PHYS379

Group Project

Lower Bound of Time for the NMR Implementation of Shor's Algorithm for Factorising 15

Wednesday 4th April, 2018

Authors:
Vlad Cărare
Alejandro Cros
Niall Mulholland
John Taylor

Supervisor:
Dr. Edward McCann

Abstract

In this paper, we describe how we simulated a quantum computer which performs Shor's algorithm on a 7-qubit system to factorise 15. Using this simulation, we calculated the lower bound of time a theoretical quantum computer could perform this calculation under the same energy conditions as current working quantum computers. We found that a nuclear magnetic resonance quantum computer would take 1.59 ± 0.04s to perform our simulated computation, whereas the lower bound of time under the same energy conditions would be 2.446 ± 0.004ms. Moreover, we found that this lower bound is inversely proportional to the energy variance of our qubit states. We concluded that the fastest quantum computers of the future would be those that allow large energy fluctuations between qubit states and can implement qubit transitions which best correlate with the time-optimal evolutions described in this paper.

Contents

1 Introduction **1**

2 Shor's Factorisation Algorithm **2**
2.1 Shor's Method for Factorising Large Numbers 2
2.2 Basics of Quantum Computing 4
2.3 Simulating Quantum Algorithms on Classical Systems 6
2.4 Analysis of the Quantum Mechanical Behaviour of Our Simulation 7
2.5 Simulating Shor's Factorisation Algorithm 9

3 Experimental Duration of Shor's Algorithm **13**
3.1 Experimental Implementation of Quantum Gates 13
3.2 Experimental Duration of Shor's Algorithm in NMR 16

4 Time Optimisation of Shor's Algorithm **18**
4.1 An Introduction to Time-Optimal Quantum Computation: the Quantum Brachistochrone Equation 18
4.2 Energetic Constraints of the Time-Optimal Hamiltonian: the Isotropic Constraint 19
4.3 Time-Optimal Translations Under Experimental NMR Constraints 20
4.4 Analysis of the Lower Bound of Time Taken to Physically Perform Shor's Algorithm 22

5 Conclusions **24**
5.1 Further Directions 25

6 Appendix **26**
6.1 Polynomial of Degree 2 (ax^2) Fit for the Data in Figure 2 26
6.2 Expanding 2×2 and 4×4 Matrices to N×N Matrices 26
6.3 Legend 27
6.4 Generating Permutation Matrices to Achieve the Function Gates for Shor's Algorithm 28
6.5 Spin Matrices 28
6.6 Changing Reference Frames for Quantum States 29
6.7 Structural Properties of the Quantum Computer Molecule 30

1 Introduction

The twentieth century provided two important milestones for science and technology: the quantum revolution early in the century and the birth of information technology with Turing in 1936 [1]. Together these spawned numerous technological advancements that have completely transformed our society. As a result, many began to speculate if the two fields could be merged by building more effective computers that utilised the strange properties of quantum mechanics. *Quantum computers* have the potential to perform tasks that are unimaginable and, in some cases, impossible on classical systems [2]. In 1995, Peter Shor proposed an ingenious quantum algorithm that allows one to deduce the prime factors of a given integer [3]. Shor's algorithm represents a step towards the solution of the P versus NP millennial problem as it is a polynomial-time algorithm [4]. Moreover, if a large-scale quantum computer could be built that successfully performed this algorithm, it would be capable of breaking modern RSA public encryption systems [5]. The RSA system was developed in the 1970's and is still the most widely used public encryption system in the world, thus the field of quantum computing has great potential.

Although large-scale quantum computers that are capable of performing Shor's algorithm are yet to be realised, small-scale versions have successfully been built [6]-[9]. In this paper, we set out to simulate such a quantum computer and investigate how it goes about factorising the number 15. We then consider how quickly our simulated quantum computer would perform Shor's algorithm using nuclear magnetic resonance (NMR) techniques [10]. Based on these findings, we ask the following question: What is the quickest possible time a theoretical quantum computer could perform the same computation bound by the laws of quantum mechanics and under the same energy constraints of an NMR quantum computer? We find that these computers perform Shor's Algorithm of the order 650 times slower than their physical limit. By answering this question we hope to portray the extraordinary uncharted potential quantum computing provides for this century and beyond.

Firstly, Section 2 uses the context of Shor's factorisation algorithm to motivate the need for quantum algorithms. Further, we offer a brief introduction into quantum computation. We introduce the idea of a qubit and its representation, and we present the most important operations that can be performed on a system of qubits. These operations are called gates, in analogy with the classical logic gates. Section 2 then moves on to describe how we simulated a quantum computer using Java. Moreover, we offer a quantitative analysis by comparing our results with theoretical predictions of quantum mechanics. Next, we demonstrate how we coded Shor's algorithm for factorising 15. We give a visual representation of the circuit, a list of the gates that act on our qubits and a flowchart that shows the process step by step. Finally, we finish this section by stating the result of factorising 15.

In Section 3 we investigate how Shor's algorithm is implemented experimentally using NMR techniques. Moreover, we derive a set of equations which we used to calculate the time such a quantum computer would take to implement the logic gates of our simulation. Using experimental data we estimate how long our simulated quantum computer would take to perform Shor's algorithm and compare this time to the results of an NMR quantum computer of a similar form [6].

Section 4 is devoted to the study of time-optimal quantum evolutions of pure states. In this section, a theoretical model of time-optimal qubit translations is developed under the same energetic conditions as the experimental NMR quantum computer of Section 3 by using the *quantum brachistochrone equation*. With this theoretical model, a lower boundary of the time it takes for Shor's algorithm to factorise 15 is set. This boundary was found to be of the order of 650 times faster than the experimental NMR results.

We conclude the paper in Section 5 with a summary of the main findings of our investigations, followed by a discussion on future directions for time-optimal quantum computing.

2 Shor's Factorisation Algorithm

This section will cover how a quantum computer uses Shor's algorithm to factorise the number 15 and will set out how we simulated such a quantum computer. This simulation will be important for the rest of the paper as it is used to calculate the experimental and time-optimal duration of Shor's algorithm.

2.1 Shor's Method for Factorising Large Numbers

It is trivial for a classical computer to multiply two large prime numbers together. However, it is difficult to do the opposite: to find the prime factors of an extremely large number. Common forms of cryptography such as the RSA algorithm [5] use arbitrarily large prime factors as keys to encrypt messages. Without knowing any of these prime factors, classical computers can only decrypt these messages via a process of trial and error. For numbers that are hundreds of digits long, even the most powerful computers today would take hundreds of years to factorise using this brute-force method [11].

Quantum computers provide a more sophisticated way of decoding such messages. For this, we need to review number theory, in particular, modular arithmetic. Modular arithmetic is a form of cyclical counting. An example would be counting the hours on the 12-hour clock as seen in Figure 1. This cyclical counting extends to the mathematical operations used in Shor's algorithm.

Let us consider what happens to the exponents of a random integer a, for example 7, in a random modulo N, for example 15:

$$\begin{aligned} 7^0 &\mod 15 = 1 \qquad & 7^4 &\mod 15 = 1 \\ 7^1 &\mod 15 = 7 \qquad & 7^5 &\mod 15 = 7 \\ 7^2 &\mod 15 = 4 \qquad & 7^6 &\mod 15 = 4 \\ 7^3 &\mod 15 = 13 \qquad & 7^7 &\mod 15 = 13. \end{aligned} \tag{1}$$

It is possible to observe that although the powers keep increasing, the modular versions of the sequence cycles periodically (1,7,4,13,1,7,4,13,1...), repeating the same pattern with the first number always being 1. It is likewise possible to say that provided a and N are relatively prime, meaning they share no common factors, this property will always hold true. We call the length of this repeating sequence the period p. Given p, we can make the following statement:

$$a^p \equiv 1 \mod N. \tag{2}$$

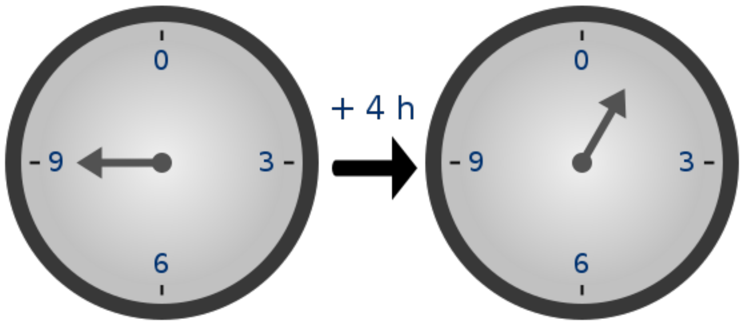


Figure 1: Figure showing how modular counting works [12]. Once 4 hours have passed after the 9:00 mark, the clock goes to 1:00, not 13:00 as expected when counting normally. The hour number here 'wraps around' once it reaches 12 so we say this is arithmetic modulo 12. As 12:00 can also be written as 0:00, we can say 12 is congruent to 0 modulo 12 ($12 \equiv 0 \mod 12$).

We can now utilise these mathematical techniques to effectively factorise large prime numbers. Consider the following equation:

$$N = b \times q \tag{3}$$

where b and q are prime numbers. Shor's algorithm uses 5 main steps to calculate the values b and q for a given value of N:

1. Pick any number a smaller than N, and check that N and a are relatively prime by computing the greatest common divisor of a and N i.e. $\gcd(a, N) = 1$ if they are relatively prime. A quick way to do this is by using the Euclidean algorithm [13].

2. Compute the period p of a mod N. Test that p is even and $a^{p/2} + 1$ is not congruent to $0 \mod N$. If either fail, return to step 1 with a new value of "a".

3. Rearrange the equation algebraically:

$$\begin{aligned} a^p &\equiv 1 \mod N \\ \therefore a^p - 1 &\equiv 0 \mod N \\ \therefore a^p - 1 &= kN \end{aligned} \tag{4}$$

   where k is an integer value. As p is even we can therefore conclude:

$$(a^{p/2} - 1)(a^{p/2} + 1) = k \times b \times q. \tag{5}$$

4. Solve b and q. Due to the periodic properties of modular arithmetic these values are given by the following equations [3]:

$$\begin{aligned} b &= \gcd(a^{p/2} - 1, N) \\ q &= \gcd(a^{p/2} + 1, N). \end{aligned} \tag{6}$$

5. Verify $\frac{N}{b}$ and $\frac{N}{q}$ are integers.

Classical computers can do all steps apart from part 2 relatively quickly. Fortunately period finding for large numbers is one process a quantum computer is capable of doing quicker than a classical computer. In the next section, we will discuss the basics of how a quantum computer works and thus how it is able to effectively decipher such a period.

2.2 Basics of Quantum Computing

In 1995 the *qubit* was officially defined by Ben Schumacher as the classical analogue to a bit of data [14]. A general pure qubit state consists of a linear superposition of two base states:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \begin{pmatrix}\alpha\\ \beta\end{pmatrix}, \tag{7}$$

where α and β are the normalised complex coefficients. Their moduli squared represent the probability of measuring that base state, thus they satisfy: $|\alpha|^2 + |\beta|^2 = 1$. These qubits are typically modelled by a two-level spin systems or polarised photons [15]. They can exhibit a peculiar quality that is unavailable to the classical system: quantum entanglement [16]. When multiple qubits are involved, this offers advantages over classical systems due to a greater level of correlation. We refer to the collection of qubits as a *qubit register*.

As with classical computing, quantum computers also have algorithms and gates. The gates used in quantum algorithms are different than their classical counterparts although can have similar functionalities. The effect of gates on qubits can be visualised using the *Bloch sphere* [17].

Three important quantum gates are the *Pauli gates* [18], defined as:

$$\hat{X} \equiv \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}, \quad \hat{Y} \equiv \begin{bmatrix}0 & -i\\ i & 0\end{bmatrix}, \quad \hat{Z} \equiv \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}, \tag{8}$$

where the $\hat{X}$ gate is also known as the *quantum* NOT *gate*. The NOT gate performs a rotation in the Bloch sphere. This corresponds to the transformation:

$$\hat{X}(\alpha\,|0\rangle + \beta\,|1\rangle) = \alpha\,|1\rangle + \beta\,|0\rangle\,. \tag{9}$$

A restriction placed on all quantum gates is that they must be unitary: $\hat{U}^\dagger\hat{U} = \hat{\mathbb{1}}$, where $\hat{\mathbb{1}}$ represents the identity matrix [15]. This implies that all quantum gates can be reversed by another gate. This constraint is one of the reasons why it is difficult to simulate quantum algorithms on a classical machine. Classical logic gates, like the NAND gate, are inherently irreversible and do not need to fulfil the unitary condition [15].

Another example of a fundamental quantum gate is the Hadamard gate, defined as

$$\hat{H} \equiv \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}. \tag{10}$$

It performs the following operation on a qubit:

$$\hat{H}(\alpha\,|0\rangle + \beta\,|1\rangle) = \alpha\frac{|0\rangle + |1\rangle}{\sqrt{2}} + \beta\frac{|0\rangle - |1\rangle}{\sqrt{2}}. \tag{11}$$

The Hadamard gate can be built from the Pauli gates:

$$\hat{H} = \frac{1}{\sqrt{2}}(X+Z) = \frac{1}{\sqrt{2}}\left(\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix} + \begin{bmatrix}1 & 0\\0 & -1\end{bmatrix}\right) = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}. \tag{12}$$

We can also construct the square root of the NOT gate, known as the $\hat{V}$ gate [18]. This gate has no classical equivalent, but can be represented in matrix form as

$$\hat{V} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & -i\\-i & 1\end{bmatrix}. \tag{13}$$

It has the property: $\hat{V}\hat{V} = \hat{X}$. We finish the discussion of 1-qubit gates by introducing the phase gate $\hat{\phi}$. This multiplies a state $|1\rangle$ by a factor $e^{i\phi}$ but leaves a state $|0\rangle$ unchanged. In terms of transformation on the Bloch sphere, this can be seen as a rotation around the z-axis[15]. It has the form:

$$\hat{\phi} \equiv \begin{bmatrix}1 & 0\\0 & e^{i\phi}\end{bmatrix}. \tag{14}$$

We will now focus on 2-qubit controlled gates. These have the property that the operation is performed on the *target* qubit if and only if the *control* qubit is in the state $|1\rangle$. The controlled-NOT (CNOT) gate is one important example that is used in our simulation. It performs a NOT gate transformation on the target qubit if the control qubit is in the state $|1\rangle$. In contrast to 1-qubit gates, 2-qubit gates require a 4×4 matrix representation, for example:

$$\text{CNOT} \equiv \begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\\0 & 0 & 1 & 0\end{bmatrix}. \tag{15}$$

A controlled $\hat{V}$-gate is constructed in a similar manner [18].

Another 2-qubit gate we need to consider is the controlled phase gate. As in the example of the CNOT gate, this gates performs a phase rotation on the target qubit when the control qubit is in the state $|1\rangle$. It has the matrix representation:

$$\hat{\phi} = \begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 1 & 0\\0 & 0 & 0 & e^{i\phi}\end{bmatrix}. \tag{16}$$

The final gate we present is the Toffoli gate. This is a 3-qubit gate with two control and one target qubit. It performs a NOT gate on the target qubit if the two control qubits are in state $|1\rangle$, for example: $|110\rangle \to |111\rangle$. Because of this, it is known as the control-control-NOT (CCNOT) gate. It is an important reason why quantum computers can perform any task a classical system can [15].

To conclude the subsection, we put forward an important property of quantum gates: *any multi-qubit logic gate can be composed from the controlled phase gates and single qubit gates*[15]. This is a powerful result. It means we have a large amount of control over our algorithms, particularly when it comes to simulating Shor's algorithms for large systems of qubits. In Section 2.5, we use this result to break down Toffoli gates into a simple sequence of CNOT and $\hat{V}$ gates.

All the gates discussed in this section have a visual representation. A list of the ones we will use in further sections is portrayed in Appendix 6.3.

2.3 Simulating Quantum Algorithms on Classical Systems

In the following subsections, we will be building the necessary knowledge about the fundamentals of quantum computer simulations on a classical machine. We will be conducting various tests to verify how well the programs follow the postulates of quantum mechanics [16]. These subsections will culminate with an overview of the implementation of Shor's factorisation algorithm and finally use it to factorise 15.

The programming discussed in this paper was mainly done in Java with figures in Sections 3 and 4 programmed in Python. Therefore there may be sections which include Java and Python jargon, although we have tried to make it accessible to everyone with a basic programming knowledge.

Firstly, we will talk about the representation and initialisation of quantum states. We used an array of binary numbers to represent qubit states. The easiest way to explain how this was done is by giving an example: Suppose we have 3 qubits. Each qubit can have two states, which can be represented by it being in either state $|0\rangle$ or $|1\rangle$. This will result in 8 possible combinations of the 3 qubits. We assume that the qubits do not permute so that the order number of any qubit is given by counting the binary digits from the left. Thus, the 1 in $|010\rangle$ corresponds to the second qubit. The 8 combinations are: $\{|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle, |111\rangle\}$. In general, given n qubits, there are 2^n possible combinations. The secret of quantum computing relies on being able to interpret those sequences of binary numbers as decimal numbers. Thus, our 8 combinations correspond to number from 0 to 7. In general, if we had n qubits, we could write numbers up to $n - 1$.

Nothing so far was unique to quantum mechanics. What distinguishes quantum mechanics from classical systems is that given those 8 combinations, which correspond to basis states [16], we can build superpositions. That means we can add the combinations together and multiply each with a complex coefficient and we would still have a valid state.

Now, the way we code a state of 3 qubits in our program is by inputting a vector, or rather arrays in Java, of dimensions 2^3. Each position in the vector corresponds to one of the 8 eigenstates, in increasing order of the decimal number they represent. The entries represent the coefficients of the corresponding eigenstate. A general 3 qubit state can therefore be written as:

$$a\,|000\rangle + b\,|001\rangle + c\,|010\rangle + d\,|011\rangle + e\,|100\rangle + f\,|101\rangle + g\,|110\rangle + h\,|111\rangle = \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{pmatrix} \tag{17}$$

One of the most important features of a quantum system that is not present in a classical system is the stochastic element. This randomness does not arise as a result of our ignorance of the system. Instead, it is ingrained into the theory by Heisenberg's Uncertainty Principle and is, therefore, unavoidable [16]. If we then want to simulate a quantum algorithm, it is essential to achieve a *probabilistic measurement.*

Our choice was to use the preset function in Java.math library. This choice can be accessed by Math.random() [19]. This generates a random number between 0 and 1. The coefficients in the superpositions are then squared and sequentially added to a sum starting from 0 until that sum exceeds the random number. Once this happens, the program picks the basis state whose last coefficient was added and sets the total state equal to it. Thus, this simulates a wave function collapse as a result of a measurement, since the superposition disappears.

Another important ingredient for making measurements is normalising the wave function beforehand. This ensures that sum of coefficients squared is equal to 1 and that our comparison with the random number between 0 and 1 works as it should.

2.4 Analysis of the Quantum Mechanical Behaviour of Our Simulation

There are various tests which check that the set-up described in the section above behaves quantum mechanically. For example, we can initialise and measure a superposition with certain coefficients a number of times. We can then check that the number of outcomes of a given eigenstate corresponds to the modulus squared of its coefficient, as given in the postulates of quantum mechanics [16]. Such a plot is shown in Figure 2.

We performed 4 sets of measurements: {10, 100, 1000, 10000, 100000}. Next, we will qualitatively analyse Figure 2. We can see that for 10 measurements, the data does not necessarily obey a certain pattern, but as we increase the number of measurements a pattern does become evident.

In order to quantify the results we used a polynomial fit of degree 2: ax^2 for our plots. A full explanation as to why a polynomial of degree 2 is chosen to fit the data is given in Appendix 6.1. In short, it is because of how probabilities scale as modulus square of coefficients [16]. The analysis in QTiPlot then gave us the value of *adjusted R squared*, which is the best indicator of how well the data fits the given function, in our case the polynomial of degree 2 [20]. Its values run from 0 to 1 with a value closer to 1 corresponding to the best fit. *Adjusted R squared* ran from 0.83663 to 0.99997 as the number of measurements increased from 10 to 100000. All the results are presented in the Table 2 of Appendix 6.1.

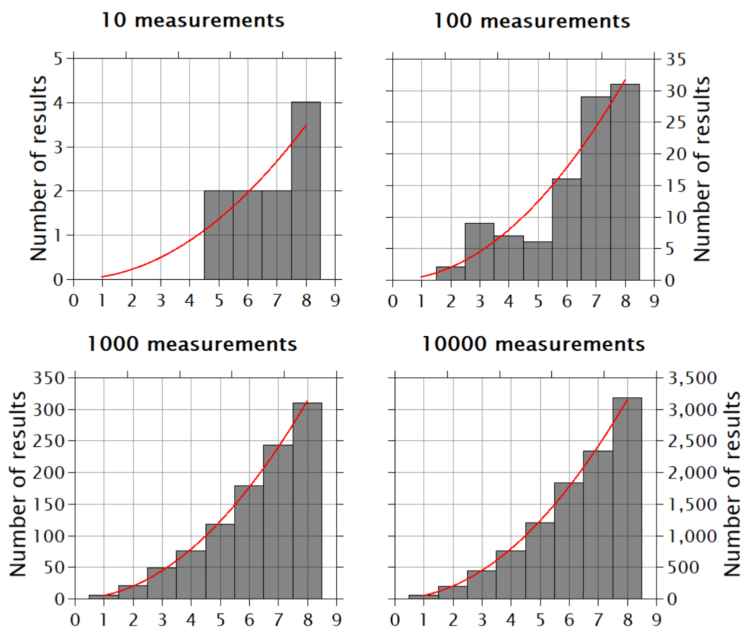


Figure 2: Plot testing that our computer program abides by the postulates of quantum mechanics. 4 sets of measurements ({10, 100, 1000, 10000} measurements) have been done on a superposition of 8 basis states, as in equation (17), with coefficients $\{1, 2, 3, 4, 5, 6, 7, 8\}$. The red curve corresponds to the theoretical behaviour, as explained in the main body of the text. Each measurement collapses the superposition to one of the 8 basis states. The vertical bar gives the number of times the superposition collapsed to the basis state given on the x-axis. No values will be recorded outside the x-range 1 to 8. Each of these digits represents, in decimal notation, one of the 8 basis states for a 3 qubits system. The basis states can be regarded as 3 digit binary numbers. For example, number 4 corresponds to the state $|100\rangle$. This situation was described in the second paragraph of Section 2.3.

Thus we have shown that our program simulates initialisation and measurement that abide by the postulates of quantum mechanics.

To best illustrate the abnormal behaviour of quantum mechanics, we chose the following example. We have already talked about the Hadamard gate in equation (10). Recall that $\hat{H}\hat{H} = \hat{\mathbb{1}}$. This implies that if a qubit experiences two Hadamard gates one after the other, it should return to its original state. Pictorially[1], this can be represented as in Figure 3(a). Our program does indeed always return $|00\rangle$ as a result for this circuit.

The example in part (a) of Figure 3 stands in stark contrast with the situation (b) of Figure 3. Superficially, we would expect the first qubit in Figure 3 (b) to have the same final state as the first qubit in Figure 3 (a). That is because although there is a new control gate in between the Hadamard gates, this does not affect the state of the controlling qubit, qubit 1. Nevertheless, our program gives as result one of the states $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ with *equal* probability. This is different from the previous example of Figure 3 (a), which always gave the state $|00\rangle$. This means the second Hadamard gate fails to put the first qubit back

[1]All circuit gate diagrams were made using the 'Quirk' program: `http://algassert.com/quirk`

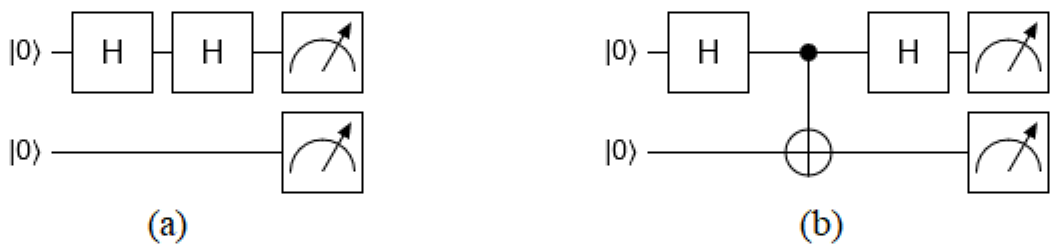


Figure 3: (a) A 2 qubit circuit in which the first qubit experiences 2 subsequent Hadamard gates. The first Hadamard gate puts the qubit in a superposition of $|0\rangle$ and $|1\rangle$ but the second Hadamard gate puts the qubit back to state $|0\rangle$. Thus it is as if there were no gates in the circuit. (b) A 2 qubit circuit where 2 Hadamard gate act on the first qubit. But, between the two Hadamard gates, a control NOT gate acts on the second qubit (see equation (15)). When the first qubit is in state $|1\rangle$ the gate performs a NOT (see equation (8)) on the second qubit. Conversely, the gate does nothing when the first qubit is in the state $|0\rangle$. The symbols at the end of the two lines represent the act of measuring the two-qubit state and collapsing any superposition.

in its original state, as it did in Figure 3 (a). The reason behind this gives another sign that our simulation behaves like a quantum mechanical system. Observation of the second qubit gives information about what the state of qubit 1 was between the two Hadamard gates. Thus, as postulated in the foundation of quantum mechanics [16], the superposition of the first qubit will collapse before the second Hadamard gate has any chance to put it back to its original state. This situation destroys any coherence and yields a random measurement of the 2-qubit state: $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$.

2.5 Simulating Shor's Factorisation Algorithm

We are now ready to discuss how Shor's algorithm for factorising 15 was coded. This code is partly based on Candela's paper [21]. In short, there were 7 qubits necessary to form a register which is split into the x-register consisting of 3 qubits and an f-register consisting of 4 qubits. We need at least 4 qubits for the f-register since we want to factorise 15, so we need to be able to have binary representation for numbers up to 15 at least. The 4 qubits give us 4 binary digits which together can represent integer numbers from 0 to 15.

After solving the initialisation and measurement problem, we had to find a way to extend 1 and 2-qubit gates to a circuit that has 7 qubits. To get an idea why this is required, recall that 7 qubits will give us $2^7 = 128$ basis states. If we were to use matrix multiplication to apply the given gates to the 128-entry vector, we needed to create 128x128 matrices. Thus a crucial part of building the code was finding a way of converting a 2x2 1-qubit gate or a 4x4 2-qubit control gate to 128x128 gates. Our method for doing this is described in Appendix 6.2. A visual representation of the quantum circuit coded is given in Figure 4.

Initially, the function gate part of the algorithm was constructed using 3 permutation matrices. The term permutation matrix refers to the fact that on each row and column there only one non-zero element, and that is 1. Generating entries for the permutation matrix is discussed in Appendix 6.4. Nevertheless, Figure 4 portrays an easier way to implement the

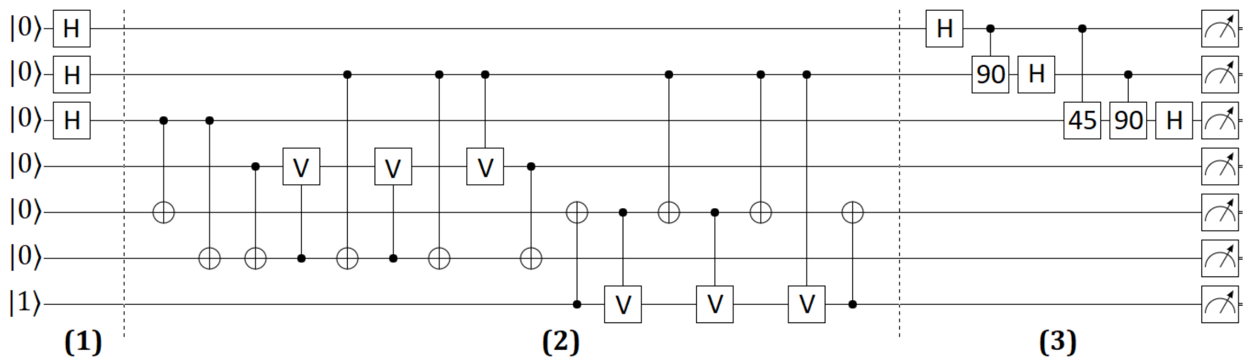
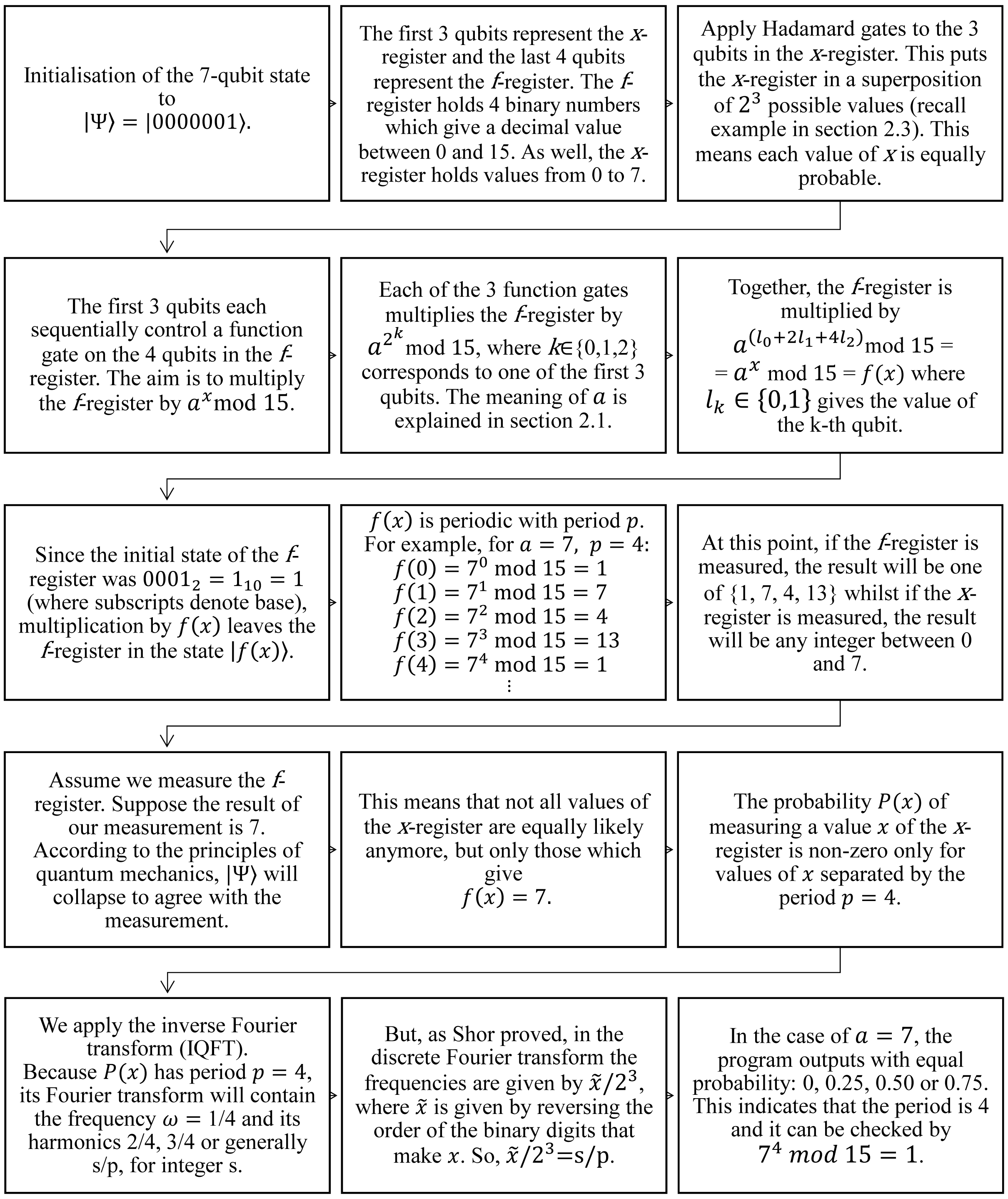


Figure 4: A gate by gate breakdown of the 7-qubit Shor's algorithm circuit used for factorising 15. All the gates used in the diagram have been discussed in detail in Section 2.2. The symbols are defined in the legend given in Appendix 6.3. The system was initialised in the state $|0000001\rangle$ as described in [21]. The first 3 qubits from the top comprise the x-register and the next 4 qubits are the f-register. Part **(1)** corresponds to putting the x-register in a superposition of all possible values of x from 0 to $2^3 - 1$, as described in Section 2.3. Part **(2)** corresponds to the gate-by-gate break down of the 3 function gates described in Candela's paper [21]. These function gates control-multiply the f-register by a^x mod 15. The resulting value of the f-register will be periodic as described in section 2.1. Part **(3)** represents the inverse Fourier transform which brings out the period in the function and we then read this out using the measurements on the far right of the diagram. A step by step process is shown in Figure 5.

function gate for 7 qubits in terms of 1 and 2-qubit gates. This was given by Vandersypen *et al.* [6]. They also used Toffoli gates, described in Section 2.2, but these are 3-qubit gates, which would be hard to implement both in our simulation and experimentally. Hence, we decompose the entirety of Shor's algorithm for 7 qubits in terms of 1 and 2-qubit gates. We know this is possible from Section 2.2. For this, we designed a program which decomposed all Toffoli gates in terms of CNOT and $\hat{V}$ gates [18]. The program also generated the full form of the inverse Fourier transform. This can be extended to an arbitrary number of qubits. Once implemented in the program, they gave the same answer as the permutation matrices counterparts. We created an output for each gate that acts on the circuit. This is shown in Figure 6. We used this output in further sections when we talked about the theoretical and time-optimal duration of Shor's 7-qubit algorithm.

Ignoring for now the IQFT part of the circuit, the output of the program was expected to be of the form $|x, f(x)\rangle$, where x is represented by a 3-digit binary number in the x-register and $f(x)$ by a 4-digit binary number in the f-register. For example, as discussed in the flowchart 5, $a = 7$ and $x = 3$ should give $f(3) = 13$. The results for all values of a that obeyed the conditions laid out in Section 2.1 are presented in Table 1.



Initialisation of the 7-qubit state to $|\Psi\rangle = |0000001\rangle$.

The first 3 qubits represent the x-register and the last 4 qubits represent the f-register. The f-register holds 4 binary numbers which give a decimal value between 0 and 15. As well, the x-register holds values from 0 to 7.

Apply Hadamard gates to the 3 qubits in the x-register. This puts the x-register in a superposition of 2^3 possible values (recall example in section 2.3). This means each value of x is equally probable.

The first 3 qubits each sequentially control a function gate on the 4 qubits in the f-register. The aim is to multiply the f-register by $a^x \text{mod } 15$.

Each of the 3 function gates multiplies the f-register by a^{2^k} mod 15, where $k \in \{0,1,2\}$ corresponds to one of the first 3 qubits. The meaning of a is explained in section 2.1.

Together, the f-register is multiplied by $a^{(l_0+2l_1+4l_2)}$mod 15 = $= a^x$ mod 15 = $f(x)$ where $l_k \in \{0,1\}$ gives the value of the k-th qubit.

Since the initial state of the f-register was $0001_2 = 1_{10} = 1$ (where subscripts denote base), multiplication by $f(x)$ leaves the f-register in the state $|f(x)\rangle$.

$f(x)$ is periodic with period p.
For example, for $a = 7,\ p = 4$:
$f(0) = 7^0$ mod 15 $= 1$
$f(1) = 7^1$ mod 15 $= 7$
$f(2) = 7^2$ mod 15 $= 4$
$f(3) = 7^3$ mod 15 $= 13$
$f(4) = 7^4$ mod 15 $= 1$
$\vdots$

At this point, if the f-register is measured, the result will be one of {1, 7, 4, 13} whilst if the x-register is measured, the result will be any integer between 0 and 7.

Assume we measure the f-register. Suppose the result of our measurement is 7. According to the principles of quantum mechanics, $|\Psi\rangle$ will collapse to agree with the measurement.

This means that not all values of the x-register are equally likely anymore, but only those which give $f(x) = 7$.

The probability $P(x)$ of measuring a value x of the x-register is non-zero only for values of x separated by the period $p = 4$.

We apply the inverse Fourier transform (IQFT).
Because $P(x)$ has period $p = 4$, its Fourier transform will contain the frequency $\omega = 1/4$ and its harmonics 2/4, 3/4 or generally s/p, for integer s.

But, as Shor proved, in the discrete Fourier transform the frequencies are given by $\tilde{x}/2^3$, where $\tilde{x}$ is given by reversing the order of the binary digits that make x. So, $\tilde{x}/2^3$=s/p.

In the case of $a = 7$, the program outputs with equal probability: 0, 0.25, 0.50 or 0.75. This indicates that the period is 4 and it can be checked by $7^4 \ mod\ 15 = 1$.

Figure 5: Flowchart portraying the process of factorising 15 using Shor's algorithm.

1. Hadamard gate acting on qubit 1.
2. Hadamard gate acting on qubit 2.
3. Hadamard gate acting on qubit 3.

---

4. CNOT gate controlled by qubit 3 acting on qubit 5.
5. CNOT gate controlled by qubit 3 acting on qubit 6.
6. CNOT gate controlled by qubit 4 acting on qubit 6.
7. V gate controlled by qubit 6 acting on qubit 4.
8. CNOT gate controlled by qubit 2 acting on qubit 6.
9. Inverse V gate controlled by qubit 6 acting on qubit 4.
10. CNOT gate controlled by qubit 2 acting on qubit 6.
11. V gate controlled by qubit 2 acting on qubit 4.
12. CNOT gate controlled by qubit 4 acting on qubit 6.
13. CNOT gate controlled by qubit 7 acting on qubit 5.
14. V gate controlled by qubit 5 acting on qubit 7.
15. CNOT gate controlled by qubit 2 acting on qubit 5.
16. Inverse V gate controlled by qubit 5 acting on qubit 7.
17. CNOT gate controlled by qubit 2 acting on qubit 5.
18. V gate controlled by qubit 2 acting on qubit 7.
19. CNOT gate controlled by qubit 7 acting on qubit 5.

---

20. Hadamard gate acting on qubit 1.
21. Phase gate PI/2 controlled by qubit 1 acting on qubit 2.
22. Phase gate PI/4 controlled by qubit 1 acting on qubit 3.
23. Hadamard gate acting on qubit 2.
24. Phase gate PI/2 controlled by qubit 2 acting on qubit 3.
25. Hadamard gate acting on qubit 3.

Figure 6: Figure displaying all the gates that act, in order, on a 7-qubit system that computes Shor's number factorisation for numbers up to 15. The qubits they act on are clearly shown. All the gates used are described in Section 2.2. The first horizontal line marks the beginning of the part which performs the 3 control function gates on the f-register as described in Figure 5. The second horizontal line marks the beginning of the inverse Fourier transform part of the program. The role of this part is also described in Figure 5.

a	p	f(x)
2	4	1, 2, 4, 8
4	2	1, 4
7	4	1, 7, 4 ,13
8	4	1, 8, 4, 2
11	2	1, 11
13	4	1, 13, 4, 7
14	2	1, 14

Table 1: Table describing the period for various values of a. The a value is an integer between 1 and 15, chosen to obey the conditions laid out in Section 2.1. p is the period of the function $f(x)$ as described in Figure 5. $f(x) = a^x \bmod 15$ is the function whose period we are calculating, in accord with Section 2.1.

After calculating the period 4, we use equation (6) from Section 2.1 to calculate the factors of 15:

$$\begin{aligned} r \times q &= 15 \\ r &= \gcd(a^{p/2} - 1, 15) = \gcd(48, 15) = 3 \\ q &= \gcd(a^{p/2} + 1, 15) = \gcd(50, 15) = 5 \end{aligned} \tag{18}$$

where r, q are co-prime numbers smaller than 15, gcd denotes greatest common denominator and p is the period of $f(x)$ as described in Figure 5. We therefore get the correct factors of 15 to be 5 and 3 which shows our factorising program works.

3 Experimental Duration of Shor's Algorithm

In this section, we investigate how Shor's algorithm is implemented experimentally using NMR techniques. We use this knowledge to calculate how long the quantum computer we have simulated would take to physically implement Shor's algorithm.

3.1 Experimental Implementation of Quantum Gates

The version of Shor's algorithm simulated in our paper is based on a physical quantum computer realised by Vandersypen *et al.* [6]. The computer uses nuclear magnetic resonance (NMR) to initialise, control, manipulate and measure the spins of nuclei within a cloud of molecules. As this paper sets out to analyse the implementation of quantum gates, we will focus on how these systems control and manipulate quantum states. More information on the initialisation and measurement of qubits in an NMR quantum computer can be found in the literature [10].

The qubit initialisation here is performed by applying a strong magnetic field in the chosen z-direction across a collection of identical molecules. Under the influence of this field, the spin-$\frac{1}{2}$ nuclei within each molecule will either align or anti-align with the z-axis due to the Zeeman effect [16]. The Hamiltonian of the spin-$\frac{1}{2}$ nucleus under the influence of a constant magnetic field is therefore

$$\hat{H}_0 = \gamma B_0 \hat{I}_z \tag{19}$$

where B_0 is the magnitude of the constant magnetic field applied across the entire system, γ is the gyromagnetic ratio of the nucleus in question and $\hat{I}_z$ is the single qubit spin-operator in the z-direction as defined in Appendix 6.5. It is also important to note that from now on we will be working in natural units ($\hbar = 1$).

As the magnetic field is constant in our system, we can simplify equation (19) by setting $\omega_0 = \gamma B_0$. We call this term the *Larmor frequency* [22]. As the electron cloud varies from atom to atom, different atoms of the same element can have different Larmor frequencies within the same molecule. It is, therefore, possible to selectively control each atom within such a molecule [23].

The Hamiltonian (19) suggests the aligned or anti-aligned spin-$\frac{1}{2}$ nuclei will split into one of the two distinct energy levels which form the qubit states of the quantum computer. A quantum computer, therefore, needs to rotate the qubit spin orientations about the orthonormal axes of the system in order to perform different gates. These transitions between states are induced by applying magnetic fields in a different direction. However, applying a second magnetic field may disrupt the field in the z-axis and as a result disrupt the basis of the qubit states. NMR techniques overcome this by applying pulses of magnetic field rotating about the z-axis of the nucleus we wish to control.

The rotating magnetic field is implemented by surrounding the cloud of nuclei with a coil connected to an AC voltage source [24]. This AC current will induce an effective magnetic field, which is proportional to the current passing through the coil, of the form

$$\mathbf{B}_r = \frac{1}{2}B_r[\cos(\omega_r t + \Phi)\hat{\mathbf{x}} + \sin(\omega_r t + \Phi)\hat{\mathbf{y}}]. \tag{20}$$

where ω_r is the frequency of the AC circuit and Φ is the initial phase of our alternating current. This magnetic field can now be used to determine a new expression for the Hamiltonian of a spin-$\frac{1}{2}$ nucleus in the lab frame under the influence of both the constant and rotating magnetic field:

$$\hat{H} = \hat{H}_0 + \hat{H}_r \tag{21}$$

where $\hat{\mathrm{H}}_0$ is defined in equation (19) and $\hat{\mathrm{H}}_r$ is the part induced by the rotating magnetic field. This part can be calculated by taking the dot product of the nucleus's magnetic moment with this rotating magnetic field, just as in equation (19). We therefore get the following expression for our Hamiltonian for a spin-$\frac{1}{2}$ nucleus in the lab frame under the influence of the stationary and rotating magnetic fields:

$$\hat{H} = \omega_0 \hat{I}_z + \omega_n[cos(\omega_r t + \Phi)\hat{I}_x + \sin(\omega_r t + \Phi)\hat{I}_y], \tag{22}$$

where $\omega_n = \frac{\gamma}{2}\mathrm{B}_r$ is what we call the nutation frequency. As we will later see it is the frequency at which the spin will nutate about the orthonormal axes. This Hamiltonian in the ω_r rotating frame, described in Appendix 6.6, becomes:

$$\tilde{H} = [\omega_0 - \omega_r]\hat{I}_z + \omega_n[\cos(\Phi)\hat{I}_x + \sin(\Phi)\hat{I}_y]. \tag{23}$$

By setting the frequency of the AC circuit such that $\omega_0 = \omega_r$, the z-spin component of the Hamiltonian will cancel as all the spin-$\frac{1}{2}$ nucleus will experience is a constant magnetic field. Thus the quantisation along the z-axis is maintained. Regardless, the rotating magnetic field pulses cause the nuclear spin to rotate about a different axis, hence putting it into a superposition of the qubit states. We are now able to control the phase Φ of our AC circuit and thus select the axis of rotation. By combining rotations about the x and y-axes, we can effectively induce any single qubit quantum gate on our system.

It is important to note that some nuclei will have very similar values of γ. It is therefore required that a sequence of magnetic field pulses are implemented to shield one nucleus from the effects of the rotating field operating on the other nucleus [17]. However, for small-scale NMR quantum computers this does not present a significant problem and so we will neglect these pulses in our simulation [24].

If we now wish to perform a single qubit operation, such as a NOT gate, we induce a magnetic field in the x-direction by setting $\Phi = 0$. This operation would thus have the following Hamiltonian:

$$\tilde{H}_{\text{NOT}} = \omega_n \hat{I}_x. \tag{24}$$

Any qubit state $|\tilde{\psi}\rangle$, under the action of this Hamiltonian, will evolve over time like

$$|\tilde{\psi}(t)\rangle = \exp\left(-i\omega_n \hat{I}_x t\right)|\tilde{\psi}(0)\rangle\,. \tag{25}$$

After a time of t = $\frac{\pi}{\omega_\text{n}}$ the state will evolve such that the exponential becomes

$$\exp\left(-i\pi\hat{I}_x\right) = \begin{pmatrix} 0 & -i \\ -i & 0 \end{pmatrix}. \tag{26}$$

This matrix corresponds to the NOT gate operation up to a global phase of $-i$. Fortunately, this can be neglected as it does not represent any observable quantity. This can be shown by looking at the density matrix of the pure state $|\psi'\rangle = e^{i\lambda}|\psi\rangle$, $\lambda \in \text{R}$, which represents our desired final state under a global phase shift:

$$|\psi'\rangle\langle\psi'| = e^{i\lambda}|\psi\rangle\langle\psi|\,e^{-i\lambda} = |\psi\rangle\langle\psi| \;\;\forall\;\lambda. \tag{27}$$

We will therefore denote this single NOT rotation as 180°_x as it is applied by inducing a rotation of 180° about the x-axis. Likewise any single rotation of an angle ϕ about either the x or y-axis will take an evolution time of:

$$t_\phi = \frac{\phi}{\omega_n}. \tag{28}$$

Thus if we apply the same pulse but for half the time span, we end up with the operation of a square-root NOT gate which is also used in the simulation. The Hadamard gate described in Section 2.2 is experimentally implemented using a sequence of three pulses:

$$45^\circ_y \rightarrow 180^\circ_x \rightarrow 45^\circ_{-y}. \tag{29}$$

This gate is implemented such that it always acts as a self-inverse operation regardless of the initial qubit state.

We have therefore introduced how gates are implemented for single qubits. An additional level of complexity is added when implementing controlled gates. Let us consider the CNOT gate. This operation is applied by sandwiching a controlled $\hat{\pi}$ phase gate between two Hadamard gates acting on the controlled qubit [18]. A controlled $\hat{\pi}$ phase gate is represented by:

$$\hat{\pi} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}. \tag{30}$$

This matrix can be expressed as the sum of the two-qubit spin operators defined in Appendix 6.5:

$$\hat{\pi} = \exp[\pm i\frac{\pi}{2}(\frac{1}{2}\hat{\mathbb{1}} - \hat{I}_z - \hat{S}_z + 2\hat{I}_z\hat{S}_z)], \tag{31}$$

with the $\frac{1}{2}\hat{\mathbb{1}}$ term included as a global phase shift which can be neglected. The coupled term $(2\hat{I}_z\hat{S}_z)$ is implemented using a spin-echo technique [24] which involves a sequence of x and

y-axis rotations acting on *both* qubits. The $\hat{I}_z$ and $\hat{S}_z$ operations are implemented by periods of free precession about the z-axis which can be interpreted as rotating the ω_r reference frame. Due to the fact that operations on different qubits commute, they can be applied in any order, thus a variety of combinations can be implemented. Some are favoured over others as rotations are cancelled. One of the optimal combination given in the literature [10] would therefore be

$$\frac{1}{4J} \to 180^\circ_x \to \frac{1}{4J} \to 90^\circ_x \to 90^\circ_{-y} \to 90^\circ_x, \tag{32}$$

with each pulse acting on both spins and the $\frac{1}{4J}$ representing the periods of free evolution given by the spin-coupling frequency J which is intrinsic to any pair of two spin-$\frac{1}{2}$ nuclei within the same molecule. Although other variations can be made on this method, each will contain a total coupling time of $\frac{1}{2J}$ such that the spin of the two qubits evolve to be in an anti-phase state. As the J-coupling is different for each pair of nuclei, this process is crucial for selective 2-qubit 'control' in this system.

To further simplify the CNOT operation, we can replace the Hadamard gates on either side of the phase gate with an inverse pseudo-Hadamard (90°_{-y}) and a pseudo-Hadamard (90°_y). Alone these operations are non-unitary and are not good substitutes for the Hadamard operation given in equation (29), but when combined with the phase gate rotations (32) form a unitary CNOT operation.

The controlled square-root NOT gate is performed similarly but with $\phi = \frac{\pi}{2}$ phase gate. This gate is implemented using the same pulse sequence as seen in equation (32) but with the 90°_{-y} pulse replaced with a 45°_{-y} and the evolution times are halved:

$$\frac{1}{8J} \to 180^\circ_x \to \frac{1}{8J} \to 90^\circ_x \to 45^\circ_{-y} \to 90^\circ_x. \tag{33}$$

The $\phi = \frac{\pi}{4}$ phase gate again follows the same pattern of halving the y-rotation and evolution time:

$$\frac{1}{16J} \to 180^\circ_x \to \frac{1}{16J} \to 90^\circ_x \to 22.5^\circ_{-y} \to 90^\circ_x. \tag{34}$$

By adding up the time to make each rotation induce these phase gates using equation (28) along with the duration of the J-coupling intervals, we get the following expressions for the time taken to complete the phase gate operation:

$$t^{\,180^\circ} = \frac{3\pi}{2\omega_n} + \frac{1}{2J}, \qquad t^{\,90^\circ} = \frac{9\pi}{4\omega_n} + \frac{1}{4J}, \qquad t^{\,45^\circ} = \frac{11\pi}{8\omega_n} + \frac{1}{8J}, \tag{35}$$

where the superscript denotes the angle of the phase gate.

We now have all the tools needed to calculate the time it would take to physically implement our version of Shor's algorithm using NMR techniques.

3.2 Experimental Duration of Shor's Algorithm in NMR

In Section 3.1 we learnt how the gate operations in Shor's algorithm can be implemented using NMR techniques. Moreover, we have introduced a way of calculating the time it takes to perform each operation using equations (28) and (35). Employing these, we only need to

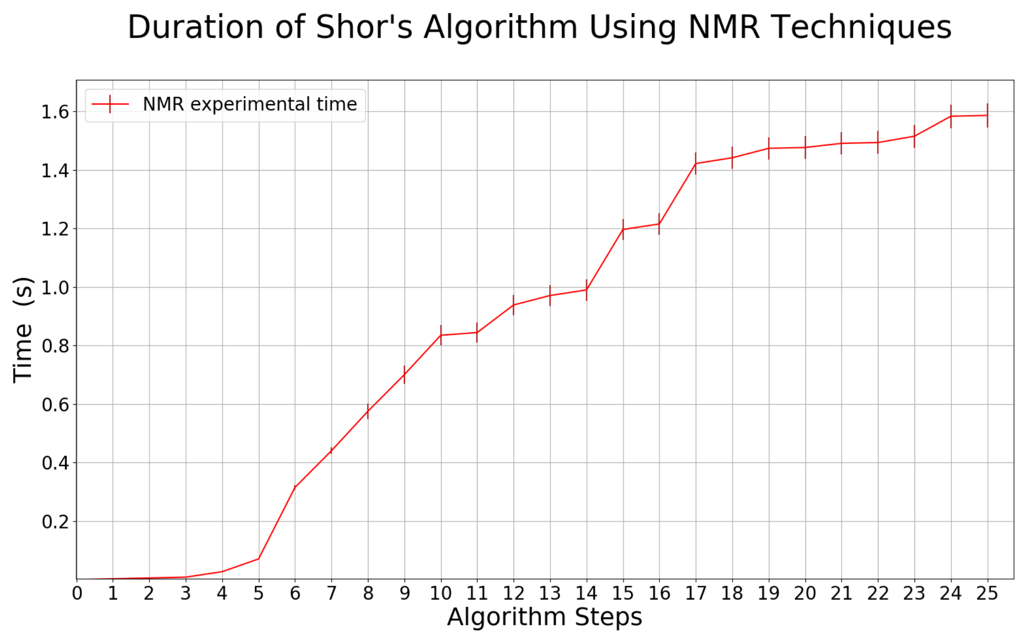


Figure 7: Plot showing the time evolution of the quantum computer we have simulated using NMR techniques. The x-axis represents each of the 25 steps of our quantum computation given in Figure 6. The y-axis shows the cumulative time it takes to perform each step in our algorithm, calculated using the equations given in Section 3.1 and using the J-coupling values given in Appendix 6.7. Errors are not quoted but we assume the error is given by the precision of the values provided. The steepest lines are due to the controlled operations between the qubits with the weakest J-coupling. This plot was made using Python.

know the ω_n and J-coupling values used in the experimental realisation [6]. Although the J-coupling values are provided, the exact values needed to calculate ω_n had to be estimated. We can use the approximate times given for all single qubit rotation to acquire a good approximation for ω_n. This approximation is made due to the fact that we are told the pulse times range from '0.22 to ∼2ms' [6]. As our shortest rotations (22.5°_{-y}) take 8 times less time than the (180°_{-y}) rotations then it implies that $t_\phi = \frac{\pi}{\phi} \times 1.88 \pm 0.12$. Therefore by using equation (28) we estimate $\omega_n = 1670 \pm 110\text{H}_z$. This is a reasonable estimation as nutation frequencies tend to be in the low radio-frequency range [24]. Using this ω_n along with the J-coupling values of our system given in Appendix 6.7, we can plot the cumulative time of each of the 25 gates (see Figure **??**).

We calculate the total time an NMR quantum computer would take to perform the quantum part of Shor's algorithm to be 1.59 ± 0.04s. In particular, the time taken to perform the inverse quantum Fourier transform part (steps 20-25) of our computation is 112 ± 2ms, as portrayed in Figure 7. We cannot compare the total time with that given in the experimental realisation as certain gates are "removed" and "replaced by simpler gates" [6], yet no information is given on what these adaptations are. However, the inverse quantum Fourier transform is performed in the same way as in our simulation and is quoted as taking of the order of 120ms to implement which compares well to our theoretical value. Discrepancies between the times are likely due to the nuclei representing the first and third qubits having similar gyromagnetic frequencies, thus a few short pulses will be added in the experiment to refocus one nucleus after the other has been operated on. For simplicity, we have omitted the times of the refocusing pulses as they do not take a significant amount of time [24]. As errors are not given in the data set in Appendix 6.7, we assume the errors are given by the precision of the values provided.

We now have a model of how all the gates of a 7-qubit NMR quantum computer which uses Shor's algorithm to factorise 15 can be implemented. We have a way of computing the time this system would take to calculate the period described in Section 2.5 based on the intrinsic characteristics of the molecule used. In the next section, we will study how much quicker these operations could be performed via time-optimal translations.

4 Time Optimisation of Shor's Algorithm

One critical aspect of any type of computation, whether it is on a classical or quantum computer, is the computation time. This is the *physical time* it takes for a certain task to be performed. Take for instance Moore's observation which states that the processors' speed will approximately double every year [25]. We aim to investigate how this transfers to quantum computers. This section is devoted to describing the basics of time-optimal computation with special regard to the implementation of Shor's algorithm using NMR techniques by employing the so-called quantum brachistochrone equation.

4.1 An Introduction to Time-Optimal Quantum Computation: the Quantum Brachistochrone Equation

Quantum computation relies on the use of *unitary operators* (quantum gates) that map an input state to some output state depending on the nature of the gate employed. This is denoted, in the Schrödinger picture, as:

$$\hat{U}(t_f, t_i)\left|\psi(t_i)\right\rangle = \left|\psi(t_f)\right\rangle, \tag{36}$$

where $\left|\psi(t_i)\right\rangle$ and $\left|\psi(t_f)\right\rangle$ denote the state at time t_i initial and t_f final respectively. We can observe that the nature of the time optimisation problem described above relies on time-optimising the transition between the initial input state $\left|\psi(t_i)\right\rangle$ to the final target output state $\left|\psi(t_f)\right\rangle$. We shall hence study *what is the fastest possible way that any transformation of the form of (36) can be done, subject to some initial constraints*. Such a task is already well described by what is known as the *quantum brachistochrone equation* [26], which solved the following variational problem [27]:

Given an initial pure state, $\left|\psi_i\right\rangle$, and a final state $\left|\psi_f\right\rangle$ find the possibly time-dependent Hamiltonian, $\hat{H}$, such that $\left|\psi_i\right\rangle$ evolves to $\left|\psi_f\right\rangle$ in the shortest possible time.

The above problem was solved by proposing the action

$$S(\psi,\phi,\hat{H},\lambda)=\int dt\left[\frac{\sqrt{\langle\dot{\psi}|\,(1-\hat{P})|\dot{\psi}\rangle}}{\Delta E}+\mathrm{Re}\left(i\,\langle\dot{\phi}|\psi\rangle+\langle\phi|\,\hat{H}\,|\psi\rangle\right)+\sum\nolimits_m\lambda_m f_m(\hat{H}')\right], \tag{37}$$

where $\hat{P}=|\psi\rangle\,\langle\psi|$ represents the projector onto the state $|\psi\rangle$, ΔE is the energy variance of the Hamiltonian $\hat{H}$, $|\phi\rangle$, λ are Lagrange multipliers and $f_m(\hat{H}')=0$ are the constraints imposed on $\hat{H}'$, the traceless part of the Hamiltonian. The over-dot represents the time derivative. For convenience, we will from now on consider the Hamiltonian to be traceless as it can always be rescaled by:

$$\hat{H}'=\hat{H}-\frac{Tr(\hat{H})}{N}\hat{\mathbb{1}}, \tag{38}$$

where $\hat{H}'$ is the traceless part of the Hamiltonian, $\hat{\mathbb{1}}$ is the identity and N is the dimension of the Hilbert space. This is because we are more interested in the spacing of the energy levels (energy spectrum) rather than the actual value of the levels themselves [26]. From now onwards we will omit the dash notation employed in (38) unless it is explicitly stated. With this action (37) in hand, and by taking the variation with respect to $|\phi\rangle$, $|\psi\rangle$ and $\hat{H}$ we obtain

$$\hat{F}=\hat{F}\hat{P}+\hat{P}\hat{F} \tag{39}$$

and the brachistochrone equation

$$\frac{d}{dt}\hat{F}=i[\hat{H},\hat{F}] \tag{40}$$

where

$$\hat{F}\triangleq\sum_m\lambda_m\left(\frac{\delta f_m}{\delta\hat{H}}-\langle\frac{\delta f_m}{\delta\hat{H}}\rangle\hat{P}\right). \tag{41}$$

Given some constraints $f_m(\hat{H})$ we find $\hat{F}$ by using (41). With $\hat{F}$, we may now solve (39) with (40) to obtain the time-optimal Hamiltonian $\hat{H}$ and the time-optimal state $|\psi\rangle$ that evolves from our initial state $|\psi_i\rangle$ to some final target state $|\psi_f\rangle$. The next subsection will be devoted to describing the most general constraint $f_m(\hat{H})$ used in the literature. We will use this constraints to adapt the solutions of (40) to the experimental NMR computer described in Section 3 to study the optimal duration of Shor's algorithm.

4.2 Energetic Constraints of the Time-Optimal Hamiltonian: the Isotropic Constraint

In this section, we will impose the most general constraint, the so-called isotropic constraint [26], that simply bounds the available energy of the system (that is perhaps limited by the nature of the experimental set-up). This constraint also avoids the trivial solution of an infinitely energetic Hamiltonian that would evidently lead to an infinitely fast transition, as for a time-invariant Hamiltonian:

$$\hat{U}(t_f, t_i) = \exp(-i(t_f - t_i)\hat{H}) \iff \exp(-i(t_f - t_i)\hat{H}) |\psi(t_i)\rangle = |\psi(t_f)\rangle , \qquad (42)$$

and by rescaling $\hat{H} \to \alpha\hat{H}$, where $\alpha > 1$ is simply a dimensionless parameter, we have:

$$\lim_{\alpha\to\infty} \exp(-i\alpha(t_f - t_i)\hat{H}) |\psi(t_i)\rangle = |\psi(t_f)\rangle \iff (t_f - t_i) \to 0. \qquad (43)$$

A mathematical way of bounding the available energy is to limit the energy spectrum 'variance' to a constant value. This can be realised by imposing

$$\left\{\frac{Tr(\hat{H}^2)}{N} = \overline{E^2}\right\} \wedge \left\{\frac{Tr(\hat{H})}{N} = \overline{E}\right\} \longrightarrow \omega^2 = \overline{E^2} - \overline{E}^2 = \overline{E^2} \quad \because \ Tr(\hat{H}) = 0 \qquad (44)$$

where $\wedge$ is the logical conjunction, $\overline{E^2}$ denotes the arithmetic mean of the eigenvalues squared of $\hat{H}$, and $\overline{E}$ denotes the arithmetic mean of the eigenvalues of $\hat{H}$. In fact, it can be mathematically shown [26] that due to the nature of the variational problem

$$\langle\hat{H}\rangle = \frac{Tr(\hat{H})}{N} \text{ and } \langle\hat{H}^2\rangle = \frac{Tr(\hat{H}^2)}{2}, \qquad (45)$$

where $\langle\hat{H}\rangle$ denotes the expectation value of $\hat{H}$ with respect to $|\psi\rangle$. Expression (45) implies

$$(\Delta E)^2 = \langle\hat{H}^2\rangle - \langle\hat{H}\rangle^2 = \langle\hat{H}^2\rangle = \omega^2, \qquad (46)$$

where ΔE is the energy variance and we have used expression (44). We are now in a position to assert the general mathematical form of our energy constraint,

$$f_m(\hat{H}) = \frac{Tr(\hat{H}^2)}{2} - \omega^2 = 0. \qquad (47)$$

4.3 Time-Optimal Translations Under Experimental NMR Constraints

We now turn back to the initial problem: what is the fastest way some quantum state $|\psi_i\rangle$ can evolve to some other target state $|\psi_f\rangle$? We shall now portray this problem in the realm of quantum computation by describing perhaps the most illustrative example, a NOT gate. Consider the two possible endomorphisms performed by a NOT gate, namely:

$$|1\rangle \longrightarrow |0\rangle \quad \text{and} \quad |0\rangle \longrightarrow |1\rangle . \qquad (48)$$

Hence the previous question is more precisely formulated as what is the fastest way the above maps can be performed. We can now obtain operator $\hat{F}$ from (41) by using (47). With $\hat{F}$, we may now solve (40) with (39) to obtain $\hat{H}_{op}$ and $|\psi\rangle_{op}$ [26]:

$$\hat{H}_{op} = i\omega(|\psi'_f\rangle \langle\psi_i| - |\psi_i\rangle \langle\psi'_f|). \tag{49}$$

and

$$|\psi(t)\rangle_{op} = cos(\omega t)\,|\psi_i\rangle + sin(\omega t)\,|\psi'_f\rangle\,, \tag{50}$$

where $|\psi'_f\rangle$ and $|\psi_i\rangle$ denotes the final and initial state which are orthonormalised by using the Gram-Schmidt method. In practice, most quantum gates, like the NOT or CNOT gate, have initial and final states already orthonormal. However some gates, like the Hadamard or a general phase gates, need to have their states orthonormalised. Employing the expression above for the NOT gate we have that for the endomorphism $|1\rangle \longrightarrow |0\rangle$ the state will optimally evolve like:

$$|\psi(t)\rangle = cos(\omega t)\,|1\rangle + sin(\omega t)\,|0\rangle\,. \tag{51}$$

Conversely, for the remaining endomorphism $|0\rangle \longrightarrow |1\rangle$ we may re-express equation (51) as:

$$|\psi(t)\rangle = cos(\omega t)\,|0\rangle + sin(\omega t)\,|1\rangle\,. \tag{52}$$

Observe that both endomorphisms take an optimal time of

$$T_{op} = \frac{\pi}{2\omega} \tag{53}$$

and we hence conclude that *any NOT gate may not be achieved in less time than T_{op}*. If we consider the NOT gate in the projective space $\mathbb{C}P^{n-1}$ ($\mathbb{C}P^1$ for a single qubit case) of the Bloch sphere (Poincaré sphere) we recognise it to be an 180° rotation. It is often useful to portray quantum gates as rotations in the Bloch sphere and we shall exploit such convention from now on.

In a manner akin to the method described above we find the optimal times for $\frac{\hat{\pi}}{2}$ and $\frac{\hat{\pi}}{4}$ gates to be:

$$T_{op}^{\,90^\circ} = \frac{\pi}{4\omega}, \qquad T_{op}^{\,45^\circ} = \frac{\pi}{8\omega}, \tag{54}$$

where we will from now on regard the superscript in $T_{op}^{\,\phi}$ as that of the rotation angle ϕ. We shall frequently refer to (54) in the following sections. Furthermore, it can be showed [26] that generally, the optimal time evolution of a pure state is given by:

$$T_{op} = \frac{1}{|\omega|} arccos(|\langle\psi_f|\psi_i\rangle|). \tag{55}$$

We can test the above results by comparing the optimal time T_{op} with that obtained experimentally in the NMR picture described in Section 3. In order to obtain ω we shall summon equation (22), the Hamiltonian for a single NMR qubit rotation, and re-write it in its matrix form:

$$\hat{H} = \begin{pmatrix} \frac{1}{2}\omega_0 & \omega_n \exp\left(-i\Lambda\right) \\ \omega_n \exp\left(i\Lambda\right) & -\frac{1}{2}\omega_0 \end{pmatrix}, \tag{56}$$

where $\Lambda = \omega_r t + \Phi$. Note that the Hamiltonian of our single qubit is therefore traceless:

$$\mathrm{Tr}(\hat{H}) = \frac{1}{2}\omega_0 - \frac{1}{2}\omega_0 = 0. \tag{57}$$

We obtain the trace of $\hat{H}^2$ by squaring expression (56):

$$\hat{H}^2 = \begin{pmatrix} \frac{1}{2}\omega_0 & \omega_n \exp(-i\Lambda) \\ \omega_n \exp(i\Lambda) & -\frac{1}{2}\omega_0 \end{pmatrix}^2 = \begin{pmatrix} \frac{1}{4}\omega_0^2 + \omega_n^2 & 0 \\ 0 & \frac{1}{4}\omega_0^2 + \omega_n^2 \end{pmatrix} \tag{58}$$

and taking the trace

$$\mathrm{Tr}(\hat{H}^2) = \frac{1}{2}\omega_0^2 + 2\omega_n^2. \tag{59}$$

We can now obtain ω by using (47):

$$\omega^2 = \frac{1}{4}\omega_0^2 + \omega_n^2 \tag{60}$$

and therefore

$$\omega = \sqrt{\frac{1}{4}\omega_0^2 + \omega_n^2}. \tag{61}$$

The above expression can be further simplified by employing a binomial expansion since $1 \gg \left(\frac{2\omega_n}{\omega_0}\right)^2$. This yields

$$\omega = \frac{\omega_0}{2} + \frac{\omega_n}{16}. \tag{62}$$

When using the typical experimental values [6], this is a really good approximation with an average agreement of 99.9%.

Expression (62) will be employed in the next section to make a quantitative comparison between the experimental NMR time obtained in Section 3 and the optimal time described in this section.

4.4 Analysis of the Lower Bound of Time Taken to Physically Perform Shor's Algorithm

After discussing the main ideas of time-optimisation we will set the lower bound of time it takes a 7-qubit NMR computer to factorise the number 15 by employing Shor's algorithm. For determining the time of such a task we will employ the time-optimal theory described above under the following main assumption:

All controlled gates will take, at minimum, the time elapsed by their time-optimal corresponding single qubit gates. Therefore the lower bound of time of any controlled gate is that of its single qubit time-optimal gate.

For instance, a CNOT will minimally take the same time as a time-optimal NOT gate and thus the time-optimal time of a CNOT is lower bounded to $T^{\,180^\circ}_{c-op} = \frac{\pi}{2\omega}$, where the subscript $c - op$ denotes the controlled nature of the gate. The above assumption relies on the fact that the implementation of controlled gates in NMR quantum computing greatly vary in technique from gate to gate, thus it is impractical to make any other assumption.

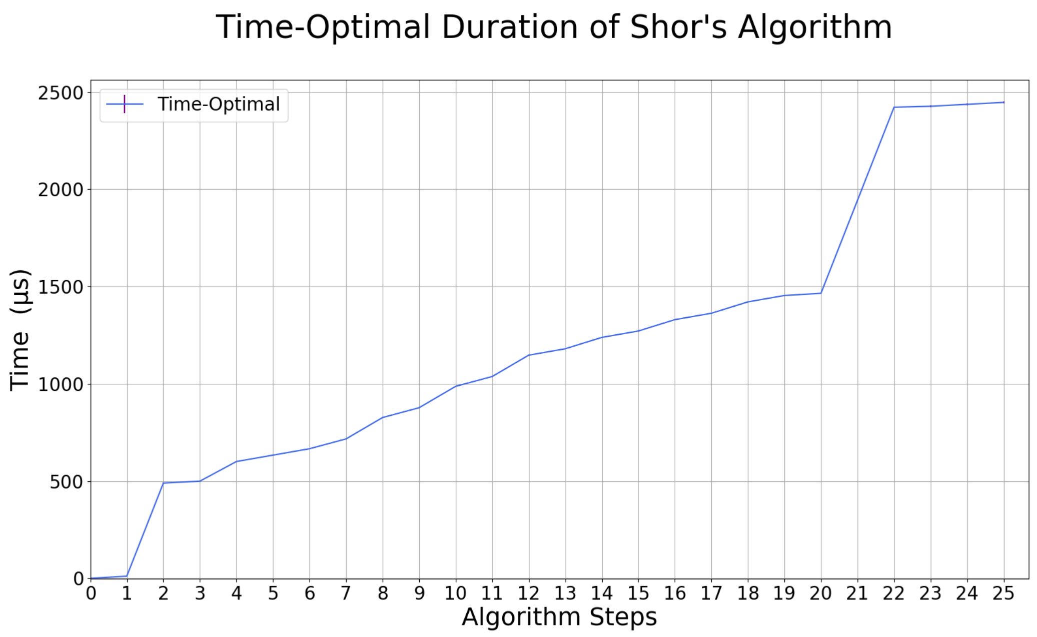


Figure 8: Plot showing the time evolution of the quantum computer using time-optimal state translations. The x-axis represents each of the 25 steps of the quantum computation given in Figure 6. The y-axis shows the cumulative time it takes to perform each step in our algorithm, calculated using the equations given in Section 4.3 and the frequencies given in Appendix 6.7. Errors are not quoted but we assume the error is given by the precision of the values provided. These errors are minute and do not appear clearly on the plot. This plot was made using Python.

By using equations (55), (62) and using the frequency values given in Appendix 6.7, we can plot the cumulative optimal time of the entirety of Shor's algorithm.

The optimal time for performing Shor's algorithm given by Figure 8 is

$$T_{\ op-shor} = 2.446 \pm 0.004\text{ms}. \tag{63}$$

We see a distinct correlation between the time required to perform each step and the Larmor frequency of the qubit being rotated e.g the peaks to step 2, 21 and 22 correspond to spin rotations of the second qubit, whose Larmor frequency is significantly lower than the other spin-$\frac{1}{2}$ nuclei in our system. We can now compare the time-optimal duration with the NMR implementation by superimposing Figure 8 onto Figure 7 with a logarithmic scale on the y-axis.

We see in Figure 9 that the experimental quantum computer would take of the order of 650 times longer to perform Shor's algorithm. The steps with the largest differences in gradient come from the controlled operations as they involve a greater number rotations in the Bloch sphere compared to the single rotation of the time-optimal trajectory.

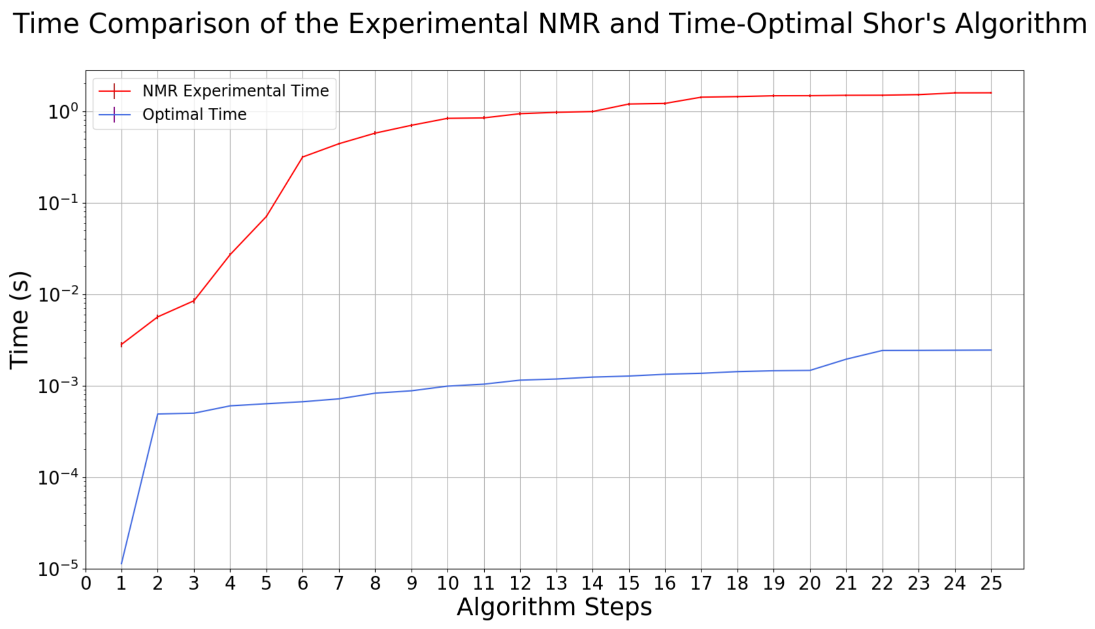


Figure 9: Plot comparing the duration of Shor's algorithm for factorising 15 using NMR techniques and time-optimal translations. The scale is logarithmic such that both data sets appear clearly on the plot. The x-axis represents each of the 25 steps of the quantum computation given in Figure 6. The y-axis shows the cumulative time it takes to perform each step in our algorithm, calculated using the frequency values given in Appendix 6.7. Errors are not quoted but we assume the error is given by the precision of the values provided, although they are not clearly visible on the logarithmic scale. The 0 point on the x-axis is omitted as it corresponds to $\log(0) = -\infty$ on the logarithmic scale. This plot was made using Python.

5 Conclusions

In this paper, we successfully simulated a quantum computer that performs Shor's factorising algorithm for numbers up to 15. We then investigated how long it would take to perform this computation using realised experimental techniques and compared this with the theoretical lower bound of time it would take to perform this computation under the same physical constraints.

Section 2 began by presenting the steps of Shor's algorithm for factorising composite numbers. It motivated the need for a quantum algorithm to find the modular period of a function. Then the discussion moved on to explaining 1 and 2-qubit gates. An important theorem was stated: any quantum logic gate can be decomposed into a sum of 1-qubit gates and controlled phase gates. The remainder of the section focused on the implementation of quantum algorithms on classical computers using Java. A random number generator was used to simulate the quantum randomness and vectors were used to represent superpositions. In Figure 2 we showed that this paradigm efficiently simulates quantum mechanical behaviour and then we used 2 examples in Figure 3 to bring out the interesting behaviour of quantum circuits. The section then culminated with the presentation of how Shor's algorithm for factorising 15 was implemented. A full diagram of the circuit is portrayed in Figure 4 and Flowchart 5 presents the process step by step.

Having verified that the combination of logic gates used in our simulation were effective in making the quantum calculation of Shor's algorithm, we investigated in Section 3.1 how such quantum gates could be implemented using an NMR quantum computer. Moreover, we presented a set of equations that can be used to calculate how long each gate would take to perform using experimental NMR techniques. With these equations, we predicted that under the same experimental values as a realised quantum computer [6], it would take 1.59 ± 0.04s to perform the algorithm. The quantum inverse Fourier transform part of this algorithm would take 112 ± 2ms, which stands in good agreement with the approximate 120ms it takes an experimental NMR quantum computer to perform this operation.

With this agreement, we proceeded in Section 4 to ask the central question of our paper: *Under the same energy constraints as the NMR quantum computer, what is the quickest time a quantum computer could perform Shor's algorithm*? This involved an investigation into quantum time-optimal theory and the quantum brachistochrone equation. By utilising this theory we proposed the lower bound of time it would take to perform each quantum gate used in our simulation and discovered this was only dependent on the energy variance of our qubit states. By deriving equations for the time duration, we found the lower bound of time to be 2.446 ± 0.004ms. This result is of the order of 650 times faster than using NMR techniques under the same energy constraints. This shows that current quantum computers only scratch the surface of their potential with regards to fast computations. As larger scale quantum computers are built, we suggest that the quickest ones will utilise qubit states with the maximum energy fluctuations allowed by the two-level system. Furthermore, the fastest computers will also be the ones that are able to implement qubit translations closest to the time-optimal proposed in this paper.

5.1 Further Directions

In recent years, physicists have begun moving away from NMR techniques for quantum computing as numerous problems arise when scaling such systems [28]. Alternative techniques have therefore been proposed and realised for performing quantum computations. These methods include manipulating trapped ions [29], solid state systems [30] and even optical systems [31]. Further research could be made into the energy constraints of such systems and, by extension of the equations proposed in this paper, a value could be derived for the time-optimal duration of such quantum computers.

To further adapt the time-optimal theory, research could be made into how the quantum brachistochrone problem extends to mixed states as opposed to the pure states we deal with in this paper. In 2016 a paper was published [32] which analysed the time-optimal evolution of mixed states. This theory could indicate a way of calculating the time-optimal duration of quantum computers that are based on mixed state systems [33].

6 Appendix

6.1 Polynomial of Degree 2 (ax^2) Fit for the Data in Figure 2

The reason we use a polynomial of degree 2 to fit the data in Figure 2 lies in the way we chose our coefficients to run from 1 to 8 and in one of the postulates of quantum mechanics [16]. This postulate states that the outcome probability of a certain results is given by the modulus squared of the coefficient in front of it. Therefore, since our coefficients are {1, 2, 3, 4, 5, 6, 7, 8} we expect the probabilities to scale as {1, 4, 9, 16, 25, 36, 49, 64}. As a consequence, a function ax^2 should perfectly fit this data. Analysis from QTiPlot gives: $a = 1$ and *adjusted R squared* = 1. These coefficients are yet to be normalised since the sum of squared do not add up to one. But Operation.normalise() method in our program solves this problem by dividing all coefficients by the square root of their sum of squares [16]. This adaptation should not affect the shape of our function, only the orientation. Instead of $a = 1$, we get the values quoted in Table 2.

	Number of measurements (powers of 10)				
	1	2	3	4	5
a	0.0542	0.494	4.88	49.2	489.99
adjusted R squared	0.83663	0.89422	0.99898	0.99846	0.99997

Table 2: Table gives the result of 5 sets of measurements of a 8 basis state superposition with coefficients { 1, 2, 3, 4, 5, 6, 7, 8}. The number of measurements for each set is given in the 5 columns beneath the first row ({10, 100, 1000, 10000, 100000} measurements). In order to quantify how well the program simulates the quantum behaviour, the data was fitted with the function ax^2. Perfect fit is obeyed by the exact answer from the postulates of quantum mechanics. The best characterisation of how well the data fits the function is given by the value of *adjusted R squared*. The closer it is to 1 the better the fit [20]. Parameter a is the coefficient in ax^2 and it is a measure of the tilt of the graph. It differs from 1 just due to our normalisation. Different powers of 10 in the values of a only reflect the increasing order in the number of measurements in each set.

6.2 Expanding 2×2 and 4×4 Matrices to N×N Matrices

We calculate the 128x128 matrices entry by entry. The row and column indexes of each entry are converted to binary notation. For 7 qubits, these indexes should be 7 digits long. Firstly we ignore the digits given by the qubits involved in the operation. If the remaining parts of the two binary numbers (for row and for column) are different, the whole entry is set to 0. If they are the same, the digits that we ignored give the decimal position of an entry in the 2x2 or 4x4 matrices (1-qubit or 2-qubit gates). This will be our answer for the entry of the 128x128. Now we needed to repeat this process 128x128-1 times. The method we used is presented in detail in Candela's paper [21].

6.3 Legend

In this section, we detail the circuit representations of the gates that we have used and represented in this paper. All gates are made using Quirk.

- Firstly, the Hadamard gate is represented:

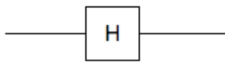


Figure 10: Hadamard Gate in a circuit.

- The CNOT gate is represented:

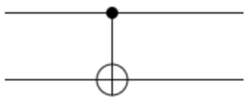


Figure 11: CNOT Gate in a circuit.

- The 45 degree controlled rotation gate is represented:

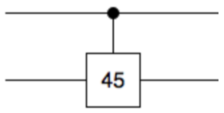


Figure 12: 45 degrees rotation Gate in a circuit.

- The 90 degree controlled rotation gate is represented:

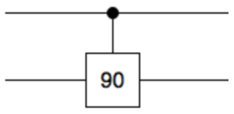


Figure 13: 90 degrees rotation Gate in a circuit.

- The controlled V gate is represented:

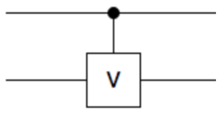


Figure 14: Controlled V gate in a circuit.

6.4 Generating Permutation Matrices to Achieve the Function Gates for Shor's Algorithm

The columns of the permutation matrices in Section 2.3 were sequentially generated by converting the 4-digit binary number f-register to a decimal number and multiplying it by a^{2^k} mod 15, as described in the flowchart, Figure 5. The decimal number was then converted back to a base 2 number. The 3 digits of the x-register were then pre-appended to it. That gave us a 7 digit binary number. Its decimal form gave us the row on which the column we started with will have an entry 1. Then we had to repeat the process for all other 127 columns and then the whole method for the other 2 function gates.

6.5 Spin Matrices

To avoid confusion, we use the symbol $\hat{\mathbb{1}}$ to represent the identity matrix in this paper:

$$\hat{\mathbb{1}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{64}$$

when considering qubit rotations, we need to consider spin operators for our system with $\hbar = 1$:

$$\hat{X} = \frac{1}{2}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \tag{65}$$

$$\hat{Y} = \frac{1}{2}\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \tag{66}$$

$$\hat{Z} = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{67}$$

For many of our calculations, we do not actually use these matrices. We instead opt for the $\hat{I}$ and $\hat{S}$ notation which refers to the first and second spin-$\frac{1}{2}$ particle in our system respectively. When describing a system with a single nucleus, the $\hat{I}_z$ operation will simply be given as

$$\hat{I}_z = \hat{Z}. \tag{68}$$

However, when dealing with more qubits we have to redefine our spin operators to denote which qubit they act upon. This is done by using tensor products.

$$\hat{I}_z = \hat{Z} \otimes \hat{\mathbb{1}} \tag{69}$$

$$\hat{S}_z = \hat{\mathbb{1}} \otimes \hat{Z} \tag{70}$$

The same method is used for the x and y spin matrices, only with the tensor products of (65) and (66) used instead.

6.6 Changing Reference Frames for Quantum States

A quantum state will evolve over time such that

$$|\psi(t)\rangle = \exp(-i\hat{H}t)\,|\psi(0)\rangle \tag{71}$$

where $|\psi(0)\rangle$ is the time-independent initial quantum state and $\hat{H}$ is the time-invariant lab frame Hamiltonian of the system. We now want to switch to a new reference frame that is rotating about the z-axis. We can define this state as

$$\begin{aligned} |\tilde{\psi}(t)\rangle &= \exp(+i\omega_r\hat{I}_z)\,|\psi(t)\rangle \\ &= \hat{R}_z(-\omega_r t)\,|\psi(t)\rangle \\ &= \hat{R}_z(-\omega_r t)e^{-i\hat{H}t}\,|\psi(0)\rangle \end{aligned} \tag{72}$$

with $-\omega_r$ denoting the frequency at which our new reference frame is rotating and the notation $\hat{R}_z(a) = \exp(-ia\hat{I}_z)$. The rotation operator $\hat{R}_z$ commutes with the Hamiltonian $\hat{H}$ and thus expression (72) is a solution to the time-dependent Schrödinger equation (TISE). Hence by inserting (72) in the TISE [16]:

$$\frac{d\,|\tilde{\psi}(t)\rangle}{dt} = -i[\hat{R}_z(-\omega_r t)\hat{H}\hat{R}_z(\omega_r t) - \omega_r\hat{I}_z]\,|\tilde{\psi}(t)\rangle\,, \tag{73}$$

we obtain the Hamiltonian in the rotating frame $\tilde{H}$ in terms of the frequency of the rotation and the lab frame Hamiltonian:

$$\tilde{H} = \hat{R}_z(-\omega_r t)\hat{H}\hat{R}_z(\omega_r t) - \omega_r\hat{I}_z. \tag{74}$$

We now want to find out what our single-qubit Hamiltonian (22) is in a rotating frame of reference. The sinusoidal terms in this Hamiltonian can be rewritten such that it is now in terms of the $\hat{I}_x$ and $\hat{I}_z$ operators using the following identity

$$\begin{aligned} \cos(\mathrm{a})\hat{I}_x + \sin(a)\hat{I}_y &= \frac{1}{2}\begin{pmatrix} 0 & cos(a) - i\sin(a) \\ \cos(a) + i\sin(a) & 0 \end{pmatrix} \\ &= \frac{1}{2}\begin{pmatrix} 0 & \exp(-ia) \\ \exp(+ia) & 0 \end{pmatrix} \\ &= \frac{1}{2}\begin{pmatrix} \exp(-\frac{1}{2}ia) & 0 \\ 0 & \exp(+\frac{1}{2}ia) \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} \exp(+\frac{1}{2}ia) & 0 \\ 0 & \exp(-\frac{1}{2}ia) \end{pmatrix} \\ &= \exp(-i\mathrm{a}\hat{I}_z)\hat{I}_x\exp(+ia\hat{I}_z) \\ &= \hat{R}_z(a)\hat{I}_x\hat{R}_z(-a). \end{aligned} \tag{75}$$

Thus, the Hamiltonian (22) can be re-written as

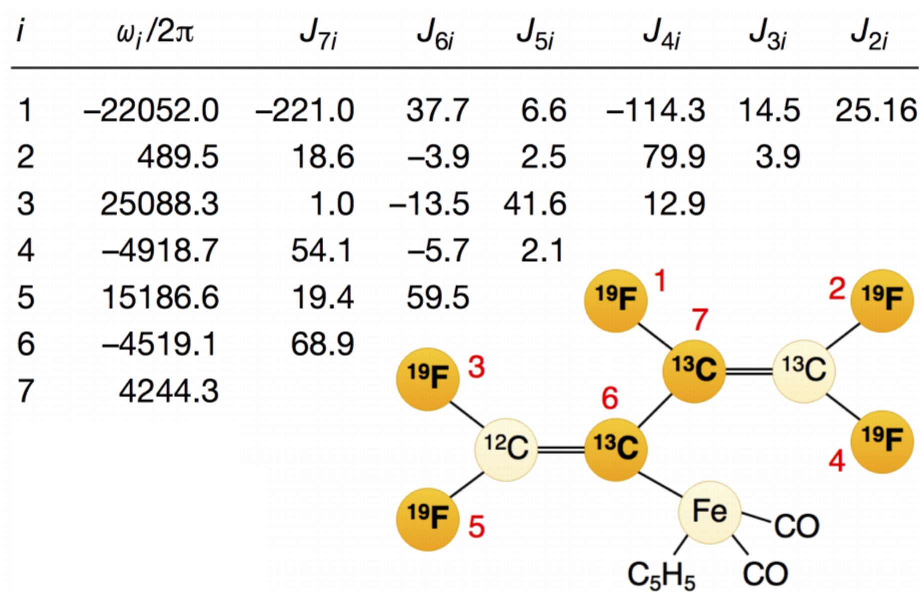
$$\hat{H} = \omega_0\hat{I}_z + \omega_n\hat{R}_z(\omega_r t + \Phi)\hat{I}_x\hat{R}_z(-\omega_r t - \Phi). \tag{76}$$

We can now insert the lab frame Hamiltonian into equation (74). If we considering the fact all $\hat{I}_z$ operators commute with each other and $\hat{R}_z(-\omega_r t)\hat{R}_z(\omega_r t) = \mathbb{1}$, we acquire the following expression for the Hamiltonian in the rotating reference frame:

$$
\begin{aligned}
\tilde{H} &= \hat{R}_z(-\omega_r t)\omega_0 \hat{I}_z \hat{R}_z(\omega_r t) + \omega_n \hat{R}_z(-\omega_r t)\hat{R}_z(\omega_r t + \Phi)\hat{I}_x \hat{R}_z(-\omega_r t - \Phi)\hat{R}_z(\omega_r t) - \omega_r \hat{I}_z \\
&= \hat{R}_z(-\omega_r t)\hat{R}_z(\omega_r t)\omega_0 \hat{I}_z + \omega_n \hat{R}_z(-\omega_r t)\hat{R}_z(\omega_r t)\hat{R}_z(\Phi)\hat{I}_x \hat{R}_z(-\Phi)\hat{R}_z(-\omega_r t)\hat{R}_z(\omega_r t) - \omega_r \hat{I}_z \\
&= \omega_0 \hat{I}_z + \hat{R}_z(\Phi)\hat{I}_x \hat{R}_z(-\Phi) - \omega_r \hat{I}_z \\
&= \omega_0 \hat{I}_z - \omega_r \hat{I}_z + \hat{R}_z(\Phi)\hat{I}_x \hat{R}_z(-\Phi) \\
&= [\omega_0 - \omega_r]\hat{I}_z + \omega_n[\cos(\Phi)\hat{I}_x + \sin(\Phi)\hat{I}_y].
\end{aligned}
\tag{77}
$$

Equation (77) can now be used in Section 3.1 when considering the spin-$\frac{1}{2}$ nuclei used in NMR techniques.

6.7 Structural Properties of the Quantum Computer Molecule



i	$\omega_i/2\pi$	J_{7i}	J_{6i}	J_{5i}	J_{4i}	J_{3i}	J_{2i}
1	–22052.0	–221.0	37.7	6.6	–114.3	14.5	25.16
2	489.5	18.6	–3.9	2.5	79.9	3.9	
3	25088.3	1.0	–13.5	41.6	12.9		
4	–4918.7	54.1	–5.7	2.1			
5	15186.6	19.4	59.5				
6	–4519.1	68.9					
7	4244.3						

Figure 15: Structural Properties of the molecule used in our NMR quantum computer simulation taken from [6]. In the bottom right, a schematic of the perfluorobutadienyl iron complex molecule used in the NMR experiment is included with numbers labelling the spin-$\frac{1}{2}$ nuclei used for our 7 qubits. The table gives Larmor frequency $\omega_0/2\pi$ values at 11.7 T; the strength of the constant magnetic field applied across the molecule. The table also gives the J-coupling values between each of the spin-$\frac{1}{2}$ nuclei in our system. Both the $\omega_0/2\pi$ and J-coupling values are given in units of [Hz].

References

[1] A. Turing, 'On computable numbers, with an application to the Entscheidungsproblem', *Proceedings of the London Mathematical Society*, Ser. 2, Vol. 42, (1937).

[2] H. Buhrman, R. Cleve, A. Wigderson, 'Quantum vs. Classical Communication and Computation', *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pg 63-68, (1998).

[3] P. Shor, 'Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer', *SIAM J. Comput.*, 26(5), 1484–1509, (1995). `https://arxiv.org/abs/quant-ph/9508027`, Accessed: March 2018.

[4] S. Cook, 'The Millennium Prize Problem', *Clay Mathematical Institute*, (2000)

[5] R. L. Rivest, A. Shamir, L. Adleman, 'A method for obtaining digital signatures and public-key cryptosystems', *Communications of the ACM*, Vol 21, pg 120-126, (1978).

[6] Vandersypen et. al, 'Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance', *Nature*, Vol 414, pg 883–887, (2001).

[7] B. P. Lanyon et. al, 'Experimental Demonstration of a Compiled Version of Shor's Algorithm with Quantum Entanglement',*Phys. Rev. Lett.*, vol 99, id: 250505, (2007).

[8] E. Lopez et. al, 'Experimental realization of Shor's quantum factoring algorithm using qubit recycling', *Nature Photonics*, Vol 6, pg 773, (2012).

[9] Yuan-yuan Zhao et. al, 'Experimental realization of generalized qubit measurements based on quantum walks', *Phys. Rev. A*, Vol. 91, id: 042101, (2015).

[10] J. A. Jones, 'Quantum Computing and Nuclear Magnetic Resonance', *Prog. Nucl. Magn. Reson. Spectrosc.*, Vol 38, pg 325–360, (2001).

[11] E. Gerjuoy, 'Shor's factoring algorithm and modern cryptography. An illustration of the capabilities inherent in quantum computers', *American Journal of Physics*, vol 73, pg 521 (2005) `https://arxiv.org/pdf/quant-ph/0411184.pdf`, Accessed: March 2018.

[12] Wikipedia, `https://en.wikipedia.org/wiki/Modular_arithmetic`, Date Accessed: March 2018.

[13] T.Moore, 'On the least absolute remainder Euclidean algorithm', *Bridgewater State College*, (1990). Accessed at: `http://vc.bridgew.edu/cgi/viewcontent.cgi?article=1010&context=math_compsci_fac`, Date: March 2018.

[14] B. Schumacher, 'Quantum Coding', *Phys. Rev.*, A 51, pg 2738- 2747, (1998).

[15] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, (2010).

[16] R. Shankar, 'Principles of Quantum Mechanics', *Plenum Press*, (2011).

[17] J. Jones, 'Quantum Information', *Oxford University*, (2010). Accessed at: `https://nmr.physics.ox.ac.uk/oxonly/C2/partIAasnotes.pdf`, Date accessed: March 2018.

[18] J. Jones, 'Quantum Logic Gates', *Journal of Magnetic Resonance*, Vol. 135, pg 353–360, (1998).

[19] Herbert Schildt, 'Java: A Beginner's Guide, Seventh Edition', *Oracle Press*, (2011).

[20] QTiPlot Handbook, `http://www.qtiplot.com/doc/manual-en/x7232.html`, Accessed: March 2018.

[21] D. Candela, 'Undergraduate computational physics projects on quantum computing', *American Journal of Physics*, Vol. 83, pg688, (2015).

[22] J .Larmor, 'On the theory of the magnetic influence on spectra; and on the radiation from moving ions', *Phil Mag*, S.5, Vol 44, pg 503-512, (1897).

[23] M. H. Levitt, 'Spin Dynamics: Basics of Nuclear Magnetic Resonance', *Wiley*, (2001).

[24] R.Freeman, 'Spin choreography : basic steps in high resolution NMR', *Spektrum*, Oxford, 143-188, (1997).

[25] G. Moore, 'Cramming more components onto integrated circuits', *Electronics*, Vol 38, No.8, (1965).

[26] A. Carlini, 'Quantum Brachistochrone', *arXiv*, Accessed at: `https://arxiv.org/abs/quant-ph/0511039`, Date: March 2018.

[27] C.Hermans, 'The Brachistochrone Problem', *Delft University of Technology*, (2017). Accessed at: `https://repository.tudelft.nl/islandora/object/uuid%3A3bc36b6d-eab1-4d20-afd2-1a1e0cc47a0f`, Accessed: Jan 2018.

[28] J. Jones, 'Quantum Computing and Nuclear Magnetic Resonance', *PhysChemComm*, Vol. 4, pg 49-56, (2004).

[29] T. Monz, et. al, 'Realization of a scalable Shor algorithm', *Science*, Vol. 351, Iss. 6277, pg. 1068-1070, (2016).

[30] D.J. Reilly, 'Engineering the quantum-classical interface of solid-state qubits', *NPJ Quantum Information*, Vol. 1, Article no: 15011, (2015).

[31] M. S. Tame, et. al, 'Experimental Realization of a One-Way Quantum Computer Algorithm Solving Simon's Problem', *Phys. Rev. Lett. 113*, id: 200501, (2014).

[32] A. Carlini, et. al, 'Time optimal quantum evolution of mixed states', *J. Phys. A: Math. Theor.*, Vol. 41, id: 045303, (2008).

[33] S. Fritzsche and M. Siomau, 'Quantum computing with mixed states', *Eur. Phys. J. D*, Vol. 62, pg 449, (2011).