Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF PHYSICS

Quantum Algorithms: A Review

Author:
Chukwudubem Umeano

Supervisor:
Jonathan Halliwell

Submitted in partial fulfillment of the requirements for the degree of Master of Science of Imperial College London

September 2021

Abstract

Quantum computers are able to solve certain problems quicker than a standard classical computer. This speedup is due to the development of a variety of efficient quantum algorithms. In this dissertation we provide a detailed review of three of the major, original quantum algorithms: Shor's algorithm for factoring, Grover's algorithm for search and quantum simulation. We also outline some alternative algorithms which have emerged from more modern research. For each procedure we discuss their performance, current and potential applications.

Acknowledgments

I would first like to thank God for giving me the strength and wellbeing to complete this MSc.

Thank you to Jonathan Halliwell for supervising this dissertation, as well as supporting me as my personal advisor throughout this masters course.

Thank you to the entire Theoretical Physics department for facilitating this stimulating course. I am also grateful for my fellow QFFF students who have made this year enjoyable, despite the unfortunate circumstances created by the pandemic.

Special thanks to Ollie, Mg, uncle Leslie and my dad for proofreading this report and providing helpful feedback.

Finally, I would like to thank my family and friends for their continuous love and support.

Contents

Chapter 1

Introduction

"I think I can safely say that nobody understands quantum mechanics"

- Richard Feynman

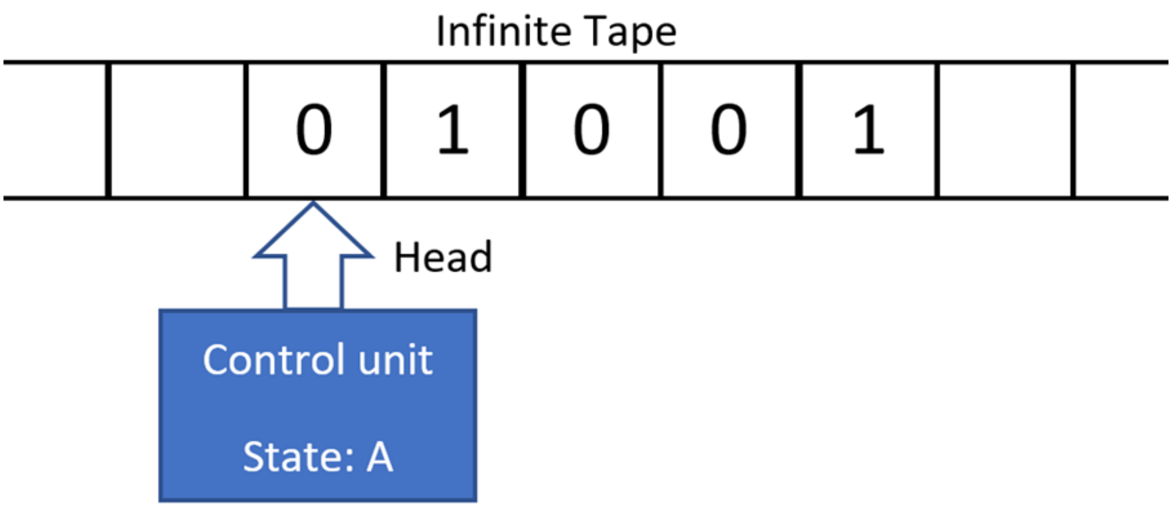
1.1 A brief history of quantum computing

The theory of quantum mechanics and all its intricacies has both intrigued and confused generations of physicists for over a century. Early observations and concepts devised by the likes of Thomas Young (wave nature of light), Max Planck (quantisation of electromagnetic radiation), Niels Bohr (model of the hydrogen atom) and Albert Einstein (photoelectric effect and the particle nature of light) set the foundations.

The 1920s saw the emergence of the now standard formulation of quantum mechanics [1]. Legendary scientists such as Werner Heisenberg, Max Born, Pascual Jordan, Erwin Schrodinger, David Hilbert, Paul Dirac and John van Neumann made their imprint, solidifying the mathematical framework still studied and applied to this day.

Just a decade later, the building blocks for modern computer science were being laid into place. In 1936, Alan Turing revealed his groundbreaking Turing Machine (TM), an abstract, idealised computing model [2]. It consists of a tape of infinite length divided into cells, each one either blank or containing a finite set of symbols (e.g. 0, 1). The control unit controls a head, which moves across the tape, reading and writing symbols in cells. The particular operation performed by the head is determined by a set of instructions; these consider the contents of the cell being scanned, as well as the current 'internal state' of the machine.

Although incredibly simple, Turing showed that for any computable problem, a TM could be constructed to solve it. His noteworthy paper also introduced the concept of a universal Turing machine; "a single machine which can be used to compute any computable sequence." Essentially, a universal Turing machine (UTM) is able to simulate any other TM. This finding is widely credited for inspiring the birth of



State	Symbol read	Symbol write	Move head	Change state to
A	BLANK	BLANK	Left	B
	0	0	Right	A
	1	1	Right	A
B	BLANK	1	Right	C
	0	1	Left	C
	1	0	Left	B
C	BLANK	BLANK	Left	HALT
	0	0	Right	C
	1	1	Right	C

Figure 1.1: A simple 3-state, 3 symbol Turing machine. With the initial state and head position shown in the diagram, this Turing machine program simply adds 1 to the initial binary number on the tape. In this case, it will change 01001 to 01010 (9 to 10).

some of the first modern computers [3], most notably the stored-program computer devised by von Neumann [4]. Turing also showed that the Entscheidungsproblem (decision problem) was incomputable [2], hence demonstrating the existence of limits of computation. Although purely conceptual machines, these models continue to play a major role in modern research, in particular computational complexity theory.

The marriage between quantum mechanics and computation was sealed in 1980, when Paul Benioff constructed the first quantum mechanical model of Turing machines. The quantum Turing machine (QTM) is conceptually very similar to the original model; just with the set of internal states replaced by states in a Hilbert space.

In his 1980 paper [5], Benioff represents the tape with a lattice of quantum spin systems. Both the symbols and states are in one-to-one correspondence with the set of possible spin projections of the spin systems. Finally, the simple transition function from the classical TM is replaced by a set of unitary operators, which are automorphisms of the Hilbert space. Benioff's work was significant as he proved the feasibility of reversible quantum computing (reversibility is required due to the nature of physical systems), which was the catalyst for the field to really emerge.

Feynman was the next to contribute; in a 1982 lecture [6] he demonstrated that it is impossible to simulate quantum systems on a classical computer. Feynman made use of a simple two-photon correlation experiment to show that 'hidden variables' could not be used to reproduce quantum mechanical results with a classical device - essentially a proof of Bell's theorem [7]. He also theorised a universal quantum simulator which, after applying some approximations, could simulate any quantum system.

David Deutsch followed with his breakthrough paper in 1985 [8], in which he presented the first real universal quantum computer. This was a quantum equivalent to the UTM; a universal quantum computer can simulate all other quantum computers. The paper's significance does not end there; Deutsch also made significant contributions to the development of quantum complexity theory and quantum algorithms, as will be explained in the following chapters.

Finally, Peter Shor blew the field of quantum computing open in 1994, with his invention of a quantum algorithm that could factor large integers rapidly [9]. If a suitable large-scale quantum computer could be constructed to apply this algorithm, public-key cryptography systems such as RSA would be under severe threat. This discovery was the trigger for the explosion of quantum computing, and twenty-seven years on it is still a burgeoning area of study, with more questions than answers.

1.2 Classical algorithms

Algorithm

A step-by-step procedure for solving a problem or accomplishing some end.

- Merriam-Webster.com Dictionary [10]

From solving a Rubik's cube to baking cupcakes, following algorithms has become a natural part of our lives. However, they are most synonymous with computer programs, and society's dependence on the internet has made them impossible to avoid in the 21st century. A computer algorithm starts in some initial state and takes in input: perhaps a word, sentence, set of numbers or a key. It then executes a finite, precise series of steps, designed to solve some problem or perform a task, before producing an output. There are often conditional steps and some loops, but as long as the program terminates in finite time, it can be considered an algorithm.

Despite following the same basic paradigms, algorithms range in their sophistication, from the relatively primitive Euclidean algorithm detailed below, to the multilayered, complicated and somewhat controversial algorithms used to keep Facebook feeds tailored to each user's tastes. Classifying algorithms is another task. We may choose to classify them by type of problem (e.g. search, sort, graph, numerical), complexity (see Chapter 3) or even method of approach. Some algorithms rely on a 'brute-force' approach with the aim to test every possible solution; some split the problem into chunks and solve each subproblem separately, while others rely on recursion to meander towards an outcome.

Algorithm 1 Euclidean algorithm for finding the highest common factor (HCF) of 2 numbers. Named after the great Euclid, who originally described the procedure in 300 BC.

```
procedure EUCLID'S ALGORITHM FOR FINDING THE HCF OF 2 NUMBERS($a$,$b$)
    while $b \neq 0$ do
        if $a > b$ then
            $a \leftarrow a - b$
        else
            $b \leftarrow b - a$
        end if
    end while
    return a
end procedure
```

1.3 The power of quantum

Quantum algorithms are simply algorithms that run on any realistic model of quantum computation [11]. The most common model is the quantum circuit model, where each program is carried out via a sequence of quantum gates; see Section 2.1 for a full breakdown. The power of these algorithms comes from the properties of quantum mechanics which they exploit: superposition, entanglement, interference and coherence. These key principles give quantum algorithms a speed and capacity that classical algorithms cannot match for many different tasks.

1.3.1 Superposition and bits vs qubits

Take an ordinary coin sitting on a table. Its 'state' is either 'Heads' (0) or 'Tails' (1). Now imagine spinning this coin so quickly that it becomes a blur of motion: this is our 'quantum coin', which is in a superposition of both states simultaneously. Similarly, bits, the most fundamental unit of information in classical computing, can only be in the states 0 or 1. Meanwhile, a quantum bit, or qubit, can be in either of the basis states $|0\rangle$ or $|1\rangle$, or a superposition of both states:

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle . \tag{1.1}$$

Here, α and β are complex amplitudes. When a measurement of the qubit is made, the state collapses into a basis state; either $|0\rangle$ with probability α^2 or $|1\rangle$ with probability β^2. A qubit can take on any superposition state with the condition $\alpha^2 + \beta^2 = 1$. This extends to multiple qubits: a pair of qubits can exist in a superposition of all four basis states:

$$|\Psi\rangle = \alpha_1 |00\rangle + \alpha_2 |01\rangle + \alpha_3 |10\rangle + \alpha_4 |11\rangle \, , \tag{1.2}$$

and measurement again collapses the superposition state into one of the basis states. Qubits can be realized physically as any two-state quantum-mechanical system. Examples of this are the two polarisations of a photon (horizontal and vertical), two spins of an electron (spin-up and spin-down), and two electronic levels in an atom (ground and excited states).

Generalising to n qubits, 2^n basis states can be represented simultaneously. With just 500 qubits, we can represent more states in superposition than the estimated number of atoms in the universe [12]. Hence it is easy to envision the enormous potential of quantum computing; having several qubits in superposition vastly increases the capacity for simultaneous calculations.

1.3.2 Quantum entanglement

Quantum entanglement, or as Einstein dubbed it, “spooky action at a distance” is a startling phenomenon. It occurs when the quantum state of two or more particles are correlated, no matter the distance between them. For an entangled pair, measuring properties of one particle, e.g. position, momentum or spin, immediately tells you the corresponding property of the other.

The instantaneity of this process troubled Einstein, who published a paper with Podolsky and Rosen discussing the ‘EPR paradox’ [13]. They suggested a local hidden variable theory as a reason: unknown properties not incorporated into quantum theory, implying incompleteness within quantum mechanics.

However, this theory was subsequently quashed by John Stewart Bell [7]. He demonstrated that a hidden variable theory only agrees with quantum physics if these variables are non-local, i.e. associated with both halves of an entangled pair. Bell’s theorem has been validated by numerous experiments, proving beyond doubt that no sort of transmission of information between particles takes place.

It is useful to outline a mathematical framework here. A pure state can be written as a state vector $|\psi\rangle$. For a composite Hilbert space $\mathcal{H}_A \otimes \mathcal{H}_B$, a separable state can be written as a tensor product of two pure states:

$$|\psi\rangle_A \otimes |\psi\rangle_B \, . \tag{1.3}$$

If a 2-qubit state cannot be written in this form, it is an entangled state. The most famous examples are the maximally entangled Bell states:

$$\begin{aligned}\left|\Phi^{\pm}\right\rangle_{AB} &= \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |0\rangle_B \pm |1\rangle_A \otimes |1\rangle_B) \\ \left|\Psi^{\pm}\right\rangle_{AB} &= \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |1\rangle_B \pm |1\rangle_A \otimes |0\rangle_B).\end{aligned} \tag{1.4}$$

It is clear to see that a measurement of the 1st qubit provides immediate knowledge of the value of the 2nd. Moving onto mixed states, we can define a density matrix:

$$\hat{\rho} = \sum_i p_i \left|\psi_i\right\rangle \left\langle\psi_i\right| \tag{1.5}$$

$\hat{\rho}$ is Hermitian, has unit trace and is a positive semi-definite operator: $\langle\phi|\hat{\rho}|\phi\rangle \geq 0$. If $\hat{\rho}$ has a single eigenvalue, it represents a pure state $|\psi\rangle\langle\psi|$, else it is a mixed state with $\sum_i p_i = 1$. A defining condition for a pure state is

$$\hat{\rho}^2 = \hat{\rho}. \tag{1.6}$$

Analogous to the form for separable pure states, a bipartite system in a mixed state is separable when it can be written in this form:

$$\hat{\rho} = \sum_i p_i \hat{\rho}_{A,i} \hat{\rho}_{B,i}. \tag{1.7}$$

If the partial transpose of the density matrix $\hat{\rho}^{PT}$ is negative, then the system is entangled; the level of negativity is a measure of entanglement [14].

1.3.3 Interference, coherence and decoherence

Linked to the properties explained above are the concepts of quantum interference and coherence. Young's famous double-slit experiment shows quantum interference in action. Light passes through the two slits before the waves interfere, producing an interference pattern with dark and light regions, due to constructive and destructive interference respectively. Mathematically, the wavefunction of this system can be written as a superposition of the contributions from each slit:

$$\Psi(x,t) = \Psi_A(x,t) + \Psi_B(x,t) \tag{1.8}$$

Hence, the probability of a photon being found at position x is given by:

$$P(x) = |\Psi(x,t)|^2 = |\Psi_A(x,t)|^2 + |\Psi_B(x,t)|^2 + \Psi_A^*(x,t)\Psi_B(x,t) + \Psi_B^*(x,t)\Psi_A(x,t) \tag{1.9}$$

The final two terms are the quantum interference terms, which could be positive or negative depending on whether there is constructive or destructive interference.

A system's state is coherent when it can be described by a distinct set of complex numbers, one for each of its basis states [15]. It is the property which facilitates quantum interference; a system remains coherent as long as there is a definite phase relation between states. Maintaining coherence is vital for prolonging the runtime of quantum computers. When the qubits are not perfectly isolated, they interact and entangle with the environment, before rapidly falling out of their quantum states. Decoherence destroys any superposition or entanglement between qubits: quantum information is lost to the surroundings.

Quantum error correction is a valuable weapon in the fight against decoherence; we will not discuss it here but Chapter 10 of Nielsen and Chuang's textbook [12] has a great introduction to this area.

1.4 Outline of the dissertation

In Chapter 2, we will review several models of quantum computing and some further preliminaries. Chapter 3 sees the introduction of the subject of Quantum Complexity Theory, as well as a breakdown of the Deutsch-Jozsa algorithm. In Chapter 4, we dive into Shor's algorithm for factoring, which also provides a scheme for solving the discrete logarithm problem. Chapter 5 includes a review of Grover's algorithm and its applications. The different schemes for quantum simulation and its fields of application are surveyed in Chapter 6. In Chapter 7, we discuss some more modern algorithms, including quantum walks, the HHL algorithm, adiabatic, and variational quantum algorithms. This chapter also covers a potential algorithm unification procedure.

Chapter 2

Models of quantum computation

Model of computation

A formal, abstract definition of a computer. Using a model one can more easily analyze the intrinsic execution time or memory space of an algorithm while ignoring many implementation issues.

- Paul E. Black [16]

Just as there are for classical computers (e.g. Turing machines, random-access machines, cellular automation), there are also several models for quantum computation. Each model has a different way of computing the output of a function from an input. We have already discussed the quantum Turing machine, but similarly to its classical counterpart, this model is not physically realisable. Four models which are in widespread use today are the quantum circuit, one-way quantum computer, topological quantum computer and adiabatic quantum computer.

2.1 Quantum circuits

2.1.1 Classical circuits

Before diving into quantum circuits, it is useful to know the basics about their classical equivalents. A key component of classical machines is the silicon chip, or integrated circuit (IC). Digital ICs are packed with transistors that communicate via digital, binary electrical signals [15]. These transistors make up logic gates; electronic devices which implement Boolean functions:

$$f : \{0,1\}^n \to \{0,1\}. \tag{2.1}$$

Classical circuits are built from a series of these logic gates; the Boolean functions take in an n-bit input and eject a single binary output. The primitive Boolean operations are summarised in Table 2.1; note that the NOT gate is a 1-bit gate while the others operate on 2 inputs to produce 1 output.

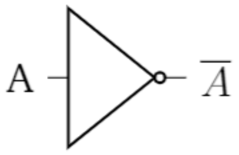
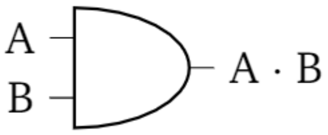
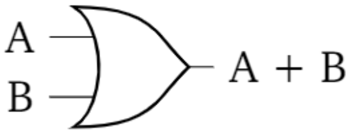
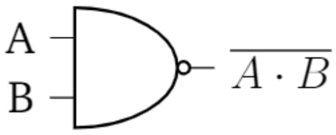
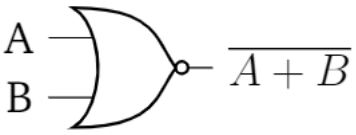
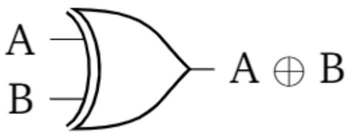
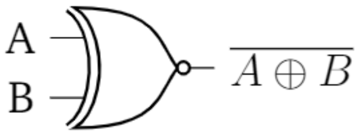
Name	**Symbol**	**Operation**
NOT (¬)		Inverts the input (0 ↔ 1)
AND (∧)		Outputs 1 only if both inputs are 1
OR (∨)		Outputs 1 if either input is 1
NAND (↑)		Outputs 0 only if both inputs are 1
NOR (↓)		Outputs 0 if either input is 1
XOR (⊻)		Outputs 1 if the two inputs are different
XNOR (⊙)		Outputs 0 if the two inputs are different

Table 2.1: Elementary logic gates

The three most basic gates, AND, OR and NOT, can be used to implement all possible Boolean functions. It is also possible to combine the AND and NOT gates to produce the OR function, likewise the OR and NOT gates can create the AND gate. Hence the sets {AND, NOT} and {OR, NOT} are known as universal gate sets; they form a complete set of logic. It has also been proven that the single gate sets NAND and NOR also form universal gate sets.

2.1.2 Reversible computation

Apart from the NOT gate, all the primitive logic gates detailed in Table 2.1 are irreversible, i.e. it is impossible to determine the inputs from the output. For example, if

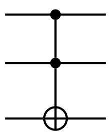


Figure 2.1: Toffoli, or Controlled-Controlled-Not gate. This gate flips the third bit only if the first two bits are 1.

the output of the OR gate is 1, the inputs could be {0,1}, {1,0} or {1,1} - information has been lost. Hence a reversible gate must be a bijective function:

$$f : \{0,1\}^n \to \{0,1\}^n. \tag{2.2}$$

In 1973, Charles Bennett made the important discovery that classical computation could be made reversible [17]. This was followed in 1980 by the proposal of the Toffoli gate by Tommaso Toffoli (Figure 2.1). This 3-bit reversible gate is also universal, proving that all classical computing can be done using reversible gates.

2.1.3 Quantum gates

By comparison, quantum gates are always reversible, since the laws of quantum mechanics does not allow for energy dissipation [15]. Hence quantum gates are unitary transformations that map the Hilbert space of n qubits onto itself. In analogy to classical circuits, these gates are combined to build quantum circuits.

It is also important to note that any reversible classical gate can also be implemented on a quantum computer. This includes the aforementioned Toffoli gate, and by nature of its universality, all classical computations can be performed on a quantum machine.

Quantum gates that act on n qubits are represented by a $2^n \times 2^n$ unitary matrix. As for classical circuits, the most common gates work on 1 or 2 qubits. Hence any single qubit gate is a 2×2 matrix, and it is useful to note that any 2×2 matrix M can be written as [14]:

$$M = a_0 \mathbb{1} + \sum_{i=x,y,z} a_i \hat{\sigma}_i, \tag{2.3}$$

where $a_0 = \frac{1}{2}$TrM and $a_i = \frac{1}{2}$Tr(M$\hat{\sigma}_i$). It is clear to see that any single qubit gate can be written as a linear superposition of Pauli operators.

2-qubit gates have a control qubit and target qubit; e.g. the Controlled-Not (CNOT) gate only flips the second (target) qubit if the first (control) qubit is $|1\rangle$. Hence the CNOT gate is the quantum generalisation of the classical XOR gate. Some basic gates are detailed in Table 2.2, as well as their action on the basis states $|0\rangle$ and $|1\rangle$.

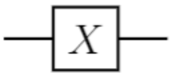
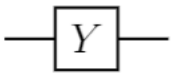
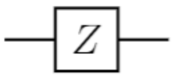
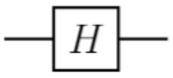
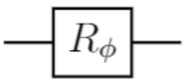
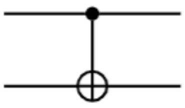
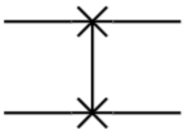
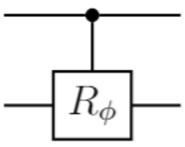
Name	**Circuit**	**Matrix**	**Operation**
Pauli-X (Quantum NOT)		$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\sigma_x\|0\rangle = \|1\rangle$ $\sigma_x\|1\rangle = \|0\rangle$
Pauli-Y		$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	$\sigma_y\|0\rangle$ = i$\|1\rangle$ $\sigma_y\|1\rangle$ = -i$\|0\rangle$
Pauli-Z		$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	$\sigma_z\|0\rangle = \|0\rangle$ $\sigma_z\|1\rangle$ = -$\|1\rangle$
Hadamard		$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	H$\|0\rangle = \frac{\|0\rangle+\|1\rangle}{\sqrt{2}}$ $(\|+\rangle)$ H$\|1\rangle = \frac{\|0\rangle-\|1\rangle}{\sqrt{2}}$ $(\|-\rangle)$
Phase shift		$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}$	$R_\phi\|0\rangle = \|0\rangle$ $R_\phi\|1\rangle = e^{i\phi}\ \|1\rangle$
CNOT		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	$U_{CN}\|00\rangle = \|00\rangle$ $U_{CN}\|01\rangle = \|01\rangle$ $U_{CN}\|10\rangle = \|11\rangle$ $U_{CN}\|11\rangle = \|10\rangle$
SWAP		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	SWAP$\|00\rangle = \|00\rangle$ SWAP$\|01\rangle = \|10\rangle$ SWAP$\|10\rangle = \|01\rangle$ SWAP$\|11\rangle = \|11\rangle$
Controlled-phase		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\phi} \end{pmatrix}$	$U_{C_\phi}\|00\rangle = \|00\rangle$ $U_{C_\phi}\|01\rangle = \|01\rangle$ $U_{C_\phi}\|10\rangle = \|10\rangle$ $U_{C_\phi}\|11\rangle = e^{i\phi}\ \|11\rangle$

Table 2.2: Elementary quantum gates

Circuit computations are often completed with a measurement. As we touched on in Section 1.3.1, measurements probabilistically collapse a quantum state into one of the basis states. This is an irreversible process, hence it is not a quantum gate; however it can still be represented in quantum circuits as in Figure 2.2.

It is important to note that the quantum circuit is computationally equivalent to the quantum Turing machine. This means that the former can simulate the latter with only polynomial overhead (number of qubits and computational steps).

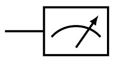


Figure 2.2: Representation of measurement in a quantum circuit

2.1.4 Universal quantum gates and the Solovay-Kitaev theorem

In analogy with classical universality, a set of quantum gates is universal if any quantum operation can be executed using a finite sequence of gates from this set. Again, there are many different universal gate sets; a common example is the 'Clifford + T' set. This includes the CNOT, Hadamard, S and T gates (the latter two are both phase shift gates with $\phi = \frac{\pi}{2}$ and $\frac{\pi}{4}$ for S and T respectively).

In general, a universal quantum gate can be made by combining arbitrary single-qubit gates with a CNOT gate, or any other two-qubit controlled gate. The three-qubit Deutsch gate (D_ϕ) makes up its own single-gate set of universal quantum gates. It is conceptually similar to the Toffoli gate:

$$|a,b,c\rangle \mapsto \begin{cases} i\cos\theta\,|a,b,c\rangle + \sin\theta\,|a,b,1\oplus c\rangle \textit{ if } a = b = 1 \\ |a,b,c\rangle \textit{ otherwise} \end{cases} \tag{2.4}$$

In fact, the Toffoli gate is equivalent to $D(\pi/2)$, another demonstration that all classical computations can be done on a quantum computer.

There is a caveat which arises from the theoretically infinite number of possible quantum operations, compared to the countable number of gate sequences achievable from a finite set of gates. This suggests that, by strict definition, a universal set of quantum gates does not exist. This issue is overcome by requiring a universal set to have the ability to merely approximate any quantum operation, rather than simulate it in full.

The Solovay-Kitaev (SK) theorem ensures that this approximation can be done efficiently. This fundamental theorem demonstrates that if a set of single-qubit quantum gates generates a dense subset of $SU(2)$, then this set is guaranteed to fill $SU(2)$ quickly [18]. In other words, any quantum operation can be approximated using unexpectedly short sequences of gates from a universal set.

The SK theorem is significant when one is restricted in the gates that can be used in computations. For example, a requirement for fault-tolerant quantum computation may restrict you to a small gate set, whereas a more complex algorithm (e.g. Shor's) may require a wider variety of gates. This theorem shows that this limited set can be used to build up all the gates in Shor's algorithm, maintaining algorithmic efficiency while meeting conditions for fault tolerance.

2.2 Other models

2.2.1 Adiabatic quantum computation

Rather than employing a finite set of gate operations, adiabatic quantum computing (AQC) involves the evolution of an initial Hamiltonian into a final Hamiltonian, whose ground state encodes the solution. This model relies on the adiabatic theorem, first conjectured by Max Born and Vladimir Fock [19]:

> "A physical system remains in its instantaneous eigenstate if a given perturbation is acting on it slowly enough and if there is a gap between the eigenvalue and the rest of the Hamiltonian's spectrum."
>
> - Born and Fock, 1928

To perform computations using AQC, 2 Hamiltonians are prepared: a problem Hamiltonian H_P, whose ground state encodes the solution, and an initial Hamiltonian H_B, specifically chosen so that its ground state is easy to find. Having prepared a set of qubits in this initial ground state, the system evolves via the Schrodinger equation with the following Hamiltonian:

$$H(t) = \left(1 - \frac{t}{T}\right) H_B + \frac{t}{T} H_P, \tag{2.5}$$

where T represents the total evolution time. As long as this Hamiltonian evolves slowly enough, and there is a nonzero energy gap between the ground and first excited states at all times, the system remains in the ground state throughout the evolution. Hence, the adiabatic theorem ensures that the final state holds the solution. Defining the minimum energy gap during the evolution as g_{min}, the theorem dictates that the size of T is governed by g_{min}^{-2} [20].

With a suitable choice of Hamiltonians, AQC can been shown to be polynomially equivalent to the quantum circuit model [15].

2.2.2 Quantum annealing

The idealistic nature of AQC makes it difficult to implement physically. For example, a truly adiabatic computation would run in perfect isolation, with no interference from the environment. Relaxing the adiabatic condition leads to the quantum annealing (QA) model.

QA is a method for finding the global minimum of an objective function (e.g. a potential energy function), and it works in a similar way to AQC. Again, there is an initial and a problem Hamiltonian, and the idea is to evolve the system from the ground state of the former to the latter. The idea behind QA is to search the space of potential solutions before driving the system towards the energy minimum. Compared to classical, or simulated annealing, which relies on thermal fluctuations

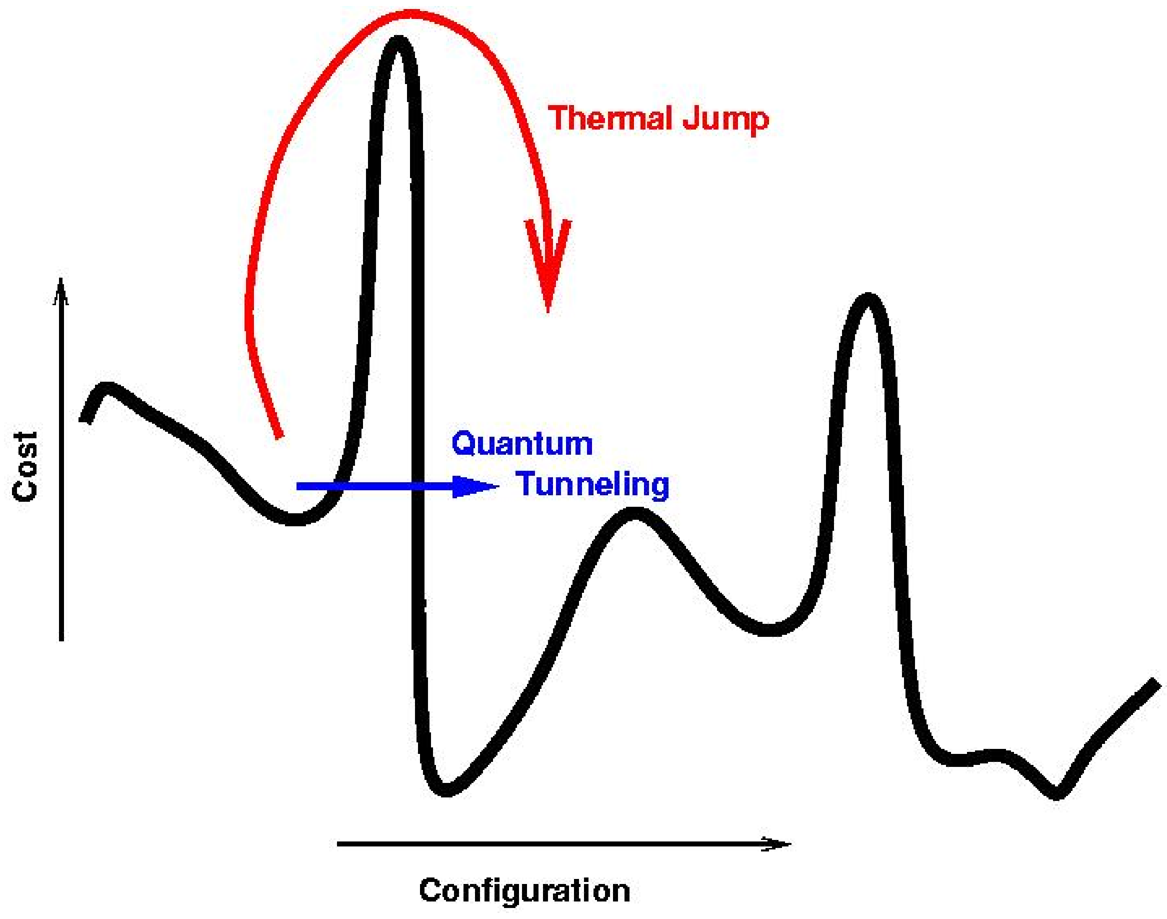


Figure 2.3: Quantum tunneling through barriers provide a more effective way of traversing the energy landscape compared to thermal jumps [21].

to 'jump' over barriers, QA can exploit quantum tunnelling to explore the solution space more efficiently (see Figure 2.3).

D-Wave systems are considered pioneers in quantum annealing, having produced several commercial annealers throughout the last decade. An example Hamiltonian used by the D-Wave system is [22]:

$$H_{ising} = -\frac{A(s)}{2}\left(\sum_i \hat{\sigma}_x^{(i)}\right) + \frac{B(s)}{2}\left(\sum_i h_i \hat{\sigma}_z^{(i)} + \sum_{i>j} J_{i,j}\hat{\sigma}_z^{(i)}\hat{\sigma}_z^{(j)}\right), \tag{2.6}$$

h_i and $J_{i,j}$ represent the strength of each qubit's coupling to the external field and to each other via entanglement. $s = t/T$ is the normalised anneal fraction, going from 0 to 1 as the anneal progresses. At $t = 0$, $A(0) \gg B(0)$, so the initial ground state is easily initialisable, with each spin delocalised. Over time, A decreases while B increases, until the end where $B(1) \gg A(1)$: at this point, the system resembles the classical Ising spin system:

$$E_{ising} = \sum_i h_i s_i + \sum_{i>j} J_{i,j} s_i s_j. \tag{2.7}$$

Unlike AQC, QA is not equivalent to the quantum circuit model. Quantum annealers cannot perform procedures such as Shor's algorithm; rather they are limited to combinatorial optimisation problems. QA has a major upside over other models since it can operate in a realistic, energy-dissipative regime. Decoherence does not have an adverse effect on the performance of QA or AQC, hence quantum annealers have proven to be much easier to scale.

2.2.3 Measurement based quantum computation

Unlike other quantum computing models, there is no classical analogue of measurement based quantum computation (MBQC). Other models such as quantum circuits utilise measurement only as a final step in the computation, whereas MBQC also includes them for intermediate steps. In general, an initial entangled state of qubits is prepared, and computations are performed by applying certain measurements to designated qubits in designated bases [23].

There are two main MBQC schemes: teleportation quantum computation and the one-way quantum computer. The former employs Bell measurements across multiple qubits, with an initial state containing bipartite entangled pairs. Meanwhile the latter uses single-qubit measurements, and an initial cluster state with many entangled qubits. More details on both schemes can be found in [23]. Again, MBQC is computationally equivalent to the quantum circuit model.

2.2.4 Topological quantum computing

We have already encountered how noise and decoherence can be a hindrance to qubit stability in quantum computing. Topological quantum computing aims to address this issue by using anyons; two-dimensional quasiparticles with peculiar properties. Bosons and fermions are characterised by their wavefunctions being symmetric or antisymmetric under identical particle exchange:

$$|\psi_1\psi_2\rangle = \pm\,|\psi_2\psi_1\rangle\,. \tag{2.8}$$

Meanwhile, for anyons, the wavefunction may be neither symmetric nor antisymmetric after particle exchange:

$$|\psi_1\psi_2\rangle = e^{2\pi\theta i}\,|\psi_2\psi_1\rangle\,. \tag{2.9}$$

This exchange of anyons is called braiding, and these braids form the logic gates used for computation. These quantum braids are much more resilient than standard quantum particles. Information is not lost as easily, since small perturbations which cause qubit decoherence do not affect the topological properties of the braids.

In this model, qubits are initialised as anyons, before braiding is done to perform quantum gate operations. Finally, the anyons are brought together (fused), and the fusion outcomes are measured to determine the output of the computation [24].

Topological quantum computation is also computationally equivalent to the quantum circuit model.

Chapter 3

Quantum complexity theory

3.1 Church-Turing-Deutsch principle

Another major result to come out of Deutsch's landmark 1985 paper [8] is the formulation of the Church-Turing-Deutsch principle. This is a stronger form of the original Church-Turing thesis conjectured by Turing and Alonzo Church in 1936. This is a statement used to formalise the idea of computability:

Church-Turing thesis

Every 'function which would naturally be regarded as computable' can be computed by the universal Turing machine.

Over the years there have been several updates or variations to this original thesis. In the 1970s, the thesis was strengthened to include the word 'efficiently': any algorithm can be simulated by a TM with polynomial overhead. The type of Turing machine was then specified as 'probabilistic', after it was realised that randomised algorithms could efficiently solve problems inaccessible to a standard deterministic program.

Deutsch was dissatisfied with all these formulations, as they lacked both the precision and physical foundation present in principles such as the law of thermodynamics. The physical version of the thesis, now known as the Church-Turing-Deutsch principle, is as follows:

Church-Turing-Deutsch principle

Every finitely realizable physical system can be perfectly simulated by a universal model computing machine operating by finite means [8].

This principle, which refers to simulating real physical processes, is strong. So strong in fact, that Turing machines do not satisfy this principle for classical physics. This is because the states of a classical system form a continuum, whereas there are only countably many ways of preparing a finite input for a TM [8]. However, the universal quantum computer, based on the principles of quantum mechanics, is able to simulate any physical system with a finite-dimensional state space.

Proving this principle is satisfied in nature is one of the major open problems within quantum computing. Until the dynamics of the universe are understood better, it remains a possibility that a physical theory may emerge that cannot be simulated by a universal device.

On the other hand, assuming that the Church-Turing-Deutsch principle holds, we can use it to guide the formulation of new physical theories. This is similar to how principles such as the conservation of energy principle have been used: any proposed new theory which does not satisfy an energy conservation law is usually rejected immediately.

We will discuss various examples of quantum simulation of physical systems in Chapter 6. As physical implementations of quantum computers continue to scale, our capacity to prove that particular theories can be simulated will be augmented. Thus, the hope is that the Church-Turing-Deutsch principle will become corroborated, and ultimately turned into a key guiding principle within physics research.

3.2 Quantum supremacy

It is illogical to design quantum algorithms without serious thought about their utility and necessity. Until quantum computers become mainstream, we must be selective about the tasks which we choose to tackle with these machines. Classical algorithms that solve a substantial volume and variety of problems already exist. Many of these perform so efficiently that searching for an improved algorithm is a rather pointless task.

What designers must search for are quantum algorithms which offer significant advantages over their classical counterparts. One of the major milestones that quantum computing researchers have been working towards is the demonstration of quantum supremacy. This is the goal of exhibiting a quantum computer's ability to solve a problem which no classical machine can solve in a reasonable amount of time. Although the concept of seeking quantum advantage has existed since the dawn of quantum computing, the term 'quantum supremacy' was introduced by John Preskill in 2012 [25].

Achieving this objective is a difficult task. Not only must a powerful quantum computer be constructed, but a suitable problem must be found, one which a quantum computer possesses an overwhelming advantage over the most powerful supercomputers. Google finally reached this benchmark in 2019 with their 53-qubit Sycamore quantum processor. Google's computer performed a random quantum circuit sampling experiment in 200 seconds; a state-of-the-art supercomputer is estimated to take 10,000 years to do the same task [26].

There are some caveats with this seemingly groundbreaking discovery. Firstly, IBM disputed Google's claim of quantum supremacy, as they believed that their supercomputer 'Summit' could actually do the calculation in just 2.5 days rather than 10,000 years. Furthermore, some scientists dispute the real significance of achieving this goal. The task which Google performed has little-to-no practical use: achieving quantum supremacy was simply a scientific goal which piqued the interest of the media and quantum enthusiasts.

There is also the possibility of classical supercomputers and algorithms improving in the future, suggesting that quantum supremacy may be only a temporary achievement. Experiments continue to be proposed and carried out to demonstrate quantum supremacy (e.g. [27]); this will prove to be a defining concept in quantum complexity theory for the foreseeable future.

3.3 Computational complexity

3.3.1 Computational problems

It is important to be able to classify and compare problems based on their difficulty. A 'difficult' problem requires a substantial number of resources (for computational complexity, time and space/memory) to find its solution.

Each computational problem has an input, known as an *instance*, and an output, or *solution*. For example, a sorting problem may have an instance $(8, 4, 6, 3, 7)$ and an output of $(3, 4, 6, 7, 8)$. These instances are usually encoded as binary strings, although when dealing with graph problems an adjacency matrix is used instead.

There are several types of problems we could assess while studying computational complexity theory, but the most commonly used are decision problems - any problem with a yes or no answer. Essentially, an input string is fed into an algorithm which either accepts (outputs yes/1) or rejects (outputs no/0) the input. The algorithm is deciding whether the input is a member of a certain *language* - e.g. 'Is this number a prime?'.

Going further, it is helpful to define promise problems, a central concept in this subject. These are a generalisation of decision problems, where the input is assumed to be taken from a subset of all possible inputs. These problems can be written as a pair $A = (A_{yes}, A_{no})$, where A_{yes} and A_{no} represent the set of yes-instances and no-instances. These sets of strings satisfy $A_{yes} \cap A_{no} = \emptyset$ [28].

3.3.2 Big O notation

Before diving into the details of complexity theory we must introduce some mathematical notation. Big O, or asymptotic notation is a tool used to classify functions or algorithms based on their growth rate as a function of input size. Its formal definition is as follows:

Big O

Define functions $f, g : \mathbb{N} \to \mathbb{R}$.
$f(n) = O(g(n))$ if $\exists\ a \in \mathbb{R}_*^+,\ \exists\ n_0 \in \mathbb{R} \mid f(n) \leq ag(n)\ \forall\ n > n_0$

In other words, $g(n)$ is an upper bound on $f(n)$ as n becomes very large. This means that we can make simplifications when analysing functions: only the highest exponent needs to be considered, and any constant factors can be ignored. To illustrate this, take $f(n) = \log n + 3n^5 + 12n^3$. The highest exponent is n^5, so ignoring other terms and dropping the constant, we can write $f(n) = O(n^5)$. This notation is very useful when studying worst-case behaviour of algorithms [12].

Notation	Name	Example problem
$O(1)$	Constant	Calculating $(-1)^n$
$O(\log n)$	Logarithmic	Binary search
$O(n)$	Linear	Linear search
$O(n \log n)$	Linearithmic	Merge sort
$O(n^2)$	Quadratic	Bubble sort
$O(n^c)$	Polynomial	AKS primality test
$O(c^n)$	Exponential	Traveling salesman problem using dynamic programming
$O(n!)$	Factorial	Traveling salesman problem using brute-force search

Table 3.1: Common classes of functions, c is a positive constant.

Big O is actually a member of a family of asymptotic notations called Bachmann-Landau notations, named after two of their inventors. We summarise this family in Table 3.2.

3.3.3 Computational complexity classes

Sets of computational problems with similar complexity are grouped into complexity classes. To define these classes we need three components; a type of problem (e.g. decision), a model of computation and a computational resource (time or space). For classical computing the Turing machine is used as the model of computation.

Name	**Notation**	**Definition**	**Description (Asymptotically)**
Big O	$f(n) = O(g(n))$	$\exists\ a > 0, \exists\ n_0 \mid \forall\ n > n_0,$ $f(n) \leq ag(n)$	f is bounded above by g
Big Omega	$f(n) = \Omega(g(n))$	$\exists\ a > 0, \exists\ n_0 \mid \forall\ n > n_0,$ $f(n) \geq ag(n)$	f is bounded below by g
Big Theta	$f(n) = \Theta(g(n))$	$\exists\ a, b > 0, \exists\ n_0 \mid \forall\ n > n_0,$ $ag(n) \leq f(n) \leq bg(n)$	f is bounded above and below by g
Small O	$f(n) = o(g(n))$	$\forall\ a > 0, \exists\ n_0 \mid \forall\ n > n_0,$ $f(n) \leq ag(n)$	f is dominated by g
Small Omega	$f(n) = \omega(g(n))$	$\forall\ a > 0, \exists\ n_0 \mid \forall\ n > n_0,$ $f(n) \geq ag(n)$	f dominates g
On the order of	$f(n) \sim (g(n))$	$\forall\ \delta > 0, \exists\ n_0 \mid \forall\ n > n_0,$ $\left\lvert \frac{f(n)}{g(n)} - 1 \right\rvert < \delta$	f is equal to g

Table 3.2: Bachmann-Landau notation. The final column describes the asymptotic behaviour of f as $n \to \infty$

The TM may be deterministic (one possible action at each step), nondeterministic (several possible actions) or probabilistic (action is determined randomly according to some probability distribution). We also use an abstract notion of time (number of steps the TM needs to solve a problem) and space (number of cells used on the TM's tape) for ease of comparison. For quantum computing either the QTM or quantum circuit model may be used, as they are computationally equivalent.

For the latter, time is quantified by circuit *depth*, which is the number of time steps required, or equivalently the maximum length of a path from input to output. Meanwhile, space is quantified by *width*; the maximum number of gates that act in any one time step [29]. Some important complexity classes for classical and quantum computation are detailed in Tables 3.3 and 3.4.

The P vs NP problem is one of the seven Millennium Prize problems, a key unresolved issue in computer science. Problems in NP can have their solutions quickly verified (in polynomial time), while a problem in P can be solved quickly: the question is whether P = NP.

This is related to the concepts of NP-hardness and NP-completeness. A problem A is NP-hard if every problem in NP can be reduced to A in polynomial time: this problem is "at least as hard as the hardest problems in NP". A caveat appears as NP-hard problems need not to be in NP; an example of this is the halting problem, which Turing proved to be undecidable [2]. NP-complete problems, such as the travelling salesman problem, are both in NP and NP-hard.

Most researchers believe P $\neq$ NP, which implies that it is impossible to find polyno-

Class	**Informal definition**	**Formal criteria**
P	Problem can be **solved** by a **deterministic** classical computer in polynomial time	Problem A for which a polynomial time deterministic TM accepts/rejects all strings in A_{yes}/A_{no}
NP	Solution can be **checked** by a **deterministic** classical computer OR problem can be **solved** by a **nondeterministic** classical computer in polynomial time	Problem A for which there exists a polynomial function p and a polynomial time deterministic TM M with the following properties: • For every string $x \in A_{yes}$, M accepts (x, y) for some string y of length $p(\|x\|)$ • For every string $x \in A_{no}$, M rejects (x, y) for all strings y of length $p(\|x\|)$
BPP	Problem can be **solved** by a **probabilistic** classical computer in polynomial time	Problem A for which a polynomial time probabilistic TM accepts all strings in A_{yes} with probability **at least 2/3**, and accepts all in strings in A_{no} with probability **at most 1/3**
P-SPACE	Problem can be **solved** by a **deterministic** classical computer in polynomial space	Problem A for which a deterministic TM running in polynomial space accepts/rejects all strings in A_{yes}/A_{no}

Table 3.3: Useful classical complexity classes. All defined based on a promise problem $A = (A_{yes}, A_{no})$. 'Polynomial time' means "in time polynomial in the input size" [30]. Table inspired by [28].

mial time solutions to NP-hard problems. However, if the contrary is proven to be true, there would be grave consequences in areas such as cryptography, which relies on certain problems such as 3-SAT being NP-complete.

Turning attention to the quantum complexity classes, BQP can be seen as the quantum analogue to BPP. A decision problem is in BQP if there is a polynomial-time quantum algorithm which solves it with a probability of at least 2/3. From the definitions in Tables 3.3 and 3.4, it is clear to see the parallels between the P-NP relationship in classical complexity, and the BQP-QMA relationship in quantum complexity. There is also a similar notion of completeness for both BQP and QMA. The k-local Hamiltonian problem (for k $\geq$ 2) is a notable example of a QMA-complete problem.

Relationships between complexity classes are still to be resolved, due to unsolved problems such as the aforementioned P vs NP issue. Since deterministic computers

Class	**Informal definition**	**Formal criteria**
BQP	Problem can be **solved** by a quantum computer in polynomial time	$A \in$ BQP(a, b) iff, for functions $a, b : \mathbb{N} \rightarrow [0, 1]$, there exists a polynomial-time generated family of quantum circuits $Q = \{Q_n : n \in \mathbb{N}\}$, where each circuit Q_n takes in n input qubits and produces one output qubit, with the following properties: • If $x \in A_{yes}$ then Pr[Q accepts x] $\geq a(\lvert x \rvert)$ • If $x \in A_{no}$ then Pr[Q accepts x] $\leq b(\lvert x \rvert)$ Define BQP = BQP(2/3, 1/3)
QMA	Solution can be **checked** by a quantum computer in polynomial time	$A \in$ QMA$_p(a, b)$ iff, for a polynomial function p and functions $a, b : \mathbb{N} \rightarrow [0, 1]$, there exists a polynomial-time generated family of quantum circuits $Q = \{Q_n : n \in \mathbb{N}\}$, where each circuit Q_n takes in $n + p(n)$ input qubits and produces one output qubit, with the following properties: • For all $x \in A_{yes}$, there exists a $p(\lvert x \rvert)$-qubit quantum state ρ such that Pr[Q accepts (x, ρ)] $\geq a(\lvert x \rvert)$ • For all $x \in A_{no}$ and all $p(\lvert x \rvert)$-qubit quantum states ρ it holds that Pr[Q accepts (x, ρ)] $\leq b(\lvert x \rvert)$ Define QMA = $\cup_p$QMA$_p$(2/3, 1/3)

Table 3.4: BQP and QMA quantum complexity classes, defined based on a promise problem $A = (A_{yes}, A_{no})$. Usually defined as BQP(2/3, 1/3) and QMA (2/3 'Polynomial time' means "in time polynomial in the input siz" [30]. Table inspired by [28].

are a special type of probabilistic machine, and quantum circuits are able to simulate all classical circuits [12], we can say:

$$P \subseteq BPP \subseteq BQP. \tag{3.1}$$

As we will see later, certain NP problems, such as integer factorisation can be solved by quantum algorithms. Hence these problems are in BQP, though the relationship between NP and BQP is still to be confirmed. Figure 3.1 visualises the suspected relationship between selected classes with the current assumption that P $\neq$ NP. Just as P $\subseteq$ NP, BQP $\subseteq$ QMA; in fact, all problems in P, NP and BQP lie within QMA.

I cannot give full justice to this extremely rich subject in such a short report; the reader is guided to [28, 31] for more details and mathematical proofs of the relationships between complexity classes.

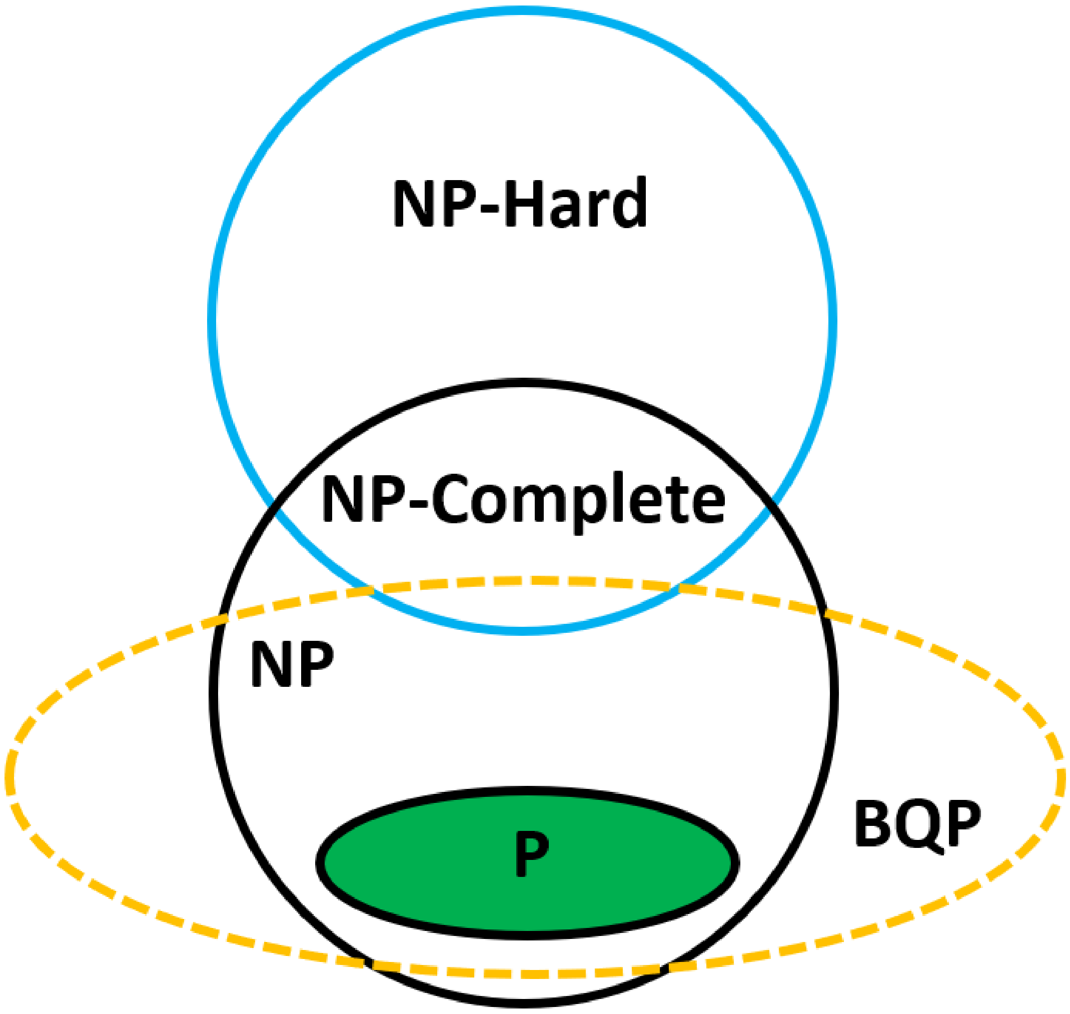


Figure 3.1: Euler diagram for P, NP, NP-complete, NP-hard and BQP problems.

3.4 Query complexity

The majority of known quantum algorithms are suitable for analysis using the query model. For these *query* algorithms, the input is no longer encoded as an initial binary string. Instead, the input is encoded in a black box which computes some function f, and each application of the black box is known as a query. A black box can only be analysed by its inputs and outputs; we have no knowledge of its inner workings. The query complexity of an algorithm is simply the number of queries made to the black box (also called oracle) to solve the problem.

The same big O notation used for computational complexity is used here, describing query complexity as a function of $N = 2^n$, where n is the number of input qubits. For example, the hidden shift problem can be solved by a quantum algorithm using $O(\log N)$ queries, compared to $O\sqrt{N}$ for the best classical algorithm.

Quantum circuits incorporate these quantum oracles as unitary transformations. For

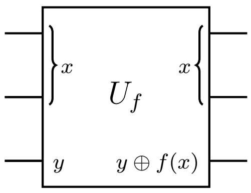


Figure 3.2: Quantum oracle where $f : \{0,1\}^2 \to \{0,1\}$. All quantum circuits in this report were drawn using [33].

instance, for a function $f : \{0,1\}^n \to \{0,1\}$, the oracle maps $|x\rangle |y\rangle \mapsto |x\rangle |y \oplus f(x)\rangle$ (for all $x \in \{0,1\}^n$ and $y \in \{0,1\}$) [32]. This means that the query complexity provides a lower bound on the overall time complexity, as the latter takes into account all gate operations, not just the black box. Note that quantum algorithms can apply the oracle to a superposition of basis states, a key property exploited by David Deutsch and Richard Jozsa in their primitive algorithm.

3.5 Deutsch-Jozsa algorithm

Despite having little practical use, the Deutsch-Jozsa (D-J) algorithm retains significance in quantum computational complexity. It is one of the first examples of a quantum algorithm solving a problem exponentially faster than the best deterministic classical algorithm. However, the problem at hand is a black-box problem, so it makes more sense to look at the query complexity. In the worst case. a classical algorithm requires exponentially more queries to the oracle than the D-J algorithm, which solves the problem with just one query.

The D-J problem involves an oracle which computes a function $f : \{0,1\}^n \to \{0,1\}$, with the promise that f is either constant (outputs either 0 for all inputs or 1 for all inputs), or balanced (outputs 0 for exactly half the possible inputs). The task is to use the oracle to work out f's state.

3.5.1 Classical solution

A classical deterministic algorithm may get lucky; in the best case only **two** queries to the oracle are needed. For example, for $n = 3$ there are $2^3 = 8$ possible inputs: $\{(0,0,0), (0,0,1), \ldots, (1,1,1)\}$ (0-7 in binary), which each need to be tested in succession. If the first two queries determine that $f(0,0,0) = 1$ and $f(0,0,1) = 0$, then we know for certain that f is balanced.

In the worst case, f needs to be evaluated five times (or $\mathbf{2^{n-1}+1}$ for general n) to be certain whether it is constant or balanced. If we checked the first four possible inputs and f output 1 each time, there is still a slight possibility that the fifth input may return a 0, meaning f is balanced, not constant. If 100% certainty is not required, less queries are needed. A classical randomised algorithm can determine f's state with error probability ϵ using only $O(\log \frac{1}{\epsilon})$ queries [11].

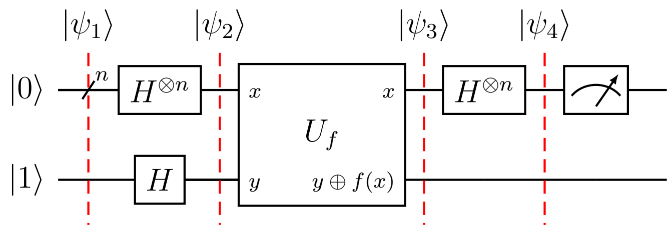


Figure 3.3: Quantum circuit for the Deutsch-Jozsa algorithm.

3.5.2 Quantum solution

The circuit used for the quantum solution is illustrated in Figure 3.3, and the corresponding steps for the D-J algorithm are summarised in Algorithm 2. Initially, two quantum registers are prepared: an n-qubit 'query' register initialised to $|0\rangle$, and a one-qubit 'answer' register set to $|1\rangle$. Hadamard gates are then applied to all $n+1$ qubits; for the query register, n Hadamard gates are applied in parallel ($H^{\otimes n}$), producing the state $\frac{1}{\sqrt{2^n}}\sum_x |x\rangle$. Using just n gates, a superposition of 2^n states is created.

Next the oracle is applied; as before it maps $|x\rangle|y\rangle \mapsto |x\rangle|y \oplus f(x)\rangle$, where $\oplus$ is addition modulo 2. Knowing that $f(x)$ is either 0 or 1, it is straightforward to see that

$$|x\rangle(|f(x)\rangle - |1 \oplus f(x)\rangle) = (-1)^{f(x)}|x\rangle(|0\rangle - |1\rangle), \tag{3.2}$$

which explains step 3 of the algorithm. Hadamard gates are then applied again to the n-qubit query register. By explicit calculation for $x=0$ and $x=1$, we can see that $H|x\rangle = \sum_y (-1)^{xy}|y\rangle/\sqrt{2}$. Generalising to n qubits:

$$H^{\otimes n}|x\rangle = \frac{\sum_y (-1)^{x.y}|y\rangle}{\sqrt{2^n}}, \tag{3.3}$$

with $x \cdot y = x_0y_0 \oplus x_1y_1 \oplus \ldots \oplus x_{n-1}y_{n-1}$ representing the bitwise inner product (modulo 2). Finally, a measurement is made on the query register. Note that the probability of measuring the state $|0\rangle^{\otimes n}$ is $\left|\frac{1}{2^n}\sum_x (-1)^{f(x)}\right|^2$. If f is constant, this probability is equal to 1 (the amplitude would be +1 for constant value 0 and -1 for constant value 1). If f is balanced, the positive and negative contributions to the amplitude cancel (destructive interference). Hence, the probability of measuring $|0\rangle^{\otimes n}$ is 0.

Therefore, with just one query to the oracle, we can determine whether f is constant or balanced.

3.5.3 Related algorithms

The D-J algorithm has been built to create algorithms to solve more complex problems. This includes the Bernstein-Vazirani (B-V) algorithm [31], which again solves a promise problem; this time $f(x) = x \cdot s$, and the task is to find this hidden string s.

Algorithm 2 Deutsch-Jozsa algorithm for determining whether a function is constant or balanced. First proposed in 1992, later improved so that only a single query of f is required.

```
1: $|\psi_1\rangle = |0\rangle^{\otimes n}|1\rangle$    ▷ Initialise $(n+1)$-qubit state
2: $|\psi_2\rangle = \sum_{x=0}^{2^n-1} \frac{|x\rangle}{\sqrt{2^n}}[\frac{|0\rangle-|1\rangle}{\sqrt{2}}]$    ▷ Create superposition state using Hadamard gates
3: $|\psi_3\rangle = \sum_{x=0}^{2^n-1} \frac{|x\rangle}{\sqrt{2^n}}[\frac{|f(x)\rangle-|1\oplus f(x)\rangle}{\sqrt{2}}]$    ▷ Apply oracle to evaluate $f$
       $= \sum_{x=0}^{2^n-1} \frac{(-1)^{f(x)}|x\rangle}{\sqrt{2^n}}[\frac{|0\rangle-|1\rangle}{\sqrt{2}}]$
4: $|\psi_4\rangle = \sum_{y=0}^{2^n-1}\sum_{x=0}^{2^n-1} \frac{(-1)^{f(x)+x.y}|y\rangle}{2^n}[\frac{|0\rangle-|1\rangle}{\sqrt{2}}]$    ▷ Apply Hadamard gate to first $n$ qubits
5: Measure $|z\rangle$    ▷ Determine if $f$ is constant or balanced
```

This algorithm solves the problem with 1 query, compared to n queries for a classical solution.

The problems solved by the D-J and B-V algorithms are not classically difficult enough to really prove an oracle separation between BQP and BPP [29]. B-V's algorithm only demonstrates a linear discrepancy between these two classes, and we showed earlier that D-J's problem is relatively easy to solve on a probabilistic classical machine. D-J's algorithm instead proves a separation between EQP, the class of problems that can be solved **exactly** on a quantum computer in polynomial time, and P.

This issue was addressed in 1993 by Dan Simon [34], who found a problem which yields an exponential oracle separation between BQP and BPP. The problem is as follows:

Simon's problem

Given a black box U_f implementing $|x\rangle|y\rangle \mapsto |x\rangle|y\oplus f(x)\rangle$ for a function $f:\{0,1\}^n \to \{0,1\}^n$, with the promise that $f(x)=f(y)$ iff $x\oplus y \in K = \{0^n, s\}$ for some $s\in\{0,1\}^n$, find s.

Alternatively, the task is to discern whether f is one-to-one ($s=0^n$, a string of zeros), or two-to-one ($s\neq 0^n$, $f(x)=f(x\oplus s)$).

This is a hard problem to solve classically: again, we could get lucky and stumble upon a solution to $f(x)=f(y)$ in the first two queries. In general, we'd have to do an exponential number of queries before the probability of finding a solution is reasonable; the best known classical algorithm has a lower bound of $\Omega(2^{n/2})$.

Meanwhile, using a succession of Hadamard gates, black box queries and measurements, Simon's algorithm solves the problem (with small error probability) with $O(n)$ queries. A good description of the full algorithm can be found in [11].

Like the other black-box algorithms discussed in this section, there are no realistic applications for Simon's algorithm. However, many of the ideas from Simon's procedure were replicated by Shor in his groundbreaking algorithm for factoring. This discovery signalled a shift towards the development of quantum algorithms with real-life, practical applications.

Chapter 4

Factoring, discrete logarithms and the abelian hidden subgroup problem

Naturally accompanying the growth of modern computing and the Internet has been the desire for secure communication protocols. Research into cryptography became popular in the 1970s, with the 1977 formulation of RSA a major early breakthrough in cryptosystem development. RSA belongs to a family of public-key cryptosystems; these systems use a pair of keys (public and private) to enable messages to be encrypted, sent and decrypted by the intended receiver.

The security of each public key algorithm is dependent on certain problems being computationally difficult (not in P). For RSA, this problem is integer factorisation. Any intruder would require the ability to factor a large number in order to recover the private key and break the decryption. For a large semiprime (product of two large primes), this is a daunting task for classical computers.

However, the creation of Shor's algorithm and development of quantum computers presents a real threat to RSA and similar cryptosystems. This is such a concern that post-quantum cryptography is a major active field of research: see [35] for a good introduction to this area.

4.1 Integer factorisation

Integer factorisation problem

Given an integer N, output positive integers $p_1, p_2, \ldots, p_n, q_1, q_2, \ldots, q_n$ where the p_i are distinct primes and $N = p_1^{q_1} p_2^{q_2} ... p_n^{q_n}$.

4.1.1 Classical algorithms

While we have yet to prove that such an algorithm cannot exist, there are no polynomial-time classical algorithms for the factorisation of a large integer N. Several algorithms do exist that are sufficiently rapid for smaller integers. A school student tasked with factoring a 2 or 3-digit number would probably employ some sort of brute force approach. This algorithm is formally known as trial division, and simply entails checking if N is divisible by each prime up to $\sqrt{N}$.

More complicated methods such as the quadratic sieve are based on finding numbers satisfying a *congruence of squares*. This is a weakened version of the famous equality for odd integers:

$$N = x^2 - y^2. \tag{4.1}$$

If an integer N can be written in this form, it is decomposible as $(x + y)(x - y)$. The weaker congruence of squares condition is as follows:

$$x^2 \pmod{N} \equiv y^2, \tag{4.2}$$

and this implies

$$(x + y)(x - y) = 0 \pmod{N}. \tag{4.3}$$

Thus the factors are finally retrieved using the Euclidean algorithm, as

$$HCF(x + y, N) \cdot HCF(x - y, N) = N. \tag{4.4}$$

The quadratic sieve is an optimised method for finding these congruences. In short, it entails finding integers a_i such that $\sqrt{N} < a_i < N$ and $b_i = a_i^2 \pmod{N}$ is B-smooth. By B-smooth, we mean that b_i's prime factors $p_i \leq B$, where B is a selected smoothness bound.

A subset of the set b_i is then found whose product is a square b^2. By multiplying the corresponding a_i together $\pmod{N}$ to find a, we have a pair of numbers satisfying (4.2). Hence using (4.4), we obtain a factor decomposition of N.

Example: Factorise $N = 1649$

To illustrate this algorithm, take $N = 1649$, with $B = 5$ (so the factor base is $\{2, 3, 5\}$). Hence $\sqrt{1649} \approx 40.6 < a_i < 1649$. Looking at 3 values of a_i, we have $41^2 \equiv 32, 42^2 \equiv 115$ and $43^2 \equiv 200$ (all $\bmod 1649$), but we ignore $115 \in b_i$ since 115 is not 5-smooth ($115 = 5 \cdot 23$).

We now have a subset of b_i $\{32, 200\}$ whose product is a square ($32 \cdot 200 = 80^2$). Hence, after multiplying the corresponding a_i: $41 \cdot 43 \pmod{1649} = 114$, we have the congruence $114^2 \pmod{1649} = 80^2$. So we can factor $1649 = HCF((114 + 80), 1649) \cdot HCF((114 - 80), 1649) = 97 \cdot 17$.

The precise algorithm actually operates on integers decomposed as a product of prime factors, written as exponent vectors (e.g. $6615 = 2^0 \cdot 3^3 \cdot 5^1 \cdot 7^2 = (0, 3, 1, 2)$) [36]. The quadratic sieve is more efficient than other similar methods (e.g. Dixon's method [37]), as it selects values of a_i close to $\sqrt{N}$. This ensures that the values b_i are small, and hence more likely to be B-smooth. This algorithm is the second fastest known method for factorisation; indeed it is the fastest for integers with less than 100 digits.

The crown for most efficient classical algorithm for factorisation of large integers goes to the general number field sieve (GNFS). This 1993 algorithm has a heuristic runtime for factoring an integer N of the form

$$exp\left(\left((\sqrt[3]{\frac{64}{9}} + o(1))(\log N)\right)^{1/3} (\log\log N)^{2/3}\right) \tag{4.5}$$

.

This is a complicated algorithm which we will not discuss in full here; see [38] for a detailed overview and example. Instead, we will outline the rational sieve; a simpler, special case of the GNFS which introduces some key ideas used in the full algorithm.

Algorithm 3 Rational sieve for factorisation of integer N

```
1: Choose bound $B$, identify factor base $P$
2: Search for integers $z$ such that both $z$ and $z + N$ are B-smooth
                                   ▷ I.e. $z = \prod_{p_i \in P} p_i^{a_i}$, $z + N = \prod_{p_i \in P} p_i^{b_i}$
3: Each z gives a relation: $\prod_{p_i \in P} p_i^{a_i} \equiv \prod_{p_i \in P} p_i^{b_i} \pmod N$
4: Combine relations so that the exponents of the primes are all even
                                   ▷ This gives a congruence of squares (4.2)
5: Use (4.4) to factorise
         ▷ If factors are trivial, repeat step 4 with a different combination of relations
```

The inefficiency of the rational sieve comes in step 2: finding B-smooth numbers is increasingly difficult for larger numbers. While the rational sieve searches for smooth numbers $\sim N$, the GNFS looks for smooth numbers subexponential in N (i.e. $O(N^{1/d})$ for some small integer d). This efficiency has allowed the number field sieve to set records for integer factorisation: a group of researchers factored 17 numbers of the form $2^n - 1$, with n between 1000 and 2000, at the cost of 7500 CPU-years [39].

4.1.2 Useful quantum subroutines

Shor's algorithm incorporates quantum subroutines which help generate the exponential speedup compared to classical algorithms. These procedures, which may be considered algorithms in their own right, also prove to be useful within other algorithms, so it makes sense to introduce them here.

Quantum Fourier Transform

The Quantum Fourier Transform (FT) is simply the quantum analogue of the classical, discrete FT. The latter acts on a N-dimensional vector $(x_0, x_1, \dots, x_{N-1})$, transforming it to the vector $(y_0, y_1, \dots, y_{N-1})$, where

$$y_q = \frac{1}{\sqrt{N}} \sum_{p=0}^{N-1} x_p e^{2\pi i pq/N}. \tag{4.6}$$

Meanwhile, the quantum FT performs the same transformation on a set of orthonormal basis states $|0\rangle, |1\rangle, \dots, |N-1\rangle$:

$$QFT\,|q\rangle = \frac{1}{\sqrt{N}} \sum_{p=0}^{N-1} e^{2\pi i pq/N} |p\rangle . \tag{4.7}$$

This transform can also be viewed as a mapping

$$|x\rangle = \sum_{p=0}^{N-1} x_p |p\rangle \mapsto |y\rangle = \sum_{q=0}^{N-1} y_q |q\rangle , \tag{4.8}$$

where the amplitudes are transformed as per (4.6). The inverse quantum FT can be simply written as

$$|p\rangle = \frac{1}{\sqrt{N}} \sum_{q=0}^{N-1} e^{-2\pi i pq/N} QFT\,|q\rangle . \tag{4.9}$$

The quantum FT is a unitary operation, hence it can be represented as a unitary matrix:

$$QFT = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \dots & \omega^{(N-1)^2} \end{pmatrix} , \tag{4.10}$$

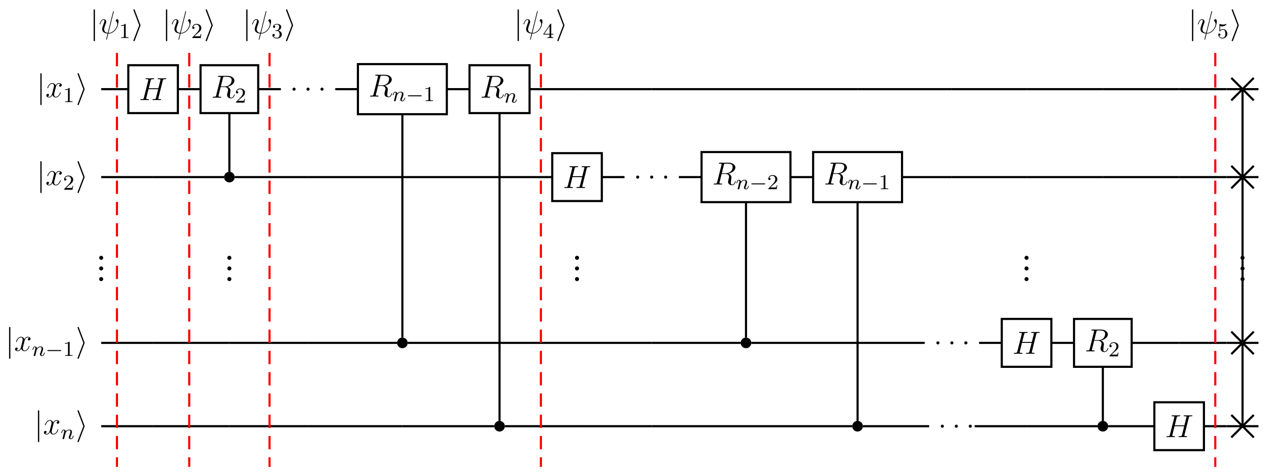


Figure 4.1: Quantum circuit for the quantum Fourier transform.

where $\omega = e^{2\pi i/N}$, the N^{th} root of unity. Noting that for n qubits, $N = 2^n$, so the quantum FT on a single qubit (QFT_2) is simply the Hadamard gate. This can be checked by calculating $QFT\,|0\rangle$ and $QFT\,|1\rangle$ using (4.7) with $N = 2$.

It is useful to write the quantum FT as a product representation. For this we must write our basis states using binary notation: $|x\rangle = |x_1, x_2, \ldots, x_n\rangle$, i.e. our input is a tensor product of n qubits. Making use of fractional binary notation:

$$0.x_l x_{l+1} \ldots x_m = \sum_{k=l}^{m} x_k 2^{-k}, \tag{4.11}$$

we can rewrite the quantum FT (4.7) as

$$|x_1, x_2, \ldots, x_n\rangle \mapsto \frac{1}{\sqrt{N}}(|0\rangle + e^{2\pi i 0.x_n}\,|1\rangle)(|0\rangle + e^{2\pi i 0.x_{n-1}x_n}\,|1\rangle)\ldots(|0\rangle + e^{2\pi i 0.x_1 x_2 \ldots x_n}\,|1\rangle), \tag{4.12}$$

noting that the tensor product between each qubit is implied.

Using this product representation, we can derive the quantum circuit for a general n-qubit quantum FT. This transform can be realised using a series of Hadamard gates and controlled-phase gates (controlled-R_k), where

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k}. \end{pmatrix} \tag{4.13}$$

The full circuit for quantum FT is shown in Figure 4.1, and the corresponding algorithm is displayed in Algorithm 4.

This circuit is built from unitary quantum gates, cementing the fact that the whole operation is unitary. This means that to implement the inverse quantum FT, the circuit simply needs to be reversed, replacing each gate with their inverse ($QFT^{-1} =$

Algorithm 4 Quantum Fourier transform

```
1: $|\psi_1\rangle = |x_1, x_2, \dots, x_n\rangle$    ▷ Input $n$-qubit state
2: $|\psi_2\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.x_1}|1\rangle)|x_2, \dots, x_n\rangle$    ▷ Apply Hadamard to 1st qubit
3: $|\psi_3\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.x_1 x_2}|1\rangle)|x_2, \dots, x_n\rangle$
                                        ▷ Apply controlled-$R_2$ gate to 1st qubit
4: $|\psi_4\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.x_1 x_2 \dots x_n}|1\rangle)|x_2, \dots, x_n\rangle$
                                        ▷ Apply all $n$ controlled-$R_n$ gates to 1st qubit
5: $|\psi_5\rangle = \frac{1}{\sqrt{N}}(|0\rangle + e^{2\pi i 0.x_1 x_2 \dots x_n}|1\rangle)(|0\rangle + e^{2\pi i 0.x_2 \dots x_n}|1\rangle)\dots(|0\rangle + e^{2\pi i 0.x_n}|1\rangle)$
                                        ▷ Apply a similar sequence of gates for qubits 2 . . . n
6: Apply SWAP gates to reverse order of qubits to obtain state as in (4.12)
```

$QFT^\dagger$).

This transform on n qubits is implemented with just $\Theta(n^2)$ gates, exponentially less operations than the most efficient classical algorithms for calculating the discrete FT. For example, the Fast Fourier Transform requires $\Theta(n2^n)$ gates. Optimised quantum FTs require even less gates; researchers in 2000 created an algorithm which completes the operation with only $O(n \log n)$ gates [40].

However, amongst other issues, differences in setup (classical FTs act on vectors while quantum FTs transform quantum states) makes the speedup of the quantum procedure difficult to realise in practice. Unless the input has been pre-encoded into a quantum state (or can be encoded in $O(\log N)$ steps), exponential speedup is not feasible [15]. Therefore, we cannot simply substitute the classical FT for the quantum FT to speedup every task; we must be more subtle to discover the real utility of this subroutine.

Quantum Phase Estimation

A key application of the quantum FT is within the phase (or eigenvalue) estimation algorithm, formalised by Alexei Kitaev [41]. The problem is simple:

Quantum phase estimation

Given a unitary operator U with an eigenvector $|u\rangle$ and eigenvalue $e^{2\pi i\phi}$, estimate ϕ.

The circuit for this algorithm is displayed in Figure 4.2, with the corresponding procedure in Algorithm 5. Two registers are used for this procedure. The number of qubits (a) in the first register determines the accuracy of the estimate, as well as the probability for the algorithm to succeed [12]. The second register simply has enough qubits to store the state $|u\rangle$.

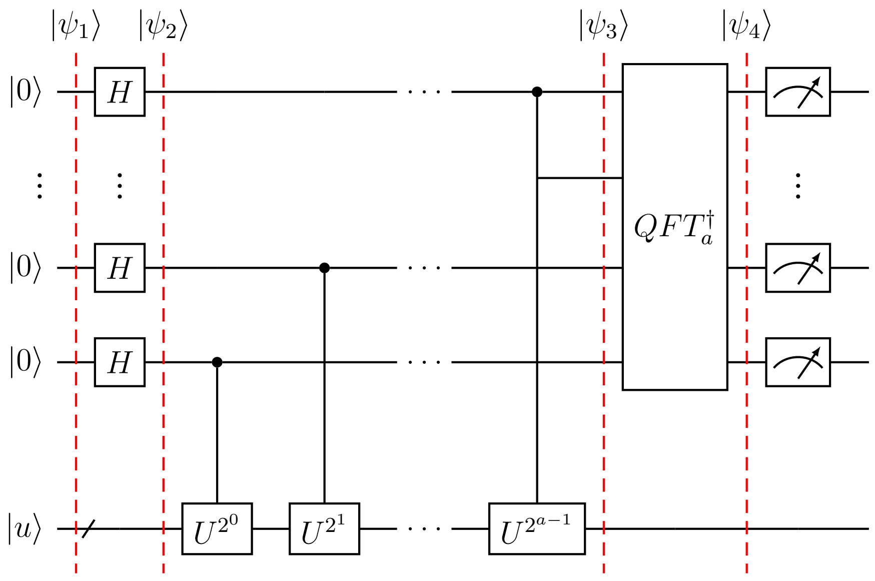


Figure 4.2: Quantum circuit for quantum phase estimation.

Algorithm 5 Quantum phase estimation. Note that here $A = 2^a$

```
1: $|\psi_1\rangle = |0\rangle^{\otimes a} |u\rangle$    ▷ Initialise qubits
2: $|\psi_2\rangle = \frac{1}{\sqrt{A}}(|0\rangle + |1\rangle)^{\otimes a} |u\rangle$    ▷ Create superposition using Hadamard gates
3: $|\psi_3\rangle = \frac{1}{\sqrt{A}}(|0\rangle + e^{2\pi i 2^{a-1}\phi} |1\rangle) \dots (|0\rangle + e^{2\pi i 2^1 \phi} |1\rangle)(|0\rangle + e^{2\pi i 2^0 \phi} |1\rangle) |u\rangle$
       $= \frac{1}{\sqrt{A}} \sum_{j=0}^{2^a-1} e^{2\pi i j \phi} |j\rangle |u\rangle$    ▷ Apply controlled-$U$ gates on 2nd register
4: $|\psi_4\rangle = \left|\tilde{\phi}\right\rangle |u\rangle$    ▷ Apply inverse quantum FT
5: Measure first register to obtain estimate ($\tilde{\phi}$)
```

As per usual, the controlled-U gate only acts on the target if the control qubit is $|1\rangle$. Step 3 of the algorithm can be understood by noting that

$$U^{2^j} |u\rangle = U^{2^j-1} U |u\rangle = U^{2^j-1} e^{2\pi i \phi} |u\rangle = \dots = e^{2\pi i 2^j \phi} |u\rangle , \tag{4.14}$$

for non-negative integers j, and applying the relation

$$|0\rangle |u\rangle + |1\rangle e^{2\pi i 2^j \phi} |u\rangle = (|0\rangle + e^{2\pi i 2^j \phi} |1\rangle) |u\rangle . \tag{4.15}$$

This step can alternatively be seen as a singular black-box implementing a controlled-U^j operation, mapping the state

$$\frac{1}{\sqrt{A}} \sum_{j=0}^{2^a-1} |j\rangle |u\rangle \mapsto \frac{1}{\sqrt{A}} \sum_{j=0}^{2^a-1} |j\rangle U^j |u\rangle . \tag{4.16}$$

In the special case where $\phi = 0.\phi_1\phi_2 \dots \phi_a$, the inverse quantum FT produces the state $|\phi_1 \dots \phi_a\rangle |u\rangle$, and a measurement of the first register produces ϕ exactly. Outside of this ideal case, applying the inverse quantum FT to $|\psi_3\rangle$ produces the state

$$\frac{1}{A}\sum_{x=0}^{2^a-1}\sum_{j=0}^{2^a-1} e^{2\pi ij\phi}e^{\frac{-2\pi ijx}{2^a}}\left|x\right\rangle = \frac{1}{A}\sum_{x=0}^{2^a-1}\sum_{j=0}^{2^a-1} e^{\frac{-2\pi ij}{2^a}(x-2^a\phi)}\left|x\right\rangle, \tag{4.17}$$

now ignoring the 2nd register, which remains as $|u\rangle$ throughout, for brevity.

Now take an integer $c \in [0..2^a-1]$ such that $\frac{c}{2^a} = 0.c_1\dots c_a$ is the best a-bit estimate of ϕ. This means that $\phi = \frac{c}{2^a}+\delta$, where $0<|\delta|\leq\frac{1}{2^{a+1}}$ [42]. Substituting into (4.17) gives the state:

$$\frac{1}{A}\sum_{x=0}^{2^a-1}\sum_{j=0}^{2^a-1} e^{\frac{2\pi i(c-x)j}{2^a}}e^{2\pi i\delta j}\left|x\right\rangle, \tag{4.18}$$

The amplitude of the best-estimate state $|c\rangle = |c_1\dots c_a\rangle$ can be written as a geometric series:

$$\frac{1}{A}\sum_{j=0}^{2^a-1}(e^{2\pi i\delta})^j = \frac{1}{A}\left(\frac{1-(e^{2\pi i\delta})^{2^a}}{1-e^{2\pi i\delta}}\right) \tag{4.19}$$

Squaring this amplitude gives the probability of obtaining the desired state $|c\rangle$, and hence obtaining a good estimate of ϕ after measurement. Note that if $\delta = 0$ as for the exact case, this probability is 100% and the approximation is precise as before. With some mathematical intuition (see [42]), we can show

$$\text{Probability of obtaining } |c\rangle \geq \frac{4}{\pi^2}. \tag{4.20}$$

I mentioned previously that increasing the number of qubits in the first register improves the effectiveness of this algorithm. More precisely, one can obtain ϕ correct to q bits with success probability $\geq 1-\epsilon$ by choosing $\mathbf{a = q + O}(\log(\mathbf{1}/\epsilon))$. Hence with $O(a^2)$ gate operations and one call to the controlled-U^j oracle, an q-bit estimate $\tilde{\phi}$ can be acquired.

4.1.3 Shor's algorithm

We now have the tools to construct Shor's algorithm for integer factorisation [9]. Part of this algorithm is actually classical, but the quantum speedup comes purely within the order-finding process, which incorporates the quantum FT. So the algorithm involves two parts: a classical reduction from factoring to order-finding, and a quantum subroutine for solving the latter problem. The general procedure is outlined in Algorithm 6.

Order-finding

The principles of Shor's algorithm are comparable to other factoring procedures already discussed. The idea is: for a particular N, find a random number x with an

Algorithm 6 Shor's algorithm for factorisation of a composite number

```
1: procedure FIND A NON-TRIVIAL FACTOR OF A COMPOSITE NUMBER($N$)
2:     if $N \mid 2$ then
3:         return 2
4:     else if $N = f^g$ for $f, g \in \mathbb{Z}, f \geq 1, g \geq 2$ then
5:         return $f$
6:     else
7:         Pick a random number $x$, $1 < x < N$
8:         Compute $H = HCF(x, N)$                      ▷ Use Euclidean algorithm
9:         if $H > 1$ then
10:            return H
11:        else
12:            Use order-finding subroutine to find order $r$ of $x \pmod N$
13:            if $r \nmid 2$ or $x^{r/2} = -1 (\text{mod} N)$ then
14:                Go back to step 6                    ▷ I.e. pick a new $x$
15:            else
16:                Calculate $F_1 = HCF(x^{r/2} - 1, N)$ & $F_2 = HCF(x^{r/2} + 1, N)$
17:                Return non-trivial factors           ▷ Either $F_1, F_2$ or $F_1$ & $F_2$
18:            end if
19:        end if
20:    end if
21: end procedure
```

order r such that $b = x^{r/2}$ is a square root of $1 \pmod N$, but $b \neq \pm 1 \pmod N$. By order, we mean the smallest positive integer r such that

$$x^r = 1 \pmod N. \tag{4.21}$$

so by definition, the positive condition is satisfied (else the order would be $r/2$). As for the negative condition, a theorem states that [12]

$$p(r \text{ is even and } b \neq -1 \pmod N) \geq \frac{1}{2}. \tag{4.22}$$

Hence, it should not take many runs of the order-finder for step 13 of the algorithm to be bypassed. If these conditions are met, one can show $b^2 - 1 = mN$ for some integer m, so step 16 of the algorithm retrieves at least one non-trivial factor of N.

But how exactly does the order-finding procedure work? No polynomial-time classical algorithm exists for this problem. However, using the phase estimation algorithm (which incorporates the quantum FT), an efficient quantum algorithm can be described.

The order-finding algorithm is simply an application of the phase estimation algorithm, with some additional subtleties. This time, the unitary operator U has the following action:

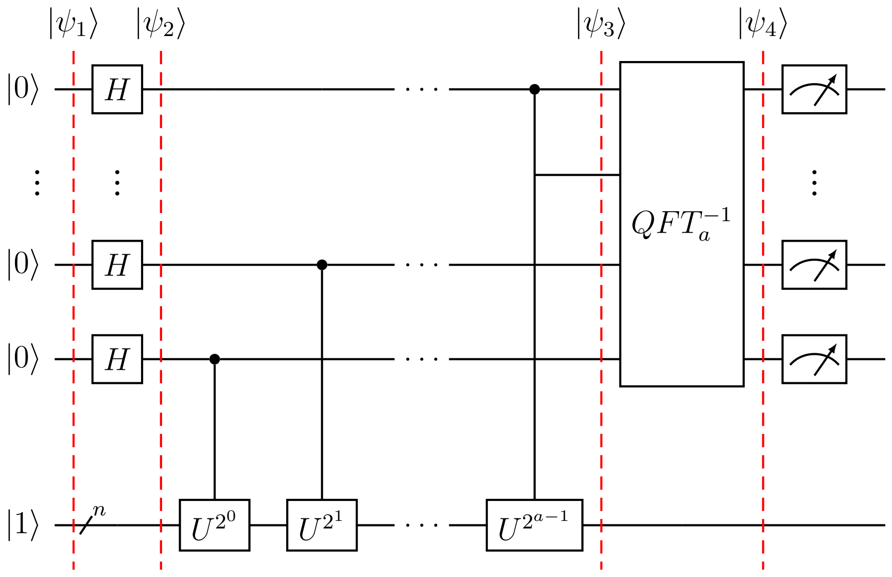


Figure 4.3: Quantum circuit for phase estimation adapted for the quantum order-finding algorithm.

$$U|y\rangle = |xy \pmod N\rangle\,. \tag{4.23}$$

Here, x is again our random number, N is our n-bit integer to be factorised and $y \in \{0,1\}^Q$. The periodicity of this operation is straightforward to see with an example:

Setting $x = 7$ and $N = 15$, it is clear that the period $r = 4$. If we begin with the initial state $|y\rangle = |1\rangle$, we can show (using integers instead of binary for clarity):

$$\begin{gathered} U^4|1\rangle = U^3|7\rangle = U^2|4\rangle = U|13\rangle = |1\rangle \\ \therefore \text{ in general, } U^r|y\rangle = |y\rangle\,. \end{gathered} \tag{4.24}$$

Therefore, using a superposition of states in this cycle gives you an eigenstate of U, for example:

$$|y\rangle = \frac{1}{2}(|1\rangle + |7\rangle + |4\rangle + |13\rangle) \mapsto U|y\rangle = \frac{1}{2}(|7\rangle + |4\rangle + |13\rangle + |1\rangle) = |y\rangle \tag{4.25}$$

This state can be generalised by adding a r-dependant phase factor. Hence the most general eigenstates of U are defined as:

$$|u_t\rangle = \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1} e^{\frac{-2\pi ikt}{r}}\left|x^k \pmod N\right\rangle, \tag{4.26}$$

where $0 \le t \le r-1$. Using a similar example to before, we can prove

$$U|u_t\rangle = e^{\frac{2\pi it}{r}}|u_t\rangle\,. \tag{4.27}$$

So we have an eigenvalue that can be retrieved by the phase estimation algorithm.

The controlled-U^j operation on the second register has a different form in this case:

$$\begin{aligned} \frac{1}{\sqrt{A}} \sum_{j=0}^{2^a-1} |j\rangle |y\rangle \mapsto \frac{1}{\sqrt{A}} \sum_{j=0}^{2^a-1} |j\rangle U^j |y\rangle \\ = \frac{1}{\sqrt{A}} \sum_{j=0}^{2^a-1} |j\rangle \left|x^j y \pmod{N}\right\rangle . \end{aligned} \tag{4.28}$$

Essentially, the oracle has the effect of scaling the contents of the second register by $x^j \pmod{N}$, a quantity known as the *modular exponential*. We can compute this by borrowing techniques from classical computing, which use repeated squaring to calculate the exponential.

Returning to our eigenstates, it is important to note that (again can be shown via example)

$$\frac{1}{\sqrt{r}} \sum_{t=0}^{r-1} |u_t\rangle = |1\rangle . \tag{4.29}$$

This means that we can prepare the second register in the trivial $|1\rangle$ state rather than the complicated $|u_t\rangle$ state (as shown in Figure 4.3). Phase estimation then retrieves an estimate $\phi \approx t/r$, where t is a random number $0 \leq t \leq r-1$, with high probability. The accuracy of this estimate is again determined by the number of qubits in the first register (a).

We almost have our period r; to obtain it we use the continued fractions algorithm to decompose our estimate ϕ. A continued fraction is an expression of the form:

$$z = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{\dots + \frac{1}{a_p}}}}. \tag{4.30}$$

where $a_0, \dots, a_p$ are integers, all but a_0 constrained to be positive. A simplified notation for this expansion is $z = [a_0; a_1, \dots, a_p]$, and we can truncate by defining the P^{th} convergent to z as $[a_0; \dots, a_P]$. As an example, the 4^{th} convergent of $\frac{1}{\sqrt{2}}$:

$$\frac{1}{\sqrt{2}} \approx \frac{1}{1 + \frac{1}{2 + \frac{1}{2 + \frac{1}{2}}}} \equiv [0; 1, 2, 2, 2] = \frac{12}{17}, \tag{4.31}$$

an approximation within 0.2% of the exact number.

So we can use the continued fractions algorithm on ϕ to produce a fraction t'/r'. r' should be the order r we are looking for (this can be checked by subbing into (4.21)), but in the (unlikely) case that t and r are not coprime, r' is instead a factor of r. Hence, several repetitions of the phase estimation algorithm with different (randomly-chosen) t may be required before the period is found [12].

The order-finding procedure is summarised in Algorithm 7.

Algorithm 7 Quantum order-finding. Again $A = 2^a$

```
1:  $|\psi_1\rangle = |0\rangle^{\otimes a}|1\rangle$                                                   ▷ Initialise qubits
2:  $|\psi_2\rangle = \frac{1}{\sqrt{N}}\sum_{j=0}^{2^a-1}|j\rangle|1\rangle$                           ▷ Create superposition
        $= \frac{1}{\sqrt{rN}}\sum_{t=0}^{2^a-1}\sum_{j=0}^{2^a-1}|j\rangle|u_t\rangle$
3:  $|\psi_3\rangle = \frac{1}{\sqrt{rN}}\sum_{t=0}^{2^a-1}\sum_{j=0}^{2^a-1}e^{\frac{2\pi itj}{r}}|j\rangle|u_t\rangle$   ▷ Apply controlled-$U^j$ operation
4:  $|\psi_4\rangle = \frac{1}{\sqrt{r}}\sum_{t=0}^{2^a-1}\left|\widetilde{t/r}\right\rangle|u_t\rangle$      ▷ Apply inverse quantum FT
5:  Measure first register to obtain estimate $(\widetilde{t/r})$
6:  Use continued fractions algorithm to find $r'$
7:  if $r'$ doesn't satisfy (4.21) then
8:      go back to step 1
9:  else
10:     return $r'(= r)$
11: end if
```

Complexity and implementation

The modular exponentiation used while applying the controlled-U^j oracle is significantly slower than the other parts of the order-finding algorithm. This sets an upper bound of $O(n^3)$ gate operations for this procedure, where $N = 2^n$.

Combining everything, Shor's algorithm factors an integer N in time polynomial in $n = \log N$. The runtime has an upper bound:

$$O((\log N)^2(\log\log N)(\log\log\log N)), \tag{4.32}$$

so the exponential speedup compared to (4.5) is clear.

Utilising this speedup is a different matter; the fault-tolerant, million-qubit quantum computers required to exploit this algorithm's power are still a while away from reality. It was originally estimated that a machine with one billion qubits is required to implement Shor's algorithm to factorise a 2048-bit RSA number.

However, researchers in 2019 cut this requirement to 20 million qubits, via an optimisation of the modular exponentiation process [43]. Their model takes just 8 hours to factorise these huge numbers; a clear threat to RSA encryption if such machines became physically feasible.

Quantum factoring records

Several research groups in the last twenty years have implemented Shor's algorithm to factor small integers on various realisations of a quantum computer. At the time of writing, the record for largest number factored by Shor's algorithm remains at 21. This was accomplished via an alteration to the standard quantum circuit, where the first register is replaced with a single qubit that is recycled throughout [44]. This significantly lowered the resource requirement hindering other implementations of the algorithm.

However, quantum devices have been demonstrated to factor much larger numbers using alternative approaches. Notably, in 2012, Nanyang Xu *et al.* created an adiabatic quantum computing procedure for factorisation, by converting the factoring problem into an optimisation problem.

Other adiabatic schemes already existed, but this group improved the algorithm by simplifying the equations used to shape the Hamiltonians. This helped reduce the number of qubits needed for factorisation, making larger numbers more accessible than before [45]. They applied their algorithm to factor 143 on an NMR quantum processor, the first 3-digit number factored on a quantum device.

It was discovered two years later that this same experiment had actually factored an entire class of much larger integers (up to 56,153). New heights were hit in 2016, when 200,009 was factored on a D-Wave 2X processor [46] using a quantum annealing procedure optimised using ideas from computational algebraic geometry.

The current record for factorisation by a quantum computer was set in 2019 by a quantum computing startup Zapata, who in conjunction with IBM found

$$1,099,551,473,989 = 1,048,589 \times 1,048,601. \tag{4.33}$$

They achieved this feat through an application of the variational quantum factoring algorithm [47]. This reduces the factoring problem to an optimisation problem solvable by the quantum approximate optimisation algorithm (see Section 7.4). This experiment demonstrates the utility of hybrid quantum-classical algorithms for making progress within quantum computing, at least in the near-term.

Even with these improvements, the factoring capabilities of quantum computers still pale in comparison to classical computers implementing the number field sieve. Significant hardware improvements are required before quantum computing fulfills its potential within this particular problem.

4.2 Related problems

4.2.1 Period finding

It is important to note that order-finding is simply a generalisation of the period-finding problem: finding the smallest integer r such that

$$f(x+r) = f(x). \tag{4.34}$$

In the case of order-finding, the period function $f(j) = x^j \pmod{N}$. A virtually identical circuit to Figure 4.3 solves this problem; the only difference is the action of the oracle:

$$U|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle \tag{4.35}$$

4.2.2 Discrete logarithms

A byproduct of Shor's algorithm for factoring was the solving of the discrete logarithm problem. More specifically:

The discrete logarithm problem

Consider a group G with order r and generator $a \in G$ such that $a^r \pmod{N} = 1$. Given $b = a^s$, where $s \in \mathbb{Z}_r$, find s.

This problem is again computationally challenging: the fastest classical algorithm, a variant of the number field sieve still runs in time exponential with p. As with integer factorisation, the difficulty of solving discrete logarithms makes them key in several cryptosystems including DSA.

Shor simply tackled this problem with an adaptation of his factoring algorithm [9], this time using two a-qubit registers rather than one. He defined a periodic function

$$f : \mathbb{Z}_r \times \mathbb{Z}_r \to G : f(x,y) = a^x b^y, \tag{4.36}$$

and made use of an oracle U with the following transform:

$$U|x_1\rangle|x_2\rangle|y\rangle = |x_1\rangle|x_2\rangle|y \oplus f(x_1,x_2)\rangle. \tag{4.37}$$

The procedure then follows the same pattern as the order-finding algorithm: Hadamard gates, unitary operator, inverse quantum FT, measurement and (generalised) continued fractions algorithm to find s.

4.3 Abelian hidden subgroup problem

Several problems, including all the ones discussed in this chapter, can be rehashed as a special case of the abelian hidden subgroup problem (HSP). This goes as follows:

The abelian hidden subgroup problem

Let $f : G \to X$ be a function mapping an abelian group G to a finite set X, such that there exists some subgroup H such that f is constant on the cosets of H and distinct on each coset.

I.e. for all $g_1, g_2 \in G, f(g_1) = f(g_2)$ iff $g_1 + H = g_2 + H$.

Given a quantum oracle for performing the unitary transform $U|g\rangle|x\rangle = |g\rangle|x \oplus f(g)\rangle$ for $g \in G, x \in X$, and $\oplus$ an appropriate binary operation on X, find a generating set for H [11, 12].

To illustrate this, consider D-J's algorithm where $f : \{0,1\}^n \to \{0,1\}$. In this case, $G = \{0,1\}^n$, or equivalently $\mathbb{Z}_2^n$, $X = \{0,1\}$ and our hidden subgroup $H = \{0\}$ or $\{0,1\}$, depending on whether f is constant or balanced.

Further examples of problems that can be viewed as HSPs are listed in Table 4.1. Abelian HSPs can all be solved using $O(\log |G|)$ operations by a quantum algorithm using similar principles to the ones discussed in this chapter. This again showcases an exponential speedup over classical approaches, which require $\Omega(\sqrt{|G/H|})$.

Polynomial-time algorithms are still yet to be found for HSPs for non-abelian groups; if this was to change, there would be significant consequences for both mathematics and cryptography. In particular, if we could solve the HSP for the dihedral group, shortest vector problems in lattices could be tackled, leaving cryptosystems such as NTRU and Ajtai-Dwork vulnerable to attack [30]. More information about the HSP and potential generalisations to non-abelian groups can be found in [11].

Name of problem	G	X	H
Deutsch-Jozsa	$\{0,1\}^n$	$\{0,1\}$	$\{0\}$ or $\{0,1\}$
Simon	$\{0,1\}^n$	$\{0,1\}^n$	$\{0,s\}, s \in \{0,1\}^n$
Order-finding	$\mathbb{Z}$	$\{a^x\}, j \in \mathbb{Z}_r, a^r = 1$	$r\mathbb{Z}$
Period-finding	$\mathbb{Z}$	Any set	$r\mathbb{Z}$
Discrete logarithm	$\mathbb{Z}_r \times \mathbb{Z}_r$	Any group	$\{k,-1\}, k \in \mathbb{Z}_r$
Hidden linear functions [48]	$\mathbb{Z} \times \mathbb{Z}$	$\mathbb{Z}_N$	$\{-a,1\}, a \in \mathbb{Z}_N$
Abelian stabliser problem [41]	Any abelian group	Any finite set	$g \in G \mid g(x) = x, \forall x \in X$

Table 4.1: A selection of hidden subgroup problems.

Chapter 5

Quantum search and amplitude amplification

Suppose that we have a list of $N = 2^n$ elements and one of them is marked as our target. A classical computer would search through each item one-by-one until the target is found: in the worst case all N items would have to be checked to find the solution.

It is more useful to view the situation as a black-box problem:

The unstructured search problem

Suppose there is a function $f : \{0,1\}^n \to \{0,1\}$, which is evaluated by a black box: $U_w \left|x\right\rangle \left|y\right\rangle \mapsto \left|x\right\rangle \left|y \oplus f(x)\right\rangle$. $f(x) = 0 \; \forall x \in \{0,1\}^n$ except for w, which satisfies $f(w) = 1$. Find w.

Lee Grover devised an algorithm [49] to solve this problem with just $O(\sqrt{N})$ queries to the oracle, and this was shortly proven to be an optimal solution [50]. Hence for this class of problems we can only achieve a quadratic, rather than exponential speedup. This is not as impressive as Shor's algorithm, but at large N this speedup becomes considerable, again highlighting the need for large-scale quantum computers to come to fruition.

The use of an oracle ensures that Grover's algorithm does not rely on the search problem in question possessing any particular structure. Hence, this algorithm can be applied to a variety of problems where the lack of structure makes exhaustive search the only option classically.

5.1 Grover's algorithm

Consider the action of the oracle when the second qubit is in the state $\frac{\left|0\right\rangle - \left|1\right\rangle}{\left|\sqrt{2}\right\rangle} = \left|-\right\rangle$:

$$\begin{aligned} f(x) = 0: &\qquad U_w |x\rangle |-\rangle \mapsto |x\rangle |-\rangle \\ f(x) = 1: &\qquad U_w |x\rangle |-\rangle \mapsto -|x\rangle |-\rangle \end{aligned} \tag{5.1}$$

It turns out that the second qubit does not change state. Therefore the operation of the black box can be written as:

$$U_w |x\rangle = (-1)^{f(x)} |x\rangle , \tag{5.2}$$

a unitary operator equivalently described as a diagonal matrix
$diag\left((-1)^{f(0)}, (-1)^{f(1)} ... (-1)^{f(2^n-1)}\right)$.

The idea of the algorithm is to iteratively increase the amplitude of the target state $|w\rangle$. This is done via the repeated application of two unitary operators:

$$U_G = U_s U_w, \tag{5.3}$$

U_w is the oracle (5.2), which flips the phase of the target state. U_s is defined as

$$U_s = 2|s\rangle\langle s| - \mathbb{1}, \tag{5.4}$$

with $|s\rangle$ representing the uniform superposition of states created by a Hadamard transform $H^{\otimes n}$.

It is insightful to apply U_s to a general state $|\Psi\rangle = \sum_i \psi_i |i\rangle$. Expressing each ψ_i in terms of its deviation $\delta\psi_i$ from the averaged amplitude $\overline{\psi}$, i.e. $\psi_i = \overline{\psi} + \delta\psi_i$, we can show:

$$U_s |\Psi\rangle = \sum_i (2\overline{\psi} - \psi_i) |i\rangle = \sum_i (\overline{\psi} - \delta\psi_i) |i\rangle . \tag{5.5}$$

Hence it is clear to see why U_s is said to induce an *inversion with respect to the mean*. Note that since the oracle U_w makes the amplitude of the target state negative, $\delta\psi_w < 0$ (and has a large magnitude). Therefore, the action of U_s increases the amplitude of $|w\rangle$ significantly. By applying $U_G \sim \sqrt{N}$ times, $\langle s|w\rangle \to 1$.

Alternatively, the two unitary operations can each be viewed as reflections, so combining them produces a rotation towards the target state (Figure 5.1).

The quantum circuit for this algorithm is depicted in Figure 5.2, noting that our second unitary operation is implemented on the circuit as $U_s = H^{\otimes n}(2|0\rangle\langle 0| - \mathbb{1})H^{\otimes n}$. The corresponding algorithm is summarised in Algorithm 5.2.

5.2 Extensions and applications of Grover's algorithm

5.2.1 Multiple solutions

We can also use Grover's algorithm to search for multiple marked items, i.e. $f(w_i) = 1$ for $i = (1, 2, \ldots, r)$. In this case the target state is a superposition of all the solutions

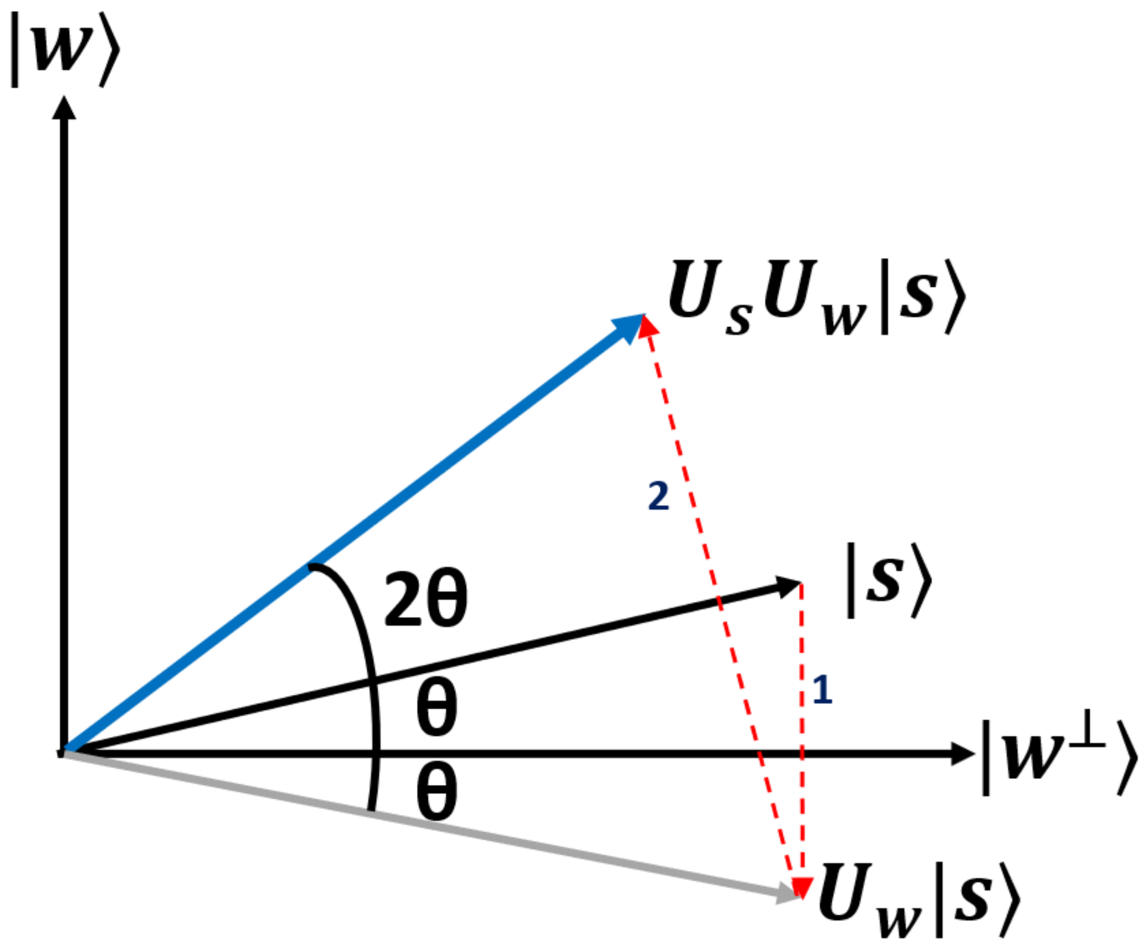


Figure 5.1: The action of the Grover iteration U_G displayed on the 2D plane spanned by the perpendicular vectors $|w\rangle$ and $|w^\perp\rangle$. 1: The oracle reflect the initial superposition state $|s\rangle$ around $|w^\perp\rangle$, then 2: U_s reflects the resultant state around $|s\rangle$. The total action is a 2θ rotation towards $|w\rangle$. Schematic adapted from [12].

to the search problem:

$$|w\rangle = \frac{1}{\sqrt{r}} \sum_{i=1}^{r} |w_i\rangle \,, \tag{5.6}$$

and the state perpendicular to this is:

$$|w^\perp\rangle = \frac{1}{\sqrt{N-r}} \sum_{x=0}^{N-r-1} |x\rangle \,. \tag{5.7}$$

The overlap of the target state with the initial superposition increases due to the expanded solution space:

$$\langle w|s\rangle = \sqrt{\frac{r}{N}}, \tag{5.8}$$

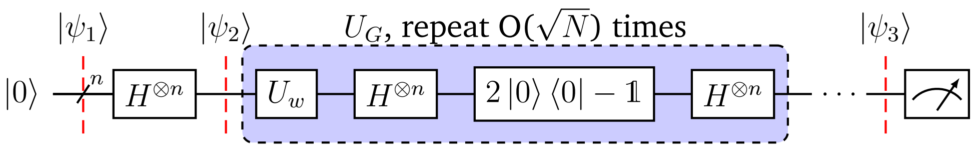


Figure 5.2: Quantum circuit for Grover's algorithm.

Algorithm 8 Grover's algorithm for quantum search

```
1: $|\psi_1\rangle = |0\rangle^{\otimes n}$                                    ▷ Initialise qubits
2: $|\psi_2\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{2^n-1} |j\rangle$          ▷ Create superposition
       $= |s\rangle$
3: $|\psi_3\rangle = U_G^{\sqrt{N}} |s\rangle$              ▷ Apply Grover iteration $\sim \sqrt{N}$ times
       $\approx |w\rangle$
4: Measure $n$ qubits to obtain $w$
```

compared to just $\frac{1}{\sqrt{N}}$ with one solution. Naturally, $O(\sqrt{N/M})$ queries to U_G are needed to acquire a solution with high probability.

5.2.2 Amplitude amplification

A generalisation of Grover's procedure was provided by Brassard and Høyer in 1997 [51]. The same ideas from Grover's original algorithm apply here: we want to boost the amplitude of a certain subspace of a Hilbert space which contains our solutions.

Their algorithm makes use of a Boolean function $\chi : \mathbb{Z} \to \{0, 1\}$ which splits a Hilbert space $\mathcal{H}$ into two orthogonal subspaces. The *good* subspace $\mathcal{H}_1$ contains all the basis states $|x\rangle \in \mathcal{H}$ for which $\chi(x) = 1$, and the states where $\chi(x) = 0$ reside in the *bad* subspace $\mathcal{H}_0$.

Every state $|\psi\rangle \in \mathcal{H}$ can be decomposed as

$$|\psi\rangle = |\psi_1\rangle + |\psi_0\rangle\,, \tag{5.9}$$

where $|\psi_i\rangle$ represents the projection onto $\mathcal{H}_i$.

Again, the repeated application of a unitary operator is key to the procedure:

$$Q(\mathcal{A}, \chi) = -\mathcal{A} S_0 \mathcal{A}^{-1} S_\chi. \tag{5.10}$$

S_χ corresponds to the Grover oracle U_W: $S_\chi |x\rangle = (-1)^{\chi(x)} |x\rangle$.

$\mathcal{A}$ and its inverse are the analogue of the Hadamard transform: it creates a superposition state $|\psi\rangle = \mathcal{A}\,|0\rangle$. Together with S_0, which only flips the sign of the amplitude of the zero state, these three operations make up the equivalent of U_s:

$$S_\psi = -\mathcal{A} S_0 \mathcal{A}^{-1} = \mathbb{1} - 2\,|\psi\rangle\langle\psi| \tag{5.11}$$

Therefore, after creating a superposition $|\psi\rangle$, applying the operator Q numerous times, then measuring the outcome, a good state is found with high probability. It can be shown that $O(1/\sqrt{a})$ queries are required to find a solution, a denoting the probability that measuring $|\psi\rangle$ produces a good state. In the case that a is known, the algorithm can be adapted so that $O(1/\sqrt{a})$ queries provide a good solution with certainty [52].

Heuristic search problem

Amplitude amplification can provide quantum speedup for a wide variety of search problems, including ones where classical heuristics exist that speed up classical search. A heuristics in this context is a polynomial-time probabilistic algorithm which outputs a good solution with nonzero probability. More formally:

The heuristic search problem

Suppose there is a family of functions $f \in \mathcal{F}$ such that $f : X \to \{0, 1\}$, and a heuristic $G : \mathcal{F} \times R \to X$, where R is a finite set. G uses a random seed $r \in R$ to generate a guess for an $x \in X$ such that $f(x) = 1$.

Using the function and heuristic, find w such that $f(w) = 1$.

Solving this problem simply involves an embedding of the heuristic into the amplitude amplification algorithm [52], which we will denote as $\mathbf{AA}(\mathcal{A}, \chi)$. The Boolean function is now defined as $\chi(r) = f(x)$, where $x = G(f, \mathbf{AA}(\mathcal{A}, \chi))$. Using the heuristic, we can find a solution $f(x) = 1$ more efficiently than a simple unstructured search.

It is useful to define these quantities; for every function $f \in \mathcal{F}$:

$$\begin{aligned} t_f &= |\{x \in X \mid f(x) = 1\}| && -\text{number of good inputs} \\ h_f &= |\{r \in R \mid f(G(f, r)) = 1\}| && -\text{number of good seeds} \end{aligned} \tag{5.12}$$

We can now specify the runtime as $O\left(\sqrt{|R|/h_f}\right)$, and the quadratic speedup over classical methods is maintained.

As a sidenote, a heuristic is labelled as *efficient* for a particular f if using G with a random seed generates good solutions with a higher probability than directly guessing with f, i.e.

$$\frac{h_f}{|R|} > \frac{t_f}{|X|}. \tag{5.13}$$

Amplitude estimation and quantum counting

There may be the situation where we are not interested in individual solutions to an N-item search problem, but rather how many solutions exist (M):

The counting problem

Given a Boolean function $f : \{0,1\}^n \to \{0,1\}$, estimate
$M = |\{x \in \{0,1\}^n \mid f(x) = 1\}|$

Going back to the state decomposition (5.9), the probability that measuring $|\psi\rangle$ produces a good state is $a = \langle\psi_1|\psi_1\rangle$. Therefore by using amplitude estimation, a, and hence, M can be determined.

It can also be shown that the amplitudes of $|\psi_1\rangle$ and $|\psi_0\rangle$, after repeated application of the operator $Q(\mathcal{A}, \chi)$ to $|\psi\rangle$, are both sinusoidal functions of period π/θ_a. A period-finding subroutine can estimate this period (or equivalently, the eigenvalues can be written in exponential form, and phase estimation used to determine θ_a). Since $a = \sin^2\theta_a$, an estimate for M can be acquired, again with quadratic speedup.

A full breakdown of this problem including proofs is provided in [52].

5.2.3 Speedup of NP-complete problems

NP-complete problems are typically characterised by their large solution space, which usually have no structure. This means that exhaustive search is a common subroutine used within algorithms for these problems. Therefore, Grover's algorithm and related procedures can be used to speed up problems in NP.

It is unfortunate that Grover's algorithm, which is optimal, is limited to quadratic speedup. An N-item space can only be searched using $O(\sqrt{N})$ queries, rather than the $O(\log N)$ query complexity required for exponential speedup. Hence, it is likely that NP-complete problems cannot be solved in polynomial time, so $NP \not\subseteq BQP$. This is not a definitive statement, however: some hidden structure may be found within these problems in the future, moving them into BQP [12].

However, this quadratic speedup is still useful for tackling a plethora of NP-complete problems. For example, let us consider the Boolean satisfiability problem, a well-surveyed problem in computational research. The task is self explanatory: take a Boolean formula and determine whether it is satisfiable. In other words, work out if there is an assignment of variables (each either TRUE/1 or FALSE/0) such that the whole formula evaluates to TRUE.

Certain versions of this problem, including the 3-SAT, use a Boolean formula Φ in conjunctive normal form. In other words, the formula is a conjunction of clauses (logical AND), with each clause a disjunction (logical OR) of literals. For example, consider a simple formula with 4 clauses:

$$\Phi(a_1, a_2, a_3) = (\neg a_1 \vee \neg a_2 \vee \neg a_3) \wedge (a_1 \vee \neg a_2 \vee a_3) \wedge (a_1 \vee a_2 \vee \neg a_3) \wedge (a_1 \vee \neg a_2 \vee \neg a_3) \tag{5.14}$$

(All k-SAT problems have k literals in each clause). It can be quickly checked that $(a_1, a_2, a_3) = (1, 0, 1)$ is one of the satisfying solutions to this problem.

It is not a stretch to see how this generalises to a search problem:

3-SAT as a search problem

Take a function $f_\phi : \mathbf{x} \to \{0, 1\}$ where $\mathbf{x} = x_1 x_2 \ldots x_n \in \{0, 1\}^n$. $f(\mathbf{x}) = 1$ if the assignment of literals $a_i = x_i$, $i = 1, 2, \ldots, n$ satisfies $\Phi(a_i)$, else $f(\mathbf{x}) = 0$.

Hence, the task is to find $\mathbf{x}$ such that $f(\mathbf{x}) = 1$ [11].

Therefore we can use our quantum search algorithms to speed up the search for satisfying solutions to 3-SAT. Other NP-complete problems that benefit from this quadratic speedup include the Hamiltonian cycle problem; determining whether a graph contains a cycle that visits each node exactly once.

Boolean satisfiability problems are part of a family of **constraint satisfaction problems** (CSPs), many of which are usually solved using a search algorithm. These are not just restricted to abstract mathematical problems; logic puzzles such as Sudoku and crosswords, as well as large-scale issues such as resource allocation can all be restructured as a CSP. Thus, the utility of the techniques explored in this chapter can be seen in real-life, commercial and recreational applications.

Chapter 6

Quantum simulation

Simulation of physical systems has been a key practical application of computers for decades. We probe systems that evolve according to differential equations such as Laplace's equation and the diffusion equation.

However, Feynman noted in his 1982 paper [6] that simulating multi-particle quantum systems on a classical computer is impossible. This is because both the number of variables describing quantum states and the number of operations required for simulation grow exponentially with system size.

Quantum devices can use the quantum mechanical properties discussed in Chapter 1 to help avoid this exponential explosion. To illustrate this, a system of $N = 40$ spin-1/2 particles requires $2^{40} \sim 10^{12}$ numbers, or 10^{13} bits to represent it, compared to just 40 qubits for a quantum machine. In 1996, Seth Lloyd proved Feynman's conjecture that a quantum computer can act as a universal quantum simulator [53].

The development of quantum simulation algorithms, and the experimental realisation of these on a variety of systems, remains an extremely active area of research to this day. One reason for this is that quantum simulations are generally not as restricted by limitations in current quantum technology (decoherence and other noise), compared to other algorithms such as Shor's. Even with a modest quantum device containing just a few dozen qubits, insightful simulations can be completed.

The quantum simulation problem

Let $|\psi\rangle$ be the state of the quantum system to be simulated. This evolves as $|\psi(t)\rangle = U|\psi(0)\rangle$, where $U = e^{-i\hbar Ht}$.

The task is to prepare the initial state $|\psi(0)\rangle$, evolve it via the unitary transform for some time t before measuring the final state $|\psi(t)\rangle$ to find the value of some relevant physical quantity.

6.1 Digital and analog quantum simulation

6.1.1 Digital quantum simulation

Digital quantum simulation (DQS) is the more direct approach to quantum simulation. The idea is to encode the system's state and implement the unitary evolution operator directly onto a circuit-based quantum computer. For example, a spin-1/2 particle in the spin-up state $|\uparrow\rangle$ can be encoded as a $|1\rangle$, while spin-down $|\downarrow\rangle$ is $|0\rangle$. Meanwhile the unitary evolution U is approximated using a series of quantum gates [54].

The first step of DQS is to **prepare the initial state** $|\psi(0)\rangle$; often not a trivial task. Several algorithms exist in the literature for the purpose of efficient state preparation for different cases. For example, Abrams and Lloyd described a quantum procedure for initialising a quantum register into an antisymmetrised superposition of states, useful for simulating fermionic systems [55].

The next step is the **unitary evolution** $U = e^{-i\hbar Ht}$. For efficient (polynomial-time) simulation, it is useful to write the Hamiltonian as a sum over local interaction terms:

$$H = \sum_{l=1}^{M} H_l \tag{6.1}$$

Each term usually represents either a one or two-body interaction, which are the dominant interactions in any physical system. The Ising model Hamiltonian is an example of this:

$$H_I = -B_x \sum_i \sigma_i^x + \sum_{i<j} J_{ij}\sigma_i^z\sigma_j^z, \tag{6.2}$$

where B_x is the external magnetic field and J_{ij} represent the spin-spin coupling coefficients.

The unitary operator with local terms $U_l = e^{-i\hbar H_l t}$ is much more straightforward to approximate using quantum gates. The problem is that in general $[H_l, H_{l'}] \neq 0$, so $e^{-i\hbar Ht} \neq \prod_l e^{-i\hbar H_l t}$. However, decomposing U into small time steps Δt, then using the first-order Trotter formula [12], we can approximate:

$$U(\Delta t) \approx \prod_l e^{-i\hbar H_l \Delta t}. \tag{6.3}$$

Hence, by constructing an appropriate quantum circuit to implement $U(\Delta t)$, then iteratively applying it:

$$|\psi_{i+1}\rangle = U(\Delta t)\,|\psi_i\rangle\,, \tag{6.4}$$

where $i = \{0, 1, \ldots, t_f/\Delta t\}$, the final state $|\psi(t_f)\rangle$ can be approximated.

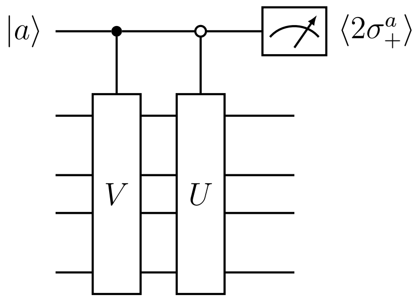


Figure 6.1: Quantum circuit for the measurement of $\langle U^\dagger V \rangle$. Ancilla qubit $|a\rangle$ is initialised as $|+\rangle$, and $2\sigma_+^a = \sigma_x^a + i\sigma_y^a$.

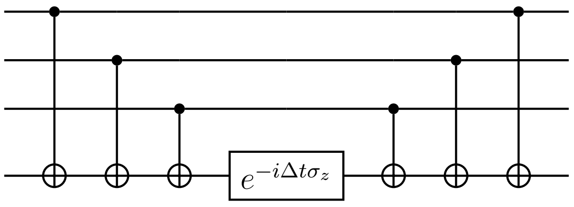


Figure 6.2: Quantum circuit for the simulation of (6.5) for $N = 3$.

The final step is to make a **measurement** of the final state to extract the value of a particular observable. The procedure varies depending on the quantity in question. For example, suppose we want to determine correlations $C_{AB}(t) = \langle A(t)B(0) \rangle = \langle e^{iHt} A e^{-iHt} B \rangle$.

The operators can be decomposed into smaller, unitary operators $A = \sum_i A_i$ and $B = \sum_j B_j$, then we can denote $U = e^{iHt} A_i^\dagger e^{-iHt}$ and $V = B_j$. These two unitary operators can be implemented on a quantum circuit (Figure 6.1), and $\langle U^\dagger V \rangle$ is obtained by measuring the expectation value of the ancilla qubit. Summing over these expectation values gives us our desired quantity C_{AB} [56].

As an example of DQS in action, consider the N-body Hamiltonian

$$H = \sigma_{z,1} \otimes \sigma_{z,2} \otimes \ldots \otimes \sigma_{z,N}, \tag{6.5}$$

where $\sigma_{z,i}$ acts on the i^{th} qubit. This Hamiltonian is not a sum over local terms, nevertheless it can still be simulated efficiently using a circuit implementing $U(\Delta t) \approx e^{-i\hbar H \Delta t}$ (Figure 6.2).

In fact, any Hamiltonian of the form

$$H = \bigotimes_{i=1}^{N} \sigma_{\alpha,i} \tag{6.6}$$

where $\alpha = \{x, y, z\}$ can be simulated on a similar quantum circuit.

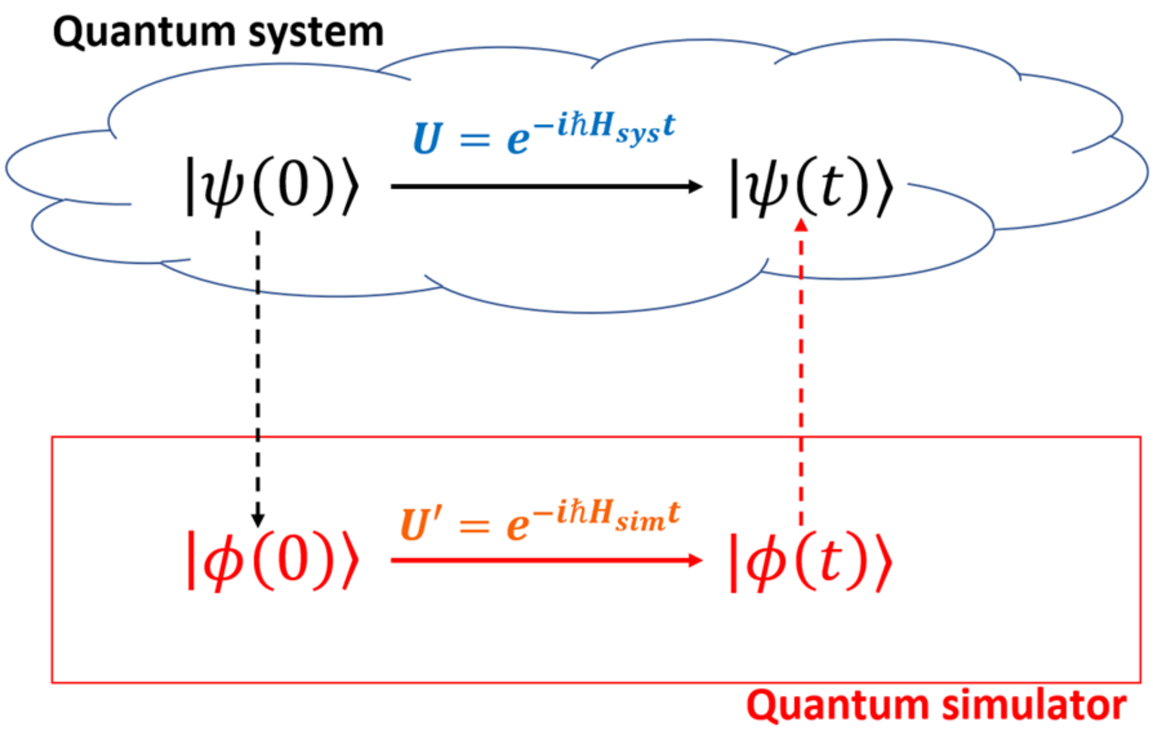


Figure 6.3: Schematic of the AQS procedure. By evolving the controllable quantum simulator using U', we can model the evolution of the system due to U. Recreated from [54].

6.1.2 Analog quantum simulation

The approach taken with analog quantum simulation (AQS) is summarised in Figure 6.3. The idea is to mimic the evolution of a complex system with a quantum simulator; a controllable model which can reproduce the system's dynamics accurately. This requires a mapping to exist between the system and simulator, e.g.

$$H_{sim} = pH_{sys}p^{-1}. \tag{6.7}$$

Here p is an operator that maps the initial state of the system to the corresponding state in the simulator: $|\phi(0)\rangle = p\,|\psi(0)\rangle$. The state is evolved on the simulator: $|\phi(t)\rangle = e^{-i\hbar H_{sim}t}\,|\phi(0)\rangle$, before it is mapped back to the system to obtain the final state of the system: $|\psi(t)\rangle = p^{-1}\,|\phi(t)\rangle$.

As an example mapping between a quantum system and a simulator, consider the Dirac equation for a spin-1/2 particle with rest mass m in (1 + 1) dimensions:

$$i\hbar\frac{\partial\psi}{\partial t} = H_D\psi = (c\hat{p}\sigma_x + mc^2\sigma_z)\psi, \tag{6.8}$$

where c denotes the speed of light, $\hat{p}$ is the momentum operator and σ_x, σ_z are the Pauli matrices. Gerritsma *et al.* demonstrated that this system could be simulated using a single trapped ion interacting with a bichromatic light field [57], with Hamiltonian:

$$H_I = 2\eta\Delta\tilde{\Omega}\sigma_x\hat{p} + \hbar\Omega\sigma_z. \tag{6.9}$$

Here, η is the Lamb-Dicke parameter and $\tilde{\Omega}$ is a parameter controlled by the intensity of the bichromatic light field. $\Delta = \sqrt{\hbar/2\tilde{m}\omega_{ax}}$ is the size of the ground-state wave function, with $\tilde{m}$ representing the trapped ion mass and ω_{ax} the axial trapping frequency.

By identifying $c = 2\eta\tilde{\Omega}\Delta$ and $mc^2 = \hbar\Omega$, it is clear to see that H_D and H_I have the same form. Hence the relativistic spin-1/2 system can be studied by simulating the non-relativistic trapped ion, which is a more controllable system. In this way, peculiar quantum effects such as Zitterbewegung can be investigated [57].

The system and simulator must be sufficiently similar for AQS to be successful. Therefore, the devices used for AQS are specialised to narrow classes of quantum systems, unlike the universal, all-purpose machines envisioned to be used for DQS in the future. Hence, AQS has the advantage over DQS in terms of near-term practical implementations.

6.2 Applications

The established and potential applications of quantum simulation span the fields of physics and chemistry. The physical implementation of quantum computers has provided a means of probing systems which were intractable on classical computers. We will discuss a selection of these applications here; for more examples see [54].

6.2.1 Condensed matter physics

Quantum many-body systems are notoriously difficult to study via classical computation, but quantum simulation can help provide insight into these systems. This could assist with the tackling of several unresolved problems within condensed matter physics, surrounding quantum phase transitions and high temperature superconductivity.

Simulations in this area are usually based around adaptations of fundamental models such as the Ising model (6.2) or the Bose-Hubbard:

$$H_{BH} = -J\sum_{i,j}\hat{b}_i^\dagger\hat{b}_j + \frac{1}{2}U\sum_i \hat{n}_i(\hat{n}_i - 1) - \mu\sum_i \hat{n}_i \tag{6.10}$$

Here, J denotes the hopping amplitude, U is the atom-atom interaction strength, $\hat{b}_i^\dagger$ and $\hat{b}_j$ are the bosonic creation and annihilation operators, $\hat{n}_i = \hat{b}_i^\dagger\hat{b}_i$ is the number of atoms on the i^{th} lattice site, and μ is the chemical potential. A similar Hamiltonian exists for the Fermi-Hubbard model; these are the simplest models for interacting particles on a lattice.

Other important models are the spin models for spin systems:

$$H_{XYZ} = \sum_{\alpha=x,y,z} \sum_{i=1}^{N} J_\alpha \sigma_i^\alpha \sigma_{i+1}^\alpha. \tag{6.11}$$

The AQS or DQS of all these models have been either proposed (e.g. [58]) or experimentally realised (e.g. [59, 60]) on numerous occasions. In terms of insightful implementations, the quantum phase transition from a superfluid to a Mott insulator, which had been predicted thirteen years prior, was finally probed through an AQS simulation of the Bose-Hubbard model [61]. In addition, the transition from paramagnet to ferromagnet has been emulated via a simulation of a spin system described by the quantum Ising model [62].

Other areas of condensed matter research where quantum simulation may prove to be invaluable include spin glasses [63] and high temperature superconductivity [64].

6.2.2 High energy physics

We have already discussed the simulation of the Dirac equation in 1 + 1 dimensions. A proposal to expand the simulation space to 3 + 1 dimensions has also been presented [65], again using a single trapped ion with a Hamiltonian akin to (6.9).

As well as the aforementioned Zitterbewegung (rapid oscillatory motion of Dirac particles), another peculiar quantum effect that physicists desire to probe is the Klein paradox. This is an effect seen when scattering electrons from a potential barrier: this barrier is virtually transparent when

$$V \sim mc^2, \tag{6.12}$$

i.e. the barrier's potential is a similar magnitude to the electron mass. Using quantum simulation, we have the capability to investigate this phenomenon [66] and solve problems beyond the capabilities of classical computing.

In terms of applications to gauge theories, vital in the scientific quest to tackle intractable aspects of QCD, see [67] for a comprehensive report of recent proposals for simulation implementations. This report also includes a review of the first proof-of-principle simulations of the Schwinger model.

6.2.3 Other fields

Other research areas that could benefit from quantum simulation implementations include:

Cosmology

Liu and Li generalised an algorithm for simulating quantum field theories, for the purpose of simulating cosmic inflation [68]. A recent article discusses the challenges of overcoming nonlinearities within cosmological simulations, problematic as quantum gates are linear. Using variational quantum computing (see Section 7.4), these challenges can be overcome, subsequently opening up a pathway towards dark matter simulations [69].

Nuclear physics

Nuclear physics is another scientific field which involves the solving of complex N-body problems. A recent workshop, entitled *Intersections between Nuclear Physics and Quantum Information* welcomed 116 experts to explore collaboration opportunities between the two fields. This produced a white paper [70], which includes a discussion of the contrasting approaches of AQS and DQS, within the context of nuclear physics.

Quantum chemistry

Several polynomial-time quantum algorithms have been proposed that can speed up calculations and simulations within quantum chemistry. These include DQS-based approaches for calculating the thermal rate constant [71] and the energy spectrum of the hydrogen atom [72]. A good review of the applications of quantum simulation within chemistry can be found in [73].

Chapter 7

Other quantum algorithms

Covering more than a minute fraction of the quantum algorithms developed to date is impossible in a report of this length. At the time of writing, Stephen Jordan's comprehensive 'Quantum Algorithm Zoo' cites 430 different papers on the subject [74]. This chapter will outline some of the more impactful algorithms from the past twenty-five years.

7.1 Quantum walks

7.1.1 Random, discrete and continuous walks

Classical random walks have proven to be significant mathematical tools for describing the behaviour of processes in a variety of fields, ranging from science and engineering to economics and sociology. Applications include estimating the size of search engines [75], modelling stock market prices [76] and predicting human response times during perceptual classification tasks [77].

Random walks are stochastic processes, where the action at each time step is determined by some probability distribution. A walk is often viewed as a graph G, with vertices V denoting the states and edges E representing the allowed transitions. The simplest example is the one-dimensional random walk on the integer line.

The 'walker' begins at 0, and at each step walks right (+1) with probability p or left (-1) with probability $q = 1 - p$; each action decided by a toss of a biased coin. In this case, the probability of the walker being at position X after N steps follows a binomial distribution [78]:

$$P_N(X) = \binom{N}{\frac{N+X}{2}} p^{\frac{N+X}{2}} q^{\frac{N-X}{2}} \tag{7.1}$$

The quantum analogue to this example is the discrete quantum walk on a line (DQWL). These are defined using a walker and coin, each of which are quantum systems living in their own Hilbert spaces, $\mathcal{H}_p$ and $\mathcal{H}_c$ respectively. Our total Hilbert space is then:

$$\mathcal{H} = \mathcal{H}_c \otimes \mathcal{H}_p, \tag{7.2}$$

therefore, the walk is defined by a product of two unitary operators. The coin operator puts the coin state into a superposition, so the Hadamard gate is often chosen for this. Meanwhile the conditional shift operator acts to shift the walker right if the coin state is $|0\rangle$, and left if it is $|1\rangle$:

$$S = |0\rangle_c \langle 0| \otimes \sum_i |i+1\rangle_p \langle i| + |1\rangle_c \langle 1| \otimes \sum_i |i-1\rangle_p \langle i| \tag{7.3}$$

Hence, the complete unitary operator applied to determine each step of the walk is

$$U = S \cdot (H \otimes \mathbb{1}_p) \tag{7.4}$$

The randomness in this process is introduced by repeatedly operating with U, only taking a measurement after several iterations. After k steps the total state of the system is succinctly given by

$$|\psi\rangle = U^k |\psi_0\rangle \tag{7.5}$$

After these iterations, states of different phases interfere constructively and destructively. Measurement of the position states produces an antisymmetric probability distribution skewing left or right if the initial coin state is $|1\rangle$ or $|0\rangle$ respectively.

This contrasts to the symmetric distribution produced by a classical walk beginning at 0, with equal probability of going left or right (Figure 7.1).

The continuous version of quantum walks was formulated by Farhi and Gutmann [80]. It is defined as follows (following [81]):

Take a graph $G = (V, E)$ where $|V| = v$. The random walk can be defined by the $v \times v$ infinitesimal generator matrix M, where

$$M_{ab} = \begin{cases} -\beta & a \neq b, a \text{ \& } b \text{ connected by an edge} \\ 0 & a \neq b, a \text{ \& } b \text{ not connected} \\ k\beta & a = b, k \text{ is the valence of vertex a} \end{cases} \tag{7.6}$$

We can define a Hamiltonian H_{cw} in a Hilbert space $\mathcal{H}$ with matrix elements $M_{ab} = \langle a|H_{cw}|b\rangle$. Using this we can construct the following Schrodinger equation for a state $|\psi\rangle \in \mathcal{H}$:

$$i\frac{d\langle a|\psi(t)\rangle}{dt} = \sum_b M_{ab} \langle b|\psi(t)\rangle \tag{7.7}$$

Hence, we can say that the unitary evolution operator $U = e^{-i\hbar H_{cw}t}$ defines the continuous quantum walk on graph G.

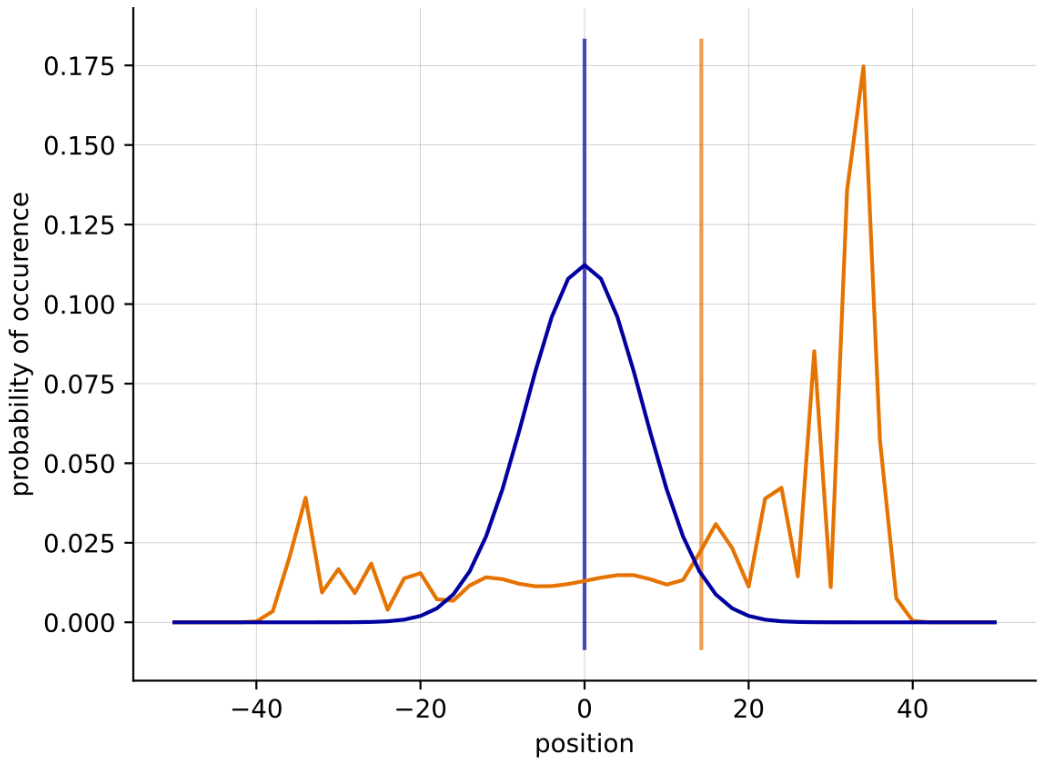


Figure 7.1: Probability distribution from one-dimensional discrete classical (blue) and quantum (orange) random walks after 50 time steps [79].

The advantage that quantum walks possess over random walks is two-fold. Algorithms based on quantum walks generally have a lower hitting time (time taken to find a particular target vertex) or lower mixing time (time taken for the walk to converge to its limiting distribution). The latter quantum speedup can be quadratic while the former can be exponential; both are useful in helping quantum walks outperform their classical counterparts.

7.1.2 Applications

One key computational problem where quantum walks have been implemented to achieve quantum speedup is the element distinctness problem. The problem is self-explanatory: given a black box implementing a function f acting on a set of inputs $M = \{x_1, \ldots, x_N\}$, determine inputs x_i and x_j such that $f(x_i) = f(x_j)$.

Ambainis solved the problem by reducing it to finding a marked vertex v_s on a graph, then used a continuous quantum walk procedure to search the graph for a marked vertex [82]. If v_s exists, then we know that $x_i = x_j$ for some $i, j \in S \subseteq N$.

This form of procedure has subsequently been generalised to general search procedures [83]; in fact, Grover's algorithm can be seen as a special type of quantum walk.

Another interesting application is the efficient evaluation of Boolean formulae. In particular, it was proven that any AND-OR Boolean formula φ on N binary inputs can be evaluated using $O(N^{1/2+O(1/\sqrt{\log N})})$ operations [84]. To achieve this result, they used a discrete quantum walk on the tree representation of the formula, before applying a quantum phase estimation procedure to evaluate φ.

This algorithm displays a polynomial speedup over the best classical algorithm ($O(N^{0.753})$). This problem is particularly interesting, as evaluating AND-OR trees can help with determining the winner of some two-player games.

The applications of quantum walks are becoming as varied as their classical counterparts. One exciting emerging area of quantum computing in general is quantum machine learning (QML). A recent study demonstrated how discrete quantum walks can be applied to speed up a deep learning process to predict how proteins fold in 3D, an extremely important problem within biochemical research today [85].

Therefore, quantum walks may prove to be one of the most powerful tools within the field. [86] provides a comprehensive review of this extremely exciting area of research, which has plenty of open challenges.

7.2 Solving linear systems

Solving linear equations is a fundamental task in almost all areas of science and engineering. Therefore, the demonstration by Harrow, Hassidim and Lloyd [87] that quantum computers can provide an exponential speedup in solving linear systems is extremely significant. It can be argued that the HHL algorithm has the greatest potential for widespread applicability.

The problem is simple: given a Hermitian $N \times N$ matrix A, and a unit vector **b**, find **x** such that $A\mathbf{x} = \mathbf{b}$.

Since A is Hermitian, we can define its decomposition in terms of its eigenvalues and eigenvectors:

$$A = \sum_{j=0}^{N-1} \lambda_j |u_j\rangle \langle u_j| \,. \tag{7.8}$$

The vector $|b\rangle$ can be decomposed using the same basis:

$$|b\rangle = \sum_{j=0}^{N-1} b_j |u_j\rangle \,, \tag{7.9}$$

so the aim of the HHL algorithm is to produce the state

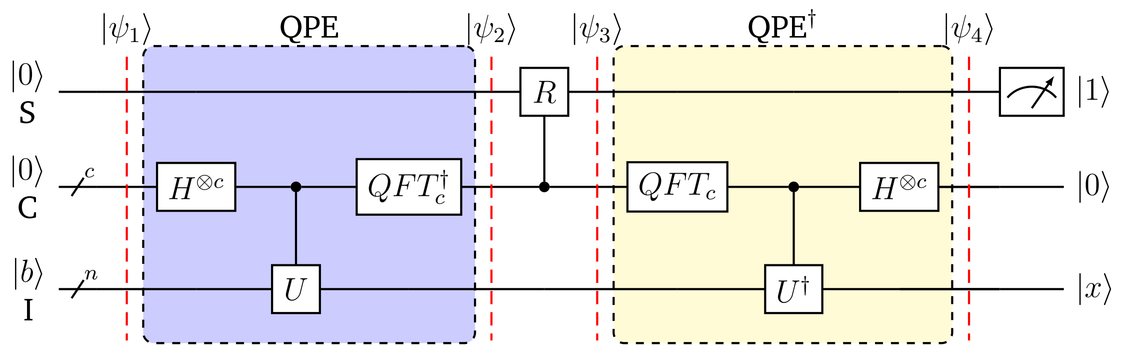


Figure 7.2: Quantum circuit for the simulation of (6.5) for $N = 3$.

$$|x\rangle = A^{-1}|b\rangle = \sum_{j=0}^{N-1} \lambda_j^{-1} b_j |u_j\rangle \tag{7.10}$$

7.2.1 HHL algorithm

The quantum circuit for this procedure is displayed in Figure 7.2. This algorithm uses three quantum registers: a c-qubit clock register C, an n-qubit input register I ($N = 2^n$), where the vector $|b\rangle$ is loaded, and an extra ancilla register S.

Algorithm 9 HHL algorithm for solving linear systems

```
1: $|\psi_1\rangle = |0\rangle|0\rangle^{\otimes n}|b\rangle$    ▷ Prepare initial state
2: $|\psi_2\rangle = \sum_{j=0}^{N-1} b_j |0\rangle_S |\lambda_j\rangle_C |u_j\rangle_I$    ▷ Perform QPE on C and I
3: $|\psi_3\rangle = \sum_{j=0}^{N-1} \left(\sqrt{1-\frac{C^2}{\lambda_j^2}}|0\rangle + \frac{C}{\lambda_j}|1\rangle\right)_S b_j |0\rangle_C^{\otimes c} |u_j\rangle_I$
       ▷ Apply controlled rotation to ancilla qubit
4: $|\psi_4\rangle = \sum_{j=0}^{N-1} \left(\sqrt{1-\frac{C^2}{\lambda_j^2}}|0\rangle + \frac{C}{\lambda_j}|1\rangle\right)_S b_j |0\rangle_C^{\otimes c} |u_j\rangle_I$
       ▷ Apply inverse QPE to uncompute C
5: Measure ancilla qubit
6: if Output is $|1\rangle$ then
7:     Return $|x\rangle$ (which is stored in I)
8: else
9:     Go back to step 1
10: end if
```

The procedure can roughly be divided into three stages. The first step is to apply the quantum phase estimation (QPE) algorithm using the unitary operator

$$U = e^{iAt} = \sum_{j=0}^{N-1} e^{i\lambda_j t} |u_j\rangle\langle u_j| \tag{7.11}$$

As explained in Algorithm 5, QPE performs the mapping

$$|0\rangle_C^{\otimes c}|b\rangle_I \rightarrow \left|\tilde{\lambda}_j\right\rangle_C |b\rangle_I\,, \tag{7.12}$$

where $\left|\tilde{\lambda}_j\right\rangle$ is the c-bit representation of the estimation of λ_j, and the subscripts denote which register each qubit belongs to. In the eigenbasis of A the state after QPE can be rewritten as (now assuming the estimation is perfect for simplicity):

$$\sum_{j=0}^{N-1} b_j |0\rangle_S |\lambda_j\rangle_C |u_j\rangle_I\,. \tag{7.13}$$

The next step is to apply a controlled rotation operator on an ancilla qubit, controlled by $|\lambda_j\rangle$, resulting in the state:

$$\sum_{j=0}^{N-1}\left(\sqrt{1-\frac{C^2}{\lambda_j^2}}\,|0\rangle + \frac{C}{\lambda_j}|1\rangle\right)_S b_j |\lambda_j\rangle_C |u_j\rangle_I \tag{7.14}$$

where C is a normalising constant.

The final step is to apply the inverse QPE to reset the clock register, leaving the state as

$$\sum_{j=0}^{N-1}\left(\sqrt{1-\frac{C^2}{\lambda_j^2}}\,|0\rangle + \frac{C}{\lambda_j}|1\rangle\right)_S b_j |0\rangle_C^{\otimes c} |u_j\rangle_I \tag{7.15}$$

We now make a measurement of the ancilla qubit; if $|1\rangle$ is measured the remaining state is (ignoring the clock register):

$$\sqrt{\frac{1}{\sum_{j=0}^{N-1} C^2|b_j|^2/|\lambda_j|^2}}\sum_{j=0}^{N-1} b_j\frac{C}{\lambda_j}|u_j\rangle_I\,, \tag{7.16}$$

This has the same form as (7.10) up to a normalisation factor, which can be worked out from the probability of obtaining $|1\rangle$.

If $|1\rangle$ is not obtained, the desired matrix inversion has not taken place, and the whole algorithm must be repeated. The expected number of repetitions is $O(\kappa)$, where κ is the condition number of A: the ratio between A's largest and smallest eigenvalues.

This algorithm produces the quantum state $|x\rangle$ rather than the vector **x**. Instead of retrieving all the components of the vector (which requires at least N repetitions), we instead seek to extract some property of **x**. This is done by mapping a linear operator M to a quantum-mechanical operator, then measuring the expectation value $\langle x|M|x\rangle$.

The HHL algorithm has a time complexity of $O(\log(N)\kappa^2 s^2/\epsilon)$, where s is the sparsity of A and ϵ is the total error in the procedure. Therefore, an exponential speedup is achieved over the best classical algorithm; the conjugative gradient method [88], which runs with $O(Ns\kappa\log(1/\epsilon))$.

7.2.2 Applications

Following the first proof-of-concept implementations in 2013 [89, 90, 91], the HHL algorithm has proven to be an invaluable tool within several practical problems. For example, Dominic Berry extended this procedure to describe an efficient algorithm for solving inhomogeneous, sparse, linear differential equations, which describe a variety of physical systems [92].

Clader, Jacobs and Sprouse generalised the HHL algorithm to compute the electromagnetic scattering cross section of an arbitrary target [93]. One important aspect of their method was the use of a preconditioner, which removes the HHL algorithm's running time's dependence on κ. This significantly expands the range of problems that can be solved using this procedure.

The cross section was calculated using the Finite Element Method (FEM), a widely-used method within engineering and mathematics for solving boundary value problems. This involves dividing the problem domain into small volume elements, before applying boundary conditions at neighbouring elements. This creates a large system of linear equations to solve, hence the HHL algorithm's utility is justified.

Similar to quantum walks, the HHL algorithm also finds applications within quantum machine learning. HHL-based QML algorithms have been created for data classification tasks, including the quantum support vector machine (QSVM) [94]. The quantum version of this well-established machine learning approach involves expressing the SVM as an approximate least-squares problem, which can be solved via matrix inversion using the HHL algorithm.

Another example of a QML application is linear regression. Wiebe, Braun and Lloyd showed how the optimal fit parameters for the fit function can be obtained using an algorithm that employs a HHL-based subroutine [95]. A further HHL-inspired procedure has shown the potential to be commercially useful within recommendation systems [96] used to generate user-tailored playlists on services such as Spotify and Netflix.

7.3 Adiabatic algorithms

We have already touched on the AQC model earlier in this dissertation, but it is useful to outline an example of its use to solve a problem. The adiabatic algorithm is applicable to almost any CSP, many of which are NP-complete, including the aforementioned 3-SAT problem.

Consider a n-bit instance of 3-SAT: a Boolean formula with clauses containing 3 bits: each bit $z_i, i = \{1, 2, \ldots, n\}$ taking the value 0 or 1 [20]. Considering each clause C, we can define an energy function

$$h_C(z_{i_C}, z_{j_C}, z_{k_C}) = \begin{cases} 0, & \text{if } (z_{i_C}, z_{j_C}, z_{k_C}) \text{ satisfies } C \\ 1, & \text{if } (z_{i_C}, z_{j_C}, z_{k_C}) \text{ violates } C. \end{cases} \tag{7.17}$$

Hence the total energy $h = \sum_C h_c = 0$ if and only if $(z_1, \ldots, z_n)$ satisfy all the clauses.

Turning from the classical to the quantum viewpoint, the bits are replaced by qubits $|z_i\rangle$, again either $|0\rangle$ or $|1\rangle$. We now have an operator for each clause

$$H_{P,C}(|z_1\rangle |z_2\rangle \ldots |z_n\rangle) = h_C(z_{i_C}, z_{j_C}, z_{k_C}) |z_1\rangle |z_2\rangle \ldots |z_n\rangle, \tag{7.18}$$

from which we define our problem Hamiltonian

$$H_P = \sum_C H_{P,C}. \tag{7.19}$$

$H_p |\psi\rangle = 0$ if and only if $|\psi\rangle$ is a superposition of the form $|z_1\rangle |z_2\rangle \ldots |z_n\rangle$, where z_i satisfy all the clauses as before.

Therefore, solving the 3-SAT problem can be reduced to finding the ground state of H_P. As explained previously, we must also describe an initial Hamiltonian H_B whose ground state is simpler to find. The 1-bit Hamiltonian acting on the i^{th} qubit is defined as

$$H_B^{(i)} = \frac{1}{2}(1 - \sigma_x), \tag{7.20}$$

Note that in this case, it is actually independent of i. This has an eigenvalue equation $H_B^{(i)} |x_i = x\rangle = x |x_i = x\rangle$, with eigenstates

$$|x_i = 0\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad |x_i = 1\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} \tag{7.21}$$

Taking $H_{B,C} = H_B^{(i_c)} + H_B^{(j_c)} + H_B^{(k_c)}$, our initial Hamiltonian is

$$H_B = \sum_C H_{B,C}, \tag{7.22}$$

with ground state

$$|x_1 = 0\rangle |x_2 = 0\rangle \ldots |x_n = 0\rangle = \frac{1}{\sqrt{N}} \sum_{z_1} \sum_{z_2} \cdots \sum_{z_n} |z_1\rangle |z_2\rangle \ldots |z_n\rangle, \tag{7.23}$$

the latter expression written in the basis of H_P as a superposition over all $N = 2^n$ basis states.

Hence, we have all the ingredients for AQC. After preparing the qubits in the ground state of H_B, the system evolves according to the Schrodinger equation with Hamiltonian (2.5). For large enough T, the final state should hold the solution to the satisfiability problem. In the case that there is no satisfying assignment, the number of violated clauses is minimised instead.

As well as the satisfiability problem and other CSPs, several fundamental algorithms already discussed can be reshaped into the AQC format. Grover's, D-J and B-V algorithms have all been implemented adiabatically; see [97] for full details and some more examples. Even the factoring problem can be tackled by the adiabatic algorithm: this involves converting factoring into an optimisation problem, before incorporating the cost function into the problem Hamiltonian [98].

One open challenge concerning the adiabatic algorithm is the lack of rigorous upper bounds on its runtime [30]. This can be calculated for specific problem instances, but to date the time complexity has not been determined in general. Finding a standard procedure to determine the required T to solve an arbitrary problem would be invaluable for this purpose.

7.4 Variational quantum algorithms

The current age of quantum technology has been dubbed the *Noisy Intermediate-Scale Quantum* (NISQ) era [99]. This is characterised by the absence of fully scalable, fault-tolerant quantum computers. Current state-of-the-art quantum processors have around 50-100 qubits and limited error correction, hence they are prone to decoherence and other quantum noise.

This is not to say that quantum devices of this size are not useful. We have already seen how NISQ computers can outperform the fastest supercomputers for certain tasks [26, 27]. A key challenge for researchers today is to discover algorithms that can achieve quantum advantage on NISQ devices, despite their obvious limitations.

Variational quantum algorithms (VQAs) could form the solution to this problem. These are typified by their hybrid quantum/classical approach: a classical optimiser is employed to optimise a parameterised quantum circuit. This is a general framework, hence VQAs can be easily adapted to solve a wide variety of problems. In fact, VQAs have been proposed for almost all the applications that have been envisioned for quantum computing [100]. This makes them possibly the most significant category of algorithms for near-term research.

The main inputs for a standard VQA are an objective function, which encodes the problem's solution, and an ansatz: a parameterised unitary operation, trained iteratively to optimise the objective function. Each loop consists of the quantum computer using the inputs to estimate the cost, before the classical computer uses this information to update the ansatz parameters. This is repeated until a satisfactory solution is found.

To illustrate this, we will now detail one of the most prominent VQAs within research today.

7.4.1 Quantum approximate optimisation algorithm

This famous algorithm, introduced by Farhi, Goldstone and Gutmann in 2014 [101], can be used to find approximate solutions for a variety of combinatorial optimisation problems. This study described its application to the NP-hard Max-Cut problem:

Max-Cut

The problem is simple: given a graph G, find a maximum cut.

By maximum cut, we mean a partition of the nodes of a graph into two sets, such that the number of edges between the two sets is as large as possible.

Consider a four-node, square graph, with each vertex either red or blue. With four nodes and two sets, there are $2^4 = 16$ possible assignments for the nodes; three of these are given below.

The first two assignments require a cut across two edges to divide the nodes by colour, whereas the third assignment requires all four edges to be cut. Hence this final assignment is the solution: denoting red as 0 and blue as 1, the bitstrings 0101 and 1010 represent the optimal solutions.

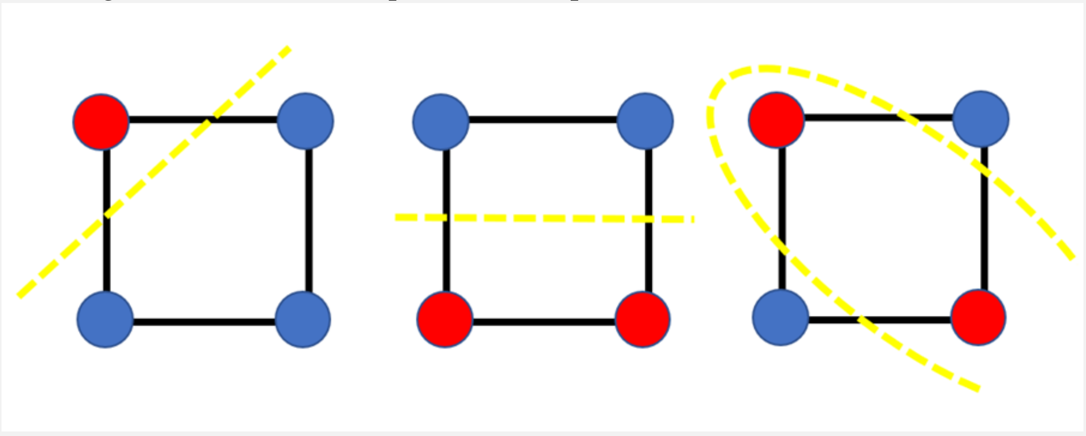


Figure 7.3: Three of the sixteen possible assignments for the Max-Cut problem for a 4-node graph.

More formally, for a graph with n vertices and a set of edges $\langle jk \rangle$, the goal is to find a string z to maximise the objective function

$$C = \sum_{\langle jk \rangle} C_{\langle jk \rangle} \tag{7.24}$$

which is a sum of local terms

$$C_{\langle jk \rangle} = \frac{1}{2}(-\sigma_j^z \sigma_k^z + 1) \tag{7.25}$$

Mirroring the procedure for AQC, we define the objective function as our problem Hamiltonian: $C = H_P$. This comes with a unitary operator

$$U(H_P, \gamma) = e^{-i\gamma H_P} = \prod_{\langle jk \rangle} e^{-i\gamma C_{\langle jk \rangle}}, \tag{7.26}$$

where γ is an angle between 0 and 2π.

The initial, or mixing Hamiltonian is defined as:

$$H_B = \sum_{j=1}^{n} \sigma_j^x \tag{7.27}$$

and the corresponding unitary operator, with β running from 0 to π:

$$U(H_B, \beta) = e^{-i\beta H_B} = \prod_{j=1}^{n} e^{-i\beta \sigma_j^x}. \tag{7.28}$$

The ground state of H_B, which is the initial state of the QAOA procedure, is simply the superposition state $|s\rangle = \frac{1}{2^n}\sum_z |z\rangle$.

Having prepared this initial state, the two unitary operators $U(H_P, \gamma)$ and $U(H_B, \beta)$ are applied p times in alternating fashion, producing the state:

$$|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle = U(H_B, \beta_p) U(H_P, \gamma_p) \ldots U(H_B, \beta_1) U(H_P, \gamma_1)) \, |s\rangle \tag{7.29}$$

Here, $\boldsymbol{\gamma} \equiv \gamma_1 \ldots \gamma_p$, and similar for $\boldsymbol{\beta}$. p is chosen as any integer ≥ 1; a larger p improves the quality of the approximation, with the downside of increasing the algorithm's complexity.

Finally, a measurement in the standard computational basis obtains a string z, hence the objective function $C(z)$ can be evaluated.

The classical optimisation for this particular problem is actually completed prior to this quantum procedure, rather than iteratively after every run. The idea is to choose angles $(\boldsymbol{\gamma}, \boldsymbol{\beta})$ to maximise the expectation value of H_P:

$$F_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \langle \boldsymbol{\gamma}, \boldsymbol{\beta} | H_P | \boldsymbol{\gamma}, \boldsymbol{\beta} \rangle \tag{7.30}$$

This quantity is useful, as maximising F_p is equivalent to maximising $C(z)$. Several classical optimisers do exist, but finding an effective method to consistently generate these optimal angles is an open challenge.

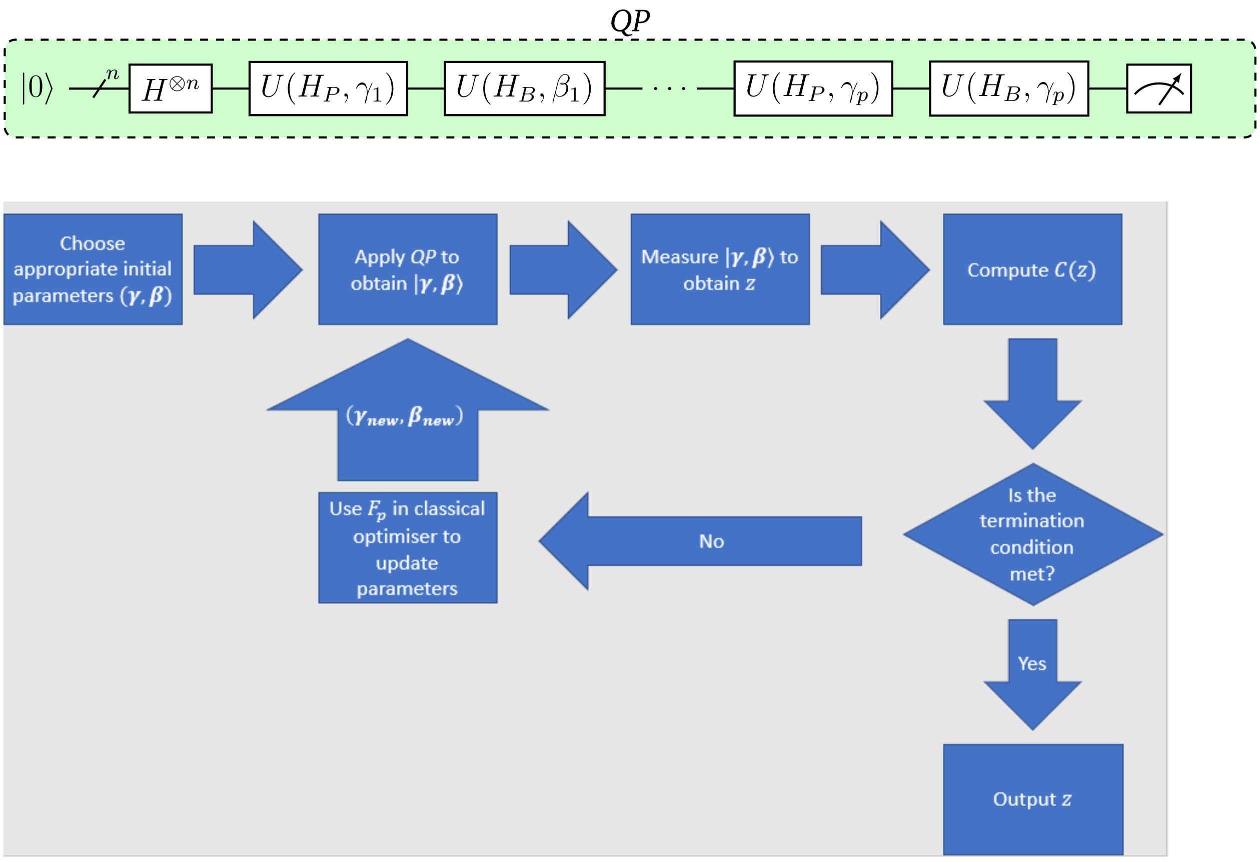

QP
$|0\rangle$
n
$H^{\otimes n}$
$U(H_P,\gamma_1)$
$U(H_B,\beta_1)$
$\cdots$
$U(H_P,\gamma_p)$
$U(H_B,\gamma_p)$
Choose appropriate initial parameters (γ,β)
Apply QP to obtain $|\gamma,\beta\rangle$
Measure $|\gamma,\beta\rangle$ to obtain z
Compute $C(z)$
$(\gamma_{new},\beta_{new})$
Use F_p in classical optimiser to update parameters
No
Is the termination condition met?
Yes
Output z

Figure 7.4: Top: High-level quantum circuit for the quantum part of the QAOA algorithm. Bottom: Simple flow diagram showing the iteration process.

Once the optimiser has found the optimal values of (γ, β), the quantum procedure can be done several times to find many sample strings z. The string that produces the highest value of $C(z)$ is taken as the solution.

A schematic of the QAOA algorithm for a general problem is shown in Figure 7.4.

7.4.2 Other variational quantum algorithms and applications

Aside from the QAOA, another well-known VQA is the variational quantum eigensolver (VQE) [102]. This algorithm seeks to find the ground state energy E_G of a Hamiltonian by minimising its expectation value over a trial state. In other words, the cost function is

$$C(\theta) = \langle\psi(\theta)|H|\psi(\theta)\rangle, \tag{7.31}$$

where $|\psi(\theta)\rangle = U(\theta)\,|\psi_0\rangle$. Hence the ansatz in this case is $U(\theta)$, and $|\psi_0\rangle$ is the initial state. The Rayleigh-Ritz variational principle constrains

$$C(\theta) \geq E_G \tag{7.32}$$

and this is an equality if $|\psi(\theta)\rangle$ is the ground state of H. The VQE incorporates a similar prepare-measure-optimise-recompute procedure as the QAOA.

Other efficient VQAs have recently been proposed for various mathematical tasks, including problems already discussed; solving linear systems of equations [103] and factoring [47]. A variational approach to quantum simulation has also been proposed: instead of implementing the unitary evolution, the state $|\psi(\theta)\rangle$ is evolved by iteratively updating the parameters θ [104].

The iterative, optimisation-based approach of VQAs fits perfectly with the QML framework, hence there is no surprise we find applications in this area. VQAs can be helpful in tasks such as supervised learning. The quantum variational classification process [105] includes applying a parameterised ansatz $W(\theta)$, where the parameters are again optimised during the training of the classifier. The cost function to be minimised is simply the error probability of assigning the wrong label.

A fantastic survey of VQAs can be found in [100]: this report references 261 papers, most of which were released within the last two years. This is a telling indicator of the current high activity within VQA research and development.

7.5 Algorithm unification

In recent years some researchers have been working on a framework for unifying some of the major algorithms. At surface level, the schemes for factoring, search and simulation share little resemblance. However, a quantum singular value transformation (QSVT) approach has been developed [106] and expanded [107] to formulate algorithms for all three of these problems.

QSVT is built on a technique called quantum signal processing (QSP). This is a method involving the alternating application of two rotation operators. The first is a signal rotation operator

$$W(a) = \begin{pmatrix} a & i\sqrt{1-a^2} \\ i\sqrt{1-a^2} & a \end{pmatrix}, \tag{7.33}$$

where $a = \cos\theta/2$ and θ is the fixed angle of rotation around the x-axis of the Bloch sphere. The second is a variable angle, signal processing rotation operator

$$S(\phi) = e^{i\phi Z}, \tag{7.34}$$

where $\boldsymbol{\phi} = (\phi_0, \phi_1, \ldots, \phi_d) \in \mathbb{R}^{d+1}$. By applying the QSP sequence

$$U_{\boldsymbol{\phi}} = S(\phi_0) \prod_{k=1}^{d} W(a) S(\phi_k), \tag{7.35}$$

a matrix is produced with each term a polynomial function of a $(P(a), Q(a))$:

$$U_{\boldsymbol{\phi}} = \begin{pmatrix} P(a) & iQ(a)\sqrt{1-a^2} \\ iQ^*(a)\sqrt{1-a^2} & P^*(a) \end{pmatrix} \tag{7.36}$$

Hence the action of QSP is to apply a polynomial transformation to each component of $W(a)$.

QSVT generalises this method by using the QSP sequence to polynomially transform all the singular values of a block-encoded matrix. Specifically, take the singular value decomposition (SVD) of an arbitrary matrix A:

$$A = M\Sigma N^\dagger. \tag{7.37}$$

M and N are unitary matrices while Σ is a diagonal matrix containing the singular values of A : $\{\sigma_k\}$. The columns of $M(|m_k\rangle)$ and $N(|n_k\rangle)$ are known as the left and right-singular vectors respectively. Hence the SVD of A can be rewritten as an eigenvalue decomposition:

$$A = \sum_{k=1}^{r} \sigma_k \left|m_k\right\rangle \left\langle n_k\right|, \tag{7.38}$$

where r is the rank of A - the number of nonzero singular values.

Therefore, after block encoding A into a unitary matrix

$$U = \begin{pmatrix} A & \cdot \\ \cdot & \cdot \end{pmatrix} \tag{7.39}$$

we can apply a QSP operation of a similar style to (7.35), with projector controlled phase shift operations Π_ϕ and $\tilde{\Pi}_\phi$ replacing $S(\phi)$. This produces a similar matrix to (7.36):

$$U_{\boldsymbol{\phi}} = \begin{pmatrix} Poly^{(SV)}(A) & \cdot \\ \cdot & \cdot \end{pmatrix}, \tag{7.40}$$

where $Poly^{(SV)}(A)$ differs slightly if d is even or odd:

$$Poly^{(SV)}(A) = \begin{cases} \sum_k Poly(\sigma_k) \left|w_k\right\rangle \left\langle v_k\right| & d \text{ odd} \\ \sum_k Poly(\sigma_k) \left|v_k\right\rangle \left\langle v_k\right| & d \text{ even} \end{cases} \tag{7.41}$$

The flexibility of the QSVT process comes from the parameterisation of the polynomial transformation. The QSP phase angles $\boldsymbol{\phi} = (\phi_0, \phi_1, \ldots, \phi_d)$ completely determine the form of the transform, so the action of QSVT can be controlled and manipulated easily.

This controllability allows us to realise a variety of different algorithms using the same model. Each algorithm simply involves intertwining quantum gates and measurements with QSVT, which is applied using a quantum oracle.

Since the three major algorithms form the basis of the majority of quantum algorithms, a QSVT subroutine can be used to construct nearly all known quantum algorithms. In that sense, a grand unification of quantum algorithms may be possible: see [107] for a complete discussion of this.

Chapter 8

Conclusion

In this review, we discussed the origins of quantum computing, from Turing to Feynman, before breaking down the key quantum mechanical concepts that provide quantum advantage. Surveys of quantum computation models and quantum complexity theory followed; a solid knowledge base in these areas allowed us to tackle the main focus of this dissertation with more precision.

We then broke down the three major ‘primordial’ algorithms in detail: Shor’s algorithm for factoring, Grover’s quantum search algorithm and the quantum simulation algorithm. For each we discussed the procedure, the origin of the quantum speedup over classical schemes, related algorithms and relevant applications.

Finally, we explored an array of more modern algorithms and their applications, including areas where research is currently being conducted. Again, the reader is guided to peruse the various reviews referenced throughout this report to fill in any gaps.

The field of quantum algorithms is relatively new compared to traditional research areas within the quantum community. The number of publications released during such a short period speaks volumes about the activity within this area, and the vast potential for these algorithms to make an impact, both academically and commercially across numerous industries.

We have already discussed the limitations of current quantum machines due to decoherence and other forms of noise. While NISQs provide an adequate architecture for near-term applications, the development of fault-tolerant devices will take the field of quantum algorithms to new heights.

Another challenge researchers are currently tackling is the problem of loading and storing data on quantum computers. The quantum analogue of RAM is qRAM (quantum random access memory), and just like its classical counterpart, qRAM will be essential for large-scale quantum computers. There are current concerns about the susceptibility of QRAM implementations to noise, as well as large overhead. See [108] for a recent study of these issues.

Fighting these practical issues, developing algorithms and finding new applications for these procedures are the three main keys to unlocking this field's true potential. If we can continue the current rate of innovation, practical quantum computers may become a staple within academic and industrial research within the next decade or two.

Bibliography

[1] D. Edwards, Synthese **42**, 1 (1979).

[2] A. M. Turing, Proceedings of the London Mathematical Society **42**, 230 (1937).

[3] M. L. Minsky, *Computation: Finite and Infinite Machines* (Prentice-Hall, Inc., USA, 1967).

[4] J. von Neumann, IEEE Annals of the History of Computing **15**, 27 (1993).

[5] P. Benioff, J Stat Phys **22**, 563 (1980).

[6] R. P. Feynman, J Stat Phys **21**, 467 (1982).

[7] J. S. Bell, Physics Physique Fizika **1**, 195 (1964).

[8] D. Deutsch, Proceedings of the Royal Society of London Series A **400**, 97 (1985).

[9] P. W. Shor, Algorithms for quantum computation: discrete logarithms and factoring, in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pp. 124–134, 1994.

[10] M.-W. Dictionary, Algorithm, https://www.merriam-webster.com/dictionary/algorithm [Retrieved 09-08-21].

[11] M. Mosca, Quantum algorithms, 2008, arXiv:0808.0369.

[12] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, Cambridge, 2010).

[13] A. Einstein, B. Podolsky, and N. Rosen, Physical Review **47**, 777 (1935).

[14] M. Kim, Quantum information – Autumn 2020, Lecture notes, Imperial College London, 2021.

[15] National Academies of Sciences, Engineering, and Medicine, *Quantum Computing: Progress and Prospects* (The National Academies Press, Washington DC, 2019).

[16] P. E. Black, model of computation, https://www.nist.gov/dads/HTML/modelOfComputation.html [Retrieved 15-08-21], 2004.

[17] C. H. Bennett, IBM Journal of Research and Development **17**, 525 (1973).

[18] C. M. Dawson and M. A. Nielsen, Quantum Info. Comput. **6**, 81 (2006).

[19] M. Born and V. Fock, Zeitschrift fur Physik **51**, 165 (1928).

[20] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, Quantum computation by adiabatic evolution, 2000.

[21] A. Das, Quantum annealing: a rugged cost/energy landscape, https://commons.wikimedia.org/wiki/File:Quant-annl.jpg [Retrieved 26-08-21], 2005, Used under a Public Domain release.

[22] D-Wave, What is quantum annealing?, https://docs.dwavesys.com/docs/latest/c_gs_2.html#how-quantum-annealing-works-in-d-wave-systems [Retrieved 26-08-21].

[23] R. Jozsa, Quantum Information Processing **199**, 1 (2005).

[24] V. Lahtinen and J. Pachos, SciPost Physics **3**, 21 (2017).

[25] J. Preskill, Quantum computing and the entanglement frontier, 2012, arXiv:1203.5813.

[26] F. Arute *et al.*, Nature **574**, 505 (2019).

[27] H.-S. Zhong *et al.*, Science **370**, 1460 (2020).

[28] J. Watrous, Quantum computational complexity, 2008, arXiv:0804.3401.

[29] J. Preskill, Lecture Notes for Ph219/CS219: Quantum Information, California Institute of Technology, 2015.

[30] A. Montanaro, npj Quantum Information **2**, 15023 (2016).

[31] E. Bernstein and U. Vazirani, SIAM Journal on Computing **26**, 1411 (1997).

[32] R. Cleve, Quantum Computation and Quantum Information Theory **1**, 103–127 (2001).

[33] A. Kay, Tutorial on the quantikz package, arXiv:1809.03842, 2018.

[34] D. R. Simon, SIAM Journal on Computing **26**, 1474 (1997).

[35] D. J. Bernstein, *Introduction to post-quantum cryptography* (Springer Berlin Heidelberg, Berlin, Heidelberg, 2009), pp. 1–14.

[36] C. Pomerance, Notices of the American Maths Society **43**, 1473 (1996).

[37] J. Dixon, Mathematics of Computation **36**, 255 (1981).

[38] M. A. Case, A beginner's guide to the general number field sieve, 2003.

[39] T. Kleinjung, J. W. Bos, and A. K. Lenstra, Mersenne factorization factory, Cryptology ePrint Archive, Report 2014/653, https://ia.cr/2014/653 [Retrieved 26-08-21], 2014.

[40] L. Hales and S. Hallgren, An improved quantum fourier transform algorithm and applications, in *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pp. 515–525, 2000.

[41] A. Y. Kitaev, Quantum measurements and the Abelian Stabilizer Problem, 1995, arXiv:9511026.

[42] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca, Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences **454**, 339 (1998).

[43] C. Gidney and M. Ekerå, Quantum **5**, 433 (2021).

[44] E. Martín-López *et al.*, Nature Photonics **6**, 773 (2012).

[45] N. Xu *et al.*, Phys. Rev. Lett. **108**, 130501 (2012).

[46] R. Dridi and H. Alghassi, Scientific Reports **7**, 43048 (2017).

[47] A. H. Karamlou *et al.*, Analyzing the performance of variational quantum factoring on a superconducting quantum processor, 2021, arXiv:2012.07825.

[48] D. Boneh and R. Lipton, Quantum cryptanalysis of hidden linear functions (extended abstract), in *CRYPTO*, p. 424–437, 1995.

[49] L. K. Grover, A fast quantum mechanical algorithm for database search, in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, p. 212–219, 1996.

[50] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, SIAM Journal on Computing **26**, 1510 (1997).

[51] G. Brassard and P. Høyer, An exact quantum polynomial-time algorithm for simon's problem, in *Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems*, pp. 12–23, 1997.

[52] G. Brassard, P. Høyer, M. Mosca, and A. Tapp, Quantum Computation and Information **305**, 53–74 (2000).

[53] S. Lloyd, Science **273**, 1073 (1996).

[54] I. M. Georgescu, S. Ashhab, and F. Nori, Reviews of Modern Physics **86**, 153 (2014).

[55] D. S. Abrams and S. Lloyd, Physical Review Letters **79**, 2586 (1997).

[56] G. Ortiz, J. E. Gubernatis, E. Knill, and R. Laflamme, Physical Review A **64**, 22319 (2001).

[57] R. Gerritsma *et al.*, Nature **463**, 68 (2010).

[58] R. Somma, G. Ortiz, J. E. Gubernatis, E. Knill, and R. Laflamme, Physical Review A **65**, 42323 (2002).

[59] B. P. Lanyon *et al.*, Science **334**, 57 (2011).

[60] J. W. Britton *et al.*, Nature **484**, 489 (2012).

[61] M. Greiner, O. Mandel, T. Esslinger, T. Hansch, and I. Bloch, Nature **415**, 39 (2002).

[62] A. Friedenauer, H. Schmitz, J. Glueckert, D. Porras, and T. Schätz, Nature Physics **4**, 757 (2008).

[63] M. Lemeshko *et al.*, Physical Review B **88**, 014426 (2013).

[64] F. Yamaguchi and Y. Yamamoto, Superlattices and Microstructures **32**, 343 (2002).

[65] L. Lamata, J. León, T. Schätz, and E. Solano, Physical Review Letters **98**, 253005 (2007).

[66] R. Gerritsma *et al.*, Physical Review Letters **106**, 060503 (2011).

[67] M. Bañuls *et al.*, The European Physical Journal D **74**, 165 (2020).

[68] J. Liu and Y.-Z. Li, On quantum simulation of cosmic inflation, 2021, arXiv:2009.10921.

[69] P. Mocz and A. Szasz, The Astrophysical Journal **910**, 29 (2021).

[70] I. C. Cloët *et al.*, Opportunities for nuclear physics & quantum information science, 2019, arXiv:1903.05453.

[71] D. A. Lidar and H. Wang, Physical Review E **59**, 2429 (1999).

[72] B. P. Lanyon *et al.*, Nature Chemistry **2**, 106 (2010).

[73] I. Kassal, J. D. Whitfield, A. Perdomo-Ortiz, M.-H. Yung, and A. Aspuru-Guzik, Annual Review of Physical Chemistry **62**, 185 (2011).

[74] S. Jordan, Quantum algorithm zoo, `https://quantumalgorithmzoo.org/` [Retrieved 18-09-21], 2021.

[75] Z. Bar-Yossef and M. Gurevich, J. ACM **55**, 24 (2008).

[76] M. D. Godfrey, C. W. J. Granger, and O. Morgenstern, Kyklos **17**, 1 (1964).

[77] R. Nosofsky and T. Palmeri, Psychological review **104**, 266 (1997).

[78] S. E. Venegas-Andraca, Quantum Information Processing **11**, 1015 (2012).

[79] shoyer, One dimensional quantum random walk, https://commons.wikimedia.org/wiki/File:One_dimensional_quantum_random_walk.svg [Retrieved 18-09-21], 2008, Used under a Creative Commons Attribution-Share Alike 3.0 Unported license.

[80] E. Farhi and S. Gutmann, Physical Review A **58**, 915 (1998).

[81] A. M. Childs, E. Farhi, and S. Gutmann, Quantum Information Processing **1**, 35 (2002).

[82] A. Ambainis, Quantum walk algorithm for element distinctness, in *45th Annual IEEE Symposium on Foundations of Computer Science*, pp. 22–31, 2004.

[83] M. Santha, Quantum walk based search algorithms, 2008, arXiv:0808.0059.

[84] A. Ambainis, A. Childs, and B. Reichardt, Siam Journal on Computing - SIAMCOMP **39**, 363 (2007).

[85] P. A. M. Casares, R. Campos, and M. A. Martin-Delgado, Qfold: Quantum walks and deep learning to solve protein folding, 2021, arXiv:2101.10279.

[86] K. Kadian, S. Garhwal, and A. Kumar, Computer Science Review **41**, 100419 (2021).

[87] A. W. Harrow, A. Hassidim, and S. Lloyd, Physical Review Letters **103**, 150502 (2009).

[88] M. Hestenes and E. Stiefel, Journal of research of the National Bureau of Standards **49**, 409 (1952).

[89] X.-D. Cai *et al.*, Physical Review Letters **110**, 230501 (2013).

[90] S. Barz *et al.*, Scientific Reports **4**, 6115 (2014).

[91] J. Pan *et al.*, Physical Review A **89**, 022313 (2014).

[92] D. W. Berry, Journal of Physics A: Mathematical and Theoretical **47**, 105301 (2014).

[93] B. D. Clader, B. C. Jacobs, and C. R. Sprouse, Physical Review Letters **110**, 250504 (2013).

[94] P. Rebentrost, M. Mohseni, and S. Lloyd, Phys. Rev. Lett. **113**, 130503 (2014).

[95] N. Wiebe, D. Braun, and S. Lloyd, Phys. Rev. Lett. **109**, 050505 (2012).

[96] I. Kerenidis and A. Prakash, Quantum recommendation systems, 2016, arXiv:1603.08675.

[97] T. Albash and D. A. Lidar, Reviews of Modern Physics **90**, 015002 (2018).

[98] B. Yan *et al.*, Quantum Engineering **3**, e59 (2021).

[99] J. Preskill, Quantum **2**, 79 (2018).

[100] M. Cerezo *et al.*, Variational quantum algorithms, 2020, arXiv:2012.09265.

[101] E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization algorithm, 2014, arXiv:1411.4028.

[102] A. Peruzzo *et al.*, Nature Communications **5**, 4213 (2014).

[103] C. Bravo-Prieto *et al.*, Variational quantum linear solver, 2020, arXiv:1909.05820.

[104] Y. Li and S. C. Benjamin, Phys. Rev. X **7**, 021050 (2017).

[105] V. Havlíček *et al.*, Nature **567**, 209 (2019).

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

[107] J. M. Martyn, Z. M. Rossi, A. K. Tan, and I. L. Chuang, A grand unification of quantum algorithms, 2021, arXiv:2105.02859.

[108] C. T. Hann, G. Lee, S. Girvin, and L. Jiang, PRX Quantum **2**, 020311 (2021).