A High Performance Computing Infrastructure for the Efficient Execution of Hybrid Quantum-Classical Algorithms

Thesis submitted by

Steven Oud

under the guidance of

Harold Meerwaldt, QuTech
Damian Podareanu, SURF
Richard Versluis, TNO

in partial fulfillment of the requirements for the degree of

Bachelor of Science

A High Performance Computing Infrastructure for the Efficient Execution of Hybrid Quantum-Classical Algorithms

Steven Oud*
500776959

Faculty of Computer Science, Information Technology, Business IT and Management
Software Engineering

Advisor: Marten Teitsma

Amsterdam University of Applied Sciences
March 16, 2021

*Tel.: +31621451016

Abstract

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Contents

List of Figures

List of Tables

1

Introduction

As an introduction to the work that was done during this thesis, this chapter starts with presenting a brief overview on the relevance of quantum computing, and describing the motivation behind this work. After that, the objective of this work is described and an outline of this report is given.

1.1 Motivation

Quantum computing is a recently proposed model of computation that promises to solve certain problems more efficiently than classical computers by making use of quantum mechanical phenomena such as superposition and interference. The idea of quantum computing originated from Benioff [1], who proposed a quantum mechanical model of the Turing machine in 1980. This idea was later extended as Manin [2] and Feynman [3] independently suggested that quantum computers have the potential to solve certain computational problems intractable by classical computers. Since then, researchers have been searching for applications for quantum computing. Some noteworthy developments in the field of quantum computing include Shor's algorithm for factoring integers [4] and Grover's algorithm for unstructured database search [5]. These quantum algorithms promise an exponential and quadratic speedup respectively over their best-known classical counterparts. The finding of such speedups have catalyzed research towards quantum computers, and more applications have since been found in fields including chemistry [6], cryptography [7], and machine learning [8].

Until recently, quantum computing had been a mainly theoretical field. These days however, thanks to recent technological advances, various quantum devices are being actively developed. Furthermore, technological giants such as IBM, Microsoft, Intel, and Google are investing heavily in the development of quantum computers. The applications of these current quantum devices are very limited however, due to their inability to store information for a long time and their sensitivity to errors. For instance, Fowler, Mariantoni, Martinis, *et al.* [9, Appendix M] estimate that to factor a 2000-bit number using Shor's algorithm would require a quantum computer with about a billion physical qubits, and error rates below 4×10^{-13}. In contrast, devices today have about 50 to 100 physical qubits with error rates above 0.1.

These small and error-prone quantum devices, referred to as noisy intermediate--scale quantum (NISQ) devices, may be unfit to run quantum algorithms like Shor's

algorithm, but they may still prove to be useful and perform tasks intractable by classical computers [10]. To deal with the limitations of NISQ devices, hybrid quantum-classical algorithms (HQCAs) are being actively researched. These HQCAs combine classical and quantum computations as visualized in Figure 1.1. Hybrid quantum-classical algorithms

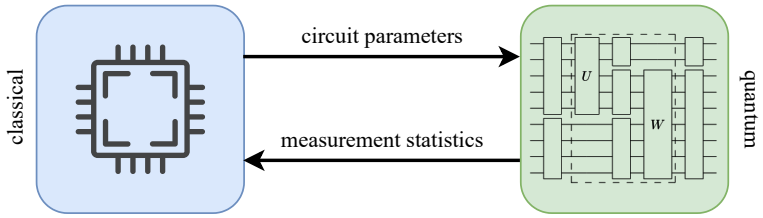


Figure 1.1: General structure of a HQCA. The data interchange between the classical and quantum part is repeated many times.

typically involve a small quantum computation inside of a classical optimization loop, greatly reducing the amount of quantum resources needed. This makes them suitable to run on NISQ devices, and are expected to be one of the first useful applications for quantum computing [11]. It is important to note that NISQ devices running HQCAs will likely not be revolutionary by itself. Instead, it should be seen as an important stepping stone towards more powerful quantum devices and algorithms.

Research towards HQCAs often involves executing quantum circuits on actual quantum chips or through quantum circuit simulation. To support this kind of research, classical and quantum computing facilities are needed. Furthermore, these facilities need to be connected and able to interchange data in a timely manner for this kind of research to be feasible. This is especially challenging given that both quantum and classical resources are shared with other users.

1.2 Objective

The purpose of this thesis is to propose an infrastructure for the efficient execution of hybrid quantum-classical algorithms. This work is in collaboration with TNO, QuTech, and SURF, and is focused on the efficient execution of HQCAs using QuTech's quantum computing platform Quantum Inspire [12] and SURF's high performance computing (HPC) center [13]. To allow for the efficient execution of HQCAs, the SURF HPC and Quantum Inspire computing facilities should be able to interchange data in a timely manner. A key part in this is figuring out sources of overhead and current bottlenecks. In a hybrid setup like this, overheads such as quantum and classic scheduling wait times, data transfers, and resource initialization can quickly increase the run time of HQCAs.

1.3 Outline

This report is structured as follows. Chapter 2 provides the reader with a background on computational complexity, quantum mechanics, and quantum computation. Chapter 3 takes a deeper look into HQCAs and their working. Chapter 4 demonstrates the practical execution of HQCAs using Quantum Inspire's quantum computing platform and SURF's HPC center.

2

Background

This chapter is focused on making the reader familiar with concepts used throughout this report. First, an introduction to computational complexity is given to establish a mathematical framework to describe the efficiency of computer algorithms. Second, the basic ideas of quantum mechanics are presented. Finally, an overview of the quantum circuit computational model is given along with a description of two important quantum algorithms to demonstrate the computing power of quantum computers.

2.1 Computational Complexity

In computer science, there seems to be a fundamental limit to what problems computers can solve. Some problems seem to be inherently uncomputable: there exists no general solution that does not go into an infinite loop for certain inputs [14, 15]. On the other hand, there are limitations to the efficiency in which problems can be solved. This section will look at how one can define the computational efficiency of certain algorithms. That is, how many resources are required to solve a certain problem?

2.1.1 Big-O Notation

The time and space required by an algorithm generally grows as the size of the input grows. Because of this, it is traditional to describe the efficiency of an algorithm as a function of the size of its input [16]. This function describes the number of primitive operations it performs for a given input size. The notion of input size here depends on the context of the problem. For example, when computing the discrete Fourier transform, the input size likely refers to the dimension of the input vector. When talking about a problem like integer multiplication, however, it is more fitting to talk about the input size as the number of bits needed to represent the input in binary.

When analyzing the efficiency of algorithms, we look at the asymptotic growth for a given input size. Consider an algorithm that given input size n takes n^2 primitive operations to run and another algorithm that takes $500n^2 + \log n$ primitive operations to run. In big-O notation, both these algorithms are said to run in $O(n^2)$ time. That is, the number of primitive operations it performs scales quadratically with the input size. Constant factors are ignored as they become negligible as $n \to \infty$. While they are practically significant — an algorithm that runs in $O(n/2)$ performs half as many

primitive operations as an algorithm that runs in $O(n)$ — they are not relevant to asymptotic analysis, as both functions grow linearly with n.

Formally, if we have functions $f(n)$ and $g(n)$ such that f eventually grows slower than some multiple of g as $n \to \infty$, we say $f(n) = O(g(n))$. For example, given $f(n) = 200n^2$ and $g(n) = n^3$, f begins to grow slower than g when $n > 200$. Thus, g bounds f from above, and $f(n) = O(g(n)) = O(n^3)$. Some common big-O run times are shown in Table 2.1, along with their written name and an example. Throughout this report, algorithms that are bounded above by a polynomial (i.e. all run times until polynomial in Table 2.1) will be referred to as polynomial-time algorithms, and algorithms that are not bounded above by a polynomial will be referred to as superpolynomial-time algorithms.

Notation	Name	Example
$O(1)$	Constant	Accessing single element from array
$O(\log n)$	Logarithmic	Binary search
$O(n)$	Linear	Unstructured database search
$O(n \log n)$	Linearithmic	Fast Fourier Transform
$O(n^2)$	Quadratic	Insertion sort
$O(n^k)$	Polynomial	Gaussian elimination
$O(k^n)$	Exponential	Graph coloring
$O(n!)$	Factorial	Brute-force search traveling salesman problem

Table 2.1: Common big-O run times from fast to slow.

2.1.2 Turing Machines

The previous section described the measurement of computational efficiency as the number of primitive operations it performs for a given input size. This abstract definition can be extended by choosing a computational model in order to define what a primitive operation means. The standard computational model used for this is the Turing machine. It is chosen as computational model for the analysis of computational efficiency because of its simplicity and because it is able to simulate most physically realizable computational models with little overhead [17].

A Turing machine is an abstract machine that manipulates symbols from a work alphabet on a finite amount of one-way infinite length tapes divided into cells [15] (Figure 2.1). Along these tapes runs a tape head that can read and write one symbol at a time. The machine has a finite set of states, which the machine executes one at a time by loading them into the state register. At any time, the machine can be in one of the finite states. A state can be thought of as a rule with the following form:

$$(q_i, a) \mapsto (q_j, b, H), \tag{2.1}$$

where q_i and q_j are states, a and b are symbols from the work alphabet, and $H \in \{L, S, R\}$ decides how to move the tape head: one cell to the left (L), stay in the same position (S), or one cell to the right (R). These states as described in Equation 2.1 can be read as "in state q_i, if the read symbol is a, go to state q_j, write symbol b, and move the tape head to H."

Everything that can be computed on models of computations we use these days can be computed on a Turing machine [19]. This hypothesis is known as the Church-Turing

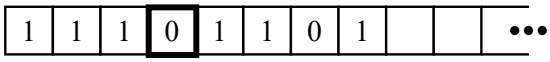


Figure 2.1: The tape of a single-tape Turing machine in an arbitrary state. Note that any multi-tape Turing machines can be efficiently simulated by a single-tape Turing machine [18], so complexity classes are not affected by changing between single-tape and multi-tape machines.

thesis. Related to the Church-Turing thesis is the extended Church-Turing thesis, which states that any physically realizable model of computation can be efficiently simulated on a Turing machine. That is, can a Turing machine simulate any model of computation in polynomial time? The quantum computational model brings doubt to this claim. It is known that quantum computers can efficiently simulate a Turing machine [20], but there appears to be no efficient algorithm for simulating a quantum computer on a Turing machine [21]. Furthermore, Arute, Arya, Babbush, *et al.* [22] experimentally demonstrated a quantum computer sampling from a probability distribution intractable by a classical computer. This gives us good reasons to believe that classical computers cannot efficiently simulate quantum computers, and that quantum computers are more computationally powerful for some problems.

2.1.3 Complexity Classes

Complexity classes are sets of computational problems that share some common feature with regard to the computational resources they need to solve some problem [17]. They are defined in terms of a type of computational problem, computational model, and a bounded resource such as time or space. In general, most complexity classes describe decision problems solvable by deterministic Turing machines — though many complexity classes are defined in terms of other types of problems and computational models. This report mainly focuses on complexity classes involving Turing machines and quantum Turing machines.

The class P contains all decision problems solvable by a deterministic Turing machine in polynomial time. Problems that fall under this class are often referred to as tractable or easy problems [16]. The class NP (non-deterministic polynomial) contains all problems *verifiable* by a deterministic Turing machine in polynomial time. Equivalently, NP can be thought of as all problems solvable in polynomial time by a non-deterministic Turing machine. A non-deterministic Turing machine is a variant of a Turing machine which is not entirely determined by its input and transition function, but can choose from a set of possible transitions when transitioning. One could then define NP as consisting of two phases: first, a non-deterministic Turing machine makes a guess about the solution, and then a second, deterministic Turing machine verifies if the guess is correct. It is clear that $P \subseteq NP$, because if you can solve a problem in polynomial time, you can also verify it in polynomial time. A still unsolved and important question in computer science is whether $P = NP$? That is, can all problems that can be verified in polynomial time also be solved in polynomial time?

A problem is said to be NP-hard when everything in NP can be reduced into it in polynomial time, even if it is outside of NP. In term, if a problem is both NP-hard and in NP, the problem is NP-complete. NP-complete problems are considered to be the hardest problems in NP. By definition of NP-hardness, if a polynomial-time algorithm can be found for any NP-complete algorithm, then all problems in NP can be solved in polynomial time and $P = NP$.

In computer science, it is sometimes possible to speed up computation using randomness. These kinds of algorithms are referred to as probabilistic algorithms and are defined in terms of a probabilistic Turing machine. A probabilistic Turing machine is a non-deterministic Turing machine that can choose from a set of possible transitions according to some probability distribution when transitioning. The probabilistic equivalent of P is BPP (bounded-error probabilistic polynomial time) and contains all decision problems solvable by a probabilistic Turing machine in polynomial time where a bounded error rate of 1/3 is allowed. Since a non-deterministic Turing machine can efficiently simulate a deterministic Turing machine, P $\subseteq$ BPP. There are problems to be known in BPP and not in P, but the number of such problems is decreasing, and Goldreich [23] and Nisan and Wigderson [24] even argue that P = BPP.

How do quantum computers relate to these complexity classes? Quantum computers are probabilistic computational devices, and its complexity class equivalent to P can be defined by replacing the probabilistic Turing machine from BPP with a quantum computer.[1] The class BQP (bounded-error quantum polynomial time) consists of all decision problems solvable by a quantum computer in polynomial time where a bounded error rate of 1/3 is allowed. It is known that there are NP problems that can be efficiently solved on a quantum computer like integer factorization, discrete logarithms, and quantum many-body simulation. As mentioned in the previous section, quantum computers can also solve all problems in P efficiently, so P $\subseteq$ BQP. Furthermore, quantum computers are more powerful than classical probabilistic computers [25], giving BPP $\subseteq$ BQP. How BQP relates to P and NP exactly is still unknown, however, it seems unlikely that BQP = NP [26]. The quantum analog of NP goes by the name of QMA (Quantum Merlin Arthur), and is related to BQP in the same way that NP is related to P. Similarly then, a problem is said to be QMA-hard when every problem in QMA can be reduced to it, and a problem is said to be QMA-complete if it is both QMA-hard and in QMA. Just like P and NP, it is clear that BQP $\subseteq$ QMA, and P $\subseteq$ NP $\subseteq$ QMA.

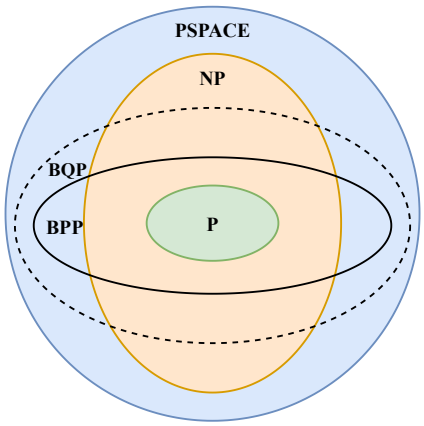


Figure 2.2: An overview of the hierarchy of some of the complexity classes discussed. This graphic assumes P $\neq$ NP, P $\neq$ BPP, and P $\subseteq$ BPP $\subseteq$ BQP. PSPACE is the space equivalent of P, containing all problems that can be solved in polynomial space by a deterministic Turing machine.

[1] Note that quantum computers are not simply probabilistic Turing machines as will be shown in the following sections.

2.2 Quantum Mechanics

Quantum mechanics is the most complete description of the physical properties of nature on the atomic scale. It is also at the core of quantum computation and information. This section describes the necessary background of quantum mechanics required for understanding quantum computing.

2.2.1 Qubits

In classic information theory, the smallest unit of information is the bit. Quantum information is built upon an analogous concept: the quantum bit, or qubit. Qubits are physical objects that appear in nature on the scale of atoms and subatomic particles. A qubit can be any two-state quantum-mechanical system such as the spin of an electron, which can be spin up or down, or the polarization of a photon, which can be horizontally of vertically polarized. In this report, qubits will be treated as abstract mathematical objects as the physical realization of qubits is beyond the scope of this work. The state of a qubit is denoted as follows:

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle. \tag{2.2}$$

Quantum states are often described using Dirac notation $|\cdot\rangle$, which describes a column vector in $\mathbb{C}^{2^n}$. The states $\{|0\rangle, |1\rangle\}$ are the computational basis states which are defined as $(1 \quad 0)^T$ and $(0 \quad 1)^T$ respectively, and form an orthonormal basis for this vector space. The values $\alpha, \beta \in \mathbb{C}$ are the state's probability amplitudes, and cannot be examined directly.[2] When measuring a qubit, it collapses probabilistically to one of the basis states. The probability of measuring 0 is given by the absolute square $|\alpha_0|^2$, and the probability of measuring 1 is given by $|\alpha_1|^2$. As these values are probabilities, they should be normalized: $|\alpha_0|^2 + |\alpha_1|^2 = 1$. Formally, a qubit can be thought of as a unit vector in a two-dimensional Hilbert space.

A qubit differs from a classical bit in that it can be in a linear combination, or superposition of states. While a bit can be only be in the state 0 or 1, a qubit can be in one of infinitely many superpositions of states. However, the laws of quantum mechanics restrict direct access to the probability amplitudes of a state. Instead, when measuring a qubit, it collapses to basis state $|j\rangle$ with probability $|\alpha_j|^2$. For example, consider the state

$$|+\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right). \tag{2.3}$$

This state has equal probability of measuring 0 and 1, as $|1/\sqrt{2}|^2 = 1/2$. Note that measurement changes the state of a qubit: if the state from Equation 2.3 is measured as 1, the superposition is lost and the state becomes $|1\rangle$.

A helpful geometric interpretation of a qubit's state can be obtained by rewriting Equation 2.2 as

$$|\psi\rangle = e^{i\delta}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle\right), \tag{2.4}$$

where $\delta, \theta, \varphi \in \mathbb{R}$. The global phase $e^{i\delta}$ can often be ignored, as $\forall\delta \in \mathbb{R} : |e^{i\delta}| = 1$, so it does not impact measurement outcome. Simplifying then, the state of a qubit can be

[2] The field of quantum tomography focuses on recovering these values through multiple measurements, but this requires prior knowledge about the system [27].

written as

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle. \tag{2.5}$$

Here, θ and φ define a point on the surface of a three-dimensional sphere referred to as the Bloch sphere (Figure 2.3). While this visualization is limited to a single qubit, it can

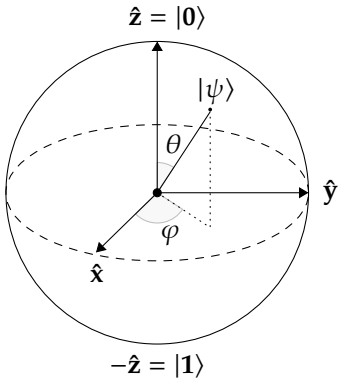


Figure 2.3: Bloch sphere representation of a qubit's state.

be a useful visual to build intuition. For example, the $|+\rangle$ state described in Equation 2.3 can be thought of as being exactly between $|0\rangle$ and $|1\rangle$ on the Bloch sphere.

The amount of probability amplitudes grows exponentially with the number of qubits: a n-qubit state has $N = 2^n$ amplitudes. Consider a two-qubit system which lives in a $2^2 = 4$-dimensional Hilbert space spanned by the computational basis states $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$. This state is defined by the linear combination

$$|\psi\rangle = \alpha_0|00\rangle + \alpha_1|01\rangle + \alpha_2|10\rangle + \alpha_3|11\rangle. \tag{2.6}$$

Again, unlike classical bits who can only be in one state at a time, this state can be in a superposition of all four states. The normalization condition still applies for Equation 2.6: $\sum_{i=0} |\alpha_i|^2 = 1$. Single-qubit states can be combined to form multi-qubit states by taking the tensor product of the two states. Given states $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ and $|\varphi\rangle = \beta_0|0\rangle + \beta_1|1\rangle$:

$$\begin{aligned}
|\psi\rangle \otimes |\varphi\rangle &= \begin{pmatrix}\alpha_0\\ \alpha_1\end{pmatrix} \otimes \begin{pmatrix}\beta_0\\ \beta_1\end{pmatrix} \\
&= \begin{pmatrix}\alpha_0\begin{pmatrix}\beta_0\\ \beta_1\end{pmatrix} \\ \alpha_1\begin{pmatrix}\beta_0\\ \beta_1\end{pmatrix}\end{pmatrix} \\
&= \begin{pmatrix}\alpha_0\beta_0\\ \alpha_0\beta_1\\ \alpha_1\beta_0\\ \alpha_1\beta_1\end{pmatrix} \\
&= \alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|10\rangle + \alpha_1\beta_1|11\rangle.
\end{aligned} \tag{2.7}$$

Often when describing multi-qubit states, the tensor product will be implied. The notations

$$|0\rangle \otimes |0\rangle = |0\rangle|0\rangle = |00\rangle \tag{2.8}$$

all describe state. The relative phases of α_0, α_1 and β_0, β_1 in Equation 2.7 are responsible for the quantum mechanical property of interference. When the phase of α_j and β_k is the same, they will interfere constructively and increase the probability amplitude for that state. On the other hand, if α_j and β_k have opposite phases, they will interfere destructively and decrease the probability amplitude for that state.

Not all multi-qubit systems can be expressed as a tensor product of individual states as shown in Equation 2.7. Consider the following state:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}} \left(|00\rangle + |11\rangle\right). \tag{2.9}$$

This state cannot be expressed as a tensor product of two individual states, as that would imply $(\alpha_0\beta_0 = \alpha_1\beta_1 = 1/\sqrt{2}) \wedge (\alpha_0\beta_1 = \alpha_1\beta_0 = 0)$, which is a contradiction. States like $|\Phi^+\rangle$ are referred to as entangled states. Entanglement is the quantum phenomena of correlation in measurement outcomes. For example, when measuring the state $|\Phi^+\rangle$ from Equation 2.9, the only two possible measurement outcomes are 00 and 11. So by measuring one qubit, one also knows the state of the other qubit.

2.2.2 State Evolution

The evolution of a closed quantum system is described by a unitary transformation [28, Section 2.2].[3] A state $|\psi\rangle$ at time t_1 is related to state $|\psi'\rangle$ at time t_2 by a unitary operator U:

$$|\psi'\rangle = U|\psi\rangle. \tag{2.10}$$

The unitary nature of these operators implies $UU^\dagger = U^\dagger U = I$, where † is the conjugate transpose and I the identity matrix. Single-qubit operators can be represented as 2×2 complex-valued unitary matrices. A common single-qubit operator is the Pauli-X operator which transforms a state $\alpha_0|0\rangle + \alpha_1|1\rangle$ to $\alpha_1|0\rangle + \alpha_0|1\rangle$. It is part of the set of Pauli matrices:

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{2.11}$$

These matrices are ubiquitous in the study of quantum computation and information. Another useful and common operator is the Hadamard operator:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \tag{2.12}$$

which maps the computational basis states to an equal superposition state.

The Pauli X, Y and Z operators give rise to the rotation operators about the $\hat{x}$, $\hat{y}$, and $\hat{z}$ axes when exponentiated:

$$R_x(\theta) = e^{-i\frac{\theta}{2}X} = \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}, \tag{2.13}$$

$$R_y(\theta) = e^{-i\frac{\theta}{2}Y} = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}, \tag{2.14}$$

$$R_z(\theta) = e^{-i\frac{\theta}{2}Z} = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}. \tag{2.15}$$

[3] In this report, a perfectly closed system is assumed, even though in reality all systems interact somewhat with other systems.

These operators can be thought of as rotating a qubit's state among its relative axis by an angle θ. Note that any single-qubit operator U can be decomposed into these operators, for example $U = R_z(\gamma)R_y(\beta)R_z(\alpha)$ where $\alpha, \beta, \gamma \in \mathbb{R}$.

Multi-qubit operators act on two or more qubits and are required for creating entangled states. Two common two-qubit operators are the CNOT and CZ operators. The CNOT operator can be thought of as a controlled-X operator, which applies a Pauli-X operation on the target qubit if the control qubit is in the $|1\rangle$ state. Equivalently, the CZ operator applies a Pauli-Z operation on the target qubit if the control qubit is $|1\rangle$. Their matrix representations are as follows:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad CZ = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}. \tag{2.16}$$

Generally, a controlled-U operator applies U on the target qubit if the control qubit is $|1\rangle$.

2.2.3 Measurement

While the evolution of quantum states in a closed system is unitary, at some point the quantum state has to interact with the outside world. This is what measurement means: any interaction from an outside system with the quantum system. Formally, measurement is defined by a set of measurement operators $\{M_m\}$, where the probability of measuring the state m is given by

$$p(m) = \langle\psi|M_m^\dagger M_m|\psi\rangle. \tag{2.17}$$

With $\langle\psi| = |\psi\rangle^\dagger$, this equation can then be read as the inner product between $M_m|\psi\rangle$ and itself. This is just a generalization of the definition of measurement given in Section 2.2.1. For example, what is the probability of measuring 0 for an arbitrary state $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$? Using $M_0 = |0\rangle\langle 0|$:

$$\begin{aligned} p(0) &= \langle\psi|M_0^\dagger M_0|\psi\rangle \\ &= \langle\psi|0\rangle\langle 0|0\rangle\langle 0|\psi\rangle \\ &= \langle\psi|0\rangle\langle 0|\psi\rangle \\ &= \begin{pmatrix} \alpha_0^* & \alpha_1^* \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} \\ &= \alpha_0^*\alpha_0 \\ &= |\alpha_0|^2. \end{aligned} \tag{2.18}$$

Similarly, using $M_1 = |1\rangle\langle 1|$ gives $p(1) = |\alpha_1|^2$. Measuring with the measurement operators $\{|0\rangle\langle 0|, |1\rangle\langle 1|\}$ is referred to as measuring in the computational basis.

After measurement, the state of the system can be described as follows:

$$|\psi\rangle = \frac{M_m|\psi\rangle}{\sqrt{p(m)}}. \tag{2.19}$$

Following the example from Equation 2.18 with $M_0 = |0\rangle\langle 0|$ and $p(0) = |\alpha_0|^2$, after measuring 0 the system is in the state

$$\begin{aligned} |\psi\rangle &= \frac{|0\rangle \langle 0|\psi\rangle}{\sqrt{|\alpha_0|^2}} \\ &= \frac{|0\rangle\alpha_0}{|\alpha_0|} \\ &= \frac{\alpha_0}{|\alpha_0|}|0\rangle. \end{aligned} \tag{2.20}$$

Note that a global phase $e^{i\delta}$ shows up as the factor $\alpha_0/|\alpha_0|$. As mentioned in Section 2.2.1, the states $e^{i\delta}|0\rangle$ and $|0\rangle$ are considered equal up to the global phase factor.

A special class of measurements called projective measurements can sometimes be used to simplify calculations. A projective measurement is defined by an observable O, which is a Hermitian operator on the state space of the system. This observable has a spectral decomposition

$$O = \sum_{\lambda} \lambda P_{\lambda}, \tag{2.21}$$

where P_λ is the projector onto the eigenspace of O with eigenvalue λ. For example, the Pauli-Z operator can be thought of as an observable with the spectral decomposition

$$Z = 1\,|0\rangle\langle 0| - 1\,|1\rangle\langle 1|\,, \tag{2.22}$$

which has eigenvectors $\{|0\rangle, |1\rangle\}$ with respective eigenvalues $\{1, -1\}$. As the observable Z has the computational basis states as eigenvectors, a measurement of Z can also be thought of a measuring in the computational basis. With this definition of projective measurements, the expectation value of an observable O for state $|\psi\rangle$ can be defined as

$$\begin{aligned} \langle O\rangle_{\psi} &= \langle\psi|O|\psi\rangle \\ &= \langle\psi|\left(\sum_{i=0}^{N-1} \lambda_i\,|\lambda_i\rangle\langle\lambda_i|\right)|\psi\rangle \\ &= \sum_{i=0}^{N-1} \lambda_i \langle\psi|\lambda_i\rangle \langle\lambda_i|\psi\rangle \\ &= \sum_{i=0}^{N-1} \lambda_i \left|\langle\lambda_i|\psi\rangle\right|^2. \end{aligned} \tag{2.23}$$

The expectation value is the sum of all possible outcomes (eigenvalues of O) weighted by their probability. Calculating the expectation value experimentally means preparing and measuring the state multiple times and is a useful way of extracting a deterministic quantity from a non-deterministic model of computation.

2.3 Quantum Computation

This section combines the fields of computer science and quantum mechanics to introduce the fundamental model of quantum computation: the quantum circuit model.

2.3.1 Quantum Circuits

Analogous to the classical circuit model, the quantum circuit model uses gates which act on data. In the classical circuit model boolean functions (logic gates) act on bits, while in the quantum circuit model quantum gates act on qubits. Most of the common quantum gates used in quantum computing were already described as operators in Section 2.2.2. To be more in line with the nomenclature of classical computing, from here on out these operators will be referred to as quantum gates in the context of quantum circuits.

In a quantum circuit time moves from left to right, where qubits are represented by wires on which gates can act, and classical bits are represented by double-lined wires. Usually the qubits are assumed to be instantiated to $|0\rangle$, unless noted otherwise. A list of frequently used quantum gates is shown in Table 2.2. Quantum gates are unitary and thus reversible, making the quantum circuit model a reversible model of computation. The inverse of a gate U is denoted as the conjugate transpose $U^\dagger$. By the definition of unitary $UU^\dagger = U^\dagger U = I$, applying the inverse $U^\dagger$ after U essentially uncomputes U and vice versa. Gates whose conjugate transpose are equal to themselves are referred to as Hermitian. For example, H is Hermitian as $H^\dagger = H$ and thus $H^2 = I$.

Figure 2.4 demonstrates a simple quantum circuit which creates the maximally entangled state $|\Phi^+\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$ from Equation 2.9 and measures both qubits. This state is one of four maximally entangled two-qubit states referred to as a Bell state. This circuit does the following. The system starts in state $|\psi\rangle = |00\rangle$. A Hadamard gate

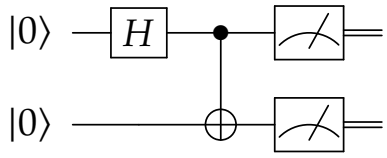


Figure 2.4: Quantum circuit for creating and measuring a Bell state $|\Phi^+\rangle$.

is applied on the first qubit:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|0\rangle \tag{2.24}$$

$$= \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle). \tag{2.25}$$

Then, a CNOT is applied with the first qubit as control and the second qubit as target, giving the final state

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle). \tag{2.26}$$

This state is then measured, which will measure 00 or 11 with equal probability.

In classical computations, copying bits is a common operation. However, in quantum computing the data one can copy is much more restricted. If a qubit is in a computational basic state, that state can be copied by a CNOT gate (Figure 2.5). However, trying to copy an arbitrary state $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ using the circuit in Figure 2.5 gives

$$|\psi\rangle = \alpha_0|00\rangle + \alpha_1|11\rangle. \tag{2.27}$$

This state does not contain two copies of $|\psi\rangle$ (unless $\alpha_0\alpha_1 = 0$ as is true with computational basis states), which should have the form

$$|\psi\rangle|\psi\rangle = \alpha_0^2|00\rangle + \alpha_0\alpha_1|01\rangle + \alpha_1\alpha_0|10\rangle + \alpha_1^2|11\rangle. \tag{2.28}$$

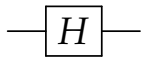
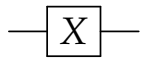
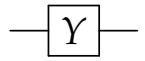
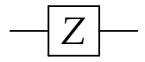
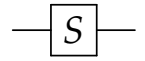
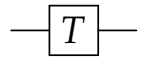
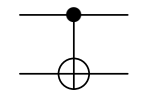
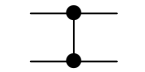
Gate name	Circuit symbol	Matrix representation
Hadamard		$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$
Pauli-X		$\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$
Pauli-Y		$\begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}$
Pauli-Z		$\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}$
Phase (S)		$\begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix}$
T		$\begin{pmatrix}1 & 0\\ 0 & e^{i\pi/4}\end{pmatrix}$
CNOT		$\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix}$
CZ		$\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & -1\end{pmatrix}$

Table 2.2: Frequently used quantum gates with their circuit symbol and matrix representation.

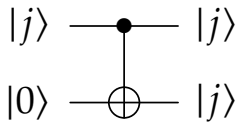


Figure 2.5: Copying a computational basis state using CNOT where $j \in \{0, 1\}$.

In fact, is it impossible to copy an unknown quantum state. There exists no solution for a unitary operator U that does the transformation

$$\alpha_0|00\rangle + \alpha_1|01\rangle \xrightarrow{U} \alpha_0^2|00\rangle + \alpha_0\alpha_1|01\rangle + \alpha_1\alpha_0|10\rangle + \alpha_1^2|11\rangle. \tag{2.29}$$

This property is known as the no-cloning theorem, and is a fundamental limit of quantum information. Note that this holds for unknown quantum states. If one has the circuit to prepare $|\psi\rangle$, you could simply create $|\psi\rangle|\psi\rangle$ by executing the circuit on a second qubit.

2.3.2 Universal Gate Sets

A universal gate set is defined as a finite set of gates that can be used to represent any other gate. In the classical circuit model, NAND is a universal gate with which all other gates can be represented. Equivalently, universal quantum gate sets are sets of quantum

gates to which any quantum operation can be reduced to. A common universal gate set is {CNOT, H, S, T}. Any quantum operation can be reduced to a combination of CNOT, H, S and T gates. For example, the three-qubit equivalent of the CNOT gate, the Toffoli gate, can be decomposed into these gates as shown in Figure 2.6.

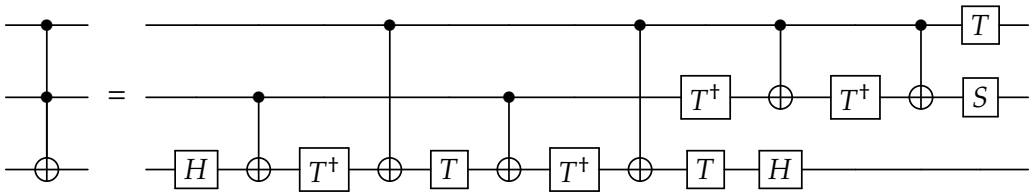


Figure 2.6: Decomposition of the Toffoli gate using CNOT, H, S and T gates.

2.3.3 Quantum Algorithms

To demonstrate the quantum circuit computational model's ability to solve certain problems more efficiently than classical computers, two important quantum algorithms which provide a speedup over their classical counterparts are now reviewed: the quantum Fourier transform (QFT) and Grover's algorithm.

Quantum Fourier Transform

The quantum Fourier transform (QFT) is the quantum analogue of the *inverse* discrete Fourier transform (DFT). The DFT is ubiquitous in the fields of engineering and computer science. In short, it transforms a signal in the time domain to the frequency domain, and vice versa with the inverse DFT. For $N = 2^n$ amplitudes, the QFT uses $O(n^2)$ gates, where n is the amount of qubits. The best-known classical algorithm for computing the DFT uses $O(n2^n)$ gates. That is, the QFT is exponentially more efficient than the best-known classical algorithm. However, the applications of the QFT are limited. The result of the QFT are stored in the amplitudes of the quantum state. These amplitudes cannot be directly accessed by measurement, so there is no way to extract the results. On the other hand, the QFT is the backbone of the phase estimation algorithm, which in term can be used to solve problems including the order-finding problem and the factoring problem efficiently.

The inverse DFT is a linear transformation on an input vector of N complex values $(x, \ldots, x_N)$. It transforms a element x_k as follows:

$$x'_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{\frac{2\pi i}{N} jk}. \tag{2.30}$$

Equivalently, the QFT is defined as a unitary operator which acts on a computational basis state $|j\rangle$ as following:

$$|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i}{N} jk} |k\rangle. \tag{2.31}$$

To describe the quantum circuit for the QFT, the following definition is useful:

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^k}} \end{pmatrix}. \tag{2.32}$$

Some useful identities are $R_1 = Z$, $R_2 = S$, and $R_3 = T$. Using this gate, the general quantum circuit for the QFT is shown in Figure 2.7. The gates at the end of the circuit are swap gates, which swaps two qubits. This is necessary to get the result in the correct order.

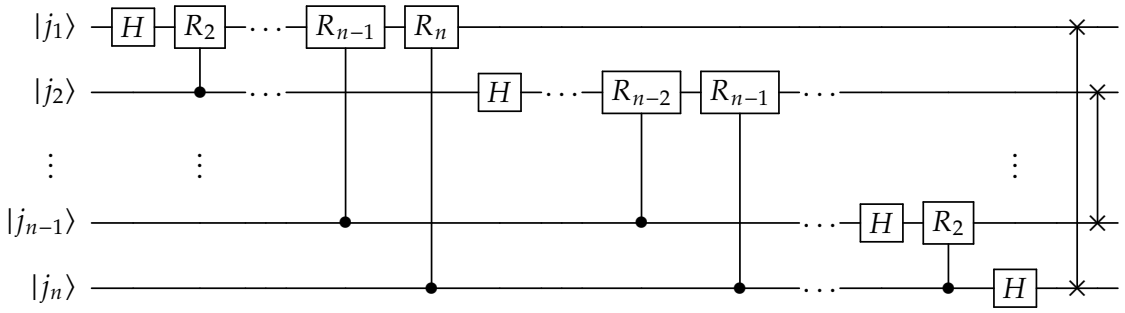


Figure 2.7: General circuit for the QFT on 2^n amplitudes with n qubits.

The QFT circuit works as follows. With $N = 2^n$ where n is the number of qubits, the computational basis exists of the states $\{|0\rangle, \ldots, |N-1\rangle\}$. We use the notation $|j\rangle$ for a decimal number j in the binary representation $j_1, \ldots, j_n$, e.g. $|15\rangle = |j_1 j_2 j_3 j_4\rangle = |1111\rangle$. Furthermore, the following notation is used to represent fractional binary numbers:

$$[0.j_1 \ldots j_m] = \sum_{k=1}^{m} j_k 2^{-k}. \tag{2.33}$$

For example, binary [0.101] is $1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} = 5/8$. Using these definitions, the QFT can be represented as follows:

$$\text{QFT}|j_1 \ldots j_n\rangle = \frac{1}{\sqrt{N}} \left(|0\rangle + e^{2\pi i[0.j_n]}|1\rangle\right) \left(|0\rangle + e^{2\pi i[0.j_{n-1}j_n]}|1\rangle\right) \cdots \left(|0\rangle + e^{2\pi i[0.j_1 j_2 \ldots j_n]}|1\rangle\right). \tag{2.34}$$

The transformed values end up in the relative phases of the amplitudes. Again, these relative phases cannot be extracted. However, the QFT is an essential part of important quantum algorithms: it is as the heart of the quantum phase estimation algorithm (QPEA), which can be used to factor integers [4], solve linear systems of equations [29], and more [28, Section 5.4].

Grover's Algorithm

Grover's algorithm is a quantum algorithm for unstructured database search. Suppose you want to search through a search space of n numbers. Given a function $f(x)$ that returns 1 for the number ω that you are looking for, and 0 for every other number:

$$f(x) = \begin{cases} 1 \text{ if } x = \omega \\ 0 \text{ if } x \neq \omega \end{cases} \tag{2.35}$$

The goal is to find a x so that $f(x) = 1$. On a classical computer, the best we can do is an exhaustive search of the entire search space, which takes $O(n)$ function calls. Using a quantum computer, this can be done with $O(\sqrt{n})$ function calls using Grover's algorithm [5]. Grover's algorithm is also asymptotically optimal: Zalka [30] showed that $O(\sqrt{n})$ is the best we can do for unstructured database search.

A common problem in computer science is examining a large number of different possibilities to see which of them satisfy a given condition. This is the same problem that Grover's algorithm looks to solve. Usually, there is some structure in the problem which allows for more efficient algorithms. For example, if a list of numbers is sorted, one can find a number in $O(\log n)$ using binary search. For hard problems that do not have such obvious structure like the boolean satisfiability problem, Grover's algorithm can provide a quadratic speedup.

The function $f(x)$ from Equation 2.35 is sometimes referred to as a black box function, as its internal workings is not of interest. Grover's algorithm is a general algorithm which can be applied to many algorithms that use search heuristics. The quantum equivalent of a black box function is called an oracle, and has the following form:

$$U_\omega|x\rangle = \begin{cases} -|x\rangle \text{ if } x = \omega \\ \ \ |x\rangle \text{ if } x \neq \omega \end{cases} \tag{2.36}$$

where $|x\rangle$ is any computational basis state, and ω is the bit string to find. That is, it marks the solution to the search problem by shifting the phase of the solution. An oracle from Equation 2.36 can be defined in terms of a black box function:

$$U_\omega|x\rangle = (-1)^{f(x)}|x\rangle, \tag{2.37}$$

which has the following matrix representation:

$$U_\omega = \begin{pmatrix} (-1)^{f(0)} & 0 & \cdots & 0 \\ 0 & (-1)^{f(1)} & \cdots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & (-1)^{f(2^n-1)} \end{pmatrix}. \tag{2.38}$$

With the oracle defined, the quantum circuit of Grover's algorithm is shown in Figure 2.8. The quantum state can be thought of as having two registers: the target qubits

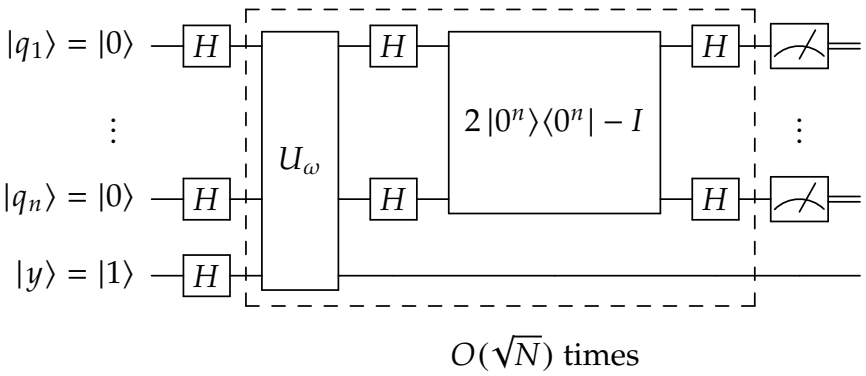
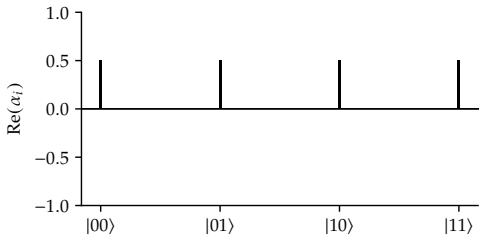


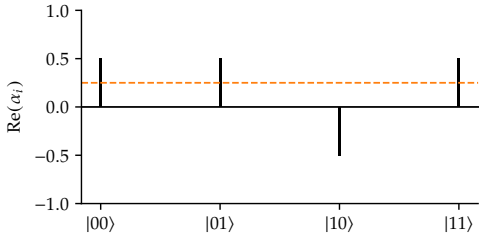
Figure 2.8: Quantum circuit representation of Grover's algorithm.

$|q1 \ldots q_n\rangle$ and the oracle qubit $|y\rangle$. The oracle qubit $|y\rangle$ starts in the state $H|1\rangle = (|0\rangle -$

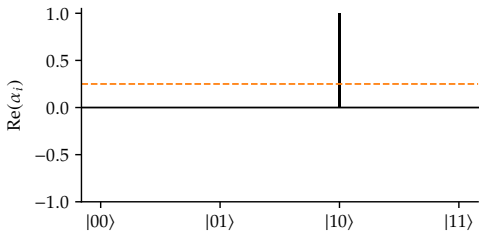
$|1\rangle)/\sqrt{2}$. When the oracle is applied to $|q1 \ldots q_n\rangle(|0\rangle - |1\rangle)/\sqrt{2}$ and x is not a solution, the state stays the same. However, if the oracle is applied to $|q1 \ldots q_n\rangle(|0\rangle - |1\rangle)/\sqrt{2}$ and x *is* a solution, the state becomes $-|q1 \ldots q_n\rangle(|0\rangle - |1\rangle)/\sqrt{2}$. After the oracle is applied, the solution is marked with a negative phase. To extract this solution, a procedure called amplitude amplification is used. Amplitude amplification can be thought of as inversion about the mean. Consider a search space of $N = 4$ with two qubits where we are looking for $|\omega\rangle = |10\rangle$. The amplitudes of the search space start out in an equal superposition:



After applying the oracle U_ω the solution state $|10\rangle$ has its phase flipped:



Note that measuring this state will still give each state with equal probabilities. Using the diffusion operator $H^{\otimes n}\,(2\,|0^n\rangle\langle 0^n| - I)\,H^{\otimes n}$, the amplitudes are inverted about their mean (displayed as the dotted orange line):



The application of the oracle and amplitude amplification is often referred to as a Grover iteration. For this example, one Grover iteration was enough to give a probability of 1 of measuring the correct solution. Generally, the upper bound on the number of required iterations R is as follows:

$$R \leq \left\lceil \frac{\pi}{4}\sqrt{\frac{M}{N}} \right\rceil, \tag{2.39}$$

where M is the number of solutions and $N = 2^n$ is the size of the search space.

3

Hybrid Quantum-Classical Algorithms

At the end of the previous chapter, two important quantum algorithms were reviewed: the QFT and Grover's algorithm. These quantum algorithms and the family of algorithms built on them require large-scale quantum computers with low error rates and high coherence. The NISQ devices that are being built now and in the near future will not be able to run these algorithms due to their limited qubit count, limited coherence times, and high error rates. To this end, hybrid quantum-classical algorithms (HQCAs) that utilize both classical and quantum resources are being researched and developed. This chapter will first introduce the basic concepts of an important family of HQCAs followed by a review of well-known and promising HQCAs.

3.1 Variational Hybrid Quantum-Classical Algorithms

Hybrid quantum-classical algorithms take into account the limited number of qubits, limited connectivity of qubits, high error rates, and limited coherence times of NISQ devices. These algorithms often make use of the variational method which consists of preparing an initial trial state $|\psi(\vec{\theta})\rangle$ parameterized by real-valued parameters $\vec{\theta}$, and finding the parameters for which the expectation value of some observable is the lowest. This family of HQCAs is sometimes also referred to as variational quantum algorithms. The rest of this chapter will be focused on variational HQCAs, as this is the main focus of research regarding HQCAs.

The general structure of a variational HQCA is shown in Figure 3.1. While the specific implementation differs between algorithms, they all share some basic elements. First, a cost function C which encodes the solution to the problem needs to be defined. Second, an ansatz needs to be chosen. The ansatz defines the operations of the quantum circuit which are parameterized by the trainable parameters $\vec{\theta}$. The parameters $\vec{\theta}$ are then trained in a quantum-classical loop to solve the optimization problem

$$\arg\min_{\vec{\theta}} C(\vec{\theta}). \tag{3.1}$$

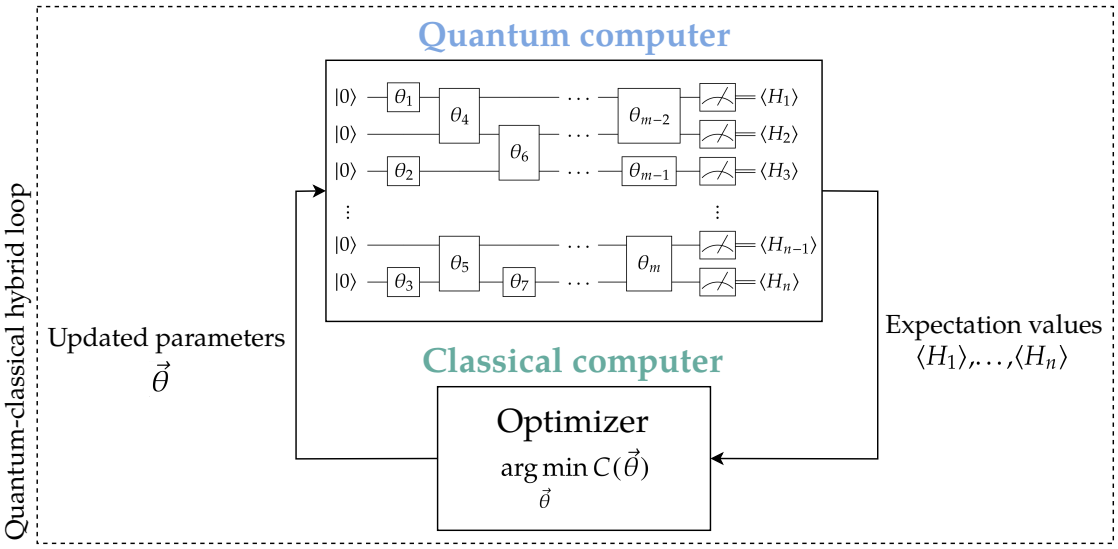


Figure 3.1: The general structure of a variational HQCA. A parameterized quantum state $|\psi(\vec{\theta})\rangle$ is prepared through a set of parameterized quantum gates $U_L(\vec{\theta}_L)\ldots U_2(\vec{\theta}_2)U_1(\vec{\theta}_1)$. These parameters are classically optimized according to a cost function $C(\vec{\theta})$. This quantum-classical loop is repeated until convergence.

3.1.1 Cost Function

In general, the cost function has the form

$$C(\vec{\theta}) = \sum_k f_k \left(\langle 0^n|U(\vec{\theta})^\dagger H_k U(\vec{\theta})|0^n\rangle\right), \tag{3.2}$$

where f_k is a function encoding the problem at hand, $U(\vec{\theta})$ is the quantum circuit parameterized by real values $\vec{\theta}$, H_k is an observable, and n is the number of qubits. A key point of variational HQCAs is that a quantum computer is used to estimate $C(\vec{\theta})$ while using a classical computer to optimize the parameters $\vec{\theta}$. Furthermore, the cost function $C(\vec{\theta})$ should be efficient to estimate by performing measurements on a quantum computer. For the variational HQCA to have a quantum advantage, the cost function should also not be efficiently computable with a classical computer.

3.1.2 Ansätze

The ansatz defines the quantum circuit $U(\vec{\theta})$ that is used to prepare the quantum state $|\psi(\vec{\theta})\rangle$ parameterized by real-valued parameters $\vec{\theta}$. To keep the algorithm realizable on NISQ devices, it is important to keep the parameterized quantum circuit (the ansatz) relatively shallow in depth. The challenge then is to choose an ansatz that is shallow in depth but has high expressibility. Sim, Johnson, and Aspuru-Guzik [31] define the expressibility of a quantum circuit as its ability to generate states that are well representative of the Hilbert space. The general form of an ansatz can be described as the product of sequentially applied unitaries:

$$U(\vec{\theta}) = U_L(\vec{\theta}_L)\ldots U_2(\vec{\theta}_2)U_1(\vec{\theta}_1), \tag{3.3}$$

where

$$U_l(\vec{\theta}_l) = \prod_m e^{-i\frac{\theta_m}{2}H_m} W_m. \tag{3.4}$$

Here H_m is a Hermitian operator and W_m is a non-parameterized unitary. Equation 3.4 can simply be thought of as alternating between parameterized and non-parameterized unitaries.

The specific structure of the ansatz usually depends on the problem, as one can sometimes employ information about the problem to tailor an ansatz [32]. Such ansätze are referred to as problem-inspired ansätze. On the other hand, there are problem-agnostic ansätze that are used when no information is available for tailoring a specific ansatz.

3.1.3 Optimization

With the cost function and ansatz defined, the next step in a variational HQCA is to train the parameters $\vec{\theta}$ using a classical optimization algorithm to solve the optimization problem from Equation 3.1. The two main types of optimization methods to consider are gradient-based methods and gradient-free methods. The former methods involve calculating first- or higher-order derivatives of the cost function, while the latter methods use only function evaluations. In general, the computational cost of gradient-based methods is larger, but the rate of convergence is significantly improved.

While gradient-free methods are straightforward to use for optimizing the quantum circuit parameters, gradient-based methods are more involved. Consider a parameterized gate of the form $G(\theta) = e^{-i\theta/2H}$ from the general ansatz description in Equation 3.4. Using the parameter-shift rule as described by Schuld, Bergholm, Gogolin, *et al.* [33], the derivative of the gate G with respect to a parameter θ can be described as follows:

$$\frac{\partial G}{\partial \theta} = \frac{1}{2}\left[G\left(\theta + \frac{\pi}{2}\right) - G\left(\theta - \frac{\pi}{2}\right)\right]. \tag{3.5}$$

That is, the derivative of G with respect to a parameter θ can be computed analytically by evaluating the quantum circuit twice with shifted parameters $\theta \pm \pi/2$.

For example, the derivative of a cost function $C(\vec{\theta}) = \sum_k \langle 0^n | U(\vec{\theta})^\dagger H_k U(\vec{\theta}) | 0^n \rangle$ with respect to a parameter θ_l can be estimated as follows:

$$\begin{aligned}\frac{\partial C}{\partial \theta_l} = \sum_k \frac{1}{2}\Bigg[&\langle 0^n | U\left(\vec{\theta} + \frac{\pi}{2}\vec{e}_l\right)^\dagger H_k U\left(\vec{\theta} + \frac{\pi}{2}\vec{e}_l\right) | 0^n \rangle \\ &- \langle 0^n | U\left(\vec{\theta} - \frac{\pi}{2}\vec{e}_l\right)^\dagger H_k U\left(\vec{\theta} - \frac{\pi}{2}\vec{e}_l\right) | 0^n \rangle\Bigg],\end{aligned} \tag{3.6}$$

where $\vec{e}_l$ is a vector with 1 as its l-th element and 0 for all other elements. Note that calculating ∂C with m parameters $\vec{\theta}$ requires you to evaluate the quantum circuit twice for every parameter, for a total of $2m$ quantum circuit evaluations per gradient evaluation. The parameter-shift rule can be further generalized to calculate higher-order derivatives of quantum gates as demonstrated by Mari, Bromley, and Killoran [34], allowing the use of optimization methods that use higher-order derivates such as Newton's method.

A common way to perform gradient-based optimization is through gradient descent based algorithms. Gradient descent uses first-order derivatives for finding a local minimum of a differentiable function. Intuitively, it takes repeated steps in the opposite

direction of the gradient of the function until the gradient is zero (Figure 3.2). Formally, for a set of parameters $\vec{\theta}_t$ and a cost function $C(\vec{\theta})$, the updated parameters $\vec{\theta}_{t+1}$ after one iteration of gradient descent are as follows:

$$\vec{\theta}_{t+1} = \vec{\theta}_t - \eta \nabla C(\vec{\theta}_t), \tag{3.7}$$

where η is the step size which determines how big of a step the algorithm makes each iteration. Finding the optimal step size is a practical challenge: if it is too large the optimization algorithm will diverge, and if it is too small it will take a long time to converge. Methods have been developed to improve the gradient descent algorithm — for example, the gradient descent based optimization algorithm Adam adapts the step size during optimization to increase the efficiency and accuracy of solutions [35].

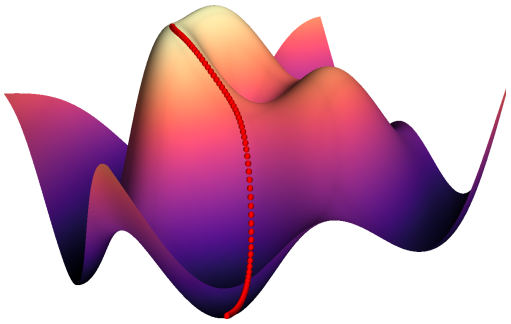


Figure 3.2: Visualization of gradient descent finding a local minimum of a two-dimensional function. The red dots represent the value of the cost function at a given iteration. The gradient descent algorithm gradually finds a local minimum by moving in the opposite direction of the gradient of the cost function.

3.2 Variational Quantum Eigensolver

The variational quantum eigensolver (VQE) was the first proposed variational HQCA. It aims to solve the problem of finding eigenvalues of a Hermitian matrix H. The problem of finding eigenvalues for large matrices has applications in quantum chemistry, but potential applications range from determining the results of search engines [36] to designing new materials and drugs [37]. A previously known quantum algorithm called the QPEA solves this problem exponentially faster than the best known classical algorithm [38]. However, the quantum resources required to run the QPEA greatly exceed the resources available in the near term. Peruzzo, McClean, Shadbolt, *et al.* [39] proposed the VQE as alternative which can run on NISQ devices.

Formally, the VQE estimates the minimum eigenvalue $\lambda_{\min}$ associated with eigenstate $|\lambda_{\min}\rangle$ for a Hermitian operator H. The cost function is defined as the expectation value of H over a trial state $|\psi(\vec{\theta})\rangle = U(\vec{\theta})|\psi_0\rangle$ where $U(\vec{\theta})$ is the ansatz and $|\psi_0\rangle$ an initial guess state:

$$C(\vec{\theta}) = \langle\psi(\vec{\theta})|H|\psi(\vec{\theta})\rangle. \tag{3.8}$$

Minimizing this cost function estimates the minimum eigenvalue of H, as the Rayleigh-Ritz variational principle provides the following bound [40]:

$$\lambda_{\min} \leq \langle\psi(\vec{\theta})|H|\psi(\vec{\theta})\rangle. \tag{3.9}$$

That is, a trial state $|\psi(\vec{\theta})\rangle$ will always give an expectation value larger than or equal to the minimum eigenvalue. In the context of quantum chemistry, where H describes the Hamiltonian of a system, finding the lowest eigenvalue of the Hamiltonian translates to finding the ground state and ground state energy of that system. By selecting an initial guess state, calculating the expectation value of H, and optimizing the cost function $C(\vec{\theta})$, one can then approximate the ground state energy of a Hamiltonian to arbitrary precision.

The problem of estimating the ground state energy is a specific instance of the k-local Hamiltonian problem, which looks to find the ground state energy of a k-local Hamiltonian. A k-local Hamiltonian is a Hamiltonian that can be expressed as a sum of Hamiltonians $H = \sum_j H_j$ where each H_j is a Hermitian operator acting non-trivially on at most k qubits [41]. This problem was shown to be QMA-complete for $k \geq 2$, while the 1-local Hamiltonian problem is in P [42]. However, quantum computers can still offer an advantage above classical computers for this problem. Any Hamiltonian H can be represented as a linear combination of Pauli operators:

$$H = \sum_j h_j P_j = \sum_j h_j \prod_k P_j^k, \tag{3.10}$$

where P_j^k is a Pauli operator $\{I, X, Y, Z\}$ and k denotes which qubit the operator acts on, and h_j is a real constant. From the linearity of quantum observables, it follows that

$$\langle H \rangle = \sum_j h_j \langle P_j \rangle. \tag{3.11}$$

The problem of estimating $\langle H \rangle$ can be done efficiently on a quantum computer, while being a hard problem to solve classically [39, 43, 44].

In order to use the VQE to estimate ground state energies of molecules, the energy of the electrons and nuclei in a molecule need to be encoded as a Hamiltonian. The standard form of such Hamiltonian is as follows:

$$H = -\sum_i \frac{\nabla^2_{\vec{R}_i}}{2M_i} - \sum_i \frac{\nabla^2_{\vec{r}_i}}{2} - \sum_{i,j} \frac{Z_i}{|\vec{R}_i - \vec{r}_j|} + \sum_{i,j>i} \frac{Z_i Z_j}{|\vec{R}_i - \vec{R}_j|} + \sum_{i,j>i} \frac{1}{|\vec{r}_i - \vec{r}_j|}, \tag{3.12}$$

where M_i, Z_i, and $\vec{R}_i$ denote the mass, atomic number, and position of nucleus i, and $\vec{r}_i$ is the position of electron i [6]. Given the fact that the nuclei are much heavier than the electrons, we can apply the Born-Oppenheimer approximation [45] to separate the Hamiltonian into electronic and nuclear terms. We are mostly interested in the electronic structure of the molecule, so we can use the Born-Oppenheimer approximation write the electronic Hamiltonian as

$$H = -\sum_i \frac{\nabla^2_{\vec{r}_i}}{2} - \sum_{i,j} \frac{Z_i}{|\vec{R}_i - \vec{r}_j|} + \sum_{i,j>i} \frac{1}{|\vec{r}_i - \vec{r}_j|}. \tag{3.13}$$

This form of Hamiltonian is often referred to as the first quantized formulation of quantum chemistry [46, Appendix A]. In order to represent electronic Hamiltonians on a quantum computer, it needs to be mapped from operators acting on indistinguishable fermions to operators acting on distinguishable qubits. There are several approaches to this [47], but the most widely used approach in quantum chemistry uses the second

quantized formulation [6]. The second quantized Hamiltonian must then be mapped into qubits in order to be implemented on a quantum computer. The most common mappings for this are the Jordan-Wigner [48] and Bravyi-Kitaev [49] transformations. The exact details on the different approaches are beyond the scope of this work, but a detailed overview can be found in [6].

Implementations of the VQE can often benefit from using a problem-inspired ansatz. A popular choice is the unitary coupled cluster (UCC) ansatz, which is a cousin of the coupled-cluster method used in quantum chemistry for describing many-body systems [50]. The UCC ansatz is parameterized by a polynomial number of parameters, while there is no known efficient classical implementation [51]. Furthermore, the UCC ansatz is believed to provide better accuracy than classical coupled cluster methods [52, 53]. Despite these advantages, the number of parameters required by the UCC ansatz might be too large to allow for practical calculations of large molecules [54]. Alternative ansätze have been proposed to reduce the quantum resources required for running the VQE on larger molecules [44, 55, 56].

3.3 Quantum Approximate Optimization Algorithm

4

Practical Hybrid Quantum-Classical Computing

This chapter is focused on the practical execution of HQCAs using the Quantum Inspire quantum computing platform and SURF's HPC center.

4.1 Quantum Inspire

Quantum Inspire is a full-stack quantum computing platform that QuTech launched last year to make quantum systems available to the general public for exploratory research [57]. Users can run quantum circuits on different back-ends through the Quantum Inspire web editor or by using the Python software development kit (SDK). The common quantum assembly language (cQASM) [58] is used for describing quantum circuits, but the popular quantum computing frameworks ProjectQ [59] and Qiskit [60] are also supported by the Python SDK. Quantum Inspire supports the execution of quantum circuits on real quantum chips and through quantum simulation. An overview of the Quantum Inspire workflow and available device back-ends is shown in Figure 4.1.

The two available quantum chips are Starmon-5 and Spin-2 which have five and two qubits respectively. These chips suffer from noise, limited coherence time, limited qubit connectivity, and each quantum chip has a specific allowed gate set. Note that these gate sets are universal, and non-native gates are decomposed into supported gates depending on the chip. Regardless of the limitations of these quantum chips, they are an essential part in research towards better quantum hardware and quantum algorithms.

In the absence of large-scale and fault-tolerant quantum computers, quantum computer simulation is critical for developing and testing quantum algorithms. Quantum Inspire offers two simulator back-ends which use QX [61] as quantum computer simulator. One of the simulator back-ends is hosted by QuTech on a commodity cloud-based server with 4GB of RAM, which can run simulations up to 26 qubits. The other simulator back-end is hosted on SURF's computer cluster Lisa which consists of several hundreds of multi-core nodes. The largest node available on Lisa has 256GB of RAM, which supports simulations up to 31 qubits.

When quantum circuits are submitted to Quantum Inspire, they are handed to a job scheduler which will attempt to schedule the job as quickly as possible when

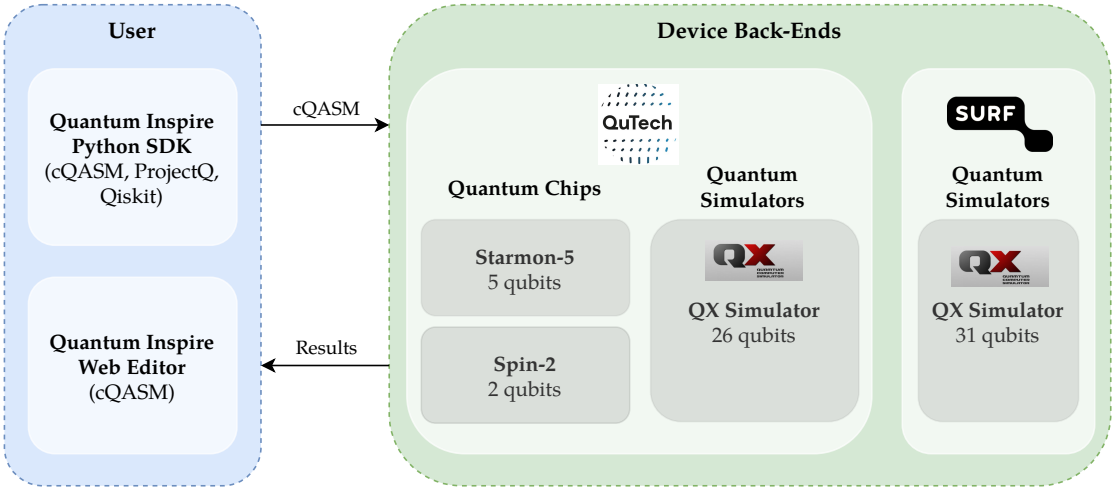


Figure 4.1: Overview of the Quantum Inspire workflow. Users can submit quantum circuits written in cQASM using the web editor or Python SDK. After the program has been run, the results are returned to the user. The quantum circuit can be executed on one of the quantum chips or simulated using one of the QX simulator back-ends. QuTech's simulator back-end supports simulations up to 26 qubits, while SURF's simulator back-end supports simulations up to 31 qubits.

the requested resources are available. For the quantum chips and QuTech simulator back-ends, the jobs are simply placed in a queue which are executed in first-in-first-out order. The wait time for these back-ends ranges from seconds to minutes. For the SURF simulator back-end, the jobs are handled by the Slurm workload manager [62]. Because SURF's infrastructure is shared with many other users, wait times can vary from minutes to hours. Especially large simulations which require a large amount of resources may take a long time to schedule.

4.2 SURF High Performance Computing

4.3 Implementation

5

Conclusion

Bibliography

[1] P. Benioff, "The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines", *Journal of statistical physics*, vol. 22, no. 5, pp. 563–591, 1980.

[2] I. I. Manin, *Vychislimoe i nevychislimoe*. Sov. radio, 1980.

[3] R. P. Feynman, "Simulating physics with computers", *Int. J. Theor. Phys*, vol. 21, no. 6/7, 1982.

[4] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer", *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[5] L. K. Grover, "A fast quantum mechanical algorithm for database search", in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 212–219.

[6] S. McArdle, S. Endo, A. Aspuru-Guzik, *et al.* (2018). Quantum computational chemistry. arXiv: `1808.10402 [quant-ph]`.

[7] C. H. Bennett and G. Brassard, "Quantum cryptography", *Theoretical Computer Science*, vol. 560, no. P1, pp. 7–11, 2014.

[8] J. Biamonte, P. Wittek, N. Pancotti, *et al.*, "Quantum machine learning", *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.

[9] A. G. Fowler, M. Mariantoni, J. M. Martinis, *et al.*, "Surface codes: Towards practical large-scale quantum computation", *Physical Review A*, vol. 86, no. 3, p. 032 324, 2012.

[10] J. Preskill, "Quantum computing in the NISQ era and beyond", *Quantum*, vol. 2, p. 79, 2018.

[11] S. Endo, Z. Cai, S. C. Benjamin, *et al.*, "Hybrid quantum-classical algorithms and quantum error mitigation", *Journal of the Physical Society of Japan*, vol. 90, no. 3, p. 032 001, 2021.

[12] QuTech. (2018). Quantum inspire home, [Online]. Available: `https://www.quantum-inspire.com/` (visited on 2021-02-24).

[13] SURF. (2021). SURFsara systems, [Online]. Available: `https://userinfo.surfsara.nl/systems/` (visited on 2021-02-24).

[14] A. Church *et al.*, "A note on the entscheidungsproblem", *J. Symb. Log.*, vol. 1, no. 1, pp. 40–41, 1936.

[15] A. M. Turing, "On computable numbers, with an application to the entscheidungsproblem", *Proceedings of the London mathematical society*, vol. 2, no. 1, pp. 230–265, 1937.

[16] T. H. Cormen, C. E. Leiserson, R. L. Rivest, *et al.*, *Introduction to algorithms*. MIT press, 2009.

[17] S. Arora and B. Barak, *Computational complexity: A modern approach*. Cambridge University Press, 2009.

[18] J. Hartmanis and R. E. Stearns, "On the computational complexity of algorithms", *Transactions of the American Mathematical Society*, vol. 117, pp. 285–306, 1965.

[19] N. Dershowitz and Y. Gurevich, "A natural axiomatization of computability and proof of church's thesis", *Bulletin of symbolic logic*, vol. 14, no. 3, pp. 299–350, 2008.

[20] C. H. Bennett, "Logical reversibility of computation", *IBM journal of Research and Development*, vol. 17, no. 6, pp. 525–532, 1973.

[21] D. Deutsch, "Quantum theory, the church–turing principle and the universal quantum computer", *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, vol. 400, no. 1818, pp. 97–117, 1985.

[22] F. Arute, K. Arya, R. Babbush, *et al.*, "Quantum supremacy using a programmable superconducting processor", *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[23] O. Goldreich, "In a world of P= BPP", in *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, Springer, 2011, pp. 191–232.

[24] N. Nisan and A. Wigderson, "Hardness vs randomness", *Journal of computer and System Sciences*, vol. 49, no. 2, pp. 149–167, 1994.

[25] E. Bernstein and U. Vazirani, "Quantum complexity theory", *SIAM Journal on computing*, vol. 26, no. 5, pp. 1411–1473, 1997.

[26] S. Aaronson, "BQP and the polynomial hierarchy", in *Proceedings of the forty-second ACM symposium on Theory of computing*, 2010, pp. 141–150.

[27] G. M. D'Ariano, M. G. Paris, and M. F. Sacchi, "Quantum tomography", *Advances in Imaging and Electron Physics*, vol. 128, pp. 206–309, 2003.

[28] M. A. Nielsen and I. Chuang, *Quantum computation and quantum information*, 2002.

[29] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations", *Physical review letters*, vol. 103, no. 15, p. 150 502, 2009.

[30] C. Zalka, "Grover's quantum searching algorithm is optimal", *Physical Review A*, vol. 60, no. 4, p. 2746, 1999.

[31] S. Sim, P. D. Johnson, and A. Aspuru-Guzik, "Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms", *Advanced Quantum Technologies*, vol. 2, no. 12, p. 1 900 070, 2019.

[32] M. Cerezo, A. Arrasmith, R. Babbush, *et al.*, "Variational quantum algorithms", 2020. arXiv: 2012.09265 [quant-ph].

[33] M. Schuld, V. Bergholm, C. Gogolin, *et al.*, "Evaluating analytic gradients on quantum hardware", *Physical Review A*, vol. 99, no. 3, p. 032 331, 2019.

[34] A. Mari, T. R. Bromley, and N. Killoran, "Estimating the gradient and higher-order derivatives on quantum hardware", *Physical Review A*, vol. 103, no. 1, p. 012 405, 2021.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", 2014. arXiv: 1412.6980 [cs].

[36] L. Page, S. Brin, R. Motwani, *et al.*, "The pagerank citation ranking: Bringing order to the web.", Stanford InfoLab, Tech. Rep., 1999.

[37] G. H. Golub and H. A. van der Vorst, "Eigenvalue computation in the 20th century", *J. Comput. Appl. Math.*, vol. 123, no. 1-2, pp. 35–65, 2000-11, ISSN: 0377-0427. DOI: 10.1016/s0377-0427(00)00413-1.

[38] D. S. Abrams and S. Lloyd, "Quantum algorithm providing exponential speed increase for finding eigenvalues and eigenvectors", *Physical Review Letters*, vol. 83, no. 24, p. 5162, 1999.

[39] A. Peruzzo, J. McClean, P. Shadbolt, *et al.*, "A variational eigenvalue solver on a photonic quantum processor", *Nature communications*, vol. 5, no. 1, pp. 1–7, 2014.

[40] W. Ritz, "Über eine neue methode zur lösung gewisser variationsprobleme der mathematischen physik.", *Journal für die reine und angewandte Mathematik*, vol. 1909, no. 135, pp. 1–61, 1909.

[41] A. D. Bookatz, "QMA-complete problems", 2012. arXiv: 1212.6312 [quant-ph].

[42] J. Kempe, A. Kitaev, and O. Regev, "The complexity of the local hamiltonian problem", *SIAM Journal on Computing*, vol. 35, no. 5, pp. 1070–1097, 2006.

[43] G. Ortiz, J. E. Gubernatis, E. Knill, *et al.*, "Quantum algorithms for fermionic simulations", *Physical Review A*, vol. 64, no. 2, p. 022 319, 2001.

[44] A. Kandala, A. Mezzacapo, K. Temme, *et al.*, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets", *Nature*, vol. 549, no. 7671, pp. 242–246, 2017.

[45] M. Born and R. Oppenheimer, "Zur quantentheorie der molekeln", *Annalen der physik*, vol. 389, no. 20, pp. 457–484, 1927.

[46] P. J. O'Malley, R. Babbush, I. D. Kivlichan, *et al.*, "Scalable quantum simulation of molecular energies", *Physical Review X*, vol. 6, no. 3, p. 031 007, 2016.

[47] I. Kassal, J. D. Whitfield, A. Perdomo-Ortiz, *et al.*, "Simulating chemistry using quantum computers", *Annual review of physical chemistry*, vol. 62, pp. 185–207, 2011.

[48] R. Somma, G. Ortiz, J. E. Gubernatis, *et al.*, "Simulating physical phenomena by quantum networks", *Physical Review A*, vol. 65, no. 4, p. 042 323, 2002.

[49] J. T. Seeley, M. J. Richard, and P. J. Love, "The bravyi-kitaev transformation for quantum computation of electronic structure", *The Journal of chemical physics*, vol. 137, no. 22, p. 224 109, 2012.

[50] I. Shavitt and R. J. Bartlett, *Many-body methods in chemistry and physics: MBPT and coupled-cluster theory*. Cambridge university press, 2009.

[51] A. G. Taube and R. J. Bartlett, "New perspectives on unitary coupled-cluster theory", *International journal of quantum chemistry*, vol. 106, no. 15, pp. 3393–3401, 2006.

[52] B. Cooper and P. J. Knowles, "Benchmark studies of variational, unitary and extended coupled cluster methods", *The Journal of chemical physics*, vol. 133, no. 23, p. 234 102, 2010.

[53] F. A. Evangelista, "Alternative single-reference coupled cluster approaches for multireference problems: The simpler, the better", *The Journal of chemical physics*, vol. 134, no. 22, p. 224 102, 2011.

[54] D. Wecker, M. B. Hastings, and M. Troyer, "Progress towards practical quantum variational algorithms", *Physical Review A*, vol. 92, no. 4, p. 042 303, 2015.

[55] J. Romero, R. Babbush, J. R. McClean, *et al.*, "Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz", *Quantum Science and Technology*, vol. 4, no. 1, p. 014 008, 2018.

[56] H. R. Grimsley, S. E. Economou, E. Barnes, *et al.*, "An adaptive variational algorithm for exact molecular simulations on a quantum computer", *Nature communications*, vol. 10, no. 1, pp. 1–9, 2019.

[57] T. Last, N. Samkharadze, P. Eendebak, *et al.*, "Quantum inspire: QuTech's platform for co-development and collaboration in quantum computing", in *Novel Patterning Technologies for Semiconductors, MEMS/NEMS and MOEMS 2020*, International Society for Optics and Photonics, vol. 11324, 2020, 113240J.

[58] N. Khammassi, G. G. Guerreschi, I. Ashraf, *et al.*, "cQASM v1. 0: Towards a common quantum assembly language", 2018. arXiv: 1805.09607 [quant-ph].

[59] D. S. Steiger, T. Häner, and M. Troyer, "Projectq: An open source software framework for quantum computing", *Quantum*, vol. 2, p. 49, 2018.

[60] H. Abraham, AduOffei, R. Agarwal, *et al.*, *Qiskit: An open-source framework for quantum computing*, 2019. DOI: 10.5281/zenodo.2562110.

[61] N. Khammassi, I. Ashraf, X. Fu, *et al.*, "Qx: A high-performance quantum computer simulation platform", in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, IEEE, 2017, pp. 464–469.

[62] A. B. Yoo, M. A. Jette, and M. Grondona, "Slurm: Simple linux utility for resource management", in *Workshop on job scheduling strategies for parallel processing*, Springer, 2003, pp. 44–60.

Acronyms

cQASM Common Quantum Assembly Language

DFT Discrete Fourier Transform

HPC High Performance Computing

HQCA Hybrid Quantum-Classical Algorithm

NISQ Noisy Intermediate-Scale Quantum

QFT Quantum Fourier Transform

QPEA Quantum Phase Estimation Algorithm

SDK Software Development Kit

UCC Unitary Coupled Cluster

VQE Variational Quantum Eigensolver