 2019). Rebentrost et al. (2019) used the application of a Hermitian operator to a quantum state in order to perform gradient descent on a homogeneous polynomial. Homogeneous polynomials have the property that the application of a gradient operator is equivalent to the application of a linear operator. Let $f(\mathbf{x})$ be a homogeneous polynomial of $\mathbf{x}=(x_1,...,x_N)^T$. Then there exists an operator $D(\mathbf{x})$ such that $\nabla f(\mathbf{x}) = D(\mathbf{x})\mathbf{x}$. Because of this property, it is possible to estimate the gradient of $f(\mathbf{x})$ using the techniques described in this section.

The gradient descent algorithm starts with an initial guess vector $\mathbf{x}^{(0)}$. Rebentrost et al. (2019) encode this vector in a quantum state $\left|\mathbf{x}^{(0)}\right\rangle$ and then use the method described in this section to apply the operator $D(\mathbf{x^{(0)}})$ to the state $\left|\mathbf{x}^{(0)}\right\rangle$ in order to evaluate the gradient $\nabla f(\mathbf{x})$. For homogeneous polynomials, the operator $D(\mathbf{x^{(0)}})$ has a relatively simple structure, which makes it possible to simulate e^{-iDt} efficiently on a quantum computer using simulation techniques from the quantum principal component analysis method (Lloyd et al., 2014) described in Section 6.6. The efficient computation of e^{-iDt} makes it possible to use QPE as in (77) and, therefore, use the Hermitian operator method described in this section to evaluate gradients for homogeneous polynomials.

6.4 The HHL Linear Systems Algorithm

The linear systems algorithm by Harrow et al. (2009) exploits the ability to apply a Hermitian operator to a quantum state in order to solve linear systems of the form $Ax=b$, where x and b are N-dimensional vectors and A is an $N\times N$ matrix. Finding the solution x requires the inversion (or pseudoiversion) of the matrix A, which is computationally expensive for a high-dimensional matrix.[19]

[19]Inversion of an $N\times N$ matrix on a classical computer requires $O(N^d)$ operations, where $2<d\leq 3$.

To introduce the core of the algorithm, we assume A is a Hermitian matrix and generalize it at the end. We also assume $N=2^n$, where n is an integer. We initialize an n qubit register and encode the state b in amplitude encoding:

$$|b\rangle = \frac{1}{\|b\|}\sum_{i=1}^{N} b_i|i\rangle, \tag{81}$$

where states $|i\rangle$ are n-qubit states in computational basis; the state of each qubit in the register can (but does not have to) correspond to the binary encoding of integers i; the amplitudes b_i are the elements of vector b; $\|b\|$ is the normalization constant, $\|b\| = \sqrt{\sum_{i=1}^{N}|b_i|^2}$.

If A is a Hermitian matrix and is invertible, the solution to the quantum linear system $A|x\rangle = |b\rangle$ is a quantum state $|x\rangle$ such that

$$|x\rangle = \frac{1}{C_1}A^{-1}|b\rangle, \tag{82}$$

where the constant C_1 ensures normalization of $|x\rangle$. To streamline notation, without loss of generality, we assume in this section that $C_1 = 1$.

Let $\{\alpha_i\}_{i=1}^N$ be the set of eigenvalues of matrix A and $\{|a_i\rangle\}_{i=1}^N$ be the set of corresponding eigenstates. The eigenstates of the matrix A^{-1}, which is Hermitian since A is Hermitian, are also $\{|a_i\rangle\}_{i=1}^N$, with eigenvalues $\{\frac{1}{\alpha_i}\}_{i=1}^N$. Using the identity matrix expressed in terms of the eigenvectors of A, $I = \sum_{j=1}^{N}|a_j\rangle\langle a_j|$, we transform the expression for $|x\rangle$ into

$$|x\rangle = A^{-1}\frac{1}{\|b\|}\sum_{i=1}^{N} b_i|i\rangle = A^{-1}\frac{1}{\|b\|}\sum_{i=1}^{N}\sum_{j=1}^{N} b_i|a_j\rangle\langle a_j|i\rangle = \frac{1}{\|b\|}\sum_{i=1}^{N}\sum_{j=1}^{N} b_i\frac{1}{\alpha_j}|a_j\rangle\langle a_j|i\rangle. \tag{83}$$

The result in (83) resembles the result of the Hermitian operator routine from Section 6.3 with one difference: in the controlled-rotation of the auxiliary qubit in (78), $\arcsin\frac{C}{\tilde{\alpha}_i}$ replaces $\arcsin\frac{\tilde{\lambda}_i}{C}$. Here, the value $\tilde{\alpha}_i$ is the approximation of α_i obtained by QPE, and the constant C is such that $\frac{C}{|\tilde{\alpha}_i|} \leq 1$. If some values $\tilde{\alpha}_i$ are very small or 0, regularization techniques, similar to those in classical matrix inversion, can provide stability (for example, Tikhonov (1963) regularization). The small values of $\tilde{\alpha}_i$ can be discarded or collected in a separate state for further analysis. We refer the reader to Harrow et al. (2009) and Dervovic et al. (2018) for details.

The routine generalizes to the case where A is non-Hermitian. In this case, in place of A, we use the Hermitian matrix $\mathbf{I}A$, where $\mathbf{I}$ is the *isometry* superoperator such that:

$$\mathbf{I}A = \begin{bmatrix} 0 & A \\ A^\dagger & 0 \end{bmatrix}. \tag{84}$$

The eigenvectors and eigenvalues of the matrix $\mathbf{I}A$ are closely related to right and left eigenvectors and singular values of matrix A, u_k, v_k, and α_k, respectively. Following Harrow et al. (2009), we append an auxiliary qubit and define $|a_k^{\pm}\rangle = \frac{1}{\sqrt{2}}(|0\rangle|u_k\rangle \pm |1\rangle|v_k\rangle)$, where $|u_k\rangle =$

$\sum_{i=1}^{N} u_{ik}|i\rangle$ and $|v_k\rangle = \sum_{j=1}^{M} v_{jk}|j\rangle$, $k=1,..,K$, and K is the rank of matrix A. The operator $\mathbf{I}A$ takes the from $\mathbf{I}A = |0\rangle\langle 1|\sum_{s=1}^{S}\alpha_s|u_s\rangle\langle v_s| + |1\rangle\langle 0|\sum_{s=1}^{S}\alpha_s|v_s\rangle\langle u_s|$.[20]

The runtime of the algorithm is $O(d^4\kappa^2 \log N/\epsilon)$, where d is the sparsity of the matrix A (the number of non-zero elements in each row or column of A), $\kappa = \alpha_{max}/\alpha_{min}$ is its condition number (the ratio of the largest to the smallest singular value), and ϵ is the admissible error. If the matrix A is sparse and well-conditioned, the algorithm runs exponentially faster than the best classical linear systems algorithm, with runtime scaling as $O(Nd\kappa \log(1/\epsilon))$ (Saad, 2003).

The result of the quantum linear systems algorithm is a quantum state, which can be passed onto another quantum subroutine. For example, Schuld et al. (2016) use the building blocks for the quantum linear systems algorithm embedded in another quantum algorithm to perform prediction by linear regression. Alternatively, the quantum state can be read out for use by a classical computer, although the information in the quantum state may need to be compressed in order to preserve the computational efficiency of the algorithm. Another approach, discussed in Section 6.6 is to characterize the state using quantum PCA.

Since HHL formulated the quantum linear systems problem, $|x\rangle = A^{-1}|b\rangle$, many efficient algorithms have been found, with computational complexity gradually improving to reduce the dependence on the error ϵ, sparsity d, and condition number κ of the matrix A. Ambainis (2012) replaced QAA with variable-time amplitude amplification to reduce the dependence on the condition number. Clader et al. (2013) to precondition the matrix A using sparse approximate inverse preconditioning and reduced the dependence on both condition number and error. Kerenidis and Prakash (2016) and Wossnig et al. (2018) replaced Hamiltonian simulation using optimized Product Formula (Section 7.2) as in Berry et al. (2007) with a Hamiltonian simulation based on a quantum walk (Section 7.3), making it possible to deliver exponential speedup to low-rank (rather than sparse) problems. Childs et al. (2017) decomposed A^{-1} as a linear combination of unitaries (Section 7.4) and eschewed QPE, achieving a logarithmic dependence on the error. Subaşi et al. (2019) and An and Lin (2019) proposed an adiabatic-inspired method. Gilyén et al. (2019a) used their QSVT method (Section 9) lowered the computational complexity to the multiplicative lower bounds in all variables, eliminating the dependence on size entirely, so that the overall complexity of solving the linear systems problem is $O(\kappa \log(\kappa/\epsilon))$.

6.5 Fast Gradient Computation

The QFT at the core of most algorithms in this section also enables efficient evaluation of gradients. The quantum gradient algorithm proposed by Jordan (2005) (and refined by Gilyén et al. (2019a)) calculates an approximate N-dimensional gradient $\nabla f(\mathbf{x})$ of a function $f:\mathbb{R}^N \to \mathbb{R}$ at point $\mathbf{x} \in \mathbb{R}^N$ using a single evaluation of f. For comparison, standard classical techniques require $N+1$ function evaluations. The algorithm is suitable for problems where the function evaluations are computationally taxing, but the number of dimensions is only moderately high, because it requires $O(N)$ qubits (and $O(N)$ measurements if the result is to be used in a classical computation).

[20] If $K < (N+M)/2$, $\mathbf{I}A$ has $N+M-2K$ zero eigenvalues, corresponding to the basis states of the orthonormal complement to the $2K$-dimensional Hilbert space spanned by $|a_k^{\pm}\rangle$.

The algorithm uses a *phase oracle* (see a brief discussion of oracles in Section 2.3). An oracle is an algorithmic "black box" assumed to perform a computational task efficiently. A quantum phase oracle O_g adds a phase to a quantum state such that, given a function $g(y)$, we have

$$O_g|y\rangle = e^{ig(y)}|y\rangle. \tag{85}$$

The algorithm stems from two observations. The first observation is that, if f is twice-differentiable then, for a vector δ with a sufficiently small norm, the expansion of $f(\mathbf{x}+\delta)$ in the vicinity of $\mathbf{x}$ takes the form $f(\mathbf{x}+\delta) = f(\mathbf{x}) + \nabla f \cdot \delta + \mathcal{O}(||\delta||^2)$. The second observation is that the phase oracle for $f(\mathbf{x}+\delta)$ takes a convenient form. To define the phase oracle, we take $g = 2\pi D f$, where where $D > 0$ is a scaling factor necessary for all values of $2\pi D f$ on the relevant domain to be less than 2π. When acting on a quantum state $|\delta\rangle$ initialized to hold the value of δ, the phase oracle adds a phase that depends on the value of f: $O_{2\pi Df} : |\delta\rangle \mapsto e^{2\pi i D f(\mathbf{x}+\delta)}|\delta\rangle$. Using the expansion of $f(\mathbf{x}+\delta)$ we can write approximately $O_{2\pi Df} : |\delta\rangle \mapsto e^{2\pi i D f(\mathbf{x})} e^{2\pi i D \nabla f \cdot \delta}|\delta\rangle$. The phase contains the dot product of the gradient ∇f and the differential δ, enabling the extraction of the gradient using inverse QFT (Section 6.1). The algorithm works under the assumption that the third and fourth derivatives of f around $\mathbf{x}$ are negligible.

The algorithm starts with a uniform superposition $|\psi\rangle = \frac{1}{\sqrt{|G_x^d|}} \sum_{\delta \in G_x^d} |\delta\rangle$ over the points of a sufficiently small discretized N-dimensional grid G_x^N around $\mathbf{x}$. Each state $|\delta\rangle$ reflects coordinates recorded in the computational basis (in an $N \times m$-qubit register, where m is the binary precision). A single call to the phase oracle $O_{2\pi Df}$ creates the state

$$O_{2\pi Df}|\psi\rangle = \frac{e^{2\pi i D f(\mathbf{x})}}{\sqrt{|G_x^d|}} \sum_{\delta \in G_x^d} e^{2\pi i D \nabla f \cdot \delta}|\delta\rangle, \tag{86}$$

ready for the application of inverse QFT, which extracts $\tilde{\nabla} f$, an m-bit approximation of the gradient ∇f. The output of the algorithm is an $N \times m$-qubit state that records the coordinates of the gradient $\tilde{\nabla} f$ in the computational basis.

For a step-by-step description of Jordan's algorithm, we refer the reader to the paper by Gilyén et al. (2019a) who review the algorithm and modify it to take advantage of central-difference formulas.

6.6 Quantum Principal Component Analysis (QPCA)

Algorithms such as HHL (Section 6.4) yield a result in the form of an unknown quantum state that has to be characterized. The most straightforward way to characterize the state is to create multiple copies and take measurements to enable statistical analysis of the state. However, this approach can be computationally taxing and inefficient, particularly for non-sparse but low-rank quantum states. Lloyd et al. (2014) propose an alternative method that uses multiple copies of an quantum system to perform Principal Component Analysis (PCA) of the system, i.e. to extract its principal components – the eigenstates corresponding to largest eigenvalues. Quantum PCA (QPCA) performs the task for any unknown low-rank N-dimensional quantum state in $O(\log N)$ runtime, exponentially faster than any existing

classical algorithm. The algorithm leverages the *density matrix* formalism of quantum mechanics, an alternative way to describe quantum states, introduced in Section 2.2.

Let ρ be a density matrix describing a quantum state. The density matrix is Hermitian and has real eigenvalues r_j corresponding to eigenstates $|\chi_j\rangle$, with $j=1,..,N$. The goal of the quantum PCA (QPCA) algorithm is to extract the eigenstates and eigenvalues of ρ. Given $|\psi\rangle$, an N-dimensional quantum state, and an m-qubit register of auxiliary qubits, QPCA performs the transformation

$$QPCA|\psi\rangle|0\rangle \mapsto \sum_j \psi_j|\chi_j\rangle|\tilde{r}_j\rangle, \tag{87}$$

where $\tilde{r}_i$ are m-qubit approximations of the eigenvalues r_i and $\psi_i = \langle\chi_i|\psi\rangle$. The state in (87) has a similar structure the intermediate result (79) of the algorithm to apply a Hermitian operator to quantum state (Section 6.3). This is because the QPCA algorithm treats the density matrix ρ as a Hermitian operator that can be applied to an arbitrary quantum state. The trouble is, for algorithm in Section 6.3 to be efficient, the Hermitian operator $\mathcal{H}$ has to be sufficiently structured for the controlled-unitary with the unitary operator $e^{-i\mathcal{H}t}$ to be realized; the density matrix ρ may lack such structure.

The critical insight at the core of the QPCA is that it is possible to construct the controlled-unitary C-U with $U = e^{-i\rho\Delta t}$ for any density matrix ρ using a series of swap operations, provided the increment Δt is sufficiently small. It is then possible to use Product Formula (Section 7.2) to develop the controlled unitary with $U = e^{-i\rho t}$ based on the controlled unitary with $U = e^{-i\rho\Delta t}$, where $t = n\Delta t$.

Consider the application of $e^{-i\rho\Delta t}$ to an arbitrary N-dimensional state described by a density matrix σ:

$$e^{-i\rho\Delta t}\sigma e^{i\rho\Delta t} = \sigma - i\Delta t[\rho,\sigma] + O(\Delta t^2). \tag{88}$$

Lloyd et al. (2014) demonstrate that this operation is equivalent to applying the swap operator to the tensor product state $\rho\otimes\sigma$ and a subsequent partial trace Tr_P of the first variable:

$$\begin{aligned}\mathrm{Tr}_P e^{-iS\Delta t}\rho\otimes\sigma e^{iS\Delta t} &= (\cos^2\Delta t)\sigma + (\sin^2\Delta t)\rho - i\sin\Delta t\cos\Delta t[\rho,\sigma] \\ &= \sigma - i\Delta t[\rho,\sigma]s + O(\Delta t^2).\end{aligned} \tag{89}$$

The swap operator S is represented by a sparse matrix and $e^{-iS\Delta t}$ is computable efficiently.

The derivation of (89) uses the identity

$$e^{-iS\Delta t} = \cos(\Delta t)I - i\sin(\Delta t)S, \tag{90}$$

so that the expression $e^{-iS\Delta t}\rho\otimes\sigma e^{iS\Delta t}$ can be rewritten as:

$$[\cos(\Delta t)I - i\sin(\Delta t)S]\rho\otimes\sigma[\cos(\Delta t)I + i\sin(\Delta t)S]. \tag{91}$$

Partial trace over the $(\cos^2\Delta t)$ term yields $\mathrm{Tr}_P I\rho\otimes\sigma I = \sigma$. Partial trace over the $(\sin^2\Delta t)$

term yields $\mathrm{Tr}_P S\rho\otimes\sigma S=\rho$. Partial trace over the $\sin\Delta t\cos\Delta t$ term results in:

$$\begin{aligned}
\mathrm{Tr}_P S\rho\otimes\sigma I &= \sum_i \langle i|_P S\rho\otimes\sigma|i\rangle_P = \sum_i\sum_{j,k}\langle i|_P S\rho\otimes\sigma|j,k\rangle\langle j,k|i\rangle_P \\
&= \sum_i\sum_{j,k}\sum_{l,m}\langle i|_P|l,m\rangle\langle l,m|S\rho\otimes\sigma|j,k\rangle\langle j,k|i\rangle_P \\
&= \sum_i\sum_{j,k}\sum_{l,m}\delta_{i,l}|m\rangle\langle l,m|S\rho\otimes\sigma|j,k\rangle\langle k|\delta_{j,i} \\
&= \sum_i\sum_{m,k}|m\rangle\langle i,m|S\rho\otimes\sigma|i,k\rangle\langle k| \\
&= \sum_i\sum_{m,k}|m\rangle\langle m,i|\rho\otimes\sigma|i,k\rangle\langle k| \\
&= \sum_i\sum_{m,k}|m\rangle\langle m|\rho|i\rangle\langle i|\sigma|k\rangle\langle k| \\
&= \sum_{m,k}|m\rangle\langle m|\rho\sigma|k\rangle\langle k| = \rho\sigma. \qquad (92)
\end{aligned}$$

The transformation in (89) therefore results in $e^{-i\rho\Delta t}\sigma e^{i\rho\Delta t}$. Given multiple copies of ρ the transformation can be applied repeatedly to provide an ϵ-approximation to $e^{-i\rho t}\sigma e^{i\rho t}$, which requires $n=O(t^2\epsilon^{-1}|\rho-\sigma|^2)\leq O(t^2\epsilon^{-1})$ steps (a consequence of Suzuki-Trotter theory, see Section 7.2). The procedure is quite flexible. For example, using matrix inversion in Harrow et al. (2009), it is possible to implement $e^{-ig(\rho)}$ for any "simply computable" function $g(\rho)$.

Armed with $e^{-i\rho t}$, the next step is to apply QPE in order to record m-bit approximations of the eigenvalues r_i in auxiliary register (Section 6.2) transforming any initial state $|\psi\rangle|0\rangle$ into the desired state $\sum_i\psi_i|\chi_i\rangle|\tilde{r}_i\rangle$.

7 Hamiltonian Simulation

7.1 Overview and Preliminaries

Introduced briefly in Section 2.2, the Hamiltonian of an N-dimensional quantum system is a Hermitian operator that guides the evolution of the system over time. Let $\mathcal{H}$ be a Hamiltonian. If a system in the initial state $|\psi\rangle$, the state evolves over time according to the unitary operator $e^{-i\mathcal{H}t}$

$$|\psi(t)\rangle = e^{-i\mathcal{H}t}|\psi\rangle, \qquad (93)$$

provided the Hamiltonian $\mathcal{H}$ stays unchanged during the period of evolution.

The time evolution of the state $|\psi\rangle$ can be expressed in terms of eigenvalues and eigenstates of the Hamiltonian $\mathcal{H}$. Let $|\lambda\rangle$ be the eigenstates of the Hamiltonian $\mathcal{H}$ with eigenvalues λ: $\mathcal{H}|\lambda\rangle=\lambda|\lambda\rangle$. Then we can express the initial state $|\psi\rangle$ in terms of the eigenstates $|\lambda\rangle$:

$$|\psi\rangle = \sum_\lambda\langle\lambda|\psi\rangle|\lambda\rangle \equiv \sum_\lambda\psi_\lambda|\lambda\rangle, \qquad (94)$$

where $\psi_\lambda \equiv \langle\lambda|\psi\rangle$. The state $|\psi(t)\rangle$ evolved under $\mathcal{H}$ over time t takes the form

$$|\psi(t)\rangle = \sum_\lambda \psi_\lambda e^{-i\lambda t}|\lambda\rangle. \tag{95}$$

Hamiltonian simulation is an approximation of the evolution of a system using the evolution of a different system – usually one that is simpler or easier to control using a simple set of controllable operations. In his classic work on universal quantum simulation, Lloyd (1996) draws a parallel between Hamiltonian simulation and parallel parking a car, which is possible even though a car is only able to move backward and forward. Using a simpler, controllable quantum system, Hamiltonian simulation approximates the result of the evolution over time t of a more complex quantum system.

The goal of Hamiltonian simulation is to evolve the initial state $|\psi\rangle$ in (94) in a way that creates pre-factors $e^{-i\lambda t}$ in front of each $|\lambda\rangle$, within ϵ-error. The creation of these pre-factors is equivalent to the evolution of $|\psi\rangle$ under Hamiltonian $\mathcal{H}$.

The ability to simulate Hamiltonian evolution efficiently on a quantum computer will not only revolutionize molecular engineering but also allow us to tackle computationally hard problems such as combinatorial optimization and high-dimensional linear systems of equations. Because of its central role in quantum computing applications, Hamiltonian simulation is an active and rapidly evolving field. Additionally, Hamiltonian simulation is BQP-complete (Low and Chuang, 2019; Berry et al., 2015b) (see Section 3.3 for a brief discussion of computational complexity). In the words of Low and Chuang (2019), Hamiltonian simulation is a "universal problem that encapsulates all the power of quantum computation."

Simulatable Hamiltonians and Simulation Limits

A *simulatable* Hamiltonian one that makes it possible to approximate the unitary evolution operator $e^{-i\mathcal{H}t}$ by a quantum circuit efficiently, i.e. to an accuracy at most polynomial in the precision of the circuit and in time at most polynomial in evolution time t (Aharonov and Ta-Shma, 2003). There is no general Hamiltonian simulation algorithm able to simulate the evolution under a general Hamiltonian in $poly(\|\mathcal{H}\|t, \log N)$, where $\|\mathcal{H}\|$ is the spectral norm (defined in Eq. 99) (Childs and Kothari, 2009). Additionally, there is no general algorithm to simulate a general sparse Hamiltonian in time less than linear in $\|\mathcal{H}\|t$ – a statement known as the *No Fast-Forwarding Theorem* (Berry et al., 2007).

Hamiltonian Input Models

Efficient Hamiltonian simulation algorithms exploit the structure of the Hamiltonian.[21] Efficient simulation algorithms have been proposed for time-independent Hamiltonians such as Hamiltonians that are linear combinations of *local* terms – terms acting on a small number of qubits (Lloyd, 1996; Berry et al., 2007), *sparse* Hamiltonians that have at most $polylog(N)$ entries in each row (Aharonov and Ta-Shma, 2003; Berry et al., 2007; Berry and Childs, 2012; Berry et al., 2014), Hamiltonians that comprise a *linear combination of unitaries*

[21] Just like it is not possible to implement an arbitrary unitary efficiently, it is not possible to efficiently to simulate an arbitrary Hamiltonian Childs and Kothari (2009).

(LCU) (Childs and Wiebe, 2012), and *low-rank* Hamiltonians (Berry and Childs, 2012; Wang and Wossnig, 2018).

Hamiltonian simulation algorithms specify input models that make the terms of the Hamiltonian accessible to the quantum computer. The most popular input models include *black-box*, *sparse access*, *QROM*, and *LCU* models.

The *black-box* input model, proposed by Grover (2000) and refined by Sanders et al. (2019), uses a unitary oracle O_H that returns matrix terms $\mathcal{H}_{jk}$ and their indices j,k in a binary format:

$$O_H|j,k\rangle|z\rangle = |j,k\rangle|z \oplus \mathcal{H}_{jk}\rangle, \tag{96}$$

where $\oplus$ represents the bit-wise XOR. The oracle O_H represents a quantum algorithm that performs the encoding of Hamiltonian terms into a quantum state. For example, Sanders et al. (2019) propose an algorithm that performs O_H using quantum computing primitives, such as Toffoli gates.

If the Hamiltonian is sparse and has at most d nonzero entries in any row, then the *sparse access* input model helps drive further efficiency (Aharonov and Ta-Shma, 2003). This model uses two unitary oracles, O_H – the black-box Hamiltonian oracle – and O_F the address oracle. Oracles

$$O_H|j,k\rangle|z\rangle = |j,k\rangle|z \oplus \mathcal{H}_{jk}\rangle \tag{97}$$

$$O_F|j,l\rangle = |j\rangle|f(j,l)\rangle \tag{98}$$

where $f(j,l)$ is a function that gives the column index of the lth non-zero element in row j. The sparse access model has been the most popular input model for sparse Hamiltonian simulation. Aharonov and Ta-Shma (2003) defined a Hamiltonian as *row-sparse* if the number of non-zero entries in each row is $O(\text{polylog}N)$ or lower. A Hamiltonian is called *row-computable* if there exists an efficient (i.e. requiring $O(\text{polylog}N)$ or fewer operations) – quantum or classical – algorithm that, given a row index i, outputs a list (j,H_{ij}) of all non-zero entries in that row. Row-sparse and row-computable Hamiltonians are simulatable, provided they have a bounded spectral norm $\|H\| \leq O(\text{polylog}N)$.[22]

For Hamiltonians decomposed into a linear combination of unitaries, the input model provides the constituent unitaries and their weights.

An increasingly popular input model is the QROM input model, based on the classical QROM structure (Kerenidis and Prakash, 2016; Chakraborty et al., 2018), outlined in Section 3.1, that provides efficient quantum access to Hamiltonian terms. This input structure is used in Hamiltonian simulation based on quantum singular value transformation (Gilyén et al., 2019b), found to be the unifying framework for the top quantum algorithms of the last two decades (Martyn et al., 2021).

Matrix Norms

For reference, we include the most common matrix norms used in Hamiltonian simulation literature. The matrix norms arise in normalizations of quantum states encoding Hamiltonian terms and in estimations of computational complexity. For rigorous definitions of matrix norms and the relationships between them see, e.g., Childs and Kothari (2009).

[22]This statement is called *the sparse Hamiltonian lemma* due to Aharonov and Ta-Shma (2003).

The *spectral norm* $\|\mathcal{H}\|$ of Hamiltonian $\mathcal{H}$ is defined as

$$\|\mathcal{H}\| = \max_{\|v\|=1} \|\mathcal{H}v\| = \sigma_{max}(\mathcal{H}), \tag{99}$$

where, for a vector v, $\|v\|$ represents the Euclidean 2-norm; $\sigma_{max}(\mathcal{H})$ denotes the largest singular value. Spectral norm is the matrix 2-norm, induced by the vector 2-norm; it sometimes denoted as $\|\mathcal{H}\|_2$. Similarly, the matrix *1-norm* induced by the vector 1-norm is $\|\mathcal{H}\|_1 = \max_j \sum_i |\mathcal{H}_{ij}|$.

The *max norm* is

$$\|\mathcal{H}\|_{max} = \max_{i,j} |\mathcal{H}_{ij}|. \tag{100}$$

The *Frobenius norm*, also called the *Hilbert-Schmidt norm*, is

$$\|\mathcal{H}\|_F = \sqrt{\sum_{i,j} |\mathcal{H}_{ij}|^2} = \sqrt{\sum_k \sigma_k^2(\mathcal{H})}, \tag{101}$$

where $\sigma_k(\mathcal{H})$ are the singular values of $\mathcal{H}$.

Overview of Hamiltonian Simulation Algorithms

As a critical potential application of quantum computers and a linchpin of other algorithms, such as quantum linear systems algorithms, Hamiltonian simulation is an active area of algorithm design. In this section, we review a few influential methods of Hamiltonian simulation. We start with a foundational method *Product Formula* that splits the Hamiltonian into easy-to-simulate parts and then uses sequences of small subsystem simulations to approximate whole-system Hamiltonian evolution (Section 7.2). We then overview Hamiltonian simulation by quantum walk, an influential method that works for general Hamiltonians and is highly efficient for sparse Hamiltonians (Section 7.3). We also summarize the method of linear combination of unitaries for Hamiltonians that can be expressed as linear combinations of unitary operators (Section 7.4). And last, we demonstrate how to use quantum signal processing in combination with the quantum walk method to perform Hamiltonian simulation with computational efficiency that corresponds multiplicatively to all known lower bounds (Section 7.5).

7.2 Product Formula

One of the foundational methods of Hamiltonian simulation is the Lie-Trotter-Suzuki Product Formula (Suzuki, 1990, 1991). This approach works for time-independent local or, more generally, separable Hamiltonians. Many Hamiltonians[23] $\mathcal{H}$ can be expressed as a sum of l Hamiltonians $\mathcal{H}_j$: $\mathcal{H} = \sum_{j=1}^{l} \mathcal{H}_j$. If the time evolution operation $e^{-i\mathcal{H}_j t}$ is relatively easy to apply to a quantum state, for example because it represents a simple sequence of rotation gates, then it would be beneficial to express the time evolution $e^{-i\mathcal{H}t}$ by the Hamiltonian H as a function of time evolution operators $e^{-i\mathcal{H}_j t}$ of the constituent Hamiltonians H_j.

[23]Particularly local Hamiltonians that usually apply to real physical systems.

Table 1: Hamiltonian simulation algorithms. The table demonstrates the gradual improvement of query complexity of Hamiltonian situation with respect to the essential parameters of the simulation: the dimension of the Hilbert space N, admissible error ϵ, and the sparsity parameter d, equal to the number of non-zero elements in each row (column) of the Hamiltonian.

Algorithm	Citation	Method	Query Complexity
Lie-Suzuki-Trotter Product Formula	Lloyd (1996)	Finite sum of local Hamiltonians	
Adiabatic Hamiltonian evolution	Aharonov and Ta-Shma (2003)	Adiabatic evolution for row-sparse, row-computable Hamiltonians	$O(\text{poly}(\log(N),d)(\|\mathcal{H}\|t)^2/\epsilon$
Optimized Product Formula	Berry et al. (2007)	Efficiently decompose Hamiltonian into a sum of local Hamiltonians	$O(d^4(\log^* N\|\mathcal{H}t\|)^{1+o(1)})$
Quantum walk-based Hamiltonian simulation	Berry and Childs (2012)	Combine quantum walk with quantum phase estimation for general Hamiltonians with black-box access	$O(d^{2/3}[(\text{loglog}d)\|\mathcal{H}\|t]^{4/3}/\epsilon^{1/3})$
Linear combination of unitaries (LCU)	Childs and Wiebe (2012)	Decompose a Hamiltonian into a finite linear combination of unitaries	$O(d\|\mathcal{H}\|_{max}t/\sqrt{\epsilon})$
Taylor-series based	Berry et al. (2015a)	Taylor series expansion of $e^{-i\mathcal{H}t}$	$O(\tau\log(\tau/\epsilon)/\text{loglog}(\tau/\epsilon))$, where $\tau = d^2\|\mathcal{H}\|_{max}t$
BCK	Berry et al. (2015b)	Combine quantum walk with linear combination of unitaries	$O(\tau\log(\tau/\epsilon)/\text{loglog}(\tau/\epsilon))$, where $\tau = d\|\mathcal{H}\|_{max}t$
Quantum signal processing	Low and Chuang (2017)	Combine quantum walk with quantum signal processing	$O(\tau + \frac{\log(1/\epsilon)}{\text{loglog}(1/\epsilon)})$ matches theoretical additive lower bound Berry et al. (2015b)

In general $e^{-i\mathcal{H}t} \neq \prod_{j=1}^{l} e^{-i\mathcal{H}_j t}$, but we can follow Suzuki (1990, 1991) and adopt the classical Lie-Trotter formula for exponentiated matrices:

$$e^{-i\mathcal{H}t} = e^{-i\sum_{j=1}^{l}\mathcal{H}_j t} = \lim_{r\to\infty}\Big(\prod_{j=1}^{l} e^{-i\mathcal{H}_j t/r}\Big)^r. \tag{102}$$

For a finite r, the product method approximates the time evolution by $\mathcal{H}$ using a sequence of time evolutions over short segments of time t/r by constituent Hamiltonians $\mathcal{H}_j$.

This method is flexible and robust, but it requires $O(lr)$ queries to the Hamiltonians $\mathcal{H}_j$. For an N-dimensional systems, the number l of simple-to-apply constituent Hamiltonian evolution operators $e^{-i\mathcal{H}_j t}$ can be of $O(N)$. This can be efficient in many situations, but would not yield exponential speedup relative to classical methods. Additionally, the algorithm scales superlinearly in simulation time t, which is above the optimal linear dependence on t. The approach also suffers from poor scaling in the sparseness $O(d^4)$.

7.3 Hamiltonian Simulation by Quantum Walk

Childs (2010) proposed a way to simulate Hamiltonian evolution using a quantum walk (Section 5.2). This method became a part of many efficient and influential algorithms, such as the Berry et al. (2015b) and Low and Chuang (2019) algorithms, considered the state of the art in Hamiltonian simulation at the time of writing this review.

The method uses two properties of quantum walks. First, it is possible to construct a quantum walk operator from a Hamiltonian, provided the elements of the Hamiltonian can be encoded in a quantum state. Second, each eigenvector of the quantum walk unitary corresponds to an eigenvector of the simulated Hamiltonian $|\lambda\rangle$, with eigenvalues $\pm e^{\pm i \arcsin\lambda}$. Childs (2010) demonstrates that it is possible to combine these two properties of quantum walks with QPE (Section 6.2) to perform Hamiltonian simulation.

Hamiltonian simulation via a quantum walk proceeds in a duplicated Hilbert space. For a Hamiltonian acting on a space spanned by $N = 2^n$ qubits, first an auxiliary qubit is appended doubling the dimension of the Hilbert space. Then another register of $n+1$ qubits is appended, expanding the Hilbert space for the quantum walk. To implement the walk, an operator T is defined that acts on the doubled Hilbert space:

$$T = \sum_{j=0}^{N-1}\sum_{b\in\{0,1\}} (|j\rangle\langle j| \otimes |b\rangle\langle b|) \otimes |\varphi_{jb}\rangle, \tag{103}$$

where the first register is the register that spans the Hilbert space of $\mathcal{H}$, the second register is the auxiliary qubit, and the state $|\varphi_{jb}\rangle$ is across the third and fourth registers that duplicate the Hilbert space of $\mathcal{H}$ and the auxiliary qubit. The state $|\varphi_{jb}\rangle$ encodes the absolute values of the non-zero elements of $\mathcal{H}$ as follows:

$$|\varphi_{j1}\rangle = |0\rangle|1\rangle \tag{104}$$

$$|\varphi_{j0}\rangle = \frac{1}{\sqrt{d}}\sum_{l\in F_j}|l\rangle\Big(\sqrt{\frac{\mathcal{H}^*_{jl}}{X}}|0\rangle + \sqrt{1-\frac{|\mathcal{H}^*_{jl}|}{X}}|1\rangle\Big), \tag{105}$$

where $X \geq \|\mathcal{H}\|_{max}$ and F_j is the set of nonzero elements of $\mathcal{H}$ in column j.

The unitary U corresponding to a Szegedy quantum walk (Section 5.2) takes the form

$$U = iS(2TT^{\dagger} - \mathbb{I}), \tag{106}$$

where the swap operator S swaps the two registers ($S|a_1\rangle|a_2\rangle|b_1\rangle|b_2\rangle = |b_1\rangle|b_2\rangle|a_1\rangle|a_2\rangle$) and $\mathbb{I}$ is identity that acts on both registers.

The random walk unitary U has properties that make it an effective building block of Hamiltonian simulation: The eigenstates of U are related to eigenstates of $\mathcal{H}$ and the eigenvalues of U are close to the desired prefactor of Hamiltonian simulation $e^{-it\lambda}$:

$$U|\mu_{\pm}\rangle = \mu_{\pm}|\mu_{\pm}\rangle \tag{107}$$

$$|\mu_{\pm}\rangle = (T + i\mu_{\pm}ST)|\lambda\rangle|0\rangle \tag{108}$$

$$\mu_{\pm} = \pm e^{\pm i \arcsin \frac{\lambda}{\|\mathcal{H}\|_1}}. \tag{109}$$

Childs (2010) was first to propose the use of the quantum walk phase $e^{\pm it\arcsin\lambda}$ to construct the exponent $e^{-it\lambda}$ that simulates Hamiltonian evolution. His Hamiltonian simulation algorithms uses QPE (Section 6.2) and a special transformation $F_t = e^{-it\sin\phi}|\theta,\phi\rangle$ to induce with high fidelity the phase $e^{-i\tilde{\lambda}t}$, where $\tilde{\lambda}$ is an approximation of λ from QPE.

Berry et al. (2015b) further optimized the method by using linear combination of unitaries (LCU) to construct the prefactor $e^{-it\lambda}$ from $e^{\pm it\arcsin\lambda}$. Instead of relying on QPE and a functional transformation, Berry et al. (2015b) decompose $e^{-it\lambda}$ into a series of exponents of $e^{\pm it\arcsin\lambda}$:

$$\sum_{m=-\infty}^{\infty} J_m(z)\mu_{\pm}^m = e^{iz\frac{\lambda}{\|\mathcal{H}\|_1}}, \tag{110}$$

where $J_m(z)$ are Bessel functions of the first kind. As a result, effectively we have:

$$\sum_{m=-\infty}^{\infty} J_m(-t)U^m = e^{-it\frac{\mathcal{H}}{\|\mathcal{H}\|_1}}. \tag{111}$$

The sum in (111) truncated to $|m| \leq k$ is an efficient approximation of the desired phase $e^{-it\frac{\mathcal{H}}{\|\mathcal{H}\|_1}}$. Success probability for U^m decays with m, limiting the efficiency of this algorithm. As will be discussed in the next Section, Low and Chuang (2017) circumvent this problem by proposing an alternative route from the prefactors $e^{\pm it\arcsin\lambda}$ to $e^{-it\lambda}$.

The query complexity of Hamiltonian simulation based on a quantum walk is linear in t and in d, the sparseness parameter.

Hamiltonian simulation is also a way to apply a unitary, since for any unitary there is a corresponding Hamiltonian. The quantum-walk approach enables the implementation of an $N \times N$ unitary in $\tilde{O}(N^{2/3})$ queries, with typical unitaries requiring $\tilde{O}(\sqrt{N})$ queries (Berry and Childs, 2012).

7.4 Linear Combination of Unitaries

The linear combination of unitaries (LCU) method uses a series of controlled unitaries combined with multi-qubit rotations to apply a complex unitary U such that

$$U = \sum_j \beta_j V_j, \tag{112}$$

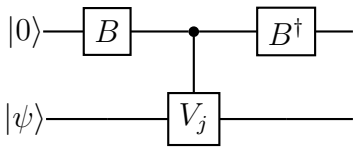
where the weights $\beta_j > 0$.

Let B be a unitary operator such that

$$B|0\rangle = \frac{1}{\sqrt{s}} \sum_j \sqrt{\beta_j} |j\rangle, \tag{113}$$

where $s = \sum_j \beta_j$.

Using the sequence of gates below:



obtain

$$\frac{1}{s}|0\rangle U|\psi\rangle + \sqrt{1 - \frac{1}{s^2}} \left|\Psi^{\perp}\right\rangle, \tag{114}$$

where the state $\left|\Psi^{\perp}\right\rangle$ spans the subspace defined by the auxiliary in state $|1\rangle$, i.e. it is orthogonal to any state of the form $|0\rangle|\bullet\rangle$, including $|0\rangle U|\psi\rangle$.

Use oblivious amplitude amplification to amplify the state $|0\rangle U|\psi\rangle$. Unlike original QAA (Section 4.2), oblivious amplitude amplification (developed by Berry et al. (2014) based on work by Marriott and Watrous (2005)) provides a way to amplify an a priori unknown state.

Linear combination of unitaries enables Hamiltonian simulation through expansion of $e^{-i\mathcal{H}t}$, for example as a Taylor series (Berry et al., 2015a) or a series of quantum walk steps (Berry et al., 2015b). For the Taylor series expansion, the Hamiltonian is expressed as a linear combination of unitaries $\mathcal{H} = \sum_l \alpha_l \mathcal{H}_l$, where each constituent Hamiltonian H_l is unitary. Then the Taylor expansion of $e^{-i\mathcal{H}t}$ is a linear combination of unitaries

$$e^{-i\mathcal{H}t} \approx \sum_{k=0}^{K} \frac{(-it)^k}{k!} \alpha_{l_1} ... \alpha_{l_k} \mathcal{H}_{l_1} ... \mathcal{H}_{l_k}. \tag{115}$$

Alternatively, Berry et al. (2015b) approximate the evolution operator $e^{-i\mathcal{H}t}$ as a series of quantum walk steps as shown in Eq. (111), where powers of the quantum walk operator U are unitary.

The expansion of the quantum evolution operator $e^{-i\mathcal{H}t}$ as a series of quantum walk step is one of the most efficient Hamiltonian simulation methods to date, requiring $O(\tau \log(\tau/\epsilon)/\log\log(\tau/\epsilon))$,

where $\tau = d\|\mathcal{H}\|_{max}t$ queries to the input oracles O_H and O_F. The dependence on time is nearly linear, close to the lower bound set by the No Fast-Forwarding theorem (Berry et al., 2007). The dependence of sparsity d is also nearly linear, while the dependence on the simulation error ϵ is sublinear. The computational complexity only depends on the dimension of the system indirectly, through the matrix norm of the Hamiltonian $\|\mathcal{H}\|_{max}$. Additionally, gate complexity of this method – the number of gates required to implement it – is only slightly larger than query complexity. A small disadvantage of the method is that, because it is based on quantum walks, it requires duplicating the register of qubits simulating the quantum systems. It is possible to extend the method to time-dependent Hamiltonians (Berry et al., 2020).

7.5 Hamiltonian Simulation by Quantum Signal Processing (QSP)

Low and Chuang (2017) proposed an alternative way to perform Hamiltonian simulation using a method they called "quantum signal processing" (QSP), because some of their methodologies are analogous to filter design in classical signal processing. (We discuss QSP in greater detail in Section 9.1.) The time complexity of the method matches the proved lower bounds for Hamiltonian simulation of sparse Hamiltonians. The method uses a single auxiliary qubit to encode the eigenvalues of the simulated Hamiltonian, and then transforms these eigenvalues using a sequence of rotation gates applied to the auxiliary qubits. Using specialized sequences of rotation gates, the method makes it possible to perform polynomial functions of degree d on the input in $O(d)$ elementary unitary operations.

QSP borrows ideas from quantum control – the area of quantum computing that deals with extending the life of qubits and improving the fidelity of quantum computation. Low and Chuang (2017) point out that Hamiltonian simulation is a mapping of a physical system onto a different physical system that is possible to control more precisely. Because of this connection between Hamiltonian simulation and quantum control, it is possible to create robust quantum simulations using the QSP framework.

Let W be a quantum walk unitary that encodes the Hamiltonian $\mathcal{H}$ as in Section 7.3. Consider a function $e^{ih(\theta)} = e^{-it\sin\theta}$, where $\theta = \arcsin\frac{\lambda}{\|\mathcal{H}\|_1}$, so that $e^{ih(\theta)} = e^{it\frac{\lambda}{\|\mathcal{H}\|_1}}$. QSP provides an efficient way to calculate an finite approximation to the Fourier transform of $e^{ih(\theta)}$.

Low and Chuang (2017) observe that $e^{ih(\theta)}$ splits into real and imaginary parts, $A(\theta)$ and $C(\theta)$, respectively

$$A(\theta) + iC(\theta) = e^{ih(\theta)} = e^{-it\sin\theta}, \tag{116}$$

and find their Fourier transforms using the Jacobi-Anger expansion

$$\cos(\tau\sin\theta) = J_0(\tau) + 2\sum_{k\text{even}>0}^{\infty} J_k(\tau)\cos(k\theta) \tag{117}$$

$$\sin(\tau\sin\theta) = 2\sum_{k\text{odd}>0}^{\infty} J_k(\tau)\sin(k\theta), \tag{118}$$

where $J_k(z)$ are Bessel functions of the first kind.

The method turns out to have query complexity that corresponds multiplicatively to all known lower bounds.

8 Quantum Optimization

Optimization is the process of finding x such that $f(x)$, where $f : \mathbb{R}^{\prime} \rightarrow \mathbb{R}$, is minimized. Optimization plays an important role in many statistical methods, including most machine learning methods. A subset of optimization problems – combinatorial optimization problems of finding an n-bit string that satisfies a number of Boolean conditions – represent some of the most computationally difficult tasks for classical computers to solve. These problems, which belong to the computational complexity class *NP*, include practical tasks such as manufacturing scheduling or finding the shortest route for delivery to multiple locations. To solve NP-hard problems classical computers need time or memory that scale, in the worst case, exponentially with n. Further, some of the problems belong to the *NP-complete* subset of NP-hard problems. These problems can be converted into one other in time polynomial in n. An efficient solution to a single NP-complete problem would solve all problems in the computational class NP.

While it is not likely that quantum computers can solve NP-complete problems, they can solve some NP-hard problems (such as prime factorization) and efficiently deliver controlled approximations to some NP-complete problems. Quantum approximate optimization algorithm (QAOA) is the most famous quantum algorithm for the approximate solution of combinatorial optimization problems. The relative efficiency of a QAOA for a general combinatorial problem compared with the best classical algorithm to solve the problem is not known (see, e.g. Zhou et al., 2020); however, we do know that it is not possible to simulate QAOA on a classical computer (Farhi and Harrow, 2016). Additionally, QAOA may be robust enough to harness the power of NISQs, the noisy quantum computers available today, and to deliver practical quantum computing advantage in the near term (McClean et al., 2016).

8.1 Adiabatic Quantum Computing (AQC)

As discussed in Section 2.2, a closed quantum system that evolves according to a Hamiltonian $\mathcal{H}$ with eigenstates $|\lambda\rangle$ and eigenvalues λ will evolve from a starting state $|\psi(0)\rangle$ into the state $|\psi(t)\rangle = \sum_{\lambda} \psi_0 e^{-i\lambda t} |\lambda\rangle$, where $\psi_0 = \langle\lambda|\psi(0)\rangle$, the inner product of $|\psi(0)\rangle$ and $|\lambda\rangle$. This property implies that, if the starting state $|\psi(0)\rangle$ is one of the eigenstates of $\mathcal{H}$, $|\lambda\rangle$, $|\psi(0)\rangle = |\lambda\rangle$, then the evolution under the Hamiltonian $\mathcal{H}$ will leave the state unchanged, $|\psi(t)\rangle = e^{-i\lambda}|\lambda\rangle \propto |\psi(0)\rangle$, since the overall phase has no impact on quantum states and is ignored. This is the essence of the *Adiabatic Theorem* of Born and Fock (1928): if the Hamiltonian of the system is changed slowly enough, the system will stay in its *ground state* (the eigenstate corresponding to the lowest eigenvalue of the Hamiltonian), provided the *spectral gap* (the gap between the lowest and the second lowest eigenvalues) is maintained throughout the evolution.

Adiabatic quantum computing (AQC) algorithms (Farhi et al., 2000) leverage the Adiabatic Theorem. An AQC algorithm starts in an easy-to-prepare starting state $|S\rangle$, usually the ground state of the Hamiltonian $\mathcal{H}_S$. The goal of the algorithm is to arrive at the ground state of the ending Hamiltonian $\mathcal{H}_E$. To transform the initial state $|S\rangle$ into the desired state $|E\rangle$, the system is evolved slowly by applying the unitary evolution operator corresponding

to a Hamiltonian $\mathcal{H}(t)$ that slowly transitions from $\mathcal{H}_S$ to $\mathcal{H}_E$ by increasing the weight $\beta(t)$:

$$\mathcal{H}(t) = \mathcal{H}_S(1-\beta(t)) + \mathcal{H}_E\beta(t). \tag{119}$$

If the spectral gap is preserved throughout the transformation from $\mathcal{H}_S$ to $\mathcal{H}_E$, the starting state evolves into the ground state of $\mathcal{H}_E$. The squared inverse of the spectral gap bounds the rate at which the Hamiltonian can evolve from $\mathcal{H}_S$ to $\mathcal{H}_E$ which, in turn, bounds the runtime efficiency of the AQC algorithm (Reichardt, 2004). Aharonov et al. (2008) proved that, in the absence of noise and decoherence, adiabatic quantum computation is theoretically equivalent to circuit based computation.

In practice, AQC has both advantages and disadvantages compared with circuit-based computation. An attractive property of adiabatic quantum computation is that, if the spectral gap is large enough (compared with the inverse of the evolution time $1/t$), the computation is robust even in the presence of noise. A disadvantage is that, for many systems, there are stringent physical limits on how fast the effective Hamiltonian can change; because of this, the required time for full transition from $\mathcal{H}_S$ to $\mathcal{H}_E$ may be slower than available system coherence time.

A recent review article by Albash and Lidar (2018) provides more detail about AQC.[24] AQC may contribute most powerfully in chemistry (see, e.g. Babbush et al., 2014; Veis and Pittner, 2014) and may unlock applications in other fields, such as machine learning (Neven et al., 2008; Denchev et al., 2012; Seddiqi and Humble, 2014; Potok et al., 2021) and combinatorial optimization (Farhi et al., 2000, 2001; Choi, 2011, 2020).

8.2 Quantum Approximate Optimization Algorithm (QAOA)

The Quantum Approximate Optimization Algorithm (QAOA) proposed by Farhi et al. (2014) finds computationally-efficient approximations for a class of NP-hard combinatorial optimization problems. Combinatorial optimization is equivalent to searching for string z of length n that satisfies m clauses. The objective function is the number of clauses string z satisfies:

$$C(z) = \sum_{k=1}^{m} C_k(z), \tag{120}$$

where $C_k(z) = 1$ if z satisfies clause k and 0 if it does not. The algorithm finds a string z – among the 2^n possible n-bit strings – such that $C(z)$ is close to $\max_z C(z)$.

The core idea of the algorithm is that it is possible to create a parameterized quantum superposition of 2^n states that represent all possible binary strings z such that the expectation of the objective function C for this state is maximized for a set of parameters. The quantum state corresponding to the maximum expectation of C contains the approximate solution to the combinatorial optimization with a high probability.

[24] A related way of performing optimization is quantum annealing (Apolloni et al., 1989). Here the optimization starts in an arbitrary initial state and then explores the cost function landscape until it finds the ground state of the system. Quantum annealing is the method deployed by the company D-Wave.

The algorithm starts with a uniform superposition of all possible n-bit binary strings z in the computational basis

$$|s\rangle = \frac{1}{\sqrt{2^n}} \sum_z |z\rangle. \tag{121}$$

Two types of parameterized unitary operators are applied to this state. The first unitary operator has the form

$$U(C,\gamma) = e^{-i\gamma C} = \prod_{k=1}^{m} e^{-i\gamma C_k}, \tag{122}$$

where γ is a scalar parameter, $\gamma \in [0,2\pi)$ The second unitary operator is based on an operator of the form

$$B = \sum_{j=1}^{n} \sigma_j^x, \tag{123}$$

where σ_j^x is a Pauli operator (Section 2.2) applied to qubit j, $j=1,..,n$. The unitary operator $U(B,\beta)$ equals

$$U(B,\beta) = e^{-i\beta B} = \prod_{j=1}^{n} e^{-i\beta\sigma_j^x}, \tag{124}$$

where $\beta \in [0,\pi)$.[25]

Alternating application of the parameterized unitary operators $U(C,\gamma)$ and $U(B,\beta)$ to the initial state $|s\rangle$ creates a parameterized quantum state

$$|\boldsymbol{\gamma},\boldsymbol{\beta}\rangle = U(B,\beta_p)U(C,\gamma_p)...U(B,\beta_1)U(C,\gamma_1)|s\rangle, \tag{125}$$

where $\boldsymbol{\gamma} = (\gamma_1,...\gamma_p)$ and $\boldsymbol{\beta} = (\gamma_1,...\gamma_p)$ are parameter vectors. The parameter p reflects the resulting quantum circuit depth, which, as Farhi et al. (2014) show, controls the quality of the approximation, converging to the exact result in the limit of $p \to \infty$. In effect, QAOA applies Product Formula (Section 7.2) to the AQC algorithm (Section 8.1).

The optimal approximation is the quantum state $|\boldsymbol{\gamma}^*,\boldsymbol{\beta}^*\rangle$ maximizing the expectation value of the objective function C

$$\text{argmax}_{\boldsymbol{\gamma},\boldsymbol{\beta}} \langle \boldsymbol{\gamma},\boldsymbol{\beta}|C|\boldsymbol{\gamma},\boldsymbol{\beta}\rangle. \tag{126}$$

The maximization proceeds in a classical outer loop using gradient descent, Nelder-Mead, or other methods. At step t, the classical computer controls parameterized unitary gates to create the state $\left|\boldsymbol{\gamma}^{(t)},\boldsymbol{\beta}^{(t)}\right\rangle$ and collects results of measuring C in this state The results of the measurements of C feed into the parameter update $(\boldsymbol{\gamma}^{(t)},\boldsymbol{\beta}^{(t)}) \mapsto (\boldsymbol{\gamma}^{(t+1)},\boldsymbol{\beta}^{(t+1)})$. The unitary gates, updated with the new set of parameters and applied to the reset initial state $|s\rangle$, create

[25]The uniform state $|s\rangle$ is the highest-eigenvalue eigenstate of B.

the updated quantum state $\left|\boldsymbol{\gamma}^{(t+1)},\boldsymbol{\beta}^{(t+1)}\right\rangle$. The algorithm stops when a stopping criterion, such as an increase in the expectation value of C below a given threshold, is reached.[26]

Because the objective function C is defined in general terms, the algorithm lends itself to a range of combinatorial approximation problems. Farhi et al. (2014) demonstrate the solution to MaxCut, the problem of cutting a graph into two parts in a way that maximizes the reduction in cost function. Proposed practical industrial applications include finance (Fernández-Lorenzo et al., 2020) and wireless scheduling (Choi et al., 2020).

8.3 Hybrid Quantum-Classical Variational Algorithms

QAOA and its counterpart for solving quantum (rather than classical) optimization problems, the Variational Quantum Eigensolver (Peruzzo et al., 2014), are examples of hybrid quantum-classical algorithms. Hybrid quantum-classical algorithms provide a way to harness the power of small quantum computers by allocating all tasks that do not deliver a provable quantum advantage, such as simple arithmetic, to a classical computer. By playing to the strengths of quantum and classical algorithms, hybrid algorithms deliver efficient optimization (McClean et al., 2016; McArdle et al., 2019).

The structure of hybrid quantum-classical variational algorithms is similar to that of QAOA: A quantum state encodes the variational (i.e. trial) solution; a quantum mechanical observable represents the cost function Rebentrost et al. (2018a); Mitarai et al. (2018); Zoufal et al. (2019); Schuld et al. (2020); Zoufal et al. (2020). A classical computer stores and updates variational parameters, which it uses to classically control the quantum gates used create the variational quantum state. The expectation value of measurements of the quantum mechanical observable in the variational quantum state is the cost associated the set of variational parameters that define the state. The classical computer collects the results of these measurements and uses them to update variational parameters. The classical computer then uses the updated variational parameters to reset the quantum gates used to prepare the next iteration of the variational quantum state. In some cases, direct measurements of the gradient of the cost function can improve convergence of hybrid variational methods (Harrow and Napp, 2019).

9 Quantum Eigenvalue and Singular Value Transformations

Quantum Singular Value Transformation (QSVT) by Gilyén et al. (2019b), a generalization of Quantum Signal Processing (QSP) by Low and Chuang (2017), has recently emerged as a unifying framework, encompassing all the major families of quantum algorithms as specific instances (Martyn et al., 2021). The algorithms leverage the fact that transformations of quantum subsystems can be non-linear even though the transformations of closed quantum systems have to be linear – more specifically, unitary – and reversible.

[26] Marsh and Wang (2020) observe that QAOA is a form of a quantum walk with phase shifts and use this observation to generalize the algorithm.

Let $\mathcal{H}$ be a Hamiltonian acting on a 2^n-dimensional space spanned by a register of n qubits. Let $|\lambda\rangle$ be the n-qubit eigenstates of the Hamiltonian $\mathcal{H}$ corresponding to eigenvalues λ: $\mathcal{H}|\lambda\rangle = \lambda|\lambda\rangle$. QSVT enables us to perform a polynomial transformation of the eigenvalues of $\mathcal{H}$:

$$Poly(\mathcal{H}) = \sum_{\lambda} Poly(\lambda)|\lambda\rangle\langle\lambda|, \tag{127}$$

assuming the operator norm of the Hamiltonian obeys $\|H\| \leq 1$ to enable block-encoding of the system in a larger system. More generally, QSVT enables polynomial transformations of singular values of a general (non-Hermitian) matrix A (Eq. 140 in Section 9)

In this section, we introduce QSP (Section 9.1), review QSVT for Hermitian matrices (Quantum Eigenvalue Transformation in Section 9.2), and describe the general form of QSVT (Section 9) and its universality (Section 9.4) as a framework for other quantum algorithms.

9.1 Quantum Signal Processing

Quantum Signal Processing (QSP) is a method to enact polynomial transformations of a "signal" x (assuming $x \in [-1,1]$) embedded in a unitary W acting on a single qubit. QSP transformations proceed as sequences of simple rotations of the qubit followed by post-selection (Section 2.3) to perform the polynomial transformations of x. To develop QSP, Low et al. (2016) drew inspiration from the signal processing methods of nuclear magnetic resonance (NMR) – a powerful and important technology widely used in medicine, chemistry, petroleum industry, materials science, and physics.

A QSP algorithm has four components. The first component is the *signal unitary* $W(x)$ – the unitary that encodes the signal x to be transformed. The second component is the *signal processing unitary* $S(\phi_j)$, usually a simple rotation by an angle ϕ_j, an element of a tuple $\vec{\phi} = (\phi_0,...,\phi_d)$. The algorithm proceeds as a sequence of alternating unitaries $S(\phi_j)$ and $W(x)$. The third component is the sequence of rotational angles in the tuple $\vec{\phi}$, which determines what polynomial transformation the "signal" x undergoes. The forth component is the *signal basis*, e.g. $\{|+\rangle,|-\rangle\}$, used to perform a measurement at the end of the algorithm.

Let W be the *signal unitary* acting on a two-state quantum system, such as a single qubit:

$$W(x) = \begin{bmatrix} x & i\sqrt{1-x^2} \\ i\sqrt{1-x^2} & x \end{bmatrix} = e^{iX\arccos(x)}, \tag{128}$$

where X is a Pauli matrix and $e^{iX\arccos(x)}$ represents rotation about the x axis by angle $2\arccos(x)$. Note that encoding of x in a unitary can take multiple forms, and the form in (128) represents a specific choice, convenient for discussion of Quantum Eigenvalue Transformation and QSVT and their applications in the following sections (Martyn et al., 2021).

Let $S(\phi)$ be a *signal processing* operator. A convenient choice is

$$S(\phi) = e^{i\phi Z}, \tag{129}$$

where Z is a Pauli matrix and $e^{i\phi Z}$ represents rotation about the z axis by angle 2ϕ. In principle, $S(\phi)$ can take other forms, as long as $S(\phi)$ does not commute with $W(x)$.

For a specific choice of a d-dimensional parameter vector $\vec{\phi} = (\phi_0,\phi_1,..\phi_d)$, a series of alternating applications of $S(\phi_j)$ and $W(x)$ results in a polynomial transformation of the signal x:

$$U_{\vec{\phi}}(x) = S(\phi_0)\prod_{j=1}^{d} W(x)S(\phi_j) = \begin{bmatrix} P(x) & iQ(x)\sqrt{1-x^2} \\ iQ^*(x)\sqrt{1-x^2} & P^*(x) \end{bmatrix}, \tag{130}$$

where $P(x)$ and $Q(x)$ are complex polynomials of degree less than or equal to d and $d-1$, and with parity of ($d \mod 2$) and ($d-1 \mod 2$), respectively. The polynomials satisfy $|P(x)|^2+(1-x^2)|Q(x)|^2=1$ for all $x\in[-1,1]$. Importantly, given polynomials $P(x)$ and $Q(x)$ that satisfy the aforementioned conditions, it is possible to find $\vec{\phi}$ that satisfies (130).

The polynomial $P(x)$ determines the probability p that the state $|0\rangle$ stays unchanged under the operation $U_{\vec{\phi}}(x)$, $P(x) = \langle 0|U_{\vec{\phi}}(x)|0\rangle$ and $p = |P(x)|^2$. The choice of $\vec{\phi}$ determines the polynomial. For example, the choice $\vec{\phi} = (0,0)$ results in $P(a) = a$; $\vec{\phi} = (0,0,0)$ results in $P(a) = 2a^2-1$, and so on, with $\vec{\phi} = (0,0,...,0)$ in d dimensions yielding Chebyshev's polynomials of the first kind, $T_d(a)$.

For greater expressiveness of $U_{\vec{\phi}}(x)$, consider the matrix element $\langle +|U_{\vec{\phi}}(x)|+\rangle$, where $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)$ is a state in the Hadamard basis, chosen in this case as the *signal basis*. The state $|+\rangle$ plays the role of the *reference state*. Post-selection on the reference state yields a polynomial transformation of x

$$\langle +|U_{\vec{\phi}}(x)|+\rangle = \mathrm{Re}(P(x)) + i\mathrm{Re}(Q(x))\sqrt{1-x^2}. \tag{131}$$

Post-selection relative to the signal basis is one of the critical steps of QSP. After all, we are able to perform a non-linear quantum transformation *because* it applies to a subsystem of a larger system. Post-selection is a way to extract the smaller system from the larger system at the end of the computation.

9.2 Quantum Eigenvalue Transformation

Gilyén et al. (2018) generalized QSP (Section 9.1) to perform polynomial transformations of matrices rather than scalar signals. They combined QSP with *block encoding* and *qubitization*.

Block encoding is a way to embed a linear non-unitary operator acting on a quantum system inside a unitary operator acting on a larger system that contains the smaller quantum system. A simple way to embed a quantum system inside a larger system is to append an auxiliary qubit. Consider a quantum system on n qubits. Its Hilbert space is $\mathbb{C}^{2^n}$. Appending the auxiliary qubit doubles the Hilbert space to $\mathbb{C}^{2^{n+1}}$. If the operator to be embedded E is unitary operator, then a simple block-encoding of E is the control-E operator, where E applies to the register of n qubits conditional on the state of the auxiliary qubit.

Consider a Hermitian operator $\mathcal{H}$ acting on the quantum system of n qubits. It is possible to embed this Hamiltonian in a unitary U acting on the expanded system of the n qubits plus the auxiliary. For example, let U take the form

$$U = I\otimes\mathcal{H} + iX\otimes\sqrt{1-\mathcal{H}^2}, \tag{132}$$

where the Pauli operators Z and X act on the auxiliary qubit and the Hermitian operators $\mathcal{H}$ and $\sqrt{1-\mathcal{H}^2}$ act on the n-qubit register. The eigenstates of $\mathcal{H}$ are n-qubit states $|\lambda\rangle$ with eigenvalues λ: $\mathcal{H}|\lambda\rangle = \lambda|\lambda\rangle$. The operator $\sqrt{1-\mathcal{H}^2}$ is Hermitian and shares its eigenstates with the Hamiltonian $\mathcal{H}$; it is easy to demonstrate using Taylor expansion that $\sqrt{1-\mathcal{H}^2}|\lambda\rangle = \sqrt{1-\lambda^2}|\lambda\rangle$. As a consequence, the operator U is unitary, $U^\dagger U = UU^\dagger = I$.[27]

Qubitization is the reduction of a multi-qubit state to a two-state qubit-like system. For example, qubitization is at the core of Grover's search algorithm (Section 4), where the "good" state and its complement effectively act as a two-state system. They span a (two-dimensional) plane, invariant under Grover iterations.

Consider the application of the unitary U from (132) to states of the extended system $|0\rangle|\lambda\rangle$ and $|1\rangle|\lambda\rangle$, where $|0\rangle$ and $|1\rangle$ represent the states of the auxiliary qubit:

$$U|0\rangle|\lambda\rangle = \lambda|0\rangle|\lambda\rangle + i\sqrt{1-\lambda^2}|1\rangle|\lambda\rangle \tag{133}$$

$$U|1\rangle|\lambda\rangle = \lambda|1\rangle|\lambda\rangle + i\sqrt{1-\lambda^2}|0\rangle|\lambda\rangle. \tag{134}$$

The space spanned by $|0\rangle|\lambda\rangle$ and $|1\rangle|\lambda\rangle$ is closed under U and acts as a two-state system. The unitary U has the effect of a rotation on each subspace that corresponds to the eigenstate of $\mathcal{H}$ $|\lambda\rangle$:

$$U = \bigoplus_\lambda \begin{bmatrix} \lambda & i\sqrt{1-\lambda^2} \\ i\sqrt{1-\lambda^2} & \lambda \end{bmatrix} \otimes |\lambda\rangle\langle\lambda| \tag{135}$$

$$= \bigoplus_\lambda e^{iX\arccos(\lambda)} \otimes |\lambda\rangle\langle\lambda|, \tag{136}$$

where $e^{iX\arccos(\lambda)}$ is effectively a rotation by $2\arccos(\lambda)$ about the x-axis of the Bloch sphere defined by $|0\rangle|\lambda\rangle$ and $|1\rangle|\lambda\rangle$ for each λ. In each subspace spanned by $|0\rangle|\lambda\rangle$ and $|1\rangle|\lambda\rangle$, the unitary operator U acts as the *signal unitary* in QSP, encoding the eigenvalue λ as the signal.

By analogy with the unitary U that extends the idea of a signal unitary to a multi-qubit state, we extend the signal processing operator. The signal processing operator is independent of the signals λ and applies to the auxiliary qubit in the same way for each subspace spanned by $|0\rangle|\lambda\rangle$ and $|1\rangle|\lambda\rangle$. Using the identity operator $I = \sum_\lambda |\lambda\rangle\langle\lambda|$, we can express the extended signal processing operator as

$$\Pi_\phi = \bigoplus_\lambda e^{i\phi Z} \otimes |\lambda\rangle\langle\lambda|, \tag{137}$$

where Z is a Pauli operator and the rotation $e^{i\phi Z}$ acts on the auxiliary qubit.

Gilyén et al. (2018) demonstrate that the sequence of alternating extended signal unitaries and signal processing operators results in an embedded polynomial transformation of the Hermitian operator $\mathcal{H}$. For an even d:

$$U_{\vec{\phi}} = \Big[\prod_{k=1}^{d/2} \Pi_{\phi_{2k-1}} U^\dagger \Pi_{\phi_{2k}} U\Big] = \begin{bmatrix} Poly(\mathcal{H}) & \cdot \\ \cdot & \cdot \end{bmatrix}, \tag{138}$$

[27] Note that U is not a unique way to embed the Hermitian operator $\mathcal{H}$ in a unitary operator acting on a larger system.

and similarly for an odd d:

$$U_{\vec{\phi}} = \Pi_{\phi_1} U \Big[\prod_{k=1}^{(d-1)/2} \Pi_{\phi_{2k}} U^\dagger \Pi_{\phi_{2k+1}} U \Big] = \begin{bmatrix} Poly(\mathcal{H}) & \cdot \\ \cdot & \cdot \end{bmatrix}, \tag{139}$$

where, post-selected on the auxiliary qubit ending in state $|0\rangle$ (if it started in state $|0\rangle$) we obtain the desired polynomial transformation $Poly(\mathcal{H})$

$$Poly(\mathcal{H}) = \sum_\lambda Poly(\lambda) |\lambda\rangle\langle\lambda|. \tag{140}$$

The resulting polynomial of $\mathcal{H}$ is of order less than or equal to d. Its exact form depends on the sequence of signal processing rotation angles $\vec{\phi}$, which can be computed efficiently using a version of the classical Remez exchange algorithm (Low et al., 2016; Martyn et al., 2021).

9.3 Quantum Singular Value Transformation

In the previous section we considered the polynomial transformation of a Hermitian operator. For non-Hermitian linear operators, it is possible to perform an analogous polynomial singular value transformation.

Consider a Non-Hermitian operator A, represented by a rectangular matrix. Any general rectangular matrix can be decomposed into a diagonal matrix of singular values Σ and unitary matrices W_Σ and V_Σ:

$$A = W_\Sigma \Sigma V_\Sigma^\dagger. \tag{141}$$

The matrix Σ contains r non-negative real singular values of matrix A along the diagonal. The columns of matrices W_Σ and V_Σ form orthonormal bases composed of left and right singular vectors respectively. In quantum mechanical notation we denote the left singular vectors $\{|w_k\rangle\}$ and right singular vectors $\{|v_k\rangle\}$ and express the matrix A as

$$A = \sum_{k=1}^{r} \sigma_k |w_k\rangle\langle v_k|. \tag{142}$$

Using a quantum computer, we can efficiently perform polynomial singular value transformation of A:

$$Poly^{(SV)}(A) = \sum_k Poly(\sigma_k) |w_k\rangle\langle v_k|. \tag{143}$$

The matrix A can be embedded in a unitary matrix that applies to a larger quantum system similarly to the way we embedded the Hermitian matrix $\mathcal{H}$ in Section 9.2. Even though the Hilbert spaces spanned by $\{|w_k\rangle\}$ and $\{|v_k\rangle\}$ in general have different dimensions, we can create an extended quantum state by appending a single auxiliary qubit. In this case, the Hilbert spaces spanned by $\{|w_k\rangle\}$ and $\{|v_k\rangle\}$ would be encoded on the same register of n qubits, where n is large enough to hold the larger of the spaces.

The extended signal unitary U then takes the form

$$U = \bigoplus_k \begin{bmatrix} \sigma_k & i\sqrt{1-\sigma_k^2} \\ i\sqrt{1-\sigma_k^2} & \sigma_k \end{bmatrix} \otimes |w_k\rangle\langle v_k|. \tag{144}$$

When the input and output spaces of A are different, we have two signal processing operators

$$\Pi_\phi = \bigoplus_k e^{i\phi Z} \otimes |v_k\rangle\langle v_k| \tag{145}$$

$$\tilde{\Pi}_\phi = \bigoplus_k e^{i\phi Z} \otimes |w_k\rangle\langle w_k|. \tag{146}$$

With these definitions we have an expression analogous to the quantum eigenvalue transformation (Section 9.2), e.g. for even d:

$$U_{\vec{\phi}} = \Big[\prod_{k=1}^{d/2} \Pi_{\phi_{2k-1}} U^\dagger \tilde{\Pi}_{\phi_{2k}} U\Big] = \begin{bmatrix} Poly^{(SV)}(A) & \cdot \\ \cdot & \cdot \end{bmatrix}, \tag{147}$$

where the polynomial transformation $Poly^{(SV)}(A)$ is postselected on the auxiliary qubit in the state $|0\rangle$ after the unitary $U_{\vec{\phi}}$ is applied.

9.4 QSVT and the "Grand Unification" of Quantum Algorithms

Quantum Eigenvalue Transformation and its generalization, the Quantum Singular Value Transformation, are powerful and expressive ways to perform a wide range of non-unitary operations on a quantum computer. For some quantum problems, such as Hamiltonian simulation, QSVT provides the most efficient known algorithm to date, nearly optimal relative to known lower bounds.

Martyn et al. (2021) point out that the QSVT framework unifies all the existing quantum algorithms. After all, a quantum algorithm is a transformation of inputs – linear or non-linear, dimension-preserving or not. The QSVT framework provides a unified way to encode any matrix transformation that has a polynomial expansion, and the algorithm itself is encoded in a sequence of real numbers – the phases in the tuple $\vec{\phi}$.

Consider, for example, the search problem described in Section 4. The problem has two natural singular vectors: the starting state $|\psi_0\rangle$ and the target state $|x_0\rangle$. The objective is to transform the small inner product between the starting and ending state $c = \langle x_0|\psi_0\rangle$ into a scalar of order unity $\langle x_0|U|\psi_0\rangle = O(1)$. Yoder et al. (2014) demonstrate that a sequence of pulse sequences can be tuned to achieve this transformation efficiently, while avoiding the Grover algorithm's "soufflé problem" – the fact that the approximation error in the Grover algorithm is periodic and, if the minimum error is missed, it starts rising with every iteration until it peaks at $O(1)$ and starts decreasing again. The method proposed by Yoder et al. (2014), called fixed-point quantum search, generalizes reflections on a plane of Grover's algorithm to rotations on a three-dimensional Block sphere using the framework of QSP and QSVT. The algorithm converges in $O(\sqrt{N})$ steps and is optimal.

The Hamiltonian simulation problem is equivalent to matrix exponentiation. Given a Hamiltonian $\mathcal{H}$ we seek to apply $e^{-i\mathcal{H}t}$ which is equivalent to applying the sum $e^{-i\mathcal{H}t} = \cos(\mathcal{H}t) - i\sin(\mathcal{H}t)$. The trigonometric functions $\cos(\mathcal{H}t)$ and $\sin(\mathcal{H}t)$ have polynomial expansions, called Jacobi-Anger expansions, and therefore it is possible to cast the Hamiltonian simulation problem as a sum of two Quantum Eigenvalue Transformations.

QSVT provides an efficient way to solve quantum linear systems problem (Section 6.4) $A|x\rangle = |b\rangle$. In the original QLSA algorithm, Harrow et al. (2009) used Quantum Phase Estimation to extract singular values of A (or its eigenvalues, if A is Hermitian) and a controlled rotation to invert these values. The QSVT framework does not require the explicit extraction of the singular values of A. The singular value inversion is performed in a single step using a polynomial approximation to A^{-1}. Assuming the singular values of A are bounded, e.g. by $1/\kappa$ from below, it is possible to construct a polynomial expansion of A^{-1}, as shown in Martyn et al. (2021), Appendix C. The query complexity of the resulting algorithm is $O(\kappa \log \kappa/\epsilon)$; remarkably, it has no dependence on N, the dimension of A, and has a logarithmic dependence on error. (It is important to note, however, that the block encoding of A may require a number of operations that scale as a function of N.)

Martyn et al. (2021) demonstrate how to apply the QSVT framework to prime factorization, phase estimation, and eigenvalue thresholding; Gilyén et al. (2018) apply QSVT to Gibbs sampling, quantum walks, and the computation of machine learning primitives. A wide variety of quantum matrix functions and quantum channel discrimination methods are being developed using QSVT. Additionally, because QSVT is based on quantum control techniques which are also used in error correction, it is possible to create naturally robust algorithms using the QSVT frameworks. For example, embedding a matrix in a large unitary may enable algorithm designers to "push" errors out into the outer blocks of the matrix – similarly to the way deep networks distribute errors through unimportant dimensions of an overparameterized model (Bartlett et al., 2020).

The QSVT framework distills the great variety of quantum algorithms to a (finite) string or real numbers – the auxiliary rotation phases $\vec{\phi}$. All the algorithms follow the same circuit – composed of alternating block-encoded unitaries and auxiliary rotations which, depending on the sequence of auxiliary rotations, transform the singular values by a nearly arbitrary polynomial. The specific sequence of auxiliary rotations delivers the multitude of useful outcomes.

The QSVT framework is powerful, but, like other quantum algorithmic frameworks, it is not without challenges. It is not trivial to design efficient block encodings – even for the simplest systems like harmonic oscillators. Computationally efficient methods to determine the sequence of phases $\vec{\phi}$ corresponding to a given polynomial are also an open challenge. Nevertheless, QSVT is a significant advance towards practical, actionable quantum algorithms.

10 Quantum Machine Learning

Quantum machine learning is a very active area of quantum computing research. Classical machine learning has seen explosive growth over the last decade, with widely ranging applications in research and industry. However, the classical machine learning paradigm

is starting to mature as current capabilities struggle with vast datasets and decelerating Moore's law. Quantum computing has the potential to bring applications of machine learning to a new level. Results in quantum learning theory point to classes of problems where quantum computing can deliver significant advantages (see, e.g. Arunachalam and de Wolf, 2017), including polynomially faster learning rates. Additionally, evidence suggests that NISQs – the noisy quantum computers becoming available in the near term – may be able to deliver quantum advantage in machine learning over classical counterparts. We refer readers interested in a more detailed review of quantum machine learning to Ciliberto et al. (2018); Schuld and Petruccione (2018); Adcock et al. (2015); and the informal review by Dunjko and Wittek (2020).

The most direct application of quantum computers to machine learning is the development of quantum neural networks – artificial neural networks encoded in quantum states that are difficult to sample from classically (see, e.g. Low et al., 2014; Rebentrost et al., 2018a; Schuld and Killoran, 2019; Schuld et al., 2020; Abbas et al., 2020; Bausch, 2020; Park and Kastoryano, 2020, and references therein). These models are designed to benefit from quantum superposition and entanglement. A critical challenge in development of quantum neural networks is that quantum transformations are fundamentally linear, and non-linearity is seen as particularly important in successful classical neural networks. Schuld et al. (2016) demonstrated that the vast majority of early proposals for quantum neural networks did not meet the non-linearity requirements of artificial neural networks. But recent models are overcoming this disadvantage using, for example, quantum measurement (see e.g. Romero et al., 2017; Wan et al., 2017) or kernel functions (Farhi and Neven, 2018; Blank et al., 2020; Liu et al., 2021) to include non-linearity. Huang et al. (2021) develop a class of kernel models that can provide rigorously demonstrable speedup over classical models in the presence of noise – i.e. potentially achievable on NISQs (Section 2.4), the noisy near-term quantum computers.

Faster optimization is another way to leverage quantum computers in machine learning. Many machine learning methods use optimization techniques for parameter learning. Quantum Optimization (Section 8), which includes annealing (Kadowaki and Nishimori, 1998, 2021) and adiabatic methods, approximate optimization, and hybrid quantum-classical optimization, has the potential to deliver quadratic or polynomial speedups (Aaronson and Ambainis, 2009; McClean et al., 2021) for a variety of machine learning tasks. For example, Miyahara and Sughiyama (2018) perform mean-field VB via quantum annealing – an alternative to gradient descent optimization. Hybrid quantum-classical variational algorithms, which combine the strengths of quantum and classical computers, can speed up variational methods (see, e.g., Farhi and Neven (2018); McClean et al. (2018); Mitarai et al. (2018) and references therein). Hybrid approaches are particularly attractive in the near term, because they may help to harness the power of NISQs.

Bayesian computation has long called for scalable techniques. Markov chain Monte Carlo (MCMC) has been the main workforce in Bayesian statistics, but it is also well-known that MCMC can be too slow in many modern applications. Quantum Markov chains (Section 5.3) hold the promise for greatly speeding up MCMC (see, e.g. Szegedy, 2004; Chowdhury and Somma, 2017; Orsucci et al., 2018). Quantum computation has also been exploited to speed up Variational Bayes (Lopatnikova and Tran, 2021) - another popular technique for Bayesian computation.

When fault-tolerant quantum computers become available, machine learning methods

may benefit from fast quantum linear algebra. For example, the algorithms to solve systems of linear equations, such as the HHL algorithm and its updates (Section 6.4), enable fast matrix inversion used widely in machine learning models. Under certain conditions, such as when quantum access to data is provided and the matrices to be inverted are sparse or low-rank, quantum computers can deliver exponential speedup relative to classical computers. Direct applications include linear regression for data fitting (Wiebe et al., 2012) and prediction (Schuld et al., 2016), ridge (Yu et al., 2021) and logistic (Liu et al., 2019) regression. Lopatnikova and Tran (2021) use quantum matrix inversion to speed up the estimation of natural gradient for Variational Bayes (VB). Related algorithms such as the Quantum PCA algorithm (Section 6.6) and quantum singular value decomposition (Wiebe et al., 2012; Lloyd et al., 2013; Rebentrost et al., 2014; Lloyd et al., 2016; Cong and Duan, 2016; Kerenidis and Prakash, 2016; Childs et al., 2017; Rebentrost et al., 2018b; Wang et al., 2019; Kerenidis and Prakash, 2020) algorithms have also been influential. For example, Quantum PCA using parameterized quantum circuits can support face recognition tasks (Xin et al., 2021).[28]

Quantum machine learning and classical machine learning have benefited from crossover ideas. Classical machine learning algorithms have been proposed based on quantum structures (Gilyén et al., 2018; Tang, 2018; Chia et al., 2020). Similarly, classical machine learning ideas, such as kernel methods, helped better understand the nature of quantum neural networks and improve their design (Schuld and Killoran, 2019).

10.1 Quantum Gradient Descent

An important method in classical machine learning is gradient descent (and stochastic gradient descent) – a popular way for training statistical models by optimizing a loss function. When gradient descent is used for training quantum neural networks, it is often performed on a classical computer. The reason is that it is difficult to iterate on a quantum computer. The vast majority of quantum computing algorithms rely on a measurement (or a series of measurements) to produce the desired result. The measurement yields the desired quantum state with a fractional probability (often around 1/2); the rest of the time it yields an incorrect state that has to be discarded. Therefore, in order to have a sufficient number of desired quantum states available at the end of the iterative process, multiple copies of the initial state have to be created, with the number of copies increasing exponentially with the number of expected iterative steps.

For optimization where the number of expected iterations is small, quantum algorithms with provable speedups have been proposed. These algorithms work for specific simple forms of the loss function. How to extend these algorithms to alternative classes of loss functions remains an open area for future research. The most general quantum gradient descent work

[28] Gilyén et al. (2018); Tang (2018); Chia et al. (2020) have recently demonstrated that, if data are made available to classical computers in structures similar to those required for efficient quantum computations, then some of the quantum linear systems algorithms can be de-quantized – i.e. significant efficiencies can be obtained using randomized classical algorithms. These quantum-inspired algorithms can bring intriguing efficiencies to machine learning; however, currently these algorithms suffer from disadvantaged scaling in critical parameters, such as condition numbers and sparsity of matrices, which may render them impractical in the near term.

is by Rebentrost et al. (2019) who propose a quantum algorithm to find the minimum of a homogeneous polynomial using gradient descent and Newton's methods. The proposed quantum algorithm leverages the matrix exponentiation method used in Quantum PCA (Section 6.6), followed by Quantum Phase Estimation (Section 6.2) and a controlled rotation (Sections 2.3 and 6.3) of an auxiliary qubit to achieve each parameter update step. The shortcoming of this method is that it requires, on average, the destruction of approximately three sets of quantum states for each updating steps (this number can be reduced to around two with an optimized sequence of steps), and is therefore only appropriate for approximate minimization with a small number of iterations.

Sweke et al. (2020) consider stochastic gradient descent for hybrid quantum-classical optimization (Section 8.3). They argue that to obtain an unbiased estimator of an integral over a probability distribution, it is sufficient to prepare a corresponding quantum state and take a measurement, repeating the process k times, where k, in some cases, can be as low as 1. Where the gradient can be expressed as a linear combination of expectation values, a "doubly-stochastic" gradient descent is possible. The paper considers cases where the cost function can be expressed as an observable that can be readily measured, such as the energy (i.e. the expectation value of the Hamiltonian of the system) in Variational Quantum Eigensolver Peruzzo et al. (2014). Stokes et al. (2020) propose a quantum algorithm to estimate the natural gradient for cost functions expressed by a block-diagonal Hamiltonian in a space spanned by parameterized unitary gates.

11 Conclusion

We have provided a quick introduction to quantum computation and reviewed a range of quantum algorithms that can be of importance in statistics and machine learning.

The intersection of quantum computing and statistics has delivered – and is likely to continue to deliver – powerful breakthroughs that unlock interesting and practical applications. Consider, for example, advances in quantum MCMC, in particular quantum Metropolis-Hastings. Speeding up the Metropolis-Hastings algorithm is critical for many important practical applications but, because of its mathematical structure, the method does not generally apply to high-dimensional statistical models. For such cases, Sequential Monte Carlo (SMC) can provide an attractive alternative to MCMC. Quantum SMC is a challenging, but potentially highly impactful research question, which remains an open question at the time of writing.

In the classical world, Variational Bayes stands out as a computationally attractive alternative to MCMC for Bayesian computation in big model and big data settings. Lopatnikova and Tran (2021) propose a Variational Bayes method based on quantum natural gradient, which can be implemented on a quantum-classical device. How to implement a quantum Variational Bayes approach entirely on a quantum computer is an interesting research question.

Potential advances stem not just from applying quantum algorithms to machine learning, but also from borrowing insights the other way around – from statistics and machine learning to quantum computing. For example, as discussed in Section 3.2, reading out a quantum state that encodes the result of a quantum computation might require too many measurements offseting quantum efficiency. In the case where the result is a quantum sample state,

we can interpret it as a probability distribution. We can then adopt the idea of normalizing flow in machine learning (Papamakarios et al., 2021) to transform the quantum sample state into a new manageable quantum sample state, e.g., the uniform state. The transformation is implemented via parameterized quantum circuits with the parameters trained using classical routines. The method would allow us to create as many (approximate) copies of the original state as needed, via the inverse transformation, which can then be used in quantum tomography.

One important area that was not reviewed in this article is quantum-inspired computation, such as quantum-inspired linear algebra (Gilyén et al., 2018; Chia et al., 2020). Quantum-inspired algorithms work on classical computers, but are designed based on quantum-inspired ideas and can still offer significant speed-ups. We leave this topic for a future work.

References

Aaronson, S. (2013). *Quantum computing since Democritus*. Cambridge University Press.

Aaronson, S. (2015). Read the fine print. *Nature Physics*, 11(4):291–293.

Aaronson, S. (2019). Shadow tomography of quantum states. *SIAM Journal on Computing*, 49(5):STOC18–368.

Aaronson, S. and Ambainis, A. (2009). The need for structure in quantum speedups. *arXiv preprint arXiv:0911.0996*.

Abbas, A., Sutter, D., Zoufal, C., Lucchi, A., Figalli, A., and Woerner, S. (2020). The power of quantum neural networks. *arXiv preprint arXiv:2011.00027*.

Abrams, D. S. and Lloyd, S. (1998). Nonlinear quantum mechanics implies polynomial-time solution for np-complete and# p problems. *Physical Review Letters*, 81(18):3992.

Adcock, J., Allen, E., Day, M., Frick, S., Hinchliff, J., Johnson, M., Morley-Short, S., Pallister, S., Price, A., and Stanisic, S. (2015). Advances in quantum machine learning. *arXiv preprint arXiv:1512.02900*.

Aharonov, D., Kitaev, A., and Nisan, N. (1998). Quantum circuits with mixed states. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 20–30.

Aharonov, D. and Ta-Shma, A. (2003). Adiabatic quantum state generation and statistical zero knowledge. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 20–29.

Aharonov, D. and Ta-Shma, A. (2007). Adiabatic quantum state generation. *SIAM Journal on Computing*, 37(1):47–82.

Aharonov, D., Van Dam, W., Kempe, J., Landau, Z., Lloyd, S., and Regev, O. (2008). Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM review*, 50(4):755–787.

Albash, T. and Lidar, D. A. (2018). Adiabatic quantum computation. *Reviews of Modern Physics*, 90(1):015002.

Ambainis, A. (2012). Variable time amplitude amplification and quantum algorithms for linear algebra problems. In *STACS'12 (29th Symposium on Theoretical Aspects of Computer Science)*, volume 14, pages 636–647. LIPIcs.

An, D. and Lin, L. (2019). Quantum linear system solver based on time-optimal adiabatic quantum computing and quantum approximate optimization algorithm. *arXiv preprint arXiv:1909.05500*.

An, D., Linden, N., Liu, J.-P., Montanaro, A., Shao, C., and Wang, J. (2021). Quantum-accelerated multilevel Monte Carlo methods for stochastic differential equations in mathematical finance. *Quantum*, 5:481.

Apolloni, B., Carvalho, C., and De Falco, D. (1989). Quantum stochastic optimization. *Stochastic Processes and their Applications*, 33(2):233–244.

Arunachalam, S. and de Wolf, R. (2017). Guest column: A survey of quantum learning theory. *ACM SIGACT News*, 48(2):41–67.

Arunachalam, S., Gheorghiu, V., Jochym-O'Connor, T., Mosca, M., and Srinivasan, P. V. (2015). On the robustness of bucket brigade quantum RAM. *New Journal of Physics*, 17(12):123010.

Babbush, R., Love, P. J., and Aspuru-Guzik, A. (2014). Adiabatic quantum simulation of quantum chemistry. *Scientific reports*, 4(1):1–11.

Bartlett, P. L., Long, P. M., Lugosi, G., and Tsigler, A. (2020). Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117(48):30063–30070.

Baumgratz, T., Gross, D., Cramer, M., and Plenio, M. B. (2013). Scalable reconstruction of density matrices. *Physical review letters*, 111(2):020401.

Bausch, J. (2020). Recurrent quantum neural networks. *Advances in Neural Information Processing Systems*, 33.

Bennett, C. H. (1989). Time/space trade-offs for reversible computation. *SIAM Journal on Computing*, 18(4):766–776.

Berry, D. W., Ahokas, G., Cleve, R., and Sanders, B. C. (2007). Efficient quantum algorithms for simulating sparse Hamiltonians. *Communications in Mathematical Physics*, 270(2):359–371.

Berry, D. W. and Childs, A. M. (2012). Black-box Hamiltonian simulation and unitary implementation. *Quantum Information and Computation*, 12(1-2):29–62.

Berry, D. W., Childs, A. M., Cleve, R., Kothari, R., and Somma, R. D. (2014). Exponential improvement in precision for simulating sparse hamiltonians. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 283–292.

Berry, D. W., Childs, A. M., Cleve, R., Kothari, R., and Somma, R. D. (2015a). Simulating Hamiltonian dynamics with a truncated Taylor series. *Physical review letters*, 114(9):090502.

Berry, D. W., Childs, A. M., and Kothari, R. (2015b). Hamiltonian simulation with nearly optimal dependence on all parameters. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 792–809. IEEE.

Berry, D. W., Childs, A. M., Su, Y., Wang, X., and Wiebe, N. (2020). Time-dependent Hamiltonian simulation with $l1$-norm scaling. *Quantum*, 4:254.

Bharti, K., Cervera-Lierta, A., Kyaw, T. H., Haug, T., Alperin-Lea, S., Anand, A., Degroote, M., Heimonen, H., Kottmann, J. S., Menke, T., et al. (2021). Noisy intermediate-scale quantum (NISQ) algorithms. *arXiv preprint arXiv:2101.08448*.

Blank, C., Park, D. K., Rhee, J.-K. K., and Petruccione, F. (2020). Quantum classifier with tailored quantum kernel. *npj Quantum Information*, 6(1):1–7.

Born, M. and Fock, V. (1928). Beweis des adiabatensatzes. *Zeitschrift für Physik*, 51(3-4):165–180.

Boyer, M., Brassard, G., Høyer, P., and Tapp, A. (1998). Tight bounds on quantum searching. *Fortschritte der Physik: Progress of Physics*, 46(4-5):493–505.

Brassard, G., Dupuis, F., Gambs, S., and Tapp, A. (2011). An optimal quantum algorithm to approximate the mean and its application for approximating the median of a set of points over an arbitrary distance. *arXiv preprint arXiv:1106.4267*.

Brassard, G., Hoyer, P., Mosca, M., and Tapp, A. (2002). Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74.

Brassard, G., Høyer, P., and Tapp, A. (1998). Quantum counting. In *International Colloquium on Automata, Languages, and Programming*, pages 820–831. Springer.

Buhrman, H., Cleve, R., Watrous, J., and De Wolf, R. (2001). Quantum fingerprinting. *Physical Review Letters*, 87(16):167902.

Chakrabarti, S., Childs, A. M., Hung, S.-H., Li, T., Wang, C., and Wu, X. (2019). Quantum algorithm for estimating volumes of convex bodies. *arXiv preprint arXiv:1908.03903*.

Chakraborty, S., Gilyén, A., and Jeffery, S. (2018). The power of block-encoded matrix powers: improved regression techniques via faster Hamiltonian simulation. *arXiv preprint arXiv:1804.01973*.

Chia, N.-H., Gilyén, A., Li, T., Lin, H.-H., Tang, E., and Wang, C. (2020). Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 387–400.

Childs, A. M. (2009). Universal computation by quantum walk. *Physical review letters*, 102(18):180501.

Childs, A. M. (2010). On the relationship between continuous-and discrete-time quantum walk. *Communications in Mathematical Physics*, 294(2):581–603.

Childs, A. M., Cleve, R., Deotto, E., Farhi, E., Gutmann, S., and Spielman, D. A. (2003). Exponential algorithmic speedup by a quantum walk. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 59–68.

Childs, A. M. and Kothari, R. (2009). Limitations on the simulation of non-sparse Hamiltonians. *arXiv preprint arXiv:0908.4398*.

Childs, A. M., Kothari, R., and Somma, R. D. (2017). Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. *SIAM Journal on Computing*, 46(6):1920–1950.

Childs, A. M. and Wiebe, N. (2012). Hamiltonian simulation using linear combinations of unitary operations. *Quantum Information & Computation*, 12(11-12):901–924.

Cho, A. (2020). The biggest flipping challenge in quantum computing. *Science. https://www. sciencemag. org/news/2020/07/biggest-flipping-challenge-quantumcomputing (retrieved Aug. 3, 2020)*.

Choi, J., Oh, S., and Kim, J. (2020). Quantum approximation for wireless scheduling. *Applied Sciences*, 10(20):7116.

Choi, V. (2011). Different adiabatic quantum optimization algorithms for the NP-complete exact cover and 3SAT problems. *Quantum Information & Computation*, 11(7-8):638–648.

Choi, V. (2020). The effects of the problem hamiltonian parameters on the minimum spectral gap in adiabatic quantum optimization. *Quantum Information Processing*, 19(3):1–25.

Chowdhury, A. N. and Somma, R. D. (2017). Quantum algorithms for Gibbs sampling and hitting-time estimation. *Quantum Information & Computation*, 17(LA-UR-16-21218).

Ciliberto, C., Herbster, M., Ialongo, A. D., Pontil, M., Rocchetto, A., Severini, S., and Wossnig, L. (2018). Quantum machine learning: a classical perspective. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2209):20170551.

Clader, B. D., Jacobs, B. C., and Sprouse, C. R. (2013). Preconditioned quantum linear system algorithm. *Physical Review Letters*, 110(25):250504.

Cleve, R., Ekert, A., Macchiavello, C., and Mosca, M. (1998). Quantum algorithms revisited. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 454(1969):339–354.

Cong, I. and Duan, L. (2016). Quantum discriminant analysis for dimensionality reduction and classification. *New Journal of Physics*, 18(7):073011.

Cramer, M., Plenio, M. B., Flammia, S. T., Somma, R., Gross, D., Bartlett, S. D., Landon-Cardinal, O., Poulin, D., and Liu, Y.-K. (2010). Efficient quantum state tomography. *Nature communications*, 1(1):1–7.

Csanky, L. (1975). Fast parallel matrix inversion algorithms. In *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)*, pages 11–12. IEEE.

Denchev, V. S., Ding, N., Vishwanathan, S., and Neven, H. (2012). Robust classification with adiabatic quantum optimization. *arXiv preprint arXiv:1205.1148*.

Dervovic, D., Herbster, M., Mountney, P., Severini, S., Usher, N., and Wossnig, L. (2018). Quantum linear systems algorithms: a primer. *arXiv preprint arXiv:1802.08227*.

Draper, T. G. (2000). Addition on a quantum computer. *arXiv preprint quant-ph/0008033*.

Dunjko, V. and Wittek, P. (2020). A non-review of quantum machine learning: trends and explorations. *Quantum Views*, 4:32.

Durr, C. and Hoyer, P. (1996). A quantum algorithm for finding the minimum. *arXiv preprint quant-ph/9607014*.

Farhi, E., Goldstone, J., and Gutmann, S. (2014). A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*.

Farhi, E., Goldstone, J., Gutmann, S., Lapan, J., Lundgren, A., and Preda, D. (2001). A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem. *Science*, 292(5516):472–475.

Farhi, E., Goldstone, J., Gutmann, S., and Sipser, M. (2000). Quantum computation by adiabatic evolution. *arXiv preprint quant-ph/0001106*.

Farhi, E. and Harrow, A. W. (2016). Quantum supremacy through the quantum approximate optimization algorithm. *arXiv preprint arXiv:1602.07674*.

Farhi, E. and Neven, H. (2018). Classification with quantum neural networks on near term processors. *arXiv preprint arXiv:1802.06002*.

Fernández-Lorenzo, S., Porras, D., and Garc´ia-Ripoll, J. J. (2020). Hybrid quantum-classical optimization for financial index tracking. *arXiv preprint arXiv:2008.12050*.

Feynman, R. P. (1981). Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6/7).

Gilyén, A., Arunachalam, S., and Wiebe, N. (2019a). Optimizing quantum optimization algorithms via faster quantum gradient computation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1425–1444. SIAM.

Gilyén, A., Lloyd, S., and Tang, E. (2018). Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension. *arXiv preprint arXiv:1811.04909*.

Gilyén, A., Su, Y., Low, G. H., and Wiebe, N. (2019b). Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 193–204.

Giovannetti, V., Lloyd, S., and Maccone, L. (2008a). Architectures for a quantum random access memory. *Physical Review A*, 78(5):052310.

Giovannetti, V., Lloyd, S., and Maccone, L. (2008b). Quantum random access memory. *Physical Review Letters*, 100(16):160501.

Gross, D., Liu, Y.-K., Flammia, S. T., Becker, S., and Eisert, J. (2010). Quantum state tomography via compressed sensing. *Physical review letters*, 105(15):150401.

Grover, L. and Rudolph, T. (2002). Creating superpositions that correspond to efficiently integrable probability distributions. *arXiv preprint quant-ph/0208112*.

Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219.

Grover, L. K. (2000). Synthesis of quantum superpositions by quantum computation. *Physical review letters*, 85(6):1334.

Häner, T., Roetteler, M., and Svore, K. M. (2018). Optimizing quantum circuits for arithmetic. *arXiv preprint arXiv:1805.12445*.

Hann, C. T., Lee, G., Girvin, S., and Jiang, L. (2021). Resilience of quantum random access memory to generic noise. *PRX Quantum*, 2(2):020311.

Harrow, A. and Napp, J. (2019). Low-depth gradient measurements can improve convergence in variational hybrid quantum-classical algorithms. *arXiv preprint arXiv:1901.05374*.

Harrow, A. W. (2020). Small quantum computers and large classical data sets. *arXiv preprint arXiv:2004.00026*.

Harrow, A. W., Hassidim, A., and Lloyd, S. (2009). Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15):150502.

Heinrich, S. (2002). Quantum summation with an application to integration. *Journal of Complexity*, 18(1):1–50.

Herbert, S. (2021). The problem with Grover-Rudolph state preparation for quantum monte-carlo. *arXiv preprint arXiv:2101.02240*.

Huang, H.-Y., Broughton, M., Mohseni, M., Babbush, R., Boixo, S., Neven, H., and McClean, J. R. (2021). Power of data in quantum machine learning. *Nature communications*, 12(1):1–9.

Huang, H.-Y., Kueng, R., and Preskill, J. (2020). Predicting many properties of a quantum system from very few measurements. *Nature Physics*, 16(10):1050–1057.

Jordan, S. P. (2005). Fast quantum algorithm for numerical gradient estimation. *Physical Review Letters*, 95(5):050501.

Kadowaki, T. and Nishimori, H. (1998). Quantum annealing in the transverse ising model. *Physical Review E*, 58(5):5355.

Kadowaki, T. and Nishimori, H. (2021). Greedy parameter optimization for diabatic quantum annealing. *arXiv preprint arXiv:2111.13287*.

Kaneko, K., Miyamoto, K., Takeda, N., and Yoshino, K. (2021). Quantum speedup of Monte Carlo integration with respect to the number of dimensions and its application to finance. *Quantum Information Processing*, 20(5):1–24.

Kerenidis, I. and Prakash, A. (2016). Quantum recommendation systems. *arXiv preprint arXiv:1603.08675*.

Kerenidis, I. and Prakash, A. (2020). Quantum gradient descent for linear systems and least squares. *Physical Review A*, 101(2):022316.

Kitaev, A. Y., Shen, A., Vyalyi, M. N., and Vyalyi, M. N. (2002). *Classical and quantum computation*. Number 47. American Mathematical Soc.

Knill, E. (1995). Approximation by quantum circuits. *arXiv preprint quant-ph/9508006*.

Kyrillidis, A., Kalev, A., Park, D., Bhojanapalli, S., Caramanis, C., and Sanghavi, S. (2018). Provable compressed sensing quantum state tomography via non-convex methods. *npj Quantum Information*, 4(1):1–7.

Lanyon, B., Maier, C., Holzäpfel, M., Baumgratz, T., Hempel, C., Jurcevic, P., Dhand, I., Buyskikh, A., Daley, A., Cramer, M., et al. (2017). Efficient tomography of a quantum many-body system. *Nature Physics*, 13(12):1158–1162.

Liu, H.-L., Yu, C.-H., Wu, Y.-S., Pan, S.-J., Qin, S.-J., Gao, F., and Wen, Q.-Y. (2019). Quantum algorithm for logistic regression. *arXiv preprint arXiv:1906.03834*.

Liu, Y., Arunachalam, S., and Temme, K. (2021). A rigorous and robust quantum speed-up in supervised machine learning. *Nature Physics*, 17(9):1013–1017.

Lloyd, S. (1996). Universal quantum simulators. *Science*, pages 1073–1078.

Lloyd, S., Garnerone, S., and Zanardi, P. (2016). Quantum algorithms for topological and geometric analysis of data. *Nature communications*, 7(1):1–7.

Lloyd, S., Mohseni, M., and Rebentrost, P. (2013). Quantum algorithms for supervised and unsupervised machine learning. *arXiv preprint arXiv:1307.0411*.

Lloyd, S., Mohseni, M., and Rebentrost, P. (2014). Quantum principal component analysis. *Nature Physics*, 10(9):631–633.

Lomont, C. (2003). Fast inverse square root. *Tech-315 nical Report*, 32.

Lopatnikova, A. and Tran, M.-N. (2021). Quantum natural gradient for Variational Bayes. *arXiv preprint arXiv:2106.05807*.

Low, G. H. and Chuang, I. L. (2017). Optimal hamiltonian simulation by quantum signal processing. *Physical review letters*, 118(1):010501.

Low, G. H. and Chuang, I. L. (2019). Hamiltonian simulation by qubitization. *Quantum*, 3:163.

Low, G. H., Yoder, T. J., and Chuang, I. L. (2014). Quantum inference on Bayesian networks. *Physical Review A*, 89(6):062315.

Low, G. H., Yoder, T. J., and Chuang, I. L. (2016). Methodology of resonant equiangular composite quantum gates. *Physical Review X*, 6(4):041067.

Magniez, F., Nayak, A., Roland, J., and Santha, M. (2011). Search via quantum walk. *SIAM journal on computing*, 40(1):142–164.

Marriott, C. and Watrous, J. (2005). Quantum arthur–merlin games. *computational complexity*, 14(2):122–152.

Marsh, S. and Wang, J. B. (2020). Combinatorial optimization via highly efficient quantum walks. *Physical Review Research*, 2(2):023302.

Martyn, J. M., Rossi, Z. M., Tan, A. K., and Chuang, I. L. (2021). A grand unification of quantum algorithms. *arXiv preprint arXiv:2105.02859*.

McArdle, S., Jones, T., Endo, S., Li, Y., Benjamin, S. C., and Yuan, X. (2019). Variational ansatz-based quantum simulation of imaginary time evolution. *npj Quantum Information*, 5(1):1–6.

McClean, J. R., Boixo, S., Smelyanskiy, V. N., Babbush, R., and Neven, H. (2018). Barren plateaus in quantum neural network training landscapes. *Nature communications*, 9(1):1–6.

McClean, J. R., Harrigan, M. P., Mohseni, M., Rubin, N. C., Jiang, Z., Boixo, S., Smelyanskiy, V. N., Babbush, R., and Neven, H. (2021). Low-depth mechanisms for quantum optimization. *PRX Quantum*, 2(3):030312.

McClean, J. R., Romero, J., Babbush, R., and Aspuru-Guzik, A. (2016). The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18(2):023023.

Mitarai, K., Negoro, M., Kitagawa, M., and Fujii, K. (2018). Quantum circuit learning. *Physical Review A*, 98(3):032309.

Miyahara, H. and Sughiyama, Y. (2018). Quantum extension of variational Bayes inference. *Physical Review A*, 98(2):022330.

Montanaro, A. (2015). Quantum speedup of Monte Carlo methods. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2181):20150301.

Moroder, T., Hyllus, P., Tóth, G., Schwemmer, C., Niggebaum, A., Gaile, S., Gühne, O., and Weinfurter, H. (2012). Permutationally invariant state reconstruction. *New Journal of Physics*, 14(10):105001.

Nayak, A. and Wu, F. (1999). The quantum query complexity of approximating the median and related statistics. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 384–393.

Neven, H., Denchev, V. S., Rose, G., and Macready, W. G. (2008). Training a binary classifier with the quantum adiabatic algorithm. *arXiv preprint arXiv:0811.0416*.

Nielsen, M. A. and Chuang, I. (2002). Quantum computation and quantum information.

O'Donnell, R. and Wright, J. (2016). Efficient quantum tomography. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 899–912.

Orsucci, D., Briegel, H. J., and Dunjko, V. (2018). Faster quantum mixing for slowly evolving sequences of Markov chains. *Quantum*, 2:105.

Papamakarios, G., Nalisnick, E., Rezende, D., Mohamed, S., and Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64.

Paparo, G. D., Dunjko, V., Makmal, A., Martin-Delgado, M. A., and Briegel, H. J. (2014). Quantum speedup for active learning agents. *Physical Review X*, 4(3):031002.

Park, C.-Y. and Kastoryano, M. J. (2020). Geometry of learning neural quantum states. *Physical Review Research*, 2(2):023232.

Peruzzo, A., McClean, J., Shadbolt, P., Yung, M.-H., Zhou, X.-Q., Love, P. J., Aspuru-Guzik, A., and O'brien, J. L. (2014). A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5:4213.

Plesch, M. and Brukner, V. (2011). Quantum-state preparation with universal gate decompositions. *Physical Review A*, 83(3):032302.

Potok, T. et al. (2021). Adiabatic quantum linear regression. *Scientific Reports*, 11(1):1–10.

Prakash, A. (2014). *Quantum algorithms for linear algebra and machine learning*. University of California, Berkeley.

Preskill, J. (2018). Quantum computing in the NISQ era and beyond. *Quantum*, 2:79.

Rebentrost, P., Bromley, T. R., Weedbrook, C., and Lloyd, S. (2018a). Quantum Hopfield neural network. *Physical Review A*, 98(4):042308.

Rebentrost, P. and Lloyd, S. (2018). Quantum computational finance: quantum algorithm for portfolio optimization. *arXiv preprint arXiv:1811.03975*.

Rebentrost, P., Mohseni, M., and Lloyd, S. (2014). Quantum support vector machine for big data classification. *Physical Review Letters*, 113(13):130503.

Rebentrost, P., Schuld, M., Wossnig, L., Petruccione, F., and Lloyd, S. (2019). Quantum gradient descent and Newton's method for constrained polynomial optimization. *New Journal of Physics*, 21(7):073023.

Rebentrost, P., Steffens, A., Marvian, I., and Lloyd, S. (2018b). Quantum singular-value decomposition of nonsparse low-rank matrices. *Physical review A*, 97(1):012327.

Reichardt, B. W. (2004). The quantum adiabatic optimization algorithm and local minima. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 502–510.

Romero, J., Olson, J. P., and Aspuru-Guzik, A. (2017). Quantum autoencoders for efficient compression of quantum data. *Quantum Science and Technology*, 2(4):045001.

Saad, Y. (2003). *Iterative methods for sparse linear systems*. SIAM.

Sanders, Y. R., Low, G. H., Scherer, A., and Berry, D. W. (2019). Black-box quantum state preparation without arithmetic. *Physical review letters*, 122(2):020502.

Schuld, M., Bocharov, A., Svore, K. M., and Wiebe, N. (2020). Circuit-centric quantum classifiers. *Physical Review A*, 101(3):032308.

Schuld, M. and Killoran, N. (2019). Quantum machine learning in feature Hilbert spaces. *Physical Review Letters*, 122(4):040504.

Schuld, M. and Petruccione, F. (2018). *Supervised learning with quantum computers*. Springer.

Schuld, M., Sinayskiy, I., and Petruccione, F. (2016). Prediction by linear regression on a quantum computer. *Physical Review A*, 94(2):022342.

Seddiqi, H. and Humble, T. S. (2014). Adiabatic quantum optimization for associative memory recall. *Frontiers in Physics*, 2:79.

Shor, P. W. (1994). Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee.

Somma, R. D., Boixo, S., Barnum, H., and Knill, E. (2008). Quantum simulations of classical annealing processes. *Physical review letters*, 101(13):130504.

Steiger, D. S. and Troyer, M. (2016). Racing in parallel: quantum versus classical. In *APS March Meeting Abstracts*, volume 2016, pages H44–010.

Stokes, J., Izaac, J., Killoran, N., and Carleo, G. (2020). Quantum natural gradient. *Quantum*, 4:269.

Subaşi, Y., Somma, R. D., and Orsucci, D. (2019). Quantum algorithms for systems of linear equations inspired by adiabatic quantum computing. *Physical review letters*, 122(6):060504.

Subramanian, S., Brierley, S., and Jozsa, R. (2019). Implementing smooth functions of a hermitian matrix on a quantum computer. *Journal of Physics Communications*, 3(6):065002.

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

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

Sweke, R., Wilde, F., Meyer, J. J., Schuld, M., Fährmann, P. K., Meynard-Piganeau, B., and Eisert, J. (2020). Stochastic gradient descent for hybrid quantum-classical optimization. *Quantum*, 4:314.

Szegedy, M. (2004). Quantum speed-up of Markov chain based algorithms. In *45th Annual IEEE symposium on foundations of computer science*, pages 32–41. IEEE.

Tang, E. (2018). Quantum-inspired classical algorithms for principal component analysis and supervised clustering. *arXiv preprint arXiv:1811.00414*.

Tikhonov, A. N. (1963). On the solution of ill-posed problems and the method of regularization. In *Doklady Akademii Nauk*, volume 151, pages 501–504. Russian Academy of Sciences.

Torlai, G., Mazzola, G., Carrasquilla, J., Troyer, M., Melko, R., and Carleo, G. (2018). Neural-network quantum state tomography. *Nature Physics*, 14(5):447–450.

Torlai, G. and Melko, R. G. (2018). Latent space purification via neural density operators. *Physical review letters*, 120(24):240503.

Tóth, G., Wieczorek, W., Gross, D., Krischek, R., Schwemmer, C., and Weinfurter, H. (2010). Permutationally invariant quantum tomography. *Physical review letters*, 105(25):250403.

Vazquez, A. C. and Woerner, S. (2021). Efficient state preparation for quantum amplitude estimation. *Physical Review Applied*, 15(3):034027.

Vedral, V., Barenco, A., and Ekert, A. (1996). Quantum networks for elementary arithmetic operations. *Physical Review A*, 54(1):147.

Veis, L. and Pittner, J. (2014). Adiabatic state preparation study of methylene. *The Journal of chemical physics*, 140(21):214111.

Venegas-Andraca, S. E. (2012). Quantum walks: a comprehensive review. *Quantum Information Processing*, 11(5):1015–1106.

Ventura, D. and Martinez, T. (2000). Quantum associative memory. *Inf. Sci.*, 124:273–296.

Wan, K. H., Dahlsten, O., Kristjánsson, H., Gardner, R., and Kim, M. (2017). Quantum generalisation of feedforward neural networks. *npj Quantum information*, 3(1):1–8.

Wang, C. and Wossnig, L. (2018). A quantum algorithm for simulating non-sparse hamiltonians. *arXiv preprint arXiv:1803.08273*.

Wang, D., Higgott, O., and Brierley, S. (2019). Accelerated variational quantum eigensolver. *Physical Review Letters*, 122(14):140504.

Watrous, J. (2001). Quantum simulations of classical random walks and undirected graph connectivity. *Journal of computer and system sciences*, 62(2):376–391.

Wiebe, N., Braun, D., and Lloyd, S. (2012). Quantum algorithm for data fitting. *Physical Review Letters*, 109(5):050505.

Wocjan, P. and Abeyesinghe, A. (2008). Speedup via quantum sampling. *Physical Review A*, 78(4):042336.

Wocjan, P., Chiang, C.-F., Nagaj, D., and Abeyesinghe, A. (2009). Quantum algorithm for approximating partition functions. *Physical Review A*, 80(2):022340.

Woerner, S. and Egger, D. J. (2019). Quantum risk analysis. *npj Quantum Information*, 5(1):1–8.

Wossnig, L., Zhao, Z., and Prakash, A. (2018). Quantum linear system algorithm for dense matrices. *Physical review letters*, 120(5):050502.

Xin, T., Che, L., Xi, C., Singh, A., Nie, X., Li, J., Dong, Y., and Lu, D. (2021). Experimental quantum principal component analysis via parametrized quantum circuits. *Physical Review Letters*, 126(11):110502.

Yoder, T. J., Low, G. H., and Chuang, I. L. (2014). Fixed-point quantum search with an optimal number of queries. *Physical review letters*, 113(21):210501.

Yu, C.-H., Gao, F., and Wen, Q. (2021). An improved quantum algorithm for ridge regression. *IEEE Transactions on Knowledge and Data Engineering*, 33(3):858.

Zhang, K., Hsieh, M.-H., Liu, L., and Tao, D. (2021). Quantum gram-schmidt processes and their application to efficient state readout for quantum algorithms. *Physical Review Research*, 3(4):043095.

Zhou, L., Wang, S.-T., Choi, S., Pichler, H., and Lukin, M. D. (2020). Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Physical Review X*, 10(2):021067.

Zoufal, C., Lucchi, A., and Woerner, S. (2019). Quantum generative adversarial networks for learning and loading random distributions. *npj Quantum Information*, 5(1):1–9.

Zoufal, C., Lucchi, A., and Woerner, S. (2020). Variational quantum Boltzmann machines. *arXiv preprint arXiv:2006.06004*.