Quantum Computing Project

Tommaso Bruggi, Andrea Husseiniova, Sagar Patel, Dan Buxton, Ricardo Del Rio Fuente, Matthaios Chouzouris, Hasancan Cifci

School of Physics and Astronomy,
The University of Edinburgh

Abstract

Quantum computing is an exciting and increasingly active field of research. Quantum computing utilises three basic counter-intuitive ingredients inherent to quantum mechanics (superposition, interference and entanglement) to enhance computing power and speed, with respect to classical computers. In this project we provide an introduction to quantum computing, aimed at physics undergraduates. Interesting theoretical insights are developed for both Grover's algorithm and Deutsch-Jozsa's algorithm. Furthermore, experimental concerns are discussed as well as very recent promising research to address these. We outline our opinion of where future research should focus on. A simulator for a circuit-model quantum computer was created on a classical computer using Python, together with an efficient implementation of Grover's algorithm. Design choices are thoroughly analysed. Overall, the project is ideally balanced including mathematical, experimental and computational considerations.

Acknowledgements

We would like to thank Professor Anthony Kennedy for the three introductory lectures in the beginning of the semester and the opportunity to learn about quantum computing.

Project Goals

This project gave us the opportunity to develop a strong understanding of the exciting field of quantum computing. The goals of the project were to:

1. Test, design and successfully implement Grover's algorithm on a classical computer in a documented program with clearly-defined interfaces. Achieve this by simulating the quantum algorithm on a classical computer.
2. Write a report providing an introduction to quantum computing aimed at physics undergraduates.
   (a) Explain how Grover's algorithm works, discussing its limitations.
   (b) Describe how our program is structured, explaining our design choices.

In fact, the project went beyond the aims initially stipulated.

In the report both Deutsch-Jozsa's algorithm and quantum error correction were also discussed, including very recent promising advances in using Steane's code particularly for Grover's algorithm. Furthermore, some discussion of Grover's algorithm in determining the optimal number of iterations involving the Lindemann - Weierstrass theorem is an interesting insight and was not found in the literature. The report also mentions the importance of other algorithms such as the HHL algorithm in the mathematical sciences, in particular to the new interdisciplinary field of quantum machine learning. As any project, there always exists margin of improvement. Our group believes finishing off the implementation for the HHL algorithm in our program would have been ideal. We are however determined to do so after the deadline.

Group work evaluation

Overall, the group worked exceptionally well. The team was diverse with physics, theoretical physics and computational physics students. The tasks were split based on interests and personal strengths. The team exploited each individual's unique background, which enabled an efficient and fun learning experience. From the first meeting we elected a team leader who was responsible for organising weekly meetings and to resolve potential issues between team members (which weren't observed). The team also elected a secretary to document (during meetings) each team member's responsibilities and tasks to make sure everyone would complete their duties on time. This established hierarchy is one of the reasons why the team was successful. All communication was through a group chat on Messenger app for enhanced productivity. This allowed to easily setup polls when organizing weekly meetings, for example. Furthermore, it is a platform extensively used by all members, so important updates were read quickly. Lastly, version control was used throughout to prevent concurrent work from conflicting.

Contents

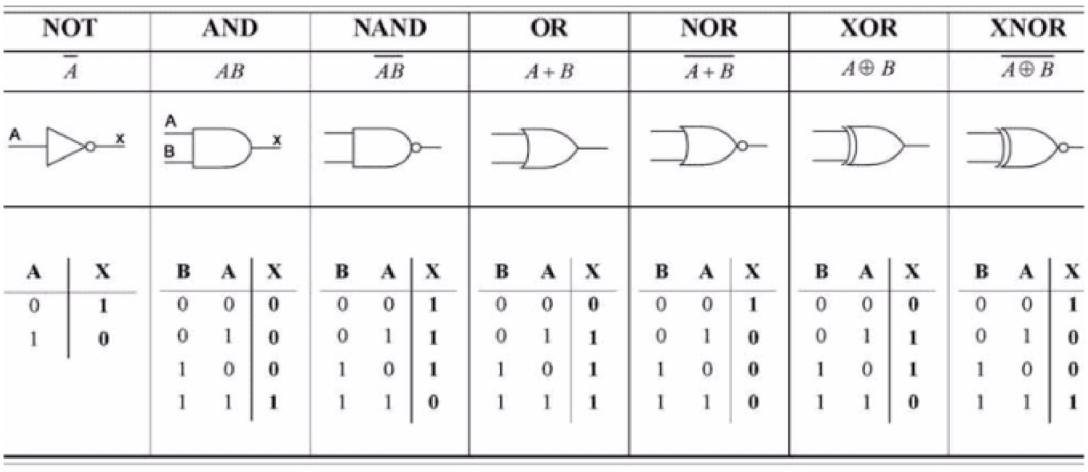
1 Introduction

1.1 The nature of quantum mechanics

Quantum mechanics is said to be the study of physics on very small length scales [1]. On this length scale the equations that govern our macroscopic world are rendered useless in the quantum world of sub-atomic particles. This is because of key differences between the macroscopic and quantum world such as the quantization of energy and angular momentum. For example, electrons can only exist in discrete energy levels around a nucleus. The other principles of quantum mechanics state that light can behave as a particle and matter can behave as a wave [2], giving way to the well-known term wave-particle duality. In the quantum world the wavelike properties of particles are described by the famous Schrodinger equation, which is used to give the allowed energy levels of the quantum mechanical system [3]. There is then the associated wavefunction which is a probability distribution and gives the probability of finding the particle in a certain position. Now you cannot know the exact position of the particle due to Heisenberg's uncertainty principle but only where it is most likely to be. This is down to the wave-particle duality of the particle and can also lead to an interesting property known as superposition. Since quantum particles can act like waves, they (like classical waves) can be added together and superposed.This means that quantum states can be added together which gives another valid quantum state. It can also be said that a quantum state can be a linear combination of distinct quantum states. This is a key concept in quantum computers which will be discussed in the next section.

1.2 From classical to quantum computing

Classical computers are what we are used to seeing and using. Information is stored in the form of 1's and 0's which are represented by a high or low voltage respectively [4]. These 1's and 0's usually represent a decimal number in a language called binary. This information can be then manipulated to perform calculations by logic gates. Logic gates are simply an electronic device with inputs and an output. Some of the most common logic gates are the AND, OR and NOT gates. The output is dependent on what the inputs are, for example the AND gate gives an output of 1 (high voltage) if both its inputs are 1 and if not then it has an output of 0 (low voltage). The NOT gate gives the opposite to the input, so if the input is a 0 then the output is a 1. All these combinations of inputs and their respective outputs can be seen in what is called a truth table. Multiple logic gates can be used together to perform a task, this could be for example illuminating a certain segment in a calculator. Nowadays logic gates in computers come in the form of transistors and integrated circuits. With the digital logic gate being the building block for most electronic devices and microprocessors [5]. However, it wasn't always like this as in the early days of classical computers, mechanical devices were used for the logic gates instead of transistors. For example vacuum tubes were used to perform Boolean operations in the Colossus computer which was used to decipher intercepted messages. In fact , modern electronics already works on the principles of quantum mechanics. Such as the origin of the energy band description of electrons which are present in periodic potentials in transistors requires quantum mechanics. The necessity of this theory has naturally led to experiments incorporating quantum physics to develop a revolutionary approach in computation: quantum computing.



NOT	AND	NAND	OR	NOR	XOR	XNOR
$\overline{A}$	AB	$\overline{AB}$	$A+B$	$\overline{A+B}$	$A\oplus B$	$\overline{A\oplus B}$

A	X
0	1
1	0

B	A	X
0	0	0
0	1	0
1	0	0
1	1	1

B	A	X
0	0	1
0	1	1
1	0	1
1	1	0

B	A	X
0	0	0
0	1	1
1	0	1
1	1	1

B	A	X
0	0	1
0	1	0
1	0	0
1	1	0

B	A	X
0	0	0
0	1	1
1	0	1
1	1	0

B	A	X
0	0	1
0	1	0
1	0	0
1	1	1

Figure 1: Truth Tables for some logic gates mentioned in [6]

Quantum computers run on a similar idea to the classical case but instead of using bits, quantum bits "qubits" are used. These like classical bits can hold the value 1 or 0 but can also hold a superposition of these values at the same time. The goal of the quantum computer is to evolve the quantum mechanical state vector and extract the information contained in the state vector. The quantum computing version of the logic gates would be any linear operator that acts on the state vector. The problem is that there isn't actually a logical framework like there is for classical computing and so one of the main goals is to find one for quantum computing.

Classical computers produce a lot of heat, this stems from the fact that they are based on irreversible logic operations. For example, lets say in a logic gate two electrons are inputted and one is outputted. This means that one electron must have been dissipated away, resulting in an entropy increase of ln2. This would mean a heat of amount kTln2 would be produced for a system operating at temperature T [7]. Now in classical computers, it is actually electrical current flowing in and out of the logic gates therefore the heat produced is much more than the example discussed above. Now over the years advances is technology have meant that the heat produced from these logic gate operations have drastically reduced however, there is a minimum that can be reached due to the fact that any many to one operation is irreversible and so has a thermodynamic cost. Moore's law, a law that stated that the number of transistors on a microchip would double every two years [8], was eventually broken down by quantum mechanics. As the transistors would eventually reach a size small enough for quantum tunnelling to occur, which would mean current could leak through these new transistors which would essentially beat the purpose of the logic gates and would result in errors in the binary number.

The origins of quantum computing arose from trying to solve the high heat production of classical computers by at first the idea of producing a reversible computer in the classical sense [9]. This opened up the idea in the 1980s for Paul Benioff and Richard Feynamnn, that to build a reversible computer quantum mechanics can be used. It was then said that a reversible quantum computer could be used to provide realistic quantum mechanical simulations. Since then, the interest in quantum computers have increased massively with the big technology companies such as Google, Intel and IBM in the race to create the first fully working quantum computer.

1.3 Mathematical formulation

1.3.1 Qubits and Quantum Registers

Quantum computing works with a set of n-qubits. This implies a Hilbert space of 2^n basis states [10]. A Hilbert space is a so-called inner product space which means an inner product between any two elements in this space associates a complex number to them with some particular properties [11]. However, rather than manipulating these states directly, it is the n qubits formed into tensor product [1] states that are then acted on by unitary operators that lead to various linear combinations of the basis states. The unitary operators within the quantum computing syntax are called gates and the combination of such operators form together to be called a circuit [10]. Thus the state vector in general is the superposition of 2^n basis states, which in principle could contain useful information for computation. Through use of interference the state vector can enhance or destroy certain states in favour of others (just as constructive and destructive interference respectively). This general process will also lead to entanglement of the qubits. Quantum entanglement occurs when qubits share a common quantum mechanical state, so cannot be expressed as a tensor product of their individual states. The results of a measurement performed on one will determine the results of future measurements on the other, even if the qubits are separated by enormous distances.

In quantum computing, similarly to classical computing, data is held in a quantum register. Such a register is represented by a state vector familiar to us from quantum mechanics [10]. For example, if we have a single qubit register, we can represent it by the single qubit itself in the following way:

$$|\Psi\rangle = c_0 |0\rangle + c_1 |1\rangle \tag{1}$$

To illustrate how multiple qubit registers are represented, we also show a 2 qubit register, from which the following representation of higher order registers follows.

$$|\Phi\rangle = c_{00} |00\rangle + c_{01} |01\rangle + c_{10} |10\rangle + c_{11} |11\rangle \tag{2}$$

To better understand the notion of a qubit, we present here the Bloch sphere representation of a qubit, which provides a visual as well as mathematical aid to understanding

[1]Tensor product is a mathematical operation used to construct a big vector space out of 2 or more vector spaces. If we start with vectors of dimension n and m, the result of their tensor product will be an nm dimensional vector space [12]

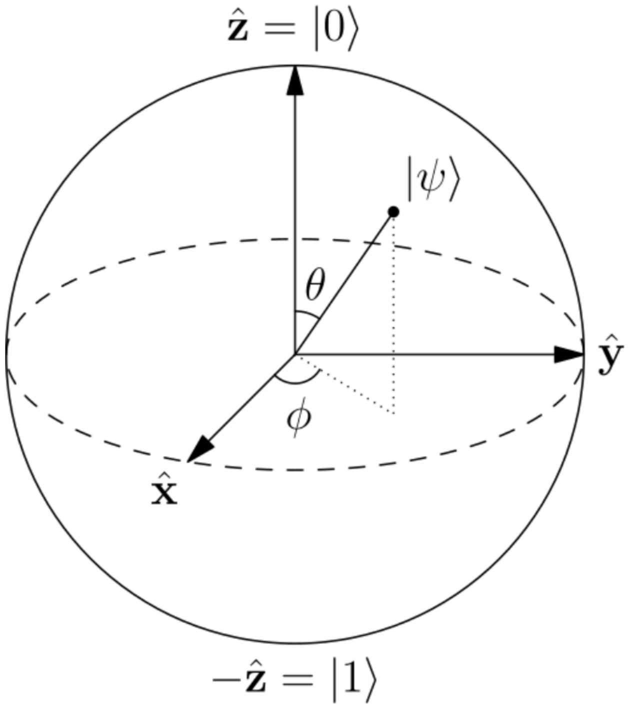


Figure 2: Diagram of a Bloch sphere showing the 2 free parameters θ and ϕ and the two orthogonal vectors at the North and South pole $|0\rangle$ and $|1\rangle$ [14].

the formalism. A qubit, i.e. the analog of a classical bit, can be represented by a vector in a complex Hilbert space. The representation can be written in the same way as in equation 1 where c_0 and c_1 are complex numbers. These complex numbers can be represented by an amplitude and a phase. We can now use the following properties of qubits to manipulate the qubit representation:

- quantum state remains unchanged if multiplied by a number of unit form
- $|c_0|^2 + |c_1|^2 = 1$

Using these properties we can manipulate equation 1 to contain only 2 unknowns. While the algebraic procedure is at this level unimportant, it leads us to the representation of a qubit on a Bloch sphere (see Figure 2). A Bloch sphere is a unit 2-sphere which means it has a radius of magnitude 1 [13]. The vectors at the two opposite poles represent orthogonal state vectors. A qubit on a Bloch sphere is represented in the following way:

$$|\Psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{j\psi}\sin\frac{\theta}{2}|1\rangle \tag{3}$$

1.3.2 Quantum Logic Gates

Quantum logic gates are analogous to classical gates, and we use them to act on a quantum register [10]. They can also be represented by matrices, which is further explored in this section. Quantum logic gates must be unitary linear operators, which means the product of the gate and its hermitian conjugate must equal the identity:

$$UU^\dagger = I \tag{4}$$

This is because we require that the quantum logic gates preserve the norm of the quantum register [10].

We will now present some basic quantum logic gates in both Dirac and matrix representations along with their circuit diagram symbols. These are also particularly relevant for the purpose of this project, as they were used in the implementation of Grover's algorithm described in section 3.1.

The Hadamard gate is frequently used to create an equal superposition of states - this means there will be an equal probability of observing state $|0\rangle$ and state $|1\rangle$ when measured [15]. The following is a matrix and Dirac notation representation of the Hadamard gate.

$$\hat{H}_d = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \frac{1}{\sqrt{2}}\left[\, |0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1| \,\right] \tag{5}$$

Another significant class of quantum logic gates are the Pauli gates [10]. These are a set of rotation gates with the following representations

$$\hat{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = |1\rangle\langle 0| + |0\rangle\langle 1| \tag{6}$$

$$\hat{Y} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = i\,|1\rangle\langle 0| - i\,|0\rangle\langle 1| \tag{7}$$

$$\hat{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1| \tag{8}$$

Since all of these are rotation gates, they can be generalized into a single generic rotation gate called the R_θ-gate [10]. Additionally it is also a modification of the Pauli-Z gate showed above.

$$\hat{R}_\theta = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix} = |0\rangle\langle 0| + e^{i\phi}\,|1\rangle\langle 1| \tag{9}$$

All of these gates are single qubit gates, which means they fundamentally act on a single qubit. There are also multiple qubit gates, which for example utilize the value of one qubit to *control* the operation on the other qubit. These are called controlled gates [10]. We will present here the most relevant gate to our algorithm and also

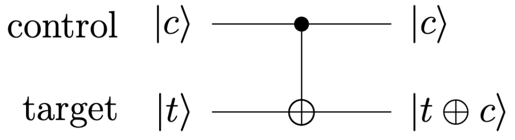


Figure 3: Schematic diagram of the CNOT gate acting on 2 qubits [15]. The symbol $\oplus$ represents the modulo 2 operation or in simple terms the AND statement.

the most commonly used 2-qubit gate - the controlled NOT-gate or better called the CNOT gate.

$$\hat{C} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = |00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 11| + |11\rangle\langle 10| \tag{10}$$

This gate does not do anything when the controlled qubit is $|0\rangle$, and flips the target qubit, if the controlled qubit is $|1\rangle$. The process of using a controlled qubit to change a target qubit is schematically shown in Figure 3.

Finally we also present the graphical representation of these basic quantum gates which are extensively used in circuit diagrams in Figure 4.

The goal is to find the suitable combination of all these basic operations that produce something of computational utility. These are known as quantum algorithms, designed for execution on quantum computers.

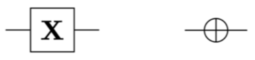
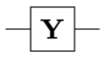
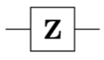
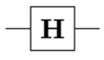
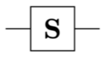
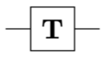
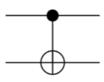
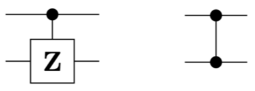
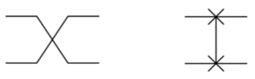
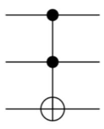
Operator	**Gate(s)**	**Matrix**
Pauli-X (X)		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)		$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S, P)		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$ (T)		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
Controlled Not (CNOT, CX)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled Z (CZ)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

Figure 4: Representation of quantum logic gates in circuit diagrams (middle column) and their matrix representations (right column). Gates not discussed in this section are included as an extension [16].

2 Quantum Algorithms

2.1 Grover's algorithm

Grover's algorithm is one of the most successful of the early algorithms to have been developed for quantum computing. Invented by Lov Grover in 1996 [17], it determines with high probability the unique input from the output of a black box function. In other words, it can find a particular state within a quantum register. In practice, Grover's algorithm is a search algorithm that can find a particular item in a list using just $\sim \sqrt{N}$ steps, where N is the length of the list thus scaling as $O(\sqrt{N})$ [18]. The best (and most trivial) classical alternative to this problem needs on average $N/2$ steps and thus scales as $O(N)$ [18]. Therefore, Grover's algorithm provides a quadratic speedup which becomes very significant for large N.

Consider an unordered list of N entries. The algorithm requires $n = \log_2 N$ qubits to construct an N-dimensional Hilbert space $\mathcal{H}$. Let $|\Psi\rangle$ be a superposition of all states. This can be achieved by applying the Hadamard operator to each qubit of a quantum register of n qubits all initialized to $|0\rangle$ (see Equation 11) [19].

From an information theory perspective we have no knowledge about any state, so we maximize the entropy by assuming each state is initially as probable as any other. This leads to the following construction and corresponding normalization coefficient,

$$|\Psi\rangle = H^{\otimes n} |0\rangle^{\otimes n} = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle \tag{11}$$

Through a series of operations which we will discuss below, Grover's algorithm will increase the coefficient of the desired state, $|w\rangle$, (the element of the list we are looking for) whilst diminishing the coefficients of all other states such that normalization is preserved. This translates to increasing the probability that our initial state, upon measurement, will yield the desired result w.

2.1.1 Step 1: The Oracle

Once the state $|\Psi\rangle$ is initialized, the first step is to apply what is known as a *quantum Oracle* $\mathcal{O}$ to it [19]. This can be thought of as a quantum black-box which observes and modifies the system without collapsing it to a classical state. Its operation on a state $|x\rangle$ is defined as,

$$|x\rangle \xrightarrow{\mathcal{O}} (-1)^{f(x)} |x\rangle, \qquad f(x) = \begin{cases} 0 & \text{for } \forall x \neq w \\ 1 & \text{for } x = w \end{cases} \tag{12}$$

This translates to rotating the desired state $|w\rangle$ by π radians, while leaving all others untouched. It is important to note that this leaves the probability that the state $|\Psi\rangle$, upon measurement, will yield a particular result the same. This is because the probability of measuring each state is proportional to the corresponding magnitude of the coefficient squared, so any phase shift will have no effect on the probability.

On the other hand, what the quantum Oracle $\mathcal{O}$ has done is effectively marked the desired state by assigning to it the only negative coefficient (see Equation 12).

Mathematically, this procedure can also be described by Equation 13.

$$\hat{\mathcal{O}} = I - 2\left|w\right\rangle\left\langle w\right| \tag{13}$$

In fact, when applying the quantum Oracle $\mathcal{O}$ to the state $|x\rangle$ we obtain indeed the same result as in Equation 12.

$$\big(I - 2\left|w\right\rangle\left\langle w\right|\big)\left|x\right\rangle = \left|x\right\rangle - 2\left|w\right\rangle\left\langle w|x\right\rangle = \begin{cases} |x\rangle & \text{for } \forall x \neq w \\ -|w\rangle & \text{for } x = w \end{cases} \tag{14}$$

This quantum Oracle operator can therefore be described as a rank 2 diagonal tensor with 1 along all diagonal entries, except for that associated with state $|w\rangle$ which has -1. This is a particularly useful representation when performing computations [20].

2.1.2 Step 2: Grover's operator

The next step in Grover's algorithm aims at exploiting the fact that the desired state $|w\rangle$ has a different (negative) amplitude to all others, so is smaller than the average of all amplitudes. Mathematically Grover's operator (also known as the diffusion transform in Figure 6) can be described as,

$$\hat{\mathcal{G}} = 2\left|\Psi\right\rangle\left\langle\Psi\right| - I \tag{15}$$

By applying Grover's operator $\mathcal{G}$ onto the resulting state after Step 1, the amplitude of the desired state increases. Similarly, all others decrease such that the total probability is conserved and is 1. Using Equation 11,

$$\begin{aligned} \hat{\mathcal{G}}\hat{\mathcal{O}}\left|\Psi\right\rangle =& \big(2\left|\Psi\right\rangle\left\langle\Psi\right| - I\big)\big(I - 2\left|w\right\rangle\left\langle w\right|\big)\left|\Psi\right\rangle \\ =& \big(2\left|\Psi\right\rangle\left\langle\Psi\right| - I\big)\big(\left|\Psi\right\rangle - \frac{2}{\sqrt{N}}\left|w\right\rangle\big) \\ =& \frac{N-4}{N}\left|\Psi\right\rangle + \frac{2}{\sqrt{N}}\left|w\right\rangle \end{aligned} \tag{16}$$

Therefore, this sequence of steps has increased the probability of finding the desired state upon measurement. Intuitively, one can repeat Steps 1 and 2 in order again to gain higher accuracy [19]. It is interesting to note the case $N = 4$, where a single iteration will yield the desired state with certainty. More generally, we quantify this by considering the ratio r_1 with which the probability of obtaining the desired result has increased after the first iteration (corresponding to the implementation of Steps 1 and 2). This is given by the magnitude squared of the respective amplitude coefficient,

$$r_1 = \frac{\left|\langle w|\,\hat{\mathcal{G}}\hat{\mathcal{O}}\,|\Psi\rangle\right|^2}{\left|\langle w|\Psi\rangle\right|^2} = \frac{\frac{9}{N}\big(1-\frac{4}{3N}\big)^2}{\frac{1}{N}} = \big(3-\frac{4}{N}\big)^2 \tag{17}$$
$$\lim_{N\to\infty} r_1(N) = 9$$

Therefore, after the first iteration, the probability of obtaining the desired state $|w\rangle$ after measurement increases by nearly an order of magnitude, for sufficiently large N.

2.1.3 Maximizing the accuracy

Unfortunately, one cannot keep repeating Steps 1 and 2 to obtain any desired accuracy. Only an integer number of iterations is physically possible and this will eventually limit the probability of obtaining the desired result w, upon measurement. Let our initial state $|\Psi\rangle$ be re-written as,

$$|\Psi\rangle = \alpha\,|w\rangle + \beta\,|w_\perp\rangle \tag{18}$$

Where $|w_\perp\rangle$ is a linear combination of all other states, which clearly must be perpendicular to $|w\rangle$. Moreover, in this case $\alpha, \beta \in \mathbb{R}$ must be real constants which satisfy the usual probability condition $|\alpha|^2 + |\beta|^2 = 1$. This equation can be thought of as that for a unit circle (see Figure 5). In particular, we define $\alpha = \sin\theta$ and correspondingly $\beta = \cos\theta$.

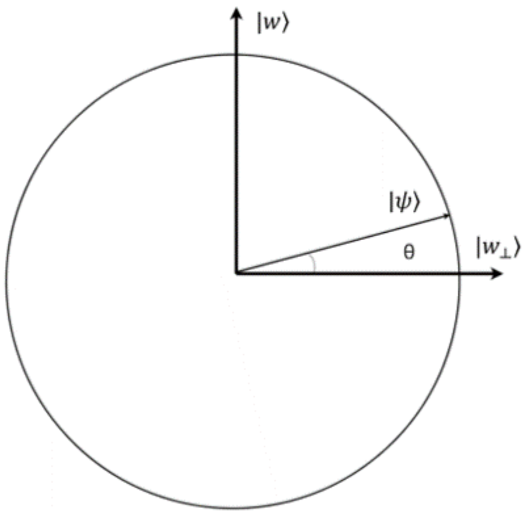


Figure 5: Unit circle visual representation we created for the decomposition of state $|\Psi\rangle$.

Where $\sin\theta = \frac{1}{\sqrt{N}}$ follows directly from a comparison with Equation 11. Therefore, intuitively, Step 1 will reflect the state about the $|w_\perp\rangle$ axis as it flips the sign of the $|w\rangle$ coefficient. Successively, Step 2, reflects this result about the (N-dimensional) vector $|\Psi\rangle$. Geometrically, repeating this iteration, will align the state closer and closer to the desired result; the $|w\rangle$ axis. However, an excessive number of iterations will overshoot this [21].

Clearly, after applying $k \in \mathbb{N}$ iterations, the resulting state becomes,

$$|\Psi_k\rangle = \sin\big((2k+1)\theta\big)\,|w\rangle + \cos\big((2k+1)\theta\big)\,|w_\perp\rangle \tag{19}$$

Since we want the coefficient of $|w\rangle$ to be as close as possible to 1, we require $(2k+1)\theta = \frac{\pi}{2} + 2\pi n$ for $n \in \mathbb{Z}$ any integer. Although mathematically these solutions are correct, physically the solution requiring the smallest number of iterations k is the most important ($n = 0$). Hence the criterion required to determine the optimal minimum number of iterations to ensure certainty in determining w upon measurement is,

$$k = \frac{\pi}{4\arcsin\frac{1}{\sqrt{N}}} - \frac{1}{2} \tag{20}$$

Importantly, by setting $N = 4$ we obtain the exact result $k = 1$. Only 1 iteration is required to determine w with absolute certainty, which agrees with the algebraic observations from Equation 16. This special case arises naturally from the rotational geometry of Grover's algorithm [21]. The other special case is the obvious solution $k = 0$ when there is only 1 state, so 1 element in the list.

It is the nature of $\pi \in \mathbb{R}/\bar{\mathbb{Q}}$ being a transcendental number by the Lindemann – Weierstrass theorem [22] (where $\bar{\mathbb{Q}}$ represents the algebraic numbers) which limits the number of exact integer solutions for k. Clearly, to seek other exact solutions, the π factors must cancel so the following conditions are necessary,

$$\begin{aligned} \arcsin\frac{1}{\sqrt{N}} = \frac{\pi}{a} \qquad & \text{for } a = 2, 6, 10, 14... = 4l-2, l \in \mathbb{Z}^+ \\ & \therefore N = \frac{1}{\sin^2\left(\frac{\pi}{4l-2}\right)} \end{aligned} \tag{21}$$

As $N \in \mathbb{N}$ is a natural number, the solutions to Equation 21 are indeed limited to the case $N = 1$ ($l = 1$) and $N = 4$ ($l = 2$). However, although, an exact solution and certainty in determining w after some integer iteration k is limited, the closest integer to maximize the probability of obtaining the desired w after measurement is given by rounding Equation 20 to the nearest integer. Using $\left[x\right]$ to symbolize rounding to the nearest integer and observing that $\left[x - \frac{1}{2}\right] = \lfloor x \rfloor$ is the floor function of x,

$$k_{optimal} = \left[\frac{\pi}{4\arcsin\frac{1}{\sqrt{N}}} - \frac{1}{2}\right] = \left\lfloor \frac{\pi}{4\arcsin\frac{1}{\sqrt{N}}} \right\rfloor \tag{22}$$

$$k_{optimal} \approx \left\lfloor \frac{\pi\sqrt{N}}{4} \right\rfloor \tag{23}$$

In the last equality the approximation $\arcsin x \approx x$ is used, which is valid for large N. This is a useful approximation as $N = 2^n$ scales exponentially with the number of qubits. Therefore, the famous result that Grover's algorithm has complexity $O(\sqrt{N})$ is recovered. Although the discussion above was valid when searching for 1 state, the generalization of Grover's algorithm for searching multiple elements $M \in \mathbb{Z}^+$ yields a similar result $O(\sqrt{\frac{N}{M}})$ [23], retaining the characteristic $\sqrt{N}$ asymptotic behaviour.

2.1.4 Considerations and limitations

Also in 1996, around the time Grover published his algorithm, Vazirani *et al* proved that any quantum solution to this problem must have at least complexity $O(\sqrt{N})$, demonstrating that Grover's algorithm is asymptotically optimal [24].

Overall, Grover's algorithm can be succinctly described by Figure 6.

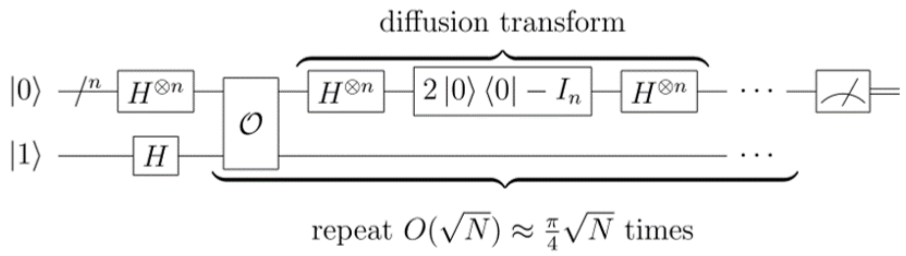


Figure 6: Quantum circuit diagram representation for Grover's algorithm [19]. Here the diffusion operator represents Grover's operator. Importantly, once the unitary operators acted on the state for all the iterations, the system is (irreversibly) measured to provide a result.

Grover's algorithm, and more generally the family of quantum algorithms that use what is known as amplitude amplification, exploit the extra information contained within the coefficients (phase shifts) of each state.

As mentioned in Section 2.1.1, Grover's algorithm is able to utilise this to mark the desired state without affecting the probability of observation. Subsequent transformations (Grover's operator) take advantage of this difference in amplitude to ultimately increase the probability of the system being in that state.

These amplitude amplification algorithms are unique to quantum computing because the concept of amplitudes has no analog in classical probability [19]. From an information theory point of view, one could intuitively understand why Grover's algorithm might have a quadratic speedup. From a classical perspective, the only information known is the probability of measuring each of the N states (say N possibilities), whilst in the quantum realm one also has the phase shift of each state (so N^2 possibilities).

One of the main limitations of this algorithm is the probabilistic nature of the result. In the classical case, it is guaranteed that the outcome of the search will be the correct result. On the other hand, as discussed earlier, this is not true for Grover's algorithm and in general one can only quantify an upper bound of the error, even with an optimal number of iterations.

Furthermore, most of the literature assumes that quantum circuits are free from decoherence (quantum noise) [25]. From a practical point of view, if quantum computers will become viable, decoherence will remain unavoidable [25]. This can be thought of as the quantum version of the Gaussian noise in classic circuits due to electrons executing Brownian motion. However, much of modern research has focused on mitigating these effects. In particular, decoherence exists because of the fundamental interactions between the qubits and the surrounding environment, which perturbs the superposition of states [26] in Equation 11. This may be achieved using quantum error correction.

2.1.5 Quantum Error Correction

Quantum error correction aims to ensure that qubits keep their coherent quantum state by mitigating the effect of quantum noise. It has been experimentally demonstrated using superconductors that this increases the coherence time of the unperturbed quantum state, to allow for computation to be more accurate [27]. Several methods have been proposed and, in fact, there is no method which is best for all algorithms [25].

For example, a very recent study argues that for Grover's algorithm in particular, what is known as Steane's Code is ideal from an implementation-oriented perspective [25]. This quantum error correction improves the probability of Grover's algorithm returning the correct result, by substantially reducing the qubit error ratio of each link between a pair of concatenated gates [25]. The results show that this probability increased from 0.22 to 0.96 when Steane's Code was implemented. The large improvement is due to the code's ability to correct both the phase flips (sign change) and bit flips (transitions in the state of qubits) [28].

In fact, this group believes the success of quantum error correction in the next years will be crucial for the development of a future quantum computer, in order to promote a widespread use of quantum algorithms.

2.2 Deutsch-Jozsa algorithm

First proposed by Richard Jozsa and David Deutsch in 1992, it is one of the first quantum algorithms to be exponentially faster than the best deterministic classical counterpart [29]. Unlike Grover's algorithm, the Deutsch-Jozsa algorithm is deterministic (without considering experimental errors), so will theoretically always produce the correct answer [29].

The problem is formulated as follows. Suppose there is an n-bit function which takes an input of n-bits and yields either the number 1 or 0:

$$f : \{0,1\}^n \rightarrow \{0,1\} \tag{24}$$

Suppose this function can be either a constant or be balanced. The former means that the output of $f \equiv f(\{x\})$ in Equation 24 is either always 0 or always 1 for every n-bit input. The latter means that $f = 0$ for precisely half of all the n-bit inputs $x \in \{0,1\}^n$ and $f = 1$ for the other half.

2.2.1 Classical Solution

Classically, in the best scenario, two queries to the oracle can determine if the hidden Boolean function f is balanced. For example, if $f(0,0,0,...) \to 0$ and $f(1,0,0,...) \to 1$, then the function must be balanced as it yielded two different outputs.

The worst case scenario will occur when the function will give the same output for each input that is given to f. In this case exactly half of all possible inputs plus one are required in order to be certain that f is constant. This corresponds to $2^{n-1}+1$ trials. A strategy would be to give random inputs to f. The probability of determining the correct result after k trials is clearly,

$$P_k = \begin{cases} 1 - \frac{1}{2^k} & \text{for } 1 \leqslant k \leqslant 2^{n-1} \\ 1 & \text{for } k = 2^{n-1}+1 \end{cases} \tag{25}$$

Hence, using a randomized classical algorithm, it is possible to truncate the process early and still obtain a high confidence in the result. The gap between the randomized classical complexity and the quantum query complexity is constant for a constant error [30]. On the other hand, there is an exponential speedup in the the quantum solution when a deterministic answer wants to be found [30].

2.2.2 Quantum Solution

By using Deutsch-Jozsa's algorithm this problem can be solved with certainty (theoretically speaking) using only 1 iteration $O(1)$, independent of the number of qubits [29]. This is a remarkable result.

The algorithm begins with the n-bit state $|0\rangle^{\otimes n}|1\rangle$ as shown from the circuit diagram, Figure 7. A Hadamard transform is applied to each bit to obtain the state (recalling its definition from Section 1.3),

$$|\Psi\rangle = H^{\otimes n}|0\rangle^{\otimes n} H|1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle H|1\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} |x\rangle \big(|0\rangle - |1\rangle\big) \tag{26}$$

2.2.3 Step 1: Applying the Oracle

Recalling the definition of f in Equation 24, define the oracle acting on a n-bit register $|x\rangle_n$ as

$$\hat{\mathcal{O}}|x\rangle_n = (-1)^{f(\{x\})}|x\rangle_n \tag{27}$$

The subscript is included to emphasize that $|x\rangle_n$ is composed of n-bits. The integer x is therefore expressed as a binary number in a tensor product state with n entries.

Therefore, the oracle maps the state $|x\rangle_n |y\rangle$ to $|x\rangle_n \left|y \oplus f(\{x\})\right\rangle$. Acting the Oracle on $|\Psi\rangle$ in Equation 26 yields,

$$\frac{1}{\sqrt{2^{n+1}}}\sum_{x=0}^{2^n-1}(-1)^{f(\{x\})}|x\rangle\big(|0\rangle-|1\rangle\big)=\frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}(-1)^{f(\{x\})}|x\rangle \tag{28}$$

Where the last qubit can now be ignored because it won't affect the rest of the calculation, in particular the probability of measuring $|0\rangle^{\otimes n}$. A useful Lemma is included to procede,

Lemma 1. *The result of a Hadamard operator $H^{\otimes n}$ acting on an n-qubit reister $|x_n...x_1\rangle_n$ for $n \in \mathbb{Z}^+$ is given by*

$$H^{\otimes n}|x_n...x_1\rangle_n=\frac{1}{2^{n/2}}\prod_{i=1}^{n}\sum_{y_i=0}^{1}(-1)^{\sum_{j=1}^{n}x_jy_j}|y_n...y_1\rangle_n \tag{29}$$

where $x_j = 0, 1$ for $j = 1, ..., n$.

Proof. To prove the lemma, the definition of the Hadamard operator from Section 1.3 is used. Therefore,

$$\begin{aligned}H^{\otimes n}|x_n...x_1\rangle_n&=\frac{1}{\sqrt{2}}\big(|0\rangle+(-1)^{x_1}|1\rangle\big)\frac{1}{\sqrt{2}}\big(|0\rangle+(-1)^{x_2}|1\rangle\big)...\frac{1}{\sqrt{2}}\big(|0\rangle+(-1)^{x_n}|1\rangle\big)\\&=\frac{1}{2^{n/2}}\prod_{i=1}^{n}\big(|0\rangle+(-1)^{x_i}|1\rangle\big)\\&=\frac{1}{2^{n/2}}\prod_{i=1}^{n}\sum_{y_i=0}^{1}(-1)^{x_iy_i}|y_i\rangle\\&=\frac{1}{2^{n/2}}\prod_{i=1}^{n}\sum_{y_i=0}^{1}(-1)^{\sum_{j=1}^{n}x_jy_j}|y_n...y_1\rangle_n\end{aligned} \tag{30}$$

□

2.2.4 Step 2: Applying the Hadamard Operator

Next, the Hadamard Operator is applied to the result of Equation 28 which gives Equation 31. This Hadamard operator is acting only on the n-qubits $|0\rangle^{\otimes n}$. By using Lemma 1,

$$H^{\otimes n}\frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}(-1)^{f(\{x\})}|x\rangle=\frac{1}{2^n}\sum_{x=0}^{2^n-1}\sum_{y=0}^{2^n-1}(-1)^{f(\{x\})+x\cdot y}|y\rangle \tag{31}$$

Where $x \cdot y = x_0y_0 \oplus x_1y_1 \oplus ... \oplus x_{n-1}y_{n-1}$ is the sum of the bitwise product.

The probability of measuring $|0\rangle^{\otimes n}$ is examined. This is given by the magnitude of the amplitude squared of the state in Equation 31 after projecting it with the bra $\langle 0|^{\otimes n}$. The orthonormality of basis states will sift the $y = 0$ term in the sum such that $x \cdot y = 0$ to yield the following [31].

$$P(|0\rangle^{\otimes n}) = \left| \frac{1}{2^n} \sum_{x=0}^{2^n-1} (-1)^{f(\{x\})} \right|^2 = \begin{cases} 0 & \text{if } f \text{ is balanced} \\ 1 & \text{if } f \text{ is constant} \end{cases} \tag{32}$$

Therefore if the outcome upon measurement is $|0\rangle^{\otimes n}$, then f must have been a constant. Importantly, if the amplitude and not the probability is computed, the precise output of the function f can also be determined. Clearly, an amplitude of 1 translates to $f = 0$ (destructive interference), whilst an amplitude of -1 must mean that $f = 1$ (constructive interference).

All this information was obtained with 1 single call of the oracle, so 1 iteration. Figure 7 visually summarizes the process involved in Deutsch-Jozsa's algorithm.

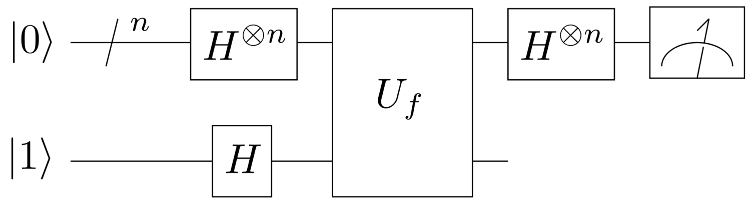


Figure 7: Quantum circuit diagram representation for Deutsch-Jozsa's algorithm [31].

2.2.5 Considerations and limitations

Although the algorithm has an exceptional space and time complexity of $O(1)$ and is exponentially faster than the best classical alternative for a deterministic result, it has limited uses. This is because the algorithm was designed to outline and demonstrate the quantum advantage, so it was not specifically thought to solve a practical problem [32].

Anyhow, the algorithm has sparked interest in the field of cryptography [33] and genetics due to its speedup [34]. A recent paper in 2018 demonstrated the utility of using this algorithm to measure what is known as the Hammond distance, which is extensively used to calculate the genetic distance in biology [35]. This is important in many branches of the subject. As an example, the distribution of genetic distances is a tool for investigating disease transmission [36]. Current methods to investigate transmission dynamics require intensive computation so are highly expensive [36].

This group personally thinks the algorithm could find application to viral evolution and disease outbreaks. With examples such as the COVID-19 coronavirus pandemic, this could become an important field of research. The speedup the Deutsch-Jozsa's offers could become vital for governments to plan containment measures timely and effectively. More future applications of quantum algorithms are discussed in detail in Section 4.

Although such an application could be possible with quantum computers, as discussed in Section 2.1.5, from a practical point of view, quantum noise (decoherence) will need to be addressed. The main limitation of Deutsch-Jozsa's algorithm is that it assumes the quantum oracle computing f from it's input does not decohere the input [37]. A proposed solution, quantum error correction, is discussed in Section 2.1.5.

Moreover, the oracle must not leave any copy of the input lying around or else computational paths with different residual information do not interfere [24]. This is important as the phenomena of interference is required to provide quantum algorithms with a speedup [38].

2.3 Other algorithms

Progress in quantum algorithms is necessary for quantum computing to succeed. This has lead extensive research in developing new ones. Although a quadratic speed-up for Grover's algorithm is substantial, Shor's algorithm (inspired from Deutsch-Jozsa's algorithm) can find the prime cofactors of an integer N (nearly) exponentially faster than the best classical alternative [39]. More recently, in 2009, Harrow, Hassidim and Lloyd created an algorithm (HHL) which provides an exponential improvement for solving linear systems of equations relative to the best classical alternative [40]. This is a task which is extensively performed in the mathematical sciences, both on its own and for solving more complex problems.

As well as for established mathematical fields, the HHL algorithm is also playing a fundamental role for the development of several new interdisciplinary fields, such as quantum machine learning (QML). HHL is used extensively in QML, as matrix operations are key to several machine learning algorithms [41]. Table 1 summarizes the speedups of some QML algorithms. Overall, the HHL algorithm has wide applicability, and therefore shows a great deal of promise in the field of quantum computing.

Quantum Machine Learning Algorithm	Speedup
Quantum Boltzmann Machines	$O(\log n)$
Quantum Principal Component Analysis	$O(\log n)$
Quantum Support Vector Machine	$O(\log n)$
Quantum reinforcement learning	$O(\sqrt{n})$

Table 1: Table summarizing various quantum machine learning algorithms and their corresponding speedup compared to their classical counterparts. This is quantified using asymptotic big O notation [42].

2.4 Devices for Quantum Computers

Finally, it is of fundamental importance to address how such algorithms would be implemented in practice on a future quantum computer. Any device must be able to [43]:

1. Prepare a fiducial initial state.

2. Perform a universal set of unitary transformations corresponding to some basic quantum gates discussed in Section 1.3.
3. Represent quantum information robustly, minimizing the effect of quantum noise/decoherence.
4. Measure the outcome in an appropriate basis.

The performance of a device is measured by the ratio of the decoherence time τ_{de} to the time scale of operation of a typical quantum gate τ_{op} [44]. The aim is to construct devices with the highest possible ratio. Clearly, a large decoherence time is beneficial as this represents a characteristic timescale for which the loss of information from the quantum system of interest into the environment becomes significant. This can be achieved by reducing the coupling with the environment as discussed in Section 2.1.5. On the other hand, any device and any measuring apparatus also couples with the quantum system of interest [44]. Therefore, an excessive reduction in coupling will hinder the control of the state of the device and its ability to measure the outcome. A trade-off is inevitable.

In recent years, several classes of devices have been proposed with varying degrees of success. Two promising methods are briefly described below.

- **Ion traps**: Ions are cooled to freeze vibrational degrees of freedom [45]. Ions are then trapped using EM fields [44]. Qubits are manipulated by optical laser beams and are represented by the low lying quantized modes of vibration of the ion chain [45].
- **Quantum dots**: Electrostatic fields are used to confine electrons and holes inside microscopic boxes (quantum dots) in metals, small molecules or semiconductors [46]. The quantized energy levels of these confined charges (electrons and holes) are used to store the qubits [46]. These are finally controlled by single mode wave guide couplers (analogous to beam splitters) and electrostatic gates (analogous to phase shifters) [44].

3 Algorithmic Implementation

3.1 The Simulator

The simulator is structured in an object-oriented way, which gives clarity to the overall organization and enables the construction of different quantum circuits from hard-coded elementary 2×2 quantum logic gates. Additionally, to fulfill the purpose of the assignment, no external packages to aid with computing e.g. `numpy` for tensor-kronecker products or sparse matrices were used.

The design we followed is relatively simple. To begin with, the simulator has the capacity to create a quantum register of arbitrary length which the user has to prepare (i.e. initialise) at a particular basis state. We decided to not allow initialisation of registers in a superposed quantum state, because it would increase the complexity of the code. Furthermore, initialising a quantum register in a superposed quantum state is not a straightforward task in real life either; given our understanding of quantum computing in the beginning of the course, we decided to avoid perplexing things further.

The user can then construct *parallel* gates that can be applied to the register sequentially (any circuit can be represented as a sequence of parallel gate operations). We are using the word *parallel* to signify that individual gates are being applied to all qubits at the same time. We do not provide a special option for the user to apply gates to qubits one-by-one, because all possibilities can be covered using our design. In case the user desires to apply a gate only to qubit i, leaving all others untouched, they can proceed by constructing a *parallel* gate with the identity applied to all qubits except for qubit i, where they can place their gate of choice.

Although controlled gates are allowed in our simulator, and can moreover be applied to subgroups of qubits in the register, there is no way possible to apply them to non-consecutive qubits. For example, in a 3-qubit register, it is impossible for the user to apply a C-NOT gate on qubit-0, controlled on qubit-2, because they are not adjacent to each other (further explored in section 3.1.2). Furthermore, there is no way of applying controlled gates to qubits more significant than their controls; a C-NOT gate on qubit-1, controlled on qubit-0 is impossible. Our design still covers a very wide range of circuits, however this functionality could be further explored and improved upon in the future.

Finally, the user can perform a measurement of the register, after they have completed the desired sequence of operations. The measurement can only be performed on all qubits at once; measuring one qubit is not allowed. This is to avoid complications in case there is entanglement.

The measurement can be completed realistically, where the register collapses to one chosen basis state and all information about the state vector before the measurement is lost. The measurement can also be carried out virtually, where one basis state from the register is returned, but the register remains in a superposition. In both cases, the probability of a basis state being chosen depends on its amplitude squared, as it should.

3.1.1 Representing Quantum States and Registers

States

Any basis state is represented by the `QubitState` class, which is the base class for `State` and serves as an interface. All methods inside this class are abstract methods which are overridden in child classes. Some methods are implemented within this class that represent useful operations. The `flip` method enables one to flip any state from a bra to a ket. Moreover, the `__string__` method returns a supplied `State` in a string representation.

The `State` class serves to represent a multi-qubit basis state in the computational (z) basis. This class uses the usual binary convention, in which the least significant qubit is on the right. The attributes of this class enable a state representation in binary, denary, ket form, its dimension and a string representation. All of these are useful in different parts of the code and for the user, which is why so many representations are available.

Registers

The class `Register` is used to represent a quantum register at any point in time. It receives one basis state as its input which corresponds to what is possible in the real world as well. The main way of representing the register in this class is `self.stateVector` in which the coefficients for each basis state are stored. One can apply any supplied gate on the objects of this class - the registers - using `applyGate`, which is widely used throughout the algorithm.

Since quantum systems are fundamentally non-deterministic, it is important to integrate this into the algorithm as well. The `measure` method chooses and returns one basis state, using a Monte-Carlo approach; the probability of a state being chosen depends on its amplitude. The method works by assigning bins to each basis state in the interval $(0, 1)$, the width of the bin being equal to the square of the amplitude of the state, i.e. the probability of it being chosen. It then generates a random number between 0 and 1, and *chooses* the state that corresponds to the bin within which the random number lands. This class also enables the use of a string representation of the register utilized for outputs to the terminal and plots.

Lastly the `measure_collapse()` method, is an extension of `measure()` that actually *collapses* the stateVector into the chosen basis state. i.e. the register stops being in a superposition after running `measure_collapse()`, and instead adopts one basis state.

3.1.2 Matrix Operations

In the `Operations.py` file one can find all the matrix operations utilised in the program, but also the 2x2 representation of all the elementary gates used: Hadamard, Identity, and Pauli rotation matrices. The quantum logic gates used in Grover's algorithm implementation and their detailed decomposition is described in section 3.2.1.

The methods in this class can calculate the sum of 2 matrices (`matrixSum`), their product (`matrixProduct`) and the determinant of a matrix (`matrixDet`). The class also contains methods for inverting a given matrix (`matrixInv`), and calculating the tensor and Kronecker product (`tensorProduct` and `kroneckerProduct` respectively). All of these operations ensure that matrices are of the correct size beforehand and can be applied to matrices of any size ensuring the universality of our program.

The `constructGate` is used to construct particular *parallel* gates. This method works by parsing a carefully formatted string, a *code*, in which each character represents the gate to be applied to each qubit. To avoid confusion in our program, we use the convention that the first character of the *code* represents the gate that is to be applied to the most significant (leftmost) qubit, the second is applied to the second most significant qubit, and so on. This *parallel* operation can be represented by a $2^n \times 2^n$ matrix, acting on the 2^n - dimensional state vector. The `constructGate` method essentially constructs this matrix, by performing a Kronecker product of the individual gates. We initially considered a more basic qubit by qubit gate application, which uses significantly less memory and time. However, this method fails to correctly simulate entanglement, because it is restricted to acting on the qubits as if they are isolated from each other; had we implemented it this way, we would have not been able to simulate controlled gates. Thus we ended up with the longer more computationally expensive alternative, yet allowing a wider range of circuit designs.

3.1.3 Sparse Matrix Implementation

The use of big matrices ($2^n \times 2^n$, where n is the number of qubits) to represent parallel gates has the effect of significantly increasing running time as n increases. As most operation matrices are sparse [2], we decided to implement a special class for them. The class `Sparse` makes use of dictionaries, and we found an immense decrease in running times when working with sparse matrices (over x1000 faster for 7-qubit registers and even halving the running time of operations for dense matrices such as the parallel Hadamard gate). This speedup is illustrated in figure 10 and further described in section 3.2.2.

This class can receive both $2D$ arrays and dictionaries as inputs and it contains two methods: one called `asMatrix` for returning the sparse matrix in $2D$ array form, and the `__str__` method, that uses `asMatrix` to print the matrix on the screen.

The sparse matrix class was implemented after we had a working simulator. This meant that we had to add functionalities to all matrix operation methods to implement sparse matrices correctly. We decided to keep both the old (dense matrix) and the new (sparse matrix) implementations, so all operation methods have conditionals that check the format of the parameters (i.e. sparse or not) before proceeding with calculations. This once again increases the universality of our program. We consider the use of sparse matrices to be something extra, as the simulator can work adequately without them (at least up to and including 7 qubit registers after which the computational time is rather excessive). Thus, we have decided to set the default mode of the simulator to be non-sparse. This can be easily controlled with a boolean

[2]Sparse matrix is such a matrix in which most of its elements are zero.

parameter. An example of how the simulator can be used can be found in Appendix A.

3.2 Grover's Algorithm

Figure 8. shows a schematic diagram of the structure of the program that runs Grover's algorithm. It starts with an input from the user asking for the target state, i.e. the state s one is looking for, and for the number of qubits n in the register. Then it moves into the *preparation* phase, in which the Hadamard, Oracle and Grover's gates are constructed. These are described in detail in section 3.2.1. Afterwards the program creates a pure register, which means the states are not yet in a superposition. At this stage all preparatory procedures will have finished and the program moves into executing Grover's Algorithm.

This starts with applying an n-dimensional Hadamard gate to the register to create a uniform superposition as required by the algorithm. Then a sequence of Oracle, Hadamard, Grover's, Hadamard gates is applied to the register in approximately $\sqrt{n}$ iterations [3].

This whole process is timed for evaluating the time efficiency of the program for different numbers of qubits. When the iterations terminate, the virtual measurement of the system is simulated k times. The uncertainty in measuring the system is modeled at this point with a Monte-Carlo-like approach. After the system is observed k times, information about the states and their observed frequency is shown both in the terminal and as a bar chart. This is where the program terminates.

3.2.1 The Gates

The three gates used throughout Grover's algorithm (Oracle, Hadamard and Grover's Gate) are formally constructed in the `grovers.py` file. These can then be applied to the quantum register in any order desired. Since we are only constructing a simulator and do not have access to an actual quantum computer, the used quantum logic gates were decomposed into elementary gates and processes that can be easily programmed on a classical computer to effectively simulate the action of the quantum computer.

Oracle Gate

The oracle gate is dynamically constructed in the `Oracle` method. For the target state `s` and for a user-specified number of qubits `nq`, `Oracle` returns a matrix which serves as the oracle gate when applied to the state vector of the quantum register. It is made out of two parallel gates, applied in 3 Layers. We will call these two gates the *Filter* (F), which consists of NOT and Identity gates, and the CZ gate, an n-dimensional Controlled-Z gate acting on the smallest qubit.

Layer 1: The filter transforms the target state $|s\rangle$ (and only the target state), into the state $|2^n - 1\rangle$ (i.e. $F(s)|s\rangle = |2^n - 1\rangle$). It is easy to infer that F has to be a function of `s`, and it is dynamically constructed according to its binary representation.

[3]The actual number of iterations used is the one dictated by the exact solution

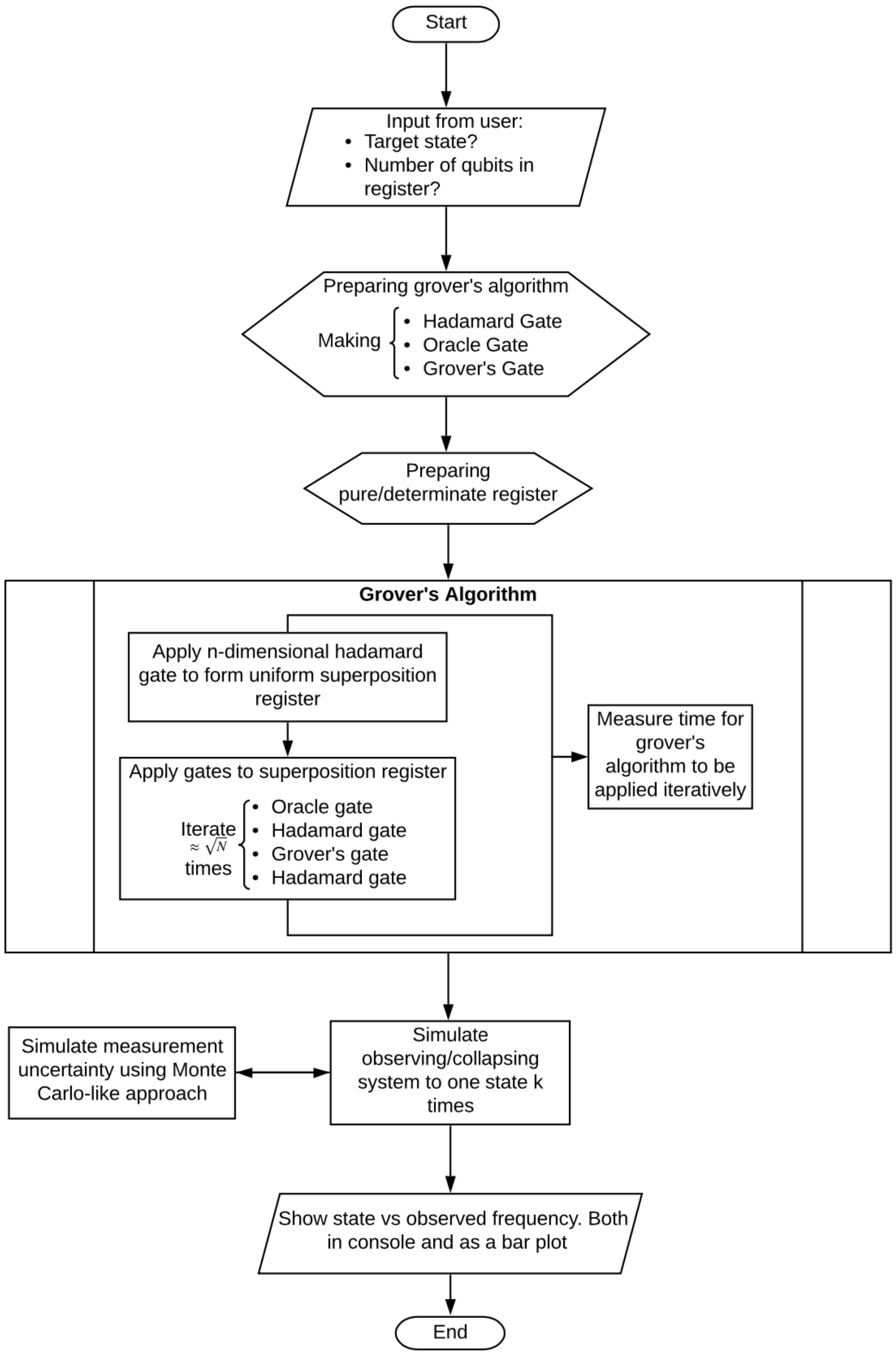

Start
Input from user:
• Target state?
• Number of qubits in register?
Preparing grover's algorithm
Making
• Hadamard Gate
• Oracle Gate
• Grover's Gate
Preparing pure/determinate register
Grover's Algorithm
Apply n-dimensional hadamard gate to form uniform superposition register
Apply gates to superposition register
Iterate $\approx \sqrt{N}$ times
• Oracle gate
• Hadamard gate
• Grover's gate
• Hadamard gate
Measure time for grover's algorithm to be applied iteratively
Simulate measurement uncertainty using Monte Carlo-like approach
Simulate observing/collapsing system to one state k times
Show state vs observed frequency. Both in console and as a bar plot
End

Figure 8: Flowchart of the structure of our program showing the key steps and procedures.

Informally, this is done by flipping all qubits of the target state that are equal to zero into ones, and leaving the ones that are equal to one unchanged. As an example, let us pretend that the target state is the basis state $|1\rangle = |001\rangle$, in a 3-qubit register. The *Filter* will then be $F = X \otimes X \otimes I$, i.e flipping qubits q_2 and q_1, and leaving q_0 unchanged. Subsequently when F acts on the target state we get, $F(|1\rangle)|001\rangle = |111\rangle = |7\rangle = \left|2^3 - 1\right\rangle$, as required. As another example, consider the *Filter* for state $|3\rangle = |0011\rangle$ (in a 4-qubit register), which is $F(|3\rangle) = X \otimes X \otimes I \otimes I$. Thus, it is clear how a general *Filter* is constructed for arbitrary `s` and `nq`.

Layer 2: The CZ gate then acts on state $|2^n - 1\rangle$ constructed in *Layer 1*, as $CZ\,|2^n - 1\rangle = -\,|2^n - 1\rangle$, flipping its sign. The action of the first 2 layers consecutively can be written as $CZ(F\,|s\rangle) = -\,|2^n - 1\rangle$.

Layer 3: Finally the *Filter* F is applied for one last time, returning the negated target state $-\,|s\rangle$, while all other basis states are returned unchanged.

Mathematically, our Oracle is decomposed as such: $O = F(CZ)F$, where F and CZ are the matrix representations of the *Filter* and CZ gates.

Hadamard Gate

The `Hadamard` method, constructs a parallel `nq`-qubit Hadamard gate, again returning its matrix representation. This gate is shown in detail in section 1.3.2.

Grover's Gate

Grover's gate is constructed in the `Diffuser` method. This method is very similar to the Oracle; with three layers and a CZ operation in the middle layer. The only difference is that the *Filter* here is independent of the target state, and is just a parallel X gate, applied to all qubits. One may observe that Grover's decomposition will always be exactly the same as the decomposition of the Oracle for state $|0\rangle$ in the same computational basis.

Combining the Gates

The `Grovers` part of the `grovers.py` file runs Grover's algorithm altogether. By taking in the number of qubits and the denary representation of a state, it iterates through the applied gates and returns the register at the end. It is capable of adapting gates dynamically depending on the mode and number of qubits selected by the user. The method also outputs the time taken to run Grover's algorithm, which is used to track the efficiency across different numbers of qubits.

3.2.2 Accuracy and efficiency

Since Grover's algorithm is not 100% accurate in reality, and will only find the desired state with probability lower than one, the user may desire to repeat (or simulate the repetition of) the algorithm, to increase their confidence in the answer.

As the state of the system before the measurement is definite (i.e. there is no uncertainty in the elements of the state vector), we do not need to run Grover's

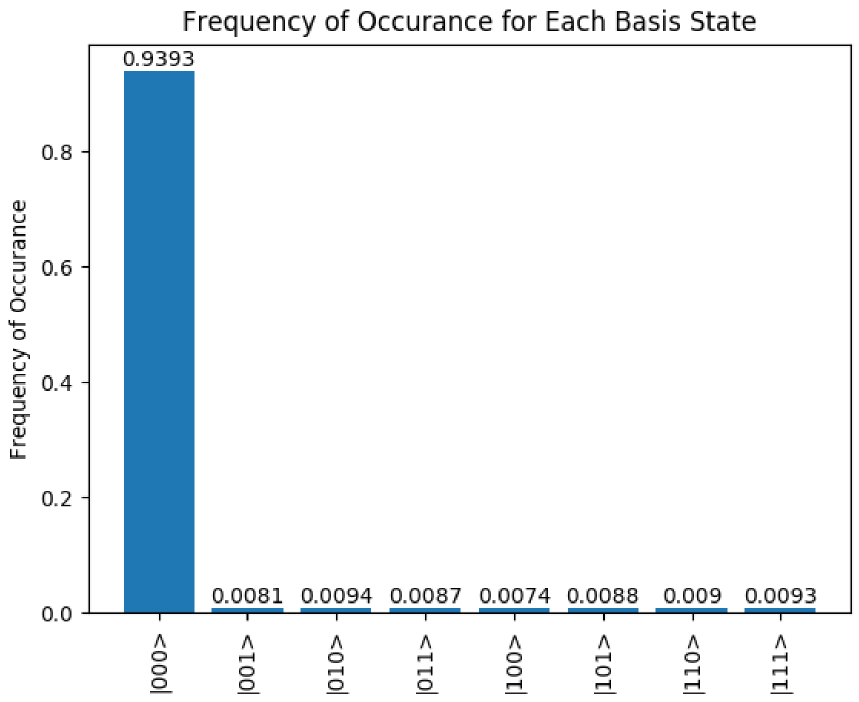


Figure 9: Plot of frequencies of measurement of each basis state for a 3 qubit register, with target state 0, for k=10000. We see that for a 3-qubit register, there is a non-zero probability of obtaining the wrong answer after running Grover's. Thus it is always important to repeat the algorithm a few times, in order to increase confidence in the correct answer.

algorithm many times. For this reason the method `Observe_System` was implemented. This method essentially simulates the final measurement k times using the `Register.measure` method.

In this part of the program we traded some realism for running time. We could have followed a different approach where we would use the `Register.measure_collapse` method, but this would collapse the state vector and all information about the individual components would be lost. This is what we would expect from a real Quantum computer after all. Unfortunately, this would mean that Grover's algorithm would have to be repeatedly run before every measurement, which would significantly increase the running time of `Observe_System`. We would need to repeat the same operation over and over to produce the same final state vector. We decided against this approach, and instead simulated the repetition of the algorithm, by performing a virtual measurement, a given number of times.

To track the accuracy of the algorithm itself, a plot is created in the `FrequencyPlot` method, which shows how many times a particular state was selected by the algorithm when ran multiple times with the same input parameter. An example of a plot for 3 qubits is shown in Figure 9, where the right answer was returneed with 93.93% accuracy.

The `efficiency.py` file was created to track the efficiency of our program while

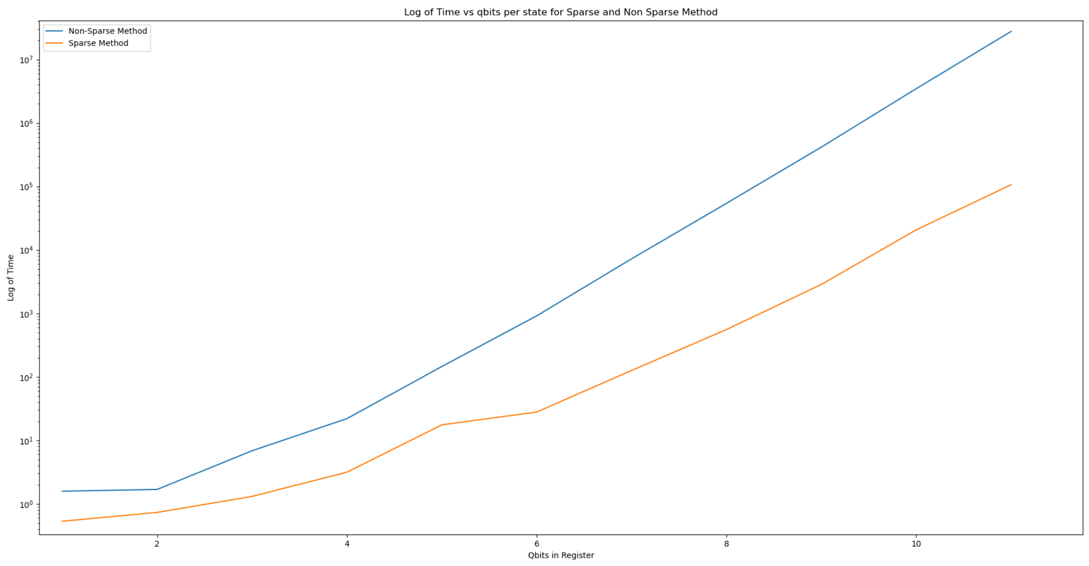


Figure 10: Plot of log(time) against number of qubits for sparse (orange) and non-sparse (blue) matrix implementation to show the speed-up produced by using sparse matrices.

running Grover's algorithm. This utilizes the `Grovers` method which tracks the time taken for the program to run. We plot the efficiency of the algorithm in Figure 10 showing the immense speedup achieved by the sparse matrix implementation compared to the regular program. The plot shows logarithmic time to better depict the difference between the two implementations.

Error Handling We also include an `Errors` class to raise any errors custom to this program, such as incompatible matrix sizes or invalid input. Because our simulator utilizes a custom datatype in the form of our register object, we decided to handle errors specific to our solution in a personalised manner. Therefore, we anticipated possible errors that could occur which are not native to `Python`: `inputErrors` and `matrixErrors`. The first is intended to handle any unexpected input to the various parts of our program, most commonly when the user is first asked for a target state and qubit number. The second, is mostly intended to constrict the data types for the matrices used in the custom matrix-handling routines so as not to break them in ways we cannot anticipate. To illustrate with an example, we might require a square matrix for a matrix inversion or require the datatype to be a `numpy` array for a Kronecker product. This last custom error was particularly useful to ensure robustness while using our `sparseMatrices` class; because of the way we chose to efficiently represent sparse matrices, we often require the solution to produce output which can be handled by said class.

Overall, these customs errors allowed many of the custom routines we designed to handle our custom datatypes to behave in a controlled manner and to account for when that was not the case. However, because of the time constraints of the project, the current error handling we support is more informative in nature. Therefore, with more time, we would have liked to extend this error handling functionality through the `try-except` blocks native to `Python` in order to maintain executing if ever one was thrown. This would also add fault-tolerance to our program, which is one of the current topics discussed in quantum computing. At present, the user and the developers are instead told information specific to the error in the trace which helps them avoid it in subsequent runs of the simulator.

3.3 Possible Extensions

Given more time in the project, we would have liked to implement algorithms, which use similar gates to the ones we already constructed for the purposes of Grover's algorithm, since their implementation would be fairly straightforward. Simulating the Deutsch-Jozsa algorithm described in detail in section 2.2 would not be unrealistic since some of the gates we already have could be reused.

We also would have liked to fully implement the HHL algorithm extensively used in Mathematics, as we believe it offers a good way of seeing how quantum computing can solve real life problems. The speedup of this algorithm in solving linear systems of equations is exponential, which underlines the importance of using it in the field. Unfortunately, we failed to fully complete the implementation, however we would like to work on it in the future given how much work we have already put into this project overall.

We also think it would be of interest and importance to the field of quantum computing, if we could implement quantum error correction into our code. As explored earlier, this could be done using Steane's code for Grover's algorithm, since Steane's error correction is ideal for the nature of Grover's algorithm in particular. This would however require an extensive amount of time as this algorithm utilizes very specific gates we have not yet implemented. Generalizing quantum error correction algorithms to an arbitrary number of qubits would also be a challenge, however definitely worth doing, since it would give us an even more realistic simulator.

4 Future Research and Applications

The potential of quantum computing and quantum computers is unprecedented. Although the construction of a fully working, commercial quantum computers is perhaps decades in the future, potential applications have already emerged and there is a boom in research, as well as start-ups who aim to pioneer the technology.

Currently, quantum computers only have the capability to utilise up to a few qubits at a time, often due to their inability to handle external noise on calculations or more specifically quantum decoherence. Only once thousands of qubits are employed can quantum computing truly take off, and all the proposed applications for the technology can be realised. Noise from the environment can come in many different forms such as temperature fluctuations, mechanical vibrations or even fluctuations in the electromagnetic fields. There are ways of reducing the effects of noise on qubit calculations. One such method described by M Otten and S K Gray [47] repeats a quantum evolution with different noise characteristics and then extrapolates what the noise-free evolution of the state would look like. The method can be implemented in sequence or in parallel, meaning through many separate systems or the same system, and the only requirement for the method is that the imposed noise is well characterised. Another way to suppress noise is through new and improved hardware designs, for example from further research on the confinement on qubits. One promising design for the production of qubits is via diamond. This is due to diamonds optical properties and how its optical centres offer access to isolated quantum systems at workable temperatures, meaning the system does not need to be cooled [48]. That being said, diamond poses its own problems of being a tough material to mould and work with, however methods for the manipulation of diamond are constantly being devised [49].

Once these methods to reduce the noise on qubit information is limited further than currently available, quantum computers can slot into their theorised positions to revolutionise a vast range of sectors and industries. New breakthroughs in the modelling of quantum systems will take place, as quantum computers will allow experts to understand properties of these materials and systems further. This knowledge in turn will allow for new devices to be made from said materials, with increased efficiencies and more complex structures [50]. There is also a vast amount that quantum computing can do in the process of clinical research into new drugs, in medicine [51]. Quantum MRI machines and predictive algorithms using quantum computers for disease diagnosis could mean a revolutionary change in the way medicine is conducted and patients are treated. Sectors such as finance and security could also be completely different after fully functioning and commercially available quantum computers are implemented. Complex and computationally demanding algorithms could be used to model the risk of financial investments, while new security systems would need to be put in place due to the new amounts of computational power available to people with quantum computing algorithms.

The future of medicine can largely be pinned on machine learning and using computers to diagnose disease to a higher accuracy than humans can. This involves having large amounts of variables and data with little, or ideally no, human input. Going beyond initial diagnosis, patients response to certain treatments could be monitored and a change in treatment can be conducted with the assistance of algorithms. This would mean a constant correction on treatment which is computationally demanding due to the amount of data that would be needed for the treatments to be accurate and useful to the patient. AI can be used to process highly detailed quantum MRI images [52] which would have the ability to find abnormalities in the patient better than a human doctor.

Cancer treatment could be affected a major way as new software to find the ideal treatment for cancer patients can devised based on a personalised data set with a higher level of detail than is used currently. Radiation plans can be simulated at high speeds to find the optimum plan which would mean the health of the patient would be in the least compromised state as less of their healthy cells would be killed to the treatment. Simulations of complex systems such as the brain are also proposed as currently computers struggle to map the neural pathways of the brain due to its vast size [53], however with the power of quantum computing these systems are more manageable and can be modelled with relative ease. Systems such as the brain can only be modelled using quantum mechanics and can be treated as a quantum system. C Zalka showed in his 1998 paper on simulating quantum systems using a quantum computer [54] and that quantum computers can obtain information on the absolute value of the wave function which in turn can be used for simulating chemical processes and overarching biological ones too.

Finance is another major industry where quantum computing has the potential to completely revolutionise the ordinary practises involved. From fraud detection to predicting stock prices, quantum computing can provide a major advantage to those that utilise it, due to its raw computational power. This is largely the reason why quantum computing has received so much financial backing over the last decade by not only big tech companies but also big banks such as JP Morgan Chase[55]. In finance arbitrage is the idea of buying and selling an asset to make profit due to price differences in different markets. A simple example of this is cross-currency arbitrage, where conversion of pounds to euros, euros to dollars and back to pounds can lead to a total profit in the process. When applied to large assets, the profits can be vast and so the quantum optimisation or quantum annealing, the concept of finding the global minimum of an objective function using quantum fluctuations, can be used to identify what the most profitable cycle is given the variables inputted[56]. An experiment was conducted using a D-Wave 2X machine, a commercially available quantum annealer for small scale operations, where a five asset sample was inputted to a function and the results were then compared against a classical solver[57]. Both avenues led to the same solution, meaning that the quantum method proved that it can fit into the exsisting frame work of financial practices.

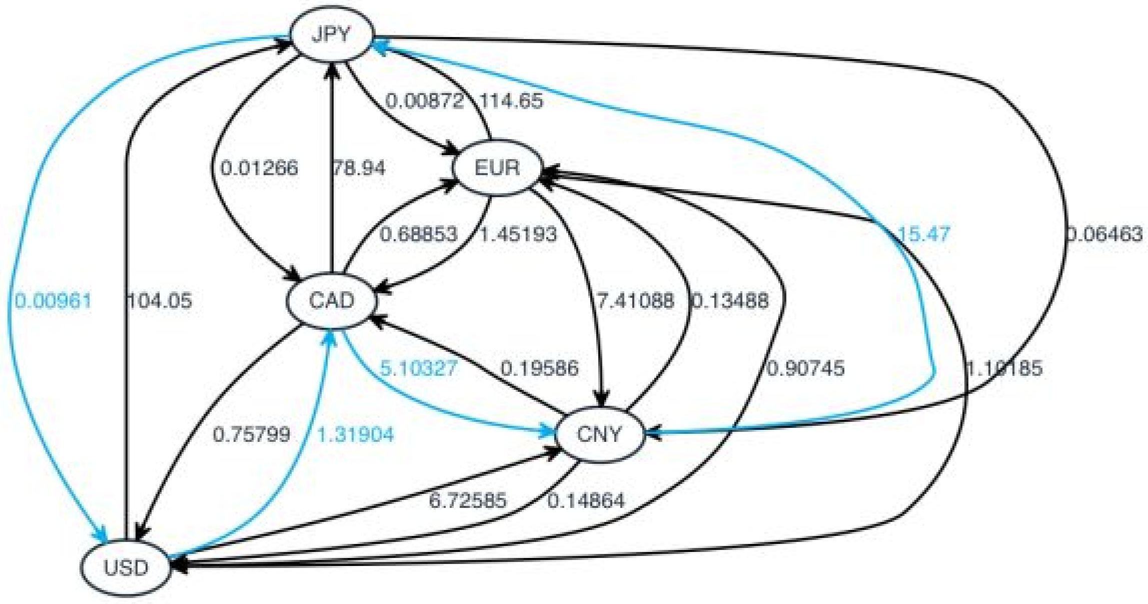


Figure 11: Most profitable arbitrage opportunity marked in blue with a five asset sample, found by both the classical solver and the quantum annealer [57]

Another important part of finance is in credit scoring and determining whether a candidate is likely to pay back the loan they are applying for. This problem can be resolved using machine learning, where the data of past applicants can be input into an algorithm, to predict the eligibility of new applicants. Using various variables with the data available to loan brokers, seemingly unrelated variables can be tested to see if this has any impact on the applicants ability to pay back their loans. This becomes difficult when there is a large amount of uncorrelated data or when there are missing chunks to the data. Due to the vast number of unknowns that need to be considered, this process can become very computationally demanding. This is where the assistance of a quantum annealer can be used where a team has done exactly that, by using a quadratic unconstrained binary optimisation(QUBO) model where features that are independent and influential are picked out from the data set[58]. Tests were conducted using German Credit Data and the QUBO model yielded results similar to standard classical packages used today in terms of accuracy, on a smaller feature subset, meaning that the model was able to find the most influential features within the data that affected the credit scoring.

Another future application of quantum computers is in chemistry. The most advanced supercomputers cannot accurately describe a chemical process. This is because classical computers find it harder to describe a system with lots of electrons. However, with the use of quantum computers, modelling chemical reactions can be done effectively. This is down to the fact that quantum computers can represent the electrons present in chemical reactions as they are, a superposition, whereas a classical computer on the other hand would represent the electrons as a binary number [59]. Therefore the quantum computer can simulate chemical reactions with a physically realistic number of qubits compared to bits. One of the main chemical reactions that scientists are keen to simulate is the Haber Bosch process. This is where ammonia is produced from atmospheric nitrogen under high temperatures and pressures. This process is extremely inefficient and scientists are trying to find a better way to produce ammonia. Its important because a significant amount of ammonia based fertilizer is used on farmland. Quantum computers can be utilised at multiple different stages of this process, one being using quantum computers to simulate the Haber Bosch process and allow chemists to see where they can make it more efficient. Another way is a substitute process of producing ammonia which uses bacteria. Bacteria use an enzyme called nitrogenase to produce ammonia at ambient conditions [59]. The mechanism for this enzyme at the moment remains elusive to scientists however Reiher et al [60] have predicted that a 100 qubit quantum computer would be able to allow scientists to understand the nitrogenase mechanism and form a new, more efficient method of producing ammonia.

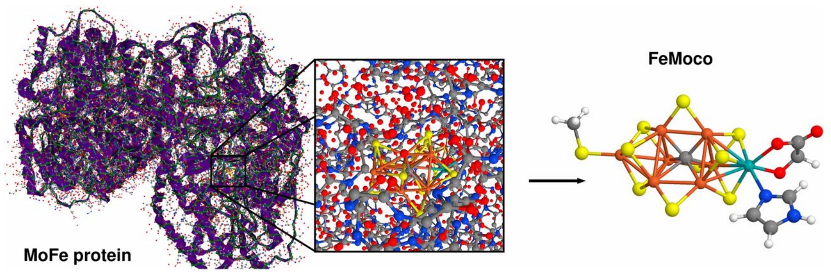


Figure 12: Nitrogenase enzyme with the iron cofactor highlighted by FeMoco . This cofactor is specifically what classical computers struggle to model [60]

There are also worries with the development of quantum computers as they will posses the computing power to solve most cryptosystems. At the moment classical cryptographs such as RSA and Diffie Hellman work on the basis of difficult calculations such as integer factorization and discrete log problems. What makes these so secure is that it takes classical computers almost billions of years to solve. However these problems would easily be solved with quantum computers using an algorithm such as Shor's algorithm. This opens the branch of post quantum cryptography. The National Institute of Standards and Technology (NIST) has been leading the work into developing new algorithms that will protect against quantum computers [61]. One such algorithm type known as lattice cryptosystems makes uses of multi-dimensional arrays. In these multi-dimensioanl arrays information is hidden and is virtually impossible to access without the key. The key, in a sense, is the route through the array that needs to be taken to access the information. As quantum computers are becoming closer to reality, the traditional cryptosystems are becoming obsolete.

Most of these applications described above have a large enough theoretical impact on their respective industries, that continued investment into quantum computers is almost certain, and that it is only a matter of time before quantum computers are made commercially available and completely functional for more complex processes.

5 Conclusion

Quantum computing is an exciting and increasingly active field of research. Using three basic counter-intuitive ingredients inherent to quantum mechanics (superposition, interference and entanglement) it has the potential to enhance computing power and speed, with respect to classical computing.

Depending on the subtleties of the respective quantum algorithms, the extent of the speedup can vary. For example, Grover's algorithm and more generally the family of quantum algorithms that use amplitude amplification, exploit the extra information contained within the coefficients (phase shifts) of each state. Because the concept of amplitudes has no analog in classical probability, these quantum algorithms provide a significant speedup with respect to classical alternatives.

The wide range of both probabilistic and deterministic quantum algorithms has sparked interest in many fields. Future potential applications are quickly starting to emerge in areas as diverse as finance, cryptography, drug discovery. In particular, with regards to medical research, quantum computing can open new avenues in predicting the best treatment for patients via an increased capacity to simulate the biological environment of the patient. Radiation therapy's efficacy can also be greatly enhanced in the same vain.

Although quantum computing has the potential to revolutionize computation, the construction of a useful quantum computer is perhaps still decades away. This group believes the success of significantly better quantum error correction codes in the next years will be crucial, in order to promote a widespread use of quantum algorithms.

The Grover's algorithm implementation on a classical computer designed as a part of this project was largely successful, as the group managed to make the code fast, efficient and general. The implementation of sparse matrices lead to a significant speed up of the program, therefore the developed code can be run for a large amount of qubits (at least 10) within seconds. Additionally, the decomposition of the quantum logic gates utilized by Grover's algorithm has enabled us to create an accurate simulator of a quantum algorithm on a classical machine without the need to explicitly use for example slow linear search methods. The design of the code allows for an easy extension to other algorithms, such as Deutsch-Josza or the HHL, which could be attempted in the future.

In December 2018, President Trump signed to fund the National Quantum Initiative Act (NQI). The law authorizes $1.2 billion to be invested in quantum information science over five years. With enormous governmental investments in quantum technologies as well as increasing investment in quantum computing companies all over the world, the future of this field is very bright. The future is quantum.

References

[1] David J Griffiths and Darrell F Schroeter. *Introduction to quantum mechanics.* Cambridge University Press, 2018.

[2] Louis De Broglie. "Recherches sur la théorie des quanta". PhD thesis. Migration-université en cours d'affectation, 1924.

[3] Erwin Schrödinger. "An undulatory theory of the mechanics of atoms and molecules". In: *Physical review* 28.6 (1926), p. 1049.

[4] Raghava Garipelly, P Madhu Kiran, and A Santhosh Kumar. "A review on reversible logic gates and their implementation". In: *International Journal of Emerging Technology and Advanced Engineering* 3.3 (2013), pp. 417–423.

[5] *Digital Logic Gates.* `https://www.electronics-tutorials.ws/logic/logic_1.html`. Accessed: 09 Mar 2020.

[6] *I will design digital logic gates, truth tables, k maps.* `https://www.fiverr.com/tejaswalvekar/design-digital-logic-gates-truth-tables-k-maps`. Accessed: 09 Mar 2020.

[7] Charles H Bennett. "The thermodynamics of computation—a review". In: *International Journal of Theoretical Physics* 21.12 (1982), pp. 905–940.

[8] James Powell. "The Quantum Limit to Moore's Law". In: *Proceedings of the IEEE* 96 (Sept. 2008), pp. 1247–1248. DOI: `10.1109/JPROC.2008.925411`.

[9] Alexis De Vos. "Reversible computer hardware". In: *electronic notes in theoretical computer science* 253.6 (2010), pp. 17–22.

[10] Arjun Berera. *Quantum Physics / Principles of Quantum Mechanics S2.* 2019, pp. 57–72.

[11] Eduard Prugovecky. *Quantum Mechanics in Hilbert Space.* 1981, pp. 1–8.

[12] Hitoshi Murayama. "221A Lecture Notes: Notes on Tensor Product". In: *Quantum Mechanics I* (2006).

[13] Remy Mossery and Rossen Dandoloff. "Geometry of Entangled States, Bloch Spheres and Hopf Fibrations". In: *Journal of Physics A: Mathematical and General* 34.47 (2001).

[14] *Introduction to quantum computing: Bloch sphere.* `http://akyrillidis.github.io/notes/quant_post_7`. Accessed: 15 Mar. 2020.

[15] Emma Strubell. "An Introduction to Quantum Algorithms". In: *COS498 - Chawathe* (2011).

[16] *Quantum Logic Gate.* `https://en.wikipedia.org/wiki/Quantum_logic_gate`. Accessed: 15 Mar. 2020.

[17] Lov K Grover. *A fast quantum mechanical algorithm for database search.* 1996, pp. 212–219.

[18] Rubens Viana Ramos, Paulo Benicio de Sousa, and David Sena Oliveira. "Solving mathematical problems with quantum search algorithm". In: *arXiv preprint quant-ph/0605003* (2006).

[19] Emma Strubell. "An introduction to quantum algorithms". In: *COS498 Chawathe Spring* 13 (2011), p. 19.

[20] N David Mermin. *Quantum computer science: an introduction.* Cambridge University Press, 2007.

[21] Dan C Marinescu. *Classical and quantum information.* Academic Press, 2011.

[22] Lindemann F. “Über die Zahl π”. In: *Mathematische Annalen* 20 (1882), pp. 213–225.

[23] Goong Chen et al. “Grover’s algorithm for multiobject search in quantum computing”. In: *Directions in Quantum Optics*. Springer, 2001, pp. 165–175.

[24] Charles H Bennett et al. “Strengths and weaknesses of quantum computing”. In: *SIAM journal on Computing* 26.5 (1997), pp. 1510–1523.

[25] Panagiotis Botsinis et al. “Quantum error correction protects quantum search algorithms against decoherence”. In: *Scientific reports* 6 (2016), p. 38095.

[26] John Preskill. “Fault-tolerant quantum computers”. In: (1998).

[27] Matthew D Reed et al. “Realization of three-qubit quantum error correction with superconducting circuits”. In: *Nature* 482.7385 (2012), pp. 382–385.

[28] Konstantinos Prousalis, Agis Iliadis, and Nikos Konofaos. “Quantum recovery protocols for stabilizer codes: Deterministic Monte-Carlo simulation”. In: *AIP Advances* 8.6 (2018), p. 065008.

[29] David Deutsch and Richard Jozsa. “Rapid solution of problems by quantum computation”. In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 439.1907 (1992), pp. 553–558.

[30] Richard Jozsa and DAMTP Cambridge. “Quantum Information and Computation Lecture notes”. In: ().

[31] *Deutsch-Josza Algorithm*. `https://qiskit.org/textbook/ch-algorithms/deutsch-josza.html`. Accessed: 12 Mar 2020.

[32] Reinhold Blumel. *Foundations of Quantum Mechanics: From Photons to Quantum Computers*. Jones & Bartlett Learning, 2010.

[33] Koji Nagata, Tadao Nakamura, and Ahmed Farouk. “Quantum cryptography based on the Deutsch-Jozsa algorithm”. In: *International Journal of Theoretical Physics* 56.9 (2017), pp. 2887–2897.

[34] D Ristè et al. “Demonstration of quantum advantage in machine learning. npj Quantum Inf”. In: (2017).

[35] José Manuel Bravo. “Calculating Hamming Distance with the IBM Q Experience”. In: *transformation* 11 (2018), p. 0.

[36] Colin J Worby et al. “The distribution of pairwise genetic distances: a tool for investigating disease transmission”. In: *Genetics* 198.4 (2014), pp. 1395–1404.

[37] Henry Semenenko and Tim Byrnes. “Implementing the Deutsch-Jozsa algorithm with macroscopic ensembles”. In: *Physical Review A* 93.5 (2016), p. 052302.

[38] Dan Stahlke. “Quantum interference as a resource for quantum speedup”. In: *Physical Review A* 90.2 (2014), p. 022302.

[39] Peter W Shor. “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer”. In: *SIAM review* 41.2 (1999), pp. 303–332.

[40] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. “Quantum algorithm for linear systems of equations”. In: *Physical review letters* 103.15 (2009), p. 150502.

[41] Marc Peter Deisenroth, A Aldo Faisal, and Cheng Soon Ong. *Mathematics for machine learning*. Cambridge University Press Cambridge, 2019.

[42] Roman Orus, Samuel Mugel, and Enrique Lizaso. “Quantum computing for finance: overview and prospects”. In: *Reviews in Physics* (2019), p. 100028.

[43] Michael A Nielsen and Isaac Chuang. *Quantum computation and quantum information.* 2002.

[44] Ashok Chatterjee. “Introduction to quantum computation”. In: *arXiv preprint quant-ph/0312111* (2003).

[45] Hartmut Häffner, Christian F Roos, and Rainer Blatt. “Quantum computing with trapped ions”. In: *Physics reports* 469.4 (2008), pp. 155–203.

[46] Daniel Loss and David P DiVincenzo. “Quantum computation with quantum dots”. In: *Physical Review A* 57.1 (1998), p. 120.

[47] Matthew Otten and Stephen K. Gray. “Recovering noise-free quantum observables”. In: *Phys. Rev. A* 99 (1 Jan. 2019), p. 012338. DOI: 10.1103/PhysRevA.99.012338. URL: https://link.aps.org/doi/10.1103/PhysRevA.99.012338.

[48] Steven Prawer and Andrew D Greentree. “Diamond for quantum computing”. In: *Science* 320.5883 (2008), pp. 1601–1602.

[49] Yossi Paltiel. *Room temperature nano quantum engineering.* 2009, pp. 5–9.

[50] Katherine L Brown, William J Munro, and Vivien M Kendon. “Using quantum computers for quantum simulation”. In: *Entropy* 12.11 (2010), pp. 2268–2307.

[51] Dmitry Solenov, Jay Brieler, and Jeffrey F Scherrer. “The potential of quantum computing and machine learning to advance clinical research and change the practice of medicine”. In: *Missouri medicine* 115.5 (2018), p. 463.

[52] Steven E Dilsizian and Eliot L Siegel. “Artificial intelligence in medicine and cardiac imaging: harnessing big data and advanced computing to provide personalized medical diagnosis and treatment”. In: *Current cardiology reports* 16.1 (2014), p. 441.

[53] John F Barry et al. “Optical magnetic detection of single-neuron action potentials using quantum defects in diamond”. In: *Proceedings of the National Academy of Sciences* 113.49 (2016), pp. 14133–14138.

[54] Christof Zalka. “Simulating quantum systems on a quantum computer”. In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454.1969 (1998), pp. 313–322.

[55] *JP Morgan Chase Prepares for Quantum Computing.* https://www.jpmorganchase.com/corporate/news/stories/jpmc-prepares-fintech-quantum-leap.htm. Accessed: 12 Mar 2020.

[56] Roman Orus, Samuel Mugel, and Enrique Lizaso. “Quantum computing for finance: overview and prospects”. In: *Reviews in Physics* (2019), p. 100028.

[57] G Rosenberg. “Finding optimal arbitrage opportunities using a quantum annealer”. In: *1QB Information Technologies White Paper* (2016), pp. 1–7.

[58] Andrew Milne, Maxwell Rounds, and Phil Goddard. “Optimal feature selection in credit scoring and classification using a quantum annealer”. In: (2017).

[59] Katherine Bourzac. “Chemistry is quantum computing’s killer app”. In: *Chemical Engineering News* (Oct. 2017). URL: https://cen.acs.org/articles/95/i43/Chemistry-quantum-computings-killer-app.html.

[60] Markus Reiher et al. “Elucidating reaction mechanisms on quantum computers”. In: *Proceedings of the National Academy of Sciences* 114.29 (2017), pp. 7555–7560.

[61] *Post-Quantum Cryptography.* `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography`. Accessed: 14 Mar. 2020.

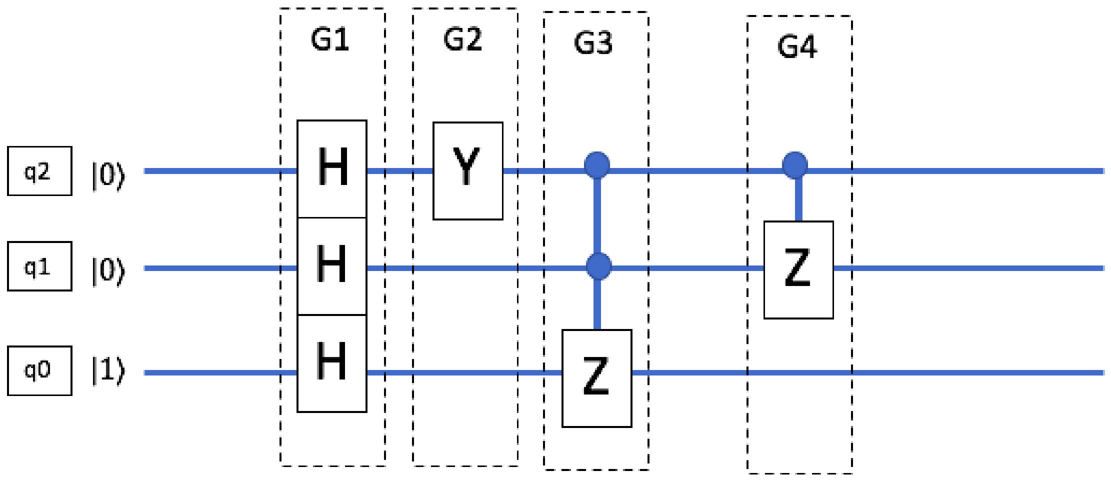


Figure 13: An example of a circuit that can be constructed using our Simulator.

A Simulator at Use

This section is aimed to aid the reader in understanding our design better, while also providing an idea of how a circuit may be built using our simulator.

The following code simulates the circuit in Figure 13. Note that the register is prepared at state |001⟩. Following that, the parallel gates (G1,G2,G3,G4) are constructed sequentially and are stored in a list. Through a loop they are then applied to the register in the correct order. The register is finally measured and the result of the measurement is printed on the screen.

```
import operations as op
import register as re
import quantum_states as qs

#the denary representation of state |001>
Initial_State = 1
#the number of qubits in the register
No_qubits = 3

#Create register and prepare it at state |001>
R = re.Register(qs.State((Initial_State, No_qubits)))

G = []
G.append(op.constructGate('HHH')) #G1
G.append(op.constructGate('YII')) #G2
G.append(op.constructGate('3Z'))  #G3
G.append(op.constructGate('2ZI')) #G4

#Apply gates
```

```
for Gate in G:
    R.applyGate(Gate)

print(R.measure_collapse())
```