Swansea University
College of Science, Department of Physics

final year theoretical project

One-Way Quantum Computation: An Introduction to Measurement Based Quantum Computation

Muhammad R.B.E. Noerrahman

under the supervision of
Dr. Markus Müller

Abstract

The laws of quantum mechanics have exposed a whole new canvas for computation. Quantum computation allows us to exploit the vast Hilbert space for computation. The quantum circuit model has been extensively studied in quantum computing. In 2001, A new model of quantum computation was proposed by Briegel and Raussendorf, where quantum computation is implemented by only applying single qubit measurements in a cluster state. This paper aims to introduce both one-way quantum computation and the general notion of quantum computation. At the end we will see whether one-way quantum computation is a viable model of universal quantum computation, by comparing one-way quantum computation model with quantum circuit model.

Contents

List of Figures

1 Introduction

'Every existing general model of computation is effectively classical', stated David Deutsch in 1985, such that at any instant the state of the information is measurable [1]. 30 years later today, the statement still holds true in general. Indeed, computational power has exponentially increased as predicted by Moore's law accordingly, but Moore's law simply comes from the observation of number of transistors could be integrated into a single chip through the years, neglecting the physical factor of the transistors themselves. As they become smaller and smaller, quantum effects have become more and more imminent. Even at this point, quantum effects have started affecting the function of these electronic devices [3]. It is true that current computers do operate under the laws of quantum mechanics, as the behaviour of all physical systems are governed by the law of quantum mechanics. Yet it is wrong to assume that current computers are quantum computers, as these laws are only exploited at hardware level (e.g. transistors) but the internal state of the computers do not exploit the operations and transformations of quantum mechanics [2].

Revisiting the statement made at the start, the fact that every current computation model allows measurement of its internal state suggests that these computational models follow the same mathematical framework as classical physics, such that the state of a physical system is deterministically measurable at any instance. Quantum mechanics has proven this to be untrue. Physical systems are governed by the non-deterministic, randomly probabilistic or stochastic quantum measurement theory. In simple words, these computation models still adhere to the *false* framework of physics. Classical computers have shown some difficulty in simulating general quantum systems, due to the fact that complex numbers needed to describe these systems generally grow exponentially [3].

Even without touching on the subject of quantum mechanics, naturally there are problems that classical computers simply cannot solve. From computational complexity theory, one of the most basic *complexity classes* is **P**. The **P** class consist of problems that are solvable in polynomial time, such that the running time of the algorithm run on a classical computer that solves that problem grows polynomially according to input size. Problems that are not in **P** class tend not to be solvable in polynomial time and thus suggests that they are *almost* unsolvable in classical computers due to the extremely

long running time or no polynomial time solution [4, 7]. One of such problems, aside from quantum mechanical problems, was turbulence simulation, as suggested by Shor [4].

The limitations that classical computation possess was the key to a new paradigm of computing, where the mathematical framework is based upon quantum mechanics. This was the realisation that quantum computing could be the future of computing. To add to that, once quantum computers are fully realised, we could truly see quantum mechanics working in our everyday life.

2 Aims and Objectives

The main aim of the project is to show that one-way quantum computation or measurement-based computation is a viable framework for quantum computing and in application can possibly be a more practical implementation of quantum computation in comparison to quantum circuit model.

Quantum computing is a product of the intersection between quantum mechanics and theoretical computer science. Before converging into the realm of one-way quantum computation or measurement-based quantum computation, fundamental ideas in quantum computing must first be expanded, such as how a qubit behaves, the mathematical concept of qubit itself and how quantum computing is conceptually different from classical computing. The laws and rules of quantum mechanics are still applied, therefore a complete understanding of quantum computing also requires fluency with quantum mechanical laws and behaviours.

Currently, quantum circuit model is the widely used framework of the quantum computation. To understand the theory of quantum computation gaining a firm grasp of how quantum circuit model operates is a crucial first step in the project. In analogy to classical computing is built upon classical circuit, quantum computing is constructed from quantum circuits consisting of wires and quantum gates [3]. Hence, understanding the action of quantum circuits is the main first step. With a better perspective on the underlying mathematical concept of quantum circuitry, any quantum circuit can be simplified into a set of unitary operators acting on a quantum state,

thus transforming the input quantum state.

Moving towards measurement based quantum computing (MBQC), with more understanding of the operations of quantum gates, the next logical step is to break these gates into their simplest unitary operations and therefore enables the replication of the operations with combination of single bit rotations. Applying quantum measurement theory, qubit rotations could be achieved by measuring qubits in a certain rotational basis, up to a correction factor. Euler's rotational theorem plays a major role in this process, of combining these rotations to possibly replicate the operation of a quantum gate.

Combining together all the ideas and understanding. A better picture of how a MBQC system behaves and operates can be drawn. Then, a side-by-side comparison of MBQC and circuit model can be made.

3 Foundations of Quantum Computation

3.1 Qubit

In 1983, Feynman put forward the idea of a quantum mechanical computer in effort to study the limitations of classical computers by asking how small can you possibly make a computer [5]. Two years later David Deutsch introduced a quantum model for computation, which consisted of a finite processor and an infinite memory of which only finite portion is used, with additional operations on top of the classical (Turing) operations. These operations are unitary transformations of a single two-dimensional Hilbert space [1]. Today, the widely used model for quantum computation is the quantum circuit model.

Analogous to classical computing and its fundamental element, bits, quantum computing relies on quantum bits or qubits. As DiVincenzo stated in his article, qubits can be simplified as classical bits shrunk to quantum (atomic) scale, defined by wavefunctions [7]. As classical bits could take the value of 1 or 0, qubits could be in a state of 1 or 0 or a superposition of both. Unlike in classical bits where the state of bits are specified by the current or voltage input, qubits can defined by the polarisation of photons or the spin of

a particle [9]. Below are the basic qubit states for quantum computing, also known as *computational basis state* [3]. The states $|0\rangle$ and $|1\rangle$ are defined as,

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad\qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{1}$$

And as stated earlier, qubits can exist in a superposition of both of these basic states, defined by the wavefunction,

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle. \tag{2}$$

Where α and β are complex numbers. Thus the general state of a qubit is a vector in a two-dimensional complex vector space or *Hilbert space* and the states $|0\rangle$ & $|1\rangle$ form an orthonormal basis to this vector space [3, 10, 9]. Similar to the quantum mechanical concept of wavefunction, the values of α and β define the *amplitudes* of the wavefunction. From quantum measurement theory, it is understood that the values of the amplitudes define the probability of the state, such that $|\alpha|^2+|\beta|^2 = 1$, where $|\alpha|^2$ is the probability of measuring the qubit in state $|0\rangle$ and $|\beta|^2$ is the probability of measuring the state in $|1\rangle$.

Aside from the numerical concept of a qubit, a geometric representation of a qubit has proven to be crucial in understanding rotational operations of a qubit. From the amplitude identity of $|\alpha|^2 + |\beta|^2 = 1$, Eq. 2 can be rewritten as [3],

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

The values of θ and ϕ are real and they define a point on the three-dimensional spherical qubit representation, namely known as *Bloch sphere*, as shown below,

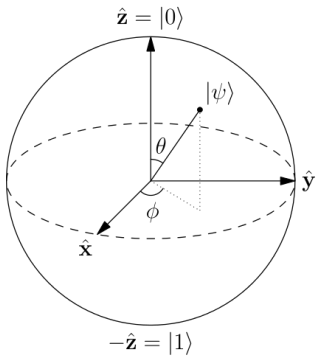


Figure 1: Bloch sphere representation of a qubit.[1]

Picturing a single qubit as in such geometrical format is useful in order to illustrate the process of unitary rotation of a single qubit, which will be the backbone of MBQC itself, as it relies upon the connecting single qubit rotations in a defined pattern to create a single qubit arbitrary rotation around a Bloch sphere.

Naive observation of the Bloch sphere could lead to the deduction that due to the infinite number of points on the sphere means that an infinite number of information could be stored into a qubit [3]. Counter-intuitively, a qubit could only represent an infinite number of informations as long as *no measurement* of its state is applied. But once a qubit is measured, it can *only* represent a single state or represent a single unit of information. This is due to the quantum mechanical nature of a qubit, such that a quantum system could exist in all of its states, but will collapse to a single state when measured or perturbed by a non-quantum system.

In classical computing, computation are done on strings of bits. Similar to strings of bits in classical computers, qubits could also exist in a 'string' form, yet its property differs from that of classical string of bits. A classical string of n bits could exist in one definite state of the possible 2^n *boolean* (true or false; 0 or 1) states of $x = 0000...0$ through 1111...1 [7, 9]. A string of n qubits is described by [9],

[1] *Bloch sphere* [Online image] (2012). Retrieved April 19, 2016 from https://commons.wikimedia.org/wiki/File:Bloch_Sphere.svg

$$|\psi\rangle = \sum_{x=0000\ldots0}^{1111\ldots1} c_x|x\rangle. \tag{4}$$

Where c_x is a complex number which describes the amplitude of state $|x\rangle$. In accordance to Eq. 2, c_x fulfils the condition of amplitudes of a quantum state, such that $\sum_x |c_x|^2 = 1$ and each state $|x\rangle$ has the probability of $|c_x|^2$. A state of n qubits could therefore exist in a superposition of all corresponding classical states described earlier. Looking back at the dimensionality of a qubit, a state of n qubits has a dimensionality of 2^n in the Hilbert space, such that an n-qubit state is described by 2^n number of amplitudes. This is where the rifts between quantum computing and classical computing exist, the exponentially large dimensionality of said space in comparison to the linearity of the expansion of space as the size of a classical system increases [3, 9, 7]. For example, to store all the complex coefficients or *amplitudes* a quantum system of just n = 50 qubits, would require 2^{50} or about 1.26×10^{15} unit of memory. This extremely large dimensionality of space holds a large potential for the improvement of computational power [3].

3.2 Quantum Circuit Model

Any model of computation, quantum or classical, relies heavily on mathematics. Circuits, aside from the actual physical realisation of circuits in classical computation, is a model of computation device. Classically, circuits are made of wires that carry information in the form of bits. While *logical gates*, such as NOT, AND, NAND, OR and XOR perform simple computational task on the bits [3].

Similarly, quantum circuit model is a way of thinking about quantum computations in the same terms as classical circuits. In simpler terms, it is in a way, classical representation of quantum computation. The fundamental basis of quantum circuit model was presented by Feynman in 1983 [5], the general model of quantum gates were finalised in 1995 [11]. To this day, quantum circuit model still remains as major contributor to the study of quantum computation. Thus, quantum circuit model has been the foundation of quantum computation, hence to study a new model of quantum computation, firm grasp on quantum circuit model is essential.

The three main parts of a quantum circuit are quite comparable to those of a classical circuit. Quantum circuits are based on *wires*, which is a representation of the flow of information in the form of qubits with respect to time, but does not necessarily corresponds to physical wire [3] and *quantum gates*, which transform qubits by applying unitary operation and also *measurements*, as the state of any qubit is not defined until measured and that measurement also changes the state of entangled qubits. The last two will be very important in MBQC.

3.2.1 Unitary Operations

To fully understand the operations of quantum gates is crucial in the process of replicating these operations with measurements only in the later stage. Single qubit operations are the simplest operations in quantum computation. These operations are defined by 2×2 unitary matrix operations defined as [11],

$$U = \begin{bmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{bmatrix}. \tag{5}$$

The main operations are the Pauli matrices, $\sigma_x = X, \sigma_y = Y, \sigma_z = Z$ [3],

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{6}$$

These operators play a major role in quantum computation, as they act similar to simple logic gates in classical circuit.

$$X|0\rangle = |1\rangle \qquad X|1\rangle = |0\rangle \tag{7}$$
$$Z|0\rangle = |0\rangle \qquad Z|1\rangle = -|1\rangle \tag{8}$$
$$Y|0\rangle = i|1\rangle \qquad Y|1\rangle = -i|0\rangle \tag{9}$$

X operator, which bears similar operation as classical NOT gate, is also known as *bit-flip* operator and the Z operator is known as the *phase-flip* operator. Y is a combination of both, with a factor of i. On top of these, Nielsen and Chuang [3] also listed three other important operators along with the Pauli matrices,

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad S = \begin{bmatrix} 1 & 0 \\ 0 & -i \end{bmatrix} \qquad T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix}. \tag{10}$$

Where H is a *Hadamard* operator, S is a *Phase* gate and T is a $\frac{\pi}{8}$ gate. Throughout this paper, H gate will be used extensively. H operator is a rather interesting operator. It is the combination of the Z and X operators with a factor of $\frac{1}{\sqrt{2}}$,

$$H = \frac{X+Z}{\sqrt{2}} = \frac{1}{\sqrt{2}} \cdot \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{11}$$

H gate also transforms a qubit by applying a $\frac{\pi}{2}$ rotation around the x-z plane of the Bloch sphere, so that applying it to a qubit in the computational basis would rotate the qubit to become an x-basis qubit,

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+_x\rangle \qquad H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-_x\rangle \tag{12}$$

Aside from quantum logical gates, qubits can also go through rotations, defined by,

$$R_P(\theta) = e^{\left[i\frac{\theta}{2}P\right]}, P = \{X, Y, Z\}, \tag{13}$$

where the rotational angle is defined by θ and P corresponds to the Pauli operators defined in Eq. 6.

Physically, these unitary operations are the result of time evolution of the time-dependent Schrödinger's equation,

$$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = \mathcal{H}|\psi\rangle \tag{14}$$

where $\mathcal{H}$ is the *Hamiltonian* operator of the Schrödinger's. This suggests that the solution to the Schrödinger's equation is,

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

therefore, a time-dependent unitary operation U_t acting on a wavefunction can be defined as,

$$U_t = e^{-i\mathcal{H}t/\hbar} \tag{16}$$

as *Hamiltonian* operator describes the *Energy eigenstates* of the wavefunction E [3, 10],

$$\mathcal{H} = E\mathbf{P} \tag{17}$$

P represents any of the Pauli matrices, defining the eigenstates of the energy. The exponential in Eq. 16, can be rewritten as $e^{-iE\mathbf{P}t/\hbar}$ and $Et/\hbar = \Phi_t$, thus U_t can be written as,

$$U_t = e^{-i\Phi_t \mathbf{P}} \tag{18}$$

Taylor expansion of the equation above gives,

$$U_t(\Phi_t) = \left(1 - \frac{\Phi_t^2}{2} + \frac{\Phi_t^4}{24} + ...\right)\mathbb{1} - i\left(\Phi_t - \frac{\Phi_t^3}{6} - \frac{\Phi_t^5}{120} + ...\right)\mathbf{P} \tag{19}$$

where $\mathbb{1}$ is the identity matrix. Using trigonometric identities this simplifies to,

$$U_t(\Phi_t) = \cos\Phi_t\mathbb{1} - i\sin\Phi_t\mathbf{P} \tag{20}$$

in matrix format,

$$U_t(\Phi_t) = \begin{bmatrix} \cos\Phi_t - i\sin\Phi_t\mathbf{P}_{00} & \cos\Phi_t - i\sin\Phi_t\mathbf{P}_{01} \\ \cos\Phi_t - i\sin\Phi_t\mathbf{P}_{10} & \cos\Phi_t - i\sin\Phi_t\mathbf{P}_{11} \end{bmatrix}. \tag{21}$$

$\mathbf{P}_{nm}$ represent the elements of the Pauli matrix. This suggests that unitary operations described previously are special cases of $U_t(\Phi_t)$–up to a global phase, which is known to be unimportant in computation [3]; or a combination of different $U_t(\Phi_t)$ concatenated. For example, an X operation is a solution of this matrix when $\Phi_t = \frac{\pi}{2}$ and $\mathbf{P} = X\, or\, \sigma_x$. Going back to the *Bloch sphere* representation, $U_t(\Phi_t)$ describes a Φ_t rotation on a qubit around the axis of **P**.

3.2.2 Quantum Gates

In quantum circuits, unitary operations are simply represented as,

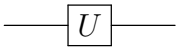
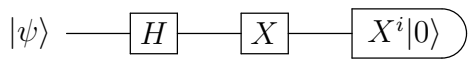


Figure 2: A quantum gate of single qubit unitary operation U.

where U is any of the operations defined in Eq. 6, 10 and 13. This is an example of a *single qubit gate*. Below is an example of a simple, single qubit circuit.



This circuit corresponds to the operation $XH|\psi\rangle$ and a measurement in the *computational basis* or *z-basis* at the end. To ease the process of showing which basis the qubit is measured, the following gate symbol will be used to define measurement,

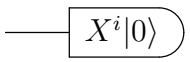


Figure 3: Z-basis measurement in a quantum circuit.

where $i \in \{0, 1\}$ such that when the qubit is in $|0\rangle$, $i = 0$, and when the qubit is in $|1\rangle$, $i = 1$ or $X|0\rangle$, as defined in Eq. 7. Different measurement basis or projectors could be applied to the measurement, such as $Z^i|+\rangle$, since Z operator acts similar to a bit-flip operator to *x-basis* qubits. In quantum mechanical term, this is defined as applying a projective measurement on a quantum state [3],

$$\frac{P_m|\psi\rangle}{\sqrt{p(m)}}. \tag{22}$$

Where $P_m = |\psi_{proj}\rangle\langle\psi_{proj}|$, the measurement operator, with $|\psi_{proj}\rangle$ as the state after the measurement and $p(m) = \langle\psi|P_m|\psi\rangle$, is the probability of measuring the qubit in that projected state. For example, when measuring in the computational basis, the projected states are simply the eigenstates of Z, $|0\rangle$ and $|1\rangle$. Similarly in the other general basis X and Y. Qubit going through measurement will be reduced by Eq. 22 to a measurement outcome state.

Quantum gates are not only limited to only single qubit operations or gates, Barenco et al. suggest that there could be up to n-qubit gates, simulated by a number of n'-qubit gates, where $n' < n$ [11]. So a large n-qubits

gate can be simulated with smaller gates. In this project, the extent of our concern are only to 2-qubit gates.

A single qubit unitary rotation is a 2-dimensional operator, thus a 2-qubit gate is a 2^2-dimensional operator. In 1995, Barenco et al. introduced a generalised format of n-qubit gates [11], $\wedge_m(U)$, where m defined as $(m+1)$-qubit or $m = n - 1$, where n is the number of qubits the gate operates on. $\wedge_m(U)$ is defined by the matrix,

$$\wedge_m(U) = \begin{pmatrix} \mathbb{1}_{m+1} & . & . & 0 \\ & . & & \\ & & . & \\ 0 & . & . & U \end{pmatrix}, \tag{23}$$

where $\mathbb{1}_{m+1}$ is the identity matrix in $m+1$ dimension and U is a single qubit unitary operation. In the case of a 2-qubit gate of 2^2-dimensional operator, where $m = 1$,

$$\wedge_1(U) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & U_{00} & U_{01} \\ 0 & 0 & U_{10} & U_{11} \end{pmatrix}, \tag{24}$$

this is also known as a *controlled*-U gate. In quantum circuit model, this is represented as,

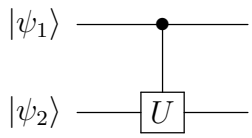


Figure 4: Controlled-U: 2-qubit quantum gate.

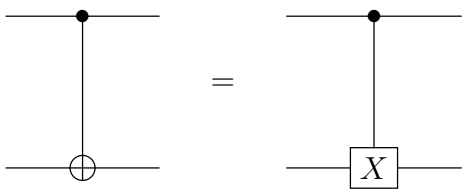
It operates on a 2-qubit system, $|\psi_1\rangle|\psi_2\rangle$, on the basis that the first, qubit $|\psi_1\rangle$ *controls* the operation on the second qubit $|\psi_2\rangle$. The operation is defined as,

$$\text{If } |\psi_1\rangle = |0\rangle, |0_1\rangle \otimes |\psi_2\rangle \qquad \text{If } |\psi_1\rangle = |1\rangle, |1_1\rangle \otimes U|\psi_2\rangle. \tag{25}$$

So that the unitary U on the second qubit is only applied when $|\psi_1\rangle = |1\rangle$. One of the main examples of a 2-qubit gate is the CNOT gate or U_{CN}. The gate U_{CN} is defined by $\wedge_1(X)$,

$$U_{CN} = \wedge_1(X) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \tag{26}$$

represented by the gate,



The logical property of a U_{CN} is similar to that of classical XOR gate, where the bit-flip operation on the second qubit is controlled by the first qubit. If the first qubit (*control* qubit, $|\psi_c\rangle$), is in $|0\rangle$ state, nothing happens to the second qubit (*target* qubit, $|\psi_t\rangle$). But if $|\psi_c\rangle$ is in $|1\rangle$ state, X operator will be applied to $|\psi_t\rangle$, thus flipping the qubit state. In *bra-ket* notation, this operation is defined as,

$$U_{CNOT} = |0\rangle_c\langle 0|_c \otimes \mathbb{1}_t + |1\rangle_c\langle 1|_c \otimes X_t, \tag{27}$$

This is just a small example of a multiple qubit gate, there are numerous more ways of implementing multiple qubit gates.

With all of these gates, there are limitless possibility of more complex quantum circuits to solve quantum algorithms

3.2.3 Universality of Quantum Gates

According to DiVincenzo[7], any unitary transformation in the 2^n-dimensional Hilbert space on n-qubits can be decomposed into a set of *universal gates* or *universal operations* applied schematically to the n-qubits and that any unitary operations on can be thought as *universal gates* of quantum computation. Therefore, for any any single qubit transformations, all qubit rotation

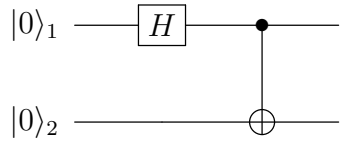
gates obviously belong in the *universal gates* set. For two-qubit transformations, Barenco et al.[11] suggests that almost any single $\wedge_1(U)$ defined in Eq. 24 is universal. DiVincenzo suggests that a "2-qubit XOR" (CNOT gate) is an option for a two-qubit universal gate [7]. From this, it is clear that any unitary transformations in a n-qubits system can be constructed from only single qubit rotation gates and CNOT gates.

3.3 Measurements on Qubits

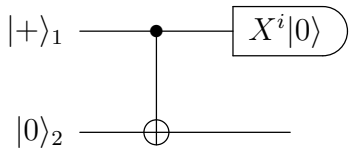
The measurement outcome of a quantum system is stochastic, meaning that measuring a qubit in a quantum circuit would provide a *non-deterministic* outcome. This is due to the intrinsic randomness of quantum mechanics.

To prove this, take the measurement outcomes of a *Bell state* $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. Bell states are special in quantum computing, as the measurements of the two qubits are completely random, yet still *correlated*, which can only be explain through *entanglement*.

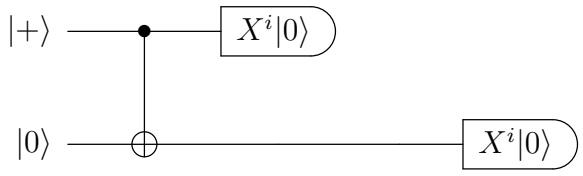
The circuit below allows us to create a Bell state from a $|0\rangle_1 \otimes |0\rangle_2$.



Applying an H gate to the first qubit changes the basis of the qubit into, $H|0\rangle = |+\rangle$, then entangling both of the qubits with a CNOT gate, $(|0\rangle_c\langle 0|_c \otimes \mathbb{1}_t + |1\rangle_c\langle 1|_c \otimes X_t)\ |+\rangle_c|0\rangle_t$, giving $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$. Measuring the first qubit in z-basis would give two outcomes, $|00\rangle$ and $|11\rangle$ with 50% probability each. Other than that, the state of the second qubit in the circuit has also been affected by the measurement, where it is no longer in a superposition of $|0\rangle$ and $|1\rangle$.



Measuring the second qubit in the same basis as the first measurement *after* the first measurement,



would give outcomes of $|0\rangle$ and $|1\rangle$ with 100% probability for $|0\rangle$ if the outcome of the first measurement was $|00\rangle$ and the same with the outcome state $|1\rangle$ if the outcome of the first measurement was $|11\rangle$. Thus giving 50% probability in total for both. Although the outcomes are random, there is still a correlation in both the first and second measurement. Yet, correlation in stochastic measurements is a special case. In this case, we also see that measurement of one qubit could affect the state and outcome of another qubit in the circuit.

With this, it is proven that measurement in quantum computation is *probabilistic*.

4 Measurement Based Quantum Computation

4.1 One-way Quantum Computation

In 2001, Raussendorf and Briegel proposed a new model of quantum computing, where information is transferred by performing an order or a pattern of single qubit measurements, thus being 'measurement based', in a certain basis alone, on a specific entangled state of large number of qubits, called *cluster state*. The outcome of the computation which consists the results of earlier measurements are passed on through onto the final qubit of the *cluster state* [6].

As the name suggests, unlike quantum circuit model, where the operations are *reversible*, due to the fact that operations in quantum circuit are *unitary*. By applying the Hermitian conjugate of the operation, $U^\dagger$, the state after operator U applied on it, $|\psi_U\rangle = U|\psi\rangle$ can be brought back to the initial state, $U^\dagger|\psi_U\rangle = |\psi\rangle$, since $UU^\dagger = U^\dagger U = \mathbb{1}$ and unitarity preserves the inner

products of the qubits [3]. In one-way quantum computation, computations on a cluster state are *non-reversible* as projective measurement on a qubit is *irreversible* and once the required measurements have been carried through the cluster state, a new cluster state must be prepared since entanglement between the qubits are destroyed due to measurements, such that the cluster state itself becomes an exhaustible physical resource of the computation [8, 12]. Thus, to repeat a certain computation, every measurements has to be repeated again on a new cluster state. Thereby as the name suggests, it is only *one-way*.

The main advantage of one-way quantum computation is that it allows *universal quantum computation* due to the fact that the computation itself does not depend on quantum gates, only measurements in different basis. From these measurements, any quantum gates could be replicated. Therefore, different algorithms only require different pattern of single qubit measurements on a specific cluster state [13].

4.2 Cluster State

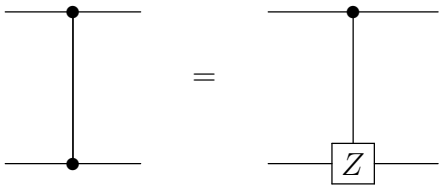
The underlining concept of a cluster state is the graph state. A graph state is an entangled state of numerous qubits, in which all the qubits are already prepared in a specific state, in most general cases the qubits are in superposition $|+\rangle$ state, an eigenstate of Pauli operator X [6], such that the initial state before entanglement looks like $|+\rangle_1 \otimes |+\rangle_2 \otimes |+\rangle_3 ... \otimes |+\rangle_n$. The entanglement interaction between the qubits are produced by applying a *Controlled-Z* or CZ operation between each qubits [12], defined by $\wedge_1(Z)$,

$$U_{CZ} = \wedge_1(Z) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{28}$$

or in a simpler *bra-ket* format,

$$U_{CZ} = |0\rangle_c\langle 0|_c \otimes \mathbb{1}_t + |1\rangle_c\langle 1|_c \otimes Z_t, \tag{29}$$

Where $_c$ and $_t$ represents on which qubit they operate on, where c and t are control qubit and target qubit respectively. The quantum gate representation of CZ is,



This operator is applied on every pair of qubits in the state [12]. The figure below gives a graphic representation of a graph state.

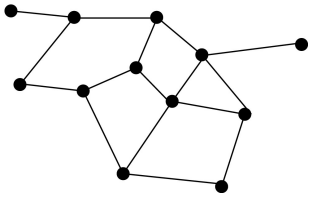


Figure 5: A graph state is made of n-qubits where each qubit is entangled to its *neighbours*. The dot represents a qubit and each of the edges represent the CZ operator applied between the two qubits.

Cluster state is defined as a graph state in an n-dimensional square grid, as shown below.

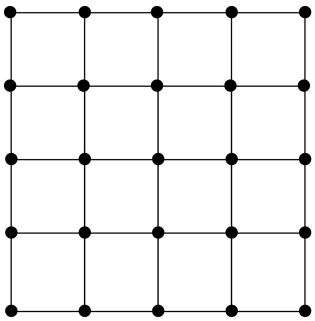


Figure 6: A cluster state is made of n-qubits where each qubit is entangled to its *specific neighbours*. This figure shows a 5×5 cluster state.

These states can be created using quantum circuit, for example a simple linear graph state of 4 qubits, such as

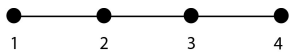
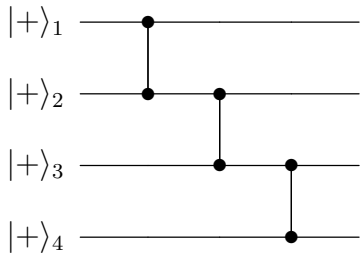


Figure 7: A 4-qubit linear graph state.

can be produced by a rather simple quantum circuit below.

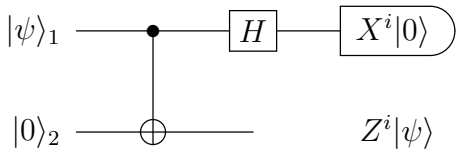


The circuit above simply applies a CZ gate between the qubits. Therefore, a larger graph or cluster state can be produced by the same method.

4.3 Single Qubit Measurements in a Cluster State

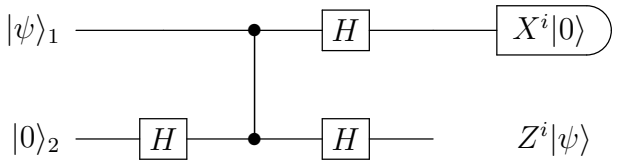
The very essence of one-way quantum computation or MBQC is measurements. Measurements allow the propagation of information and computation through the cluster state [6]. As shown in Section 3.3, measurement of a qubit in an entangled state can affect another qubit entangled to the measured qubit. Expanding the notion to a larger scale, where measuring a single qubit in a 5×5 cluster could affect the state of the other 24 entangled qubits.

How measurements could propagate information through a cluster state can be represented by looking at a single qubit teleportation circuit [14],

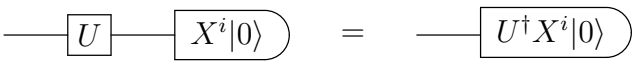


where $|\psi\rangle$ is an arbitrary input state $\alpha|0\rangle + \beta|1\rangle$. With this circuit, the arbitrary state is *teleported* to the second qubit with a *by-product* factor of Z^i which depends on the measurement outcome.

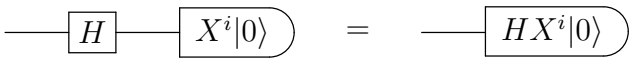
Another interesting property of a Hadamard or H gate is that it could swap X and Z gate, such that $HZH = X$ and $HXH = Z$. This allows the decomposition of the CNOT gate to a CZ gate with 2 Hadamards.



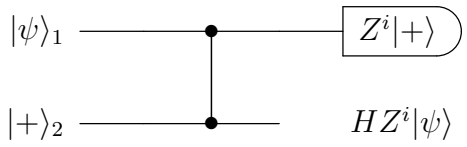
To further simplify the circuit, the two Hs on the second qubit can be 'removed' by applying one to the initial state $|0\rangle$ and the other can be *absorbed* into the outcome. On the other hand, the H on first qubit can be included in measurement with the following property. A measurement operator consists of $|\psi\rangle\langle\psi|$, as defined in Section 3.2.2. Applying a unitary U onto the projector would give, $U|\psi\rangle$, thus the measurement operator becomes $U^\dagger|\psi\rangle\langle\psi|U$. This is the same as projecting onto $U^\dagger|\psi\rangle$ [14].



Since the Hermitian conjugate of H is itself,

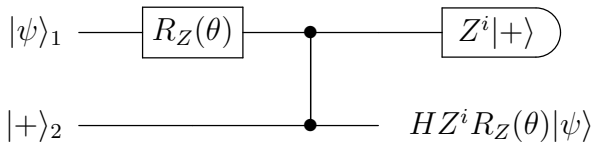


This could be further decomposed from $HX^i|0\rangle$ to $HHZ^iH|0\rangle$, using the identity between H, Z and X mentioned earlier. As $HH = \mathbb{1}$ and $H|0\rangle = |+\rangle$, this now becomes $Z^i|+\rangle$. The previous circuit can now be simplified as,

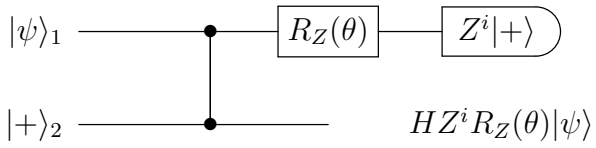


This bears a striking resemblance to the circuit previously discussed circuit for the preparation of a cluster state, that is because *it is* the circuit for a 2-qubit linear graph state with a measurement on the first qubit, if the initial arbitrary state was to be set as $|+\rangle$.

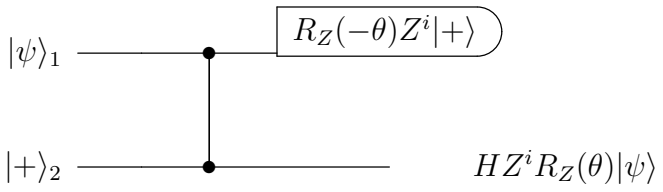
Measuring just on the standard basis has its limitation, as it does not allow the arbitrary rotation around the Bloch sphere to occur. Yet, rotation can also be replicated in measurements. Taking the previous circuit and applying a Z-rotation, $R_Z(\theta)$ on the first qubit,



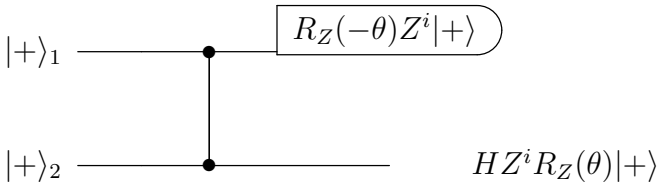
Since $R_Z(\theta)$ and Z commutes, as they are of the same Pauli matrix, Z. The rotation could be moved past the CZ gate.



Again, the Z-rotation could be absorbed into the measurement by taking its conjugate. Since $R_Z(\theta) = e^{\left[i\frac{\theta}{2}Z\right]}$, the conjugate is simply$R_Z^\dagger(\theta) = R_Z(-\theta) = e^{\left[-i\frac{\theta}{2}Z\right]}$. The circuit now becomes,



This shows that measuring in a *rotated basis* rotates the teleported qubit. If this was to be applied to a cluster state of $|+\rangle$ qubits,

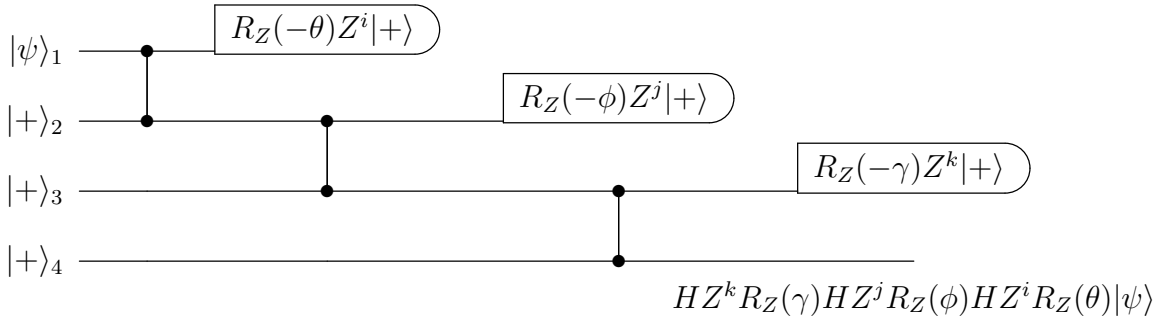


and this is how computation and operations are propagated through a cluster state.

4.4 Measurement Patterns - Arbitrary Single Qubit Rotation

Euler's theorem states that any rotation in a three-dimensional space, such as the rotation of a qubit in a Bloch sphere, could be decomposed as a product of three rotations along two orthogonal axes. This suggests that any single qubit rotation in a Bloch sphere can be decomposed into $U = R_Z(\theta)R_X(\phi)R_Z(\gamma)$ [6, 8, 12, 14]. Unlike in circuit where rotation could just be applied to a single qubit, measurement method requires 4 qubits in a graph state.

Concatenating three copies of the previously discussed circuit in Section 4.3 [14],



$$i, j, k \in \{0, 1\}$$

Thus, by just applying three measurements in rotated basis, the unitary,

$$U = HZ^kR_Z(\gamma)HZ^jR_Z(\phi)HZ^iR_Z(\theta) \tag{30}$$

is implemented. As one might have noticed, there are a series of Pauli *by-products* along with factors of Hs in between the outcome. These by-products can be permuted through, thus leaving them to the end of the operations.

4.4.1 Permutation of By-products and Adaptive Measurement

Due to the stochasticity of the by-products of the outcome of the measurements, the desired outcome state only has 1/8 probability. To have a deterministic outcome, the measurement must be made *adaptive*, meaning that the next measurement is adapted by the previous measurements [14].

First, there are two permutation rules that needs to be followed, *Pauli operator* permutation rule and *Clifford group* permutation rule [14].

Pauli permutation rule is based on the commutative relations of Pauli matrices. Take $\mathbf{p}$ and $\mathbf{P}$ as two operators, if they *commute*, then $\mathbf{pP} = \mathbf{Pp}$. If the two operators *anti-commute*, then $\mathbf{pP} = -\mathbf{Pp}$. This rule is applicable to rotations and Pauli operators. For example, $R_Z(\theta)Z = ZR_Z(\theta)$, but $R_Z(\theta)X = -XR_Z(\theta)$ which also equals to $XR_Z(-\theta)$.

Clifford group is a set of unitaries that *map Pauli matrices to Pauli matrices*, such as CNOT, CZ and H.

$$CP_1C^\dagger = P_2 \tag{31}$$

where C is a Clifford gate and P_1 and P_2 are two different Paulis. For example, $HXH = Z$. Rearranging the identity gives,

$$CP_1 = P_2C \tag{32}$$

Taking the previous example, $HX = ZH$. This is the permutation rule of a Clifford group and a Pauli.

Now applying these rules to the unitary from the measurements, defined in Eq. 30. First moving all the by-products to the end,

$$U = X^kZ^jX^iHR_Z((-1)^j\gamma)HR_Z((-1)^i\phi)HR_Z(\theta) \tag{33}$$

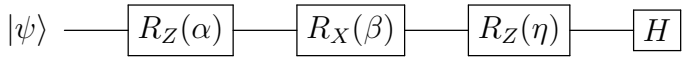
now, removing the two H in the middle by absorbing it into the second rotation,

$$U = X^kZ^jX^iHR_Z((-1)^j\gamma)R_X((-1)^i\phi)R_Z(\theta). \tag{34}$$

Measurements after the first is made adaptive, by taking the bit equivalent of the measurements, so that if the first measurement outcome is the desired state, then $i = 0$ or if not $i = 1$ and so on for every measurements that come after. Such that, taking $\theta = \alpha$, $(-1)^i\phi = \beta$ and $(-1)^j\gamma = \eta$ would give,

$$U = HR_Z(\eta)R_X(\beta)R_Z(\alpha). \tag{35}$$

which has the same function as the circuit,



.

By applying the correct angle values, any single qubit rotation can be replicated, to a global phase factor. Here, it has been proven that it is possible to apply arbitrary single qubit rotation by implementing measurements in a rotated basis.

4.4.2 Implementation on a Cluster State

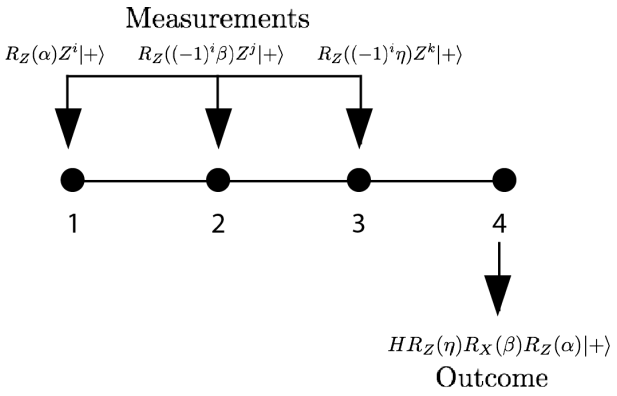


Figure 8: Shows how the measurements implemented on a 4-qubit linear graph state.

As the figure above shows, to implementing the measurements in a linear graph state requires the same pattern as shown before. For a computation with multiple operations, the scale of the linear graph state could be expanded to allow all the measurements to take place. For example a 2 consecutive operations would require a 7-qubit linear graph state, thus the outcome state from the figure above goes through three more rotations and the 7th qubit would be have the required operations acting on it.

4.5 Two-Qubit Gates in MBQC

To have the full universal gates set, a two-qubit gate is required. But as described in Section 4.2, the qubits in a cluster state are already interacted through CZ gate, which is a two-qubit gate. Thus CZ gate is built-in to the system [14]. Therefore, to achieve different two-qubit gates, measurements on individual qubits can be applied before and after the CZ gate. A possible cluster state for different two-qubit gates is shown below,

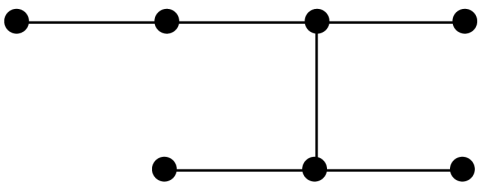


Figure 9: Possible cluster state for a two-qubit gate.

The CZ gate between the third qubit on the first linear state and the second qubit on the second linear state is what allows two-qubit gate operation to take place. As MBQC can replicate both single-qubit operations and two-qubit gates [8], MBQC is a universal quantum computation model [8, 13].

5 Simulation of Measurement Based Quantum Computation

A simple Mathematica programme was written as a simulation to the concepts introduced in MBQC and to study how errors in measurements and imperfections in the preparation of cluster state would affect the outcome. In this section, most of the pre-defined functions are not shown, to have a thorough view of the simulation, refer to Appendix 9.1.

5.1 Preparing a 4-qubit Linear Graph State

```
4 - Qubit Linear GraphState Construction

Start with |0000⟩ state

q4linear = KroneckerProduct[k0, k0, k0, k0]

{{1}, {0}, {0}, {0}, {0}, {0}, {0}, {0}, {0}, {0}, {0}, {0}, {0}, {0}, {0}, {0}}

Apply Hadamard To All and apply CZ between all

q4Linear = KroneckerProduct[Had, Had, Had, Had].q4linear

q4LinEntg = KroneckerProduct[ctrl2[Z], Id, Id].KroneckerProduct[Id, ctrl2[Z], Id].
  KroneckerProduct[Id, Id, ctrl2[Z]].q4Linear

q4LinEntg2 = 1 / 2 (KroneckerProduct[kxp, k0, kxp, k0]) + 1 / 2 (KroneckerProduct[kxp, k0, kxm, k1]) +
  1 / 2 (KroneckerProduct[kxm, k1, kxm, k0]) + 1 / 2 (KroneckerProduct[kxm, k1, kxp, k1])
```

Figure 10: Code from Mathematica to simulate the preparation of a 4-qubit linear graph state.

The simulation starts with 4 qubits in $|0\rangle$ state, $|0\rangle \otimes |0\rangle \otimes |0\rangle \otimes |0\rangle$. Applying H operator, defined as `Had` in the programme, on all the qubits to give $|+\rangle \otimes |+\rangle \otimes |+\rangle \otimes |+\rangle$. In `q4LinEntg`, CZ gates are applied between them to produce entanglement between the qubits. `q4LinEntg2` serves as a method to check if the state produced by the simulation is correct, as it should produce a state of

$$|C_4\rangle = \frac{1}{2}|+\rangle_1|0\rangle_2|+\rangle_3|0\rangle_4 + \frac{1}{2}|+\rangle_1|0\rangle_2|-\rangle_3|1\rangle_4 + \frac{1}{2}|-\rangle_1|1\rangle_2|-\rangle_3|0\rangle_4 + \frac{1}{2}|-\rangle_1|1\rangle_2|+\rangle_3|1\rangle_4. \tag{36}$$

This state was defined by Raussendorf, Browne and Briegel [8] to be the state of a 4-qubit linear cluster state.

The output of `q4LinEntg` does match the expected state, as shown below.

```
q4LinEntg = KroneckerProduct[ctrl2[Z], Id, Id].KroneckerProduct[Id, ctrl2[Z], Id].
  KroneckerProduct[Id, Id, ctrl2[Z]].q4Linear

q4LinEntg2 = 1 / 2 (KroneckerProduct[kxp, k0, kxp, k0]) + 1 / 2 (KroneckerProduct[kxp, k0, kxm, k1]) +
  1 / 2 (KroneckerProduct[kxm, k1, kxm, k0]) + 1 / 2 (KroneckerProduct[kxm, k1, kxp, k1])
```

Out[216]=

Out[217]= $\{\{\frac{1}{4}\}, \{\frac{1}{4}\}, \{\frac{1}{4}\}, \{-\frac{1}{4}\}, \{\frac{1}{4}\}, \{\frac{1}{4}\}, \{-\frac{1}{4}\}, \{\frac{1}{4}\}, \{\frac{1}{4}\}, \{\frac{1}{4}\}, \{\frac{1}{4}\}, \{-\frac{1}{4}\}, \{-\frac{1}{4}\}, \{-\frac{1}{4}\}, \{\frac{1}{4}\}, \{-\frac{1}{4}\}\}$

Out[218]= $\{\{\frac{1}{4}\}, \{\frac{1}{4}\}, \{\frac{1}{4}\}, \{-\frac{1}{4}\}, \{\frac{1}{4}\}, \{\frac{1}{4}\}, \{-\frac{1}{4}\}, \{\frac{1}{4}\}, \{\frac{1}{4}\}, \{\frac{1}{4}\}, \{\frac{1}{4}\}, \{-\frac{1}{4}\}, \{-\frac{1}{4}\}, \{-\frac{1}{4}\}, \{\frac{1}{4}\}, \{-\frac{1}{4}\}\}$

Figure 11: Output of the preparation simulation, the first output comes from `q4LinEntg` and the second output comes from `q4LinEntg2`.

5.2 Adaptive Measurements of 4-Qubit Linear Graph State

```
(*Angles to be measured*)
α = -Pi;
β = -Pi / 2;
γ = -Pi;

p0 = KroneckerProduct[proj[Rot[Z, α].kxp], Id, Id, Id];
p1 = KroneckerProduct[proj[Rot[Z, α].kxm], Id, Id, Id];
(*Projectors of the measurement*)

mR1 = stochasticOutputEntg[p0, p1, q4LinEntg2]

b1 = corrFactor[mR1, p0]
(*Binary measurement outcome,
by checking whether the probability of the outcome state and the required projector = 1,
if it is b =0, if not b=1 *)

c1 = (-1) ^ (b1)
(*Adaptive value for angle*)

p2 = KroneckerProduct[Id, proj[Rot[Z, (c1 * (β)) ].kxp], Id, Id];
p3 = KroneckerProduct[Id, proj[Rot[Z, (c1 * (β)) ].kxm], Id, Id];

mR2 = stochasticOutputEntg[p2, p3, mR1]

b2 = corrFactor[mR2, p2]

c2 = (-1) ^ (b2)

p4 = KroneckerProduct[Id, Id, proj[Rot[Z, (c2 * (γ)) ].kxp], Id];
p5 = KroneckerProduct[Id, Id, proj[Rot[Z, (c2 * (γ)) ].kxm], Id];

mR3 = stochasticOutputEntg[p4, p5, mR2]

b3 = corrFactor[mR3, p4]

c3 = (-1) ^ (b3)

psicheck = KroneckerProduct[Rot[Z, α].If[b1 == 0, Id, Z].kxp, Rot[Z, (c1 * (β))].If[b2 == 0, Id, Z].kxp,
  Rot[Z, (c2 * (γ))].If[b3 == 0, Id, Z].kxp,
  If[b3 == 0, Id, X].If[b2 == 0, Id, Z].If[b1 == 0, Id, X].Had.Rot[Z, -γ].Rot[X, -β].Rot[Z, -α].kxp]

Abs[ConjugateTranspose[psicheck].(mR3)] ^ 2
(*Check if outcome is the same as expected,
if {{1}} means its correct*)
```

Figure 12: Mathematica simulation of both randomness in measurement outcome and adaptive measurement patterns in MBQC.

This simulation takes in the prepared graph state from the previous section and applies the measurement patterns described in Section 4.4. The rotation operations simulated here are,

$$U = X^k Z^j X^i H R_Z[(-1)^j(-\pi)] R_X[(-1)^i(-\frac{\pi}{2}))] R_Z[-\pi] \tag{37}$$

as defined by α, β, γ in the simulation. These measurements should give an outcome state of

$$H R_Z(\pi) R_X(\pi/2) R_Z(\pi) |+\rangle. \tag{38}$$

Here, `p`n are the projectors of each measurement, while `b`n represents the bit equivalent of each measurement outcome and `c`n defines the measurement correction value. `mR`n holds the state after the measurement. The function `psicheck` serves as a checking method, such that if the simulation produces the desired outcome, $|\langle\psi_{out}||\psi_{check}\rangle|^2 = 1$. The function `stochasticOutputEntg` is pre-defined to apply the projectors defined by `p`n randomly, thus giving random outcome.

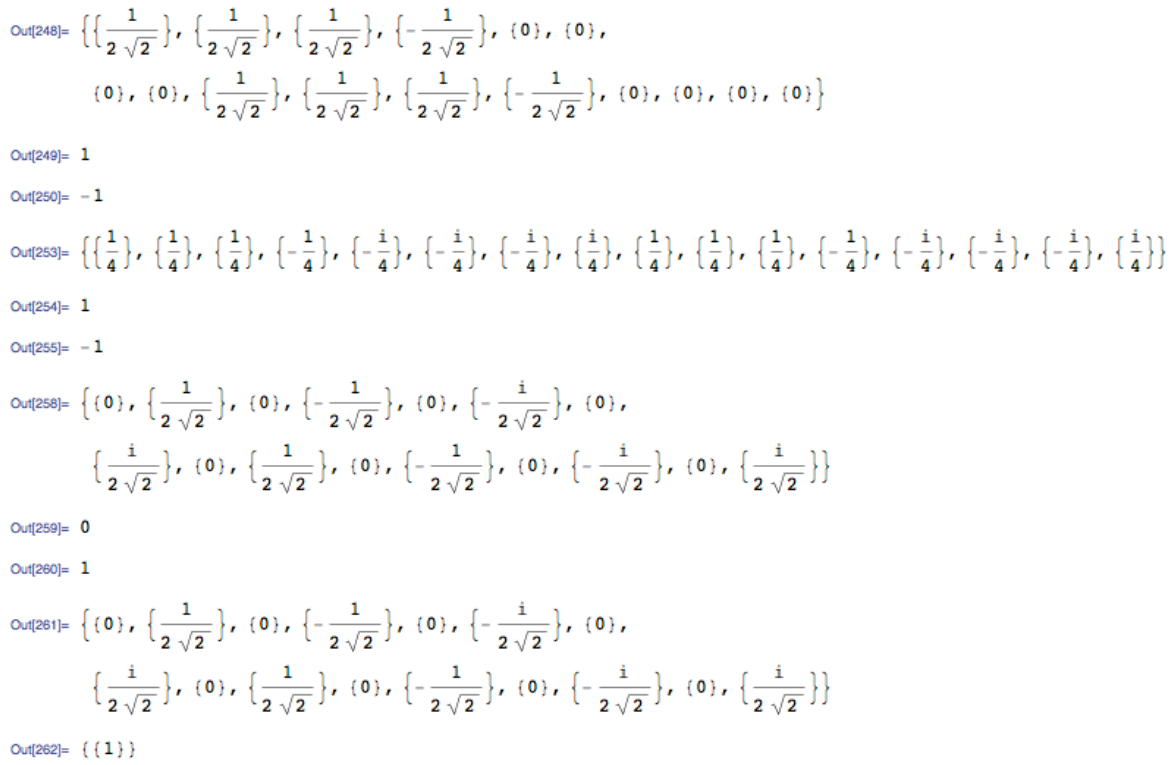


Figure 13: Output of Stochastic Measurements.

As shown in the figure above, the outcomes of the first and second measurement are not the desired state, proving that measurements are indeed *random*, but due to the adaptive measurement, the outcome is still deterministic as the output of $||\psi_{out}\rangle\langle\psi_{check}||^2$ is 1 as shown at the end. Different angles were used in the simulation, the result remained the same.

5.3 Errors and Imperfections in MBQC

Measurements cannot always be constant or perfect, especially when dealing with measurements in the atomic scale of physical qubits such as single photons in a laser[14] or neutral atoms in optical lattices [12, 13]. Imperfections in the initial cluster state could also occur if CZ gate was not applied properly between the qubits. Referring back to Eq. 21, the 'angle' Φ_t depends on

energy E and time t, which experimentally means applying a certain energy for a certain period of time. Any error in these two factors would lead to a 'flawed' operation, which could produce imperfectly prepared cluster state.

To see how errors and imperfections affect the outcome of MBQC, the *fidelity* of the quantum states, $\mathcal{F}$ between the outcome state and the expected state can be calculated. Fidelity is the measurement of the *distance* between two quantum states, not in the sense of *physical distance* but the distance between the distributions of the quantum states in the Hilbert space [3]. The simplest method to calculate the fidelity of two quantum states is to calculate $|\langle\psi_2||\psi_1\rangle|^2$, such that if $|\psi_1\rangle$ is identical to $|\psi_2\rangle$, $|\langle\psi_2||\psi_1\rangle|^2 = 1$

5.3.1 Error in Measurements

```
In[651]:= Err = {0, 0.025, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1};
          Fid = Range[13];

          α1 = -Pi;
          β1 = -Pi / 2;
          γ1 = -Pi;

          psic1 = KroneckerProduct[Rot[Z, α1].kxp, Rot[Z, β 1].kxp, Rot[Z, γ1].kxp,
              Had.Rot[Z, -γ1].Rot[X, -β1].Rot[Z, -α1].kxp];
          For[k = 1, k <= 13, k++, (ϵ = Err[[k]];
            Fid[[k]] =
             Abs[ConjugateTranspose[psic1].
                 outputStateEntg[KroneckerProduct[proj[Rot[Z, α1 * (1 + ϵ) ].kxp], Id, Id, Id],
                  outputStateEntg[KroneckerProduct[Id, proj[Rot[Z, β1 * (1 + ϵ) ].kxp], Id, Id],
                   outputStateEntg[KroneckerProduct[Id, Id, proj[Rot[Z, γ1 * (1 + ϵ) ].kxp], Id], q4LinEntg2]]]]^2 //
              N)]
          Fid = Flatten[Fid];
          data = Transpose[{Err * 100, Fid}];

          TableForm[data, TableHeadings → {None, {"Error in Measurements (%)", "Fidelity"}}]

          ListLinePlot[data, Axes → None, Frame → True, FrameLabel → {"Error (%)", "Fidelity"},
           PlotLabel → "Effect of Error in Measurements  (π,π/2,π) on Fidelity "]
```

Figure 14: Simulation for error in measurements.

This simulation models the possible errors that occur during measurements due to inaccurate measurement control over the rotation of the projectors. The angles of the measurement projectors are given percentage error values ϵ ranging from 0 to 100% error. The linear graph state is measured by the

same measurement operators defined in Section 5.2. The `psic1` serves as a checking factor as it is the expected outcome if measurements are of full accuracy. The fidelity is calculated by applying the output with the error with `psic1`. The simulation produced the graph below,

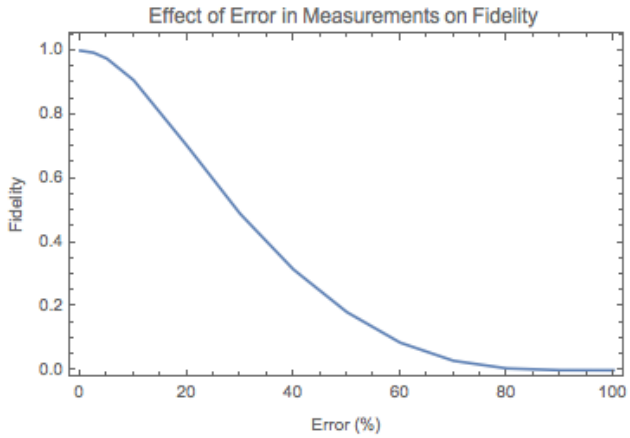


Figure 15: Fidelity affected by error in measurements.[2]

Another simulation was run for a different set of angles, where $\alpha = 0$, $\beta = \pi/2$ and $\gamma = \pi/2$, the result was almost similar,

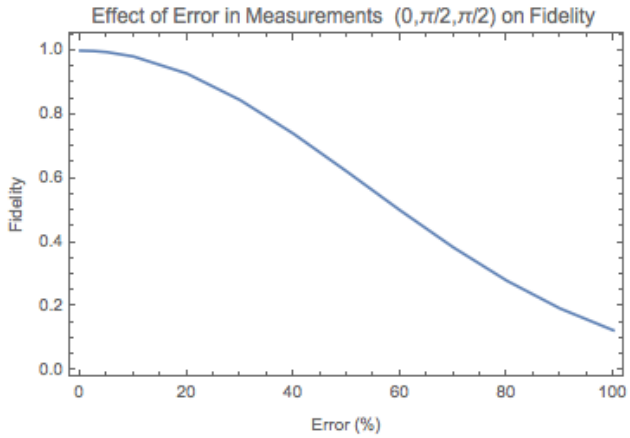


Figure 16: Fidelity affected by error in a different measurements.[2]

In both cases, fidelity starts to drop below 0.9 as error rises to around 20%. Meaning that up to around 20% error in measurements, the outcome still retain up to 90% of the information of the expected state.

5.3.2 Imperfections in Cluster State

```
In[785]:= Err1 = {0, 0.025, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1};
          Fid1 = Range[13];

          α2 = -Pi;
          β2 = -Pi / 2;
          γ2 = -Pi;

          psic2 = KroneckerProduct[Rot[Z, α2].kxp, Rot[Z, β2].kxp, Rot[Z, γ2].kxp,
              Had.Rot[Z, -γ2].Rot[X, -β2].Rot[Z, -α2].kxp];
          For[k = 1, k <= 13, k++, (ϵ = Err1[[k]];
            Fid1[[k]] =
             Abs[ConjugateTranspose[psic2].(outputStateEntg[KroneckerProduct[proj[Rot[Z, α2 ].kxp], Id, Id, Id],
                   outputStateEntg[KroneckerProduct[Id, proj[Rot[Z, β2 ].kxp], Id, Id],
                    outputStateEntg[KroneckerProduct[Id, Id, proj[Rot[Z, γ2 ].kxp], Id],
                     KroneckerProduct[ctrl2[Ph[Pi / 2 * (1 + ϵ)].Rot[Z, Pi * (1 + ϵ)]], Id, Id].
                      KroneckerProduct[Id, ctrl2[Ph[Pi / 2 * (1 + ϵ)].Rot[Z, Pi * (1 + ϵ)]], Id].
                      KroneckerProduct[Id, Id, ctrl2[Ph[Pi / 2 * (1 + ϵ)].Rot[Z, Pi * (1 + ϵ)]]].q4Linear]]])]^2 // N)]
          Fid1 = Flatten[Fid1];
          data1 = Transpose[{Err1 * 100, Fid1}];
```

Figure 17: Imperfections in the preparation of cluster state simulated in Mathematica.

In contrast to the previous simulation, this simulation models errors in the *initial state*, hence the computation will be inherently erroneous. This simulation attempts to replicate the imperfections that could occur during the preparation of a cluster state where errors ϵ are applied to the CZ gate, where in this case the Z gate is reduced into a *Phase* operator, set at $\frac{\pi}{2}$, and Z-rotation operator, set at an angle of π, to accommodate errors. Again, `psic2` is the function used for calculating fidelity. In this case, the angles of the measurement projectors are set to π,$\pi/2$ and π assuming that there is no error in measurements. The relationship between the imperfections and fidelity is shown in the graph below,

[2]The values for the graphs are included in the Appendix 9.2.

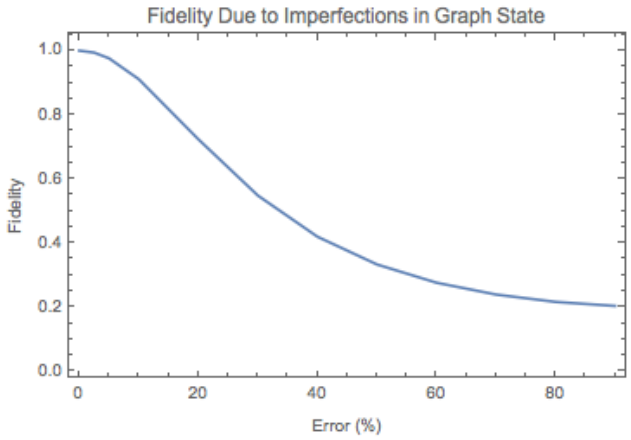


Figure 18: Fidelity affected by imperfections in the cluster state.[3]

The fidelity decreases in the similar pattern as the previous graphs, At 100% error the state cannot be measured any more since the CZ gate no longer operates.

6 Discussion and Conclusion

6.1 Implementation of Quantum Computation

One of the main objectives of quantum computing is to build a quantum computer, defined by Ladd et al. [15] as a machine that could truly exploit the many-particle quantum system to solve computational problems.

A quantum computer must allow accessibility and manipulation of the qubits, but the qubits must be isolated to retain the quantum properties [3]. This proves to be problematic experimentally, which will be discussed in the next section.

The requirements of a quantum computer are concisely listed in [3, 15]. Both suggest that in quantum computation, the system must be isolated from the rest of nature, so that fragile quantum information is not disturbed by informations from the outside leaking into the system, therefore it can still be represented properly. Second, that it must be able to access the vast Hilbert space of the qubits with only finite operations. Next, it must

[3]The values for the graph are included in the Appendix 9.2.

allow preparation of inital state, for example a certain polarisation state of a photon [13, 15] or trapped atoms of different energy levels [15]. Last and most importantly, that it must allow the measurement of output of the computations.

Some possible systems for MBQC are optical lattice, where neutral atoms trapped in periodic potential and one of the recent method, through cavity quantum electrodynamics. Neutral atoms in optical lattice could efficiently generate cluster state by superposing these optical lattices, allowing the neutral atoms in them to interact with the neutral atoms in the other lattice for a certain time [14, 12].

6.2 Limitations of Quantum Computation

The main concern in any quantum computing system is *decoherence*. Decoherence occurs information from outside the quantum system disturbs it. Thus, the quantum system loses its coherence or quantum properties.

One of the requirements stated earlier was that the system must be accessible and manipulatable, but also isolated from the surroundings. Taking account decoherence from the previous statement, a quantum system accessible to us, means that it is accessible by its surroundings, therefore there is a potential for *quantum noise* or decoherence. To find the balance between isolation and accessibility is a challenge itself. This is why photons are preferred in some cases, as they are relatively free of decoherence [15].

The scalability of a quantum computer is still currently limited. At this stage, quantum computation system requires lasers and optics for photons, cryogenic to make neutral atoms and more complex equipments to list. For quantum computers to be scalable, these instruments must be made scalable first [15].

In [13], they have found that there are a few challenges with the optical approach to MBQC. These are, the means to generate cluster states on demand, fast feedforward of measurements outcome for adaptive measurements and scaling up the cluster state.

6.3 Universality and Viability of MBQC and Quantum Circuit Model

The universality of both models have been discussed earlier. In quantum circuit model, universal gates set is utilised to achieve universal computation

[11]. On the other hand, MBQC requires no universal gates, it is inherently universal due to the fact that any operations could be replicated through different patterns of measurements [8]. Indeed, MBQC seems methodically simpler, but it requires the preparation of a cluster state and as shown in Section 5.3, it requires accurate measurements as well. Although, Walther et al. [13] stated that one-way quantum computation methods are more resource efficient than quantum circuit model.

6.4 Conclusion

Studying both models of quantum computation has offered two different ways of thinking about quantum computation. Both quantum circuit model [11] and MBQC [8, 13] are universal, therefore any quantum algorithms can be solved using any of the models.

While quantum circuit model has more of the classical notion of computation, where a qubit passes though wires and goes through logical gates. MBQC offers a more quantum mechanical approach by applying stochastic quantum measurements on a large entangled qubit states. On top of that, it is understood that MBQC uses physical resources consisting of qubits to perform the computation.

Yet, we have not reached a point where these computers are commercialised due to their scalability both in hardware and in the system itself.

Even with its great potential, it has been shown that MBQC can produce erroneous computation due to inaccuracy in controlling measurements or preparing the cluster state.

There are improvements that could be made to this project if time constraint was not a factor. Mainly, to further study the different implementation methods of MBQC on top of the theoretical groundwork, to provide more information on the efficiency and accuracy rate, thus providing more numerical data for the project. But so far, the project has provided a comprehensive and concise insight into quantum computing, especially on one-way quantum computation.

References

[1] D. Deutsch, *Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer*, Proc. R. Soc. Lond. A **400**, 1818, (1985).

[2] D. Mermin, *Quantum Computer Science: An Introduction* (Cambridge University Press, Cambridge, 2007).

[3] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, 2000).

[4] P. W. Shor, *Introduction to Quantum Algorithms*, e-print:arXiv/quant-ph/0005003v2, (2000)

[5] T. Hey, R.P. Feynman, R. Allen, *Feynman Lectures on Computation*, (Addison-Wesley Publishing, Massachusetts, 1996).

[6] R. Raussendorf and H. J. Briegel, *A One-Way Quantum Computer*, Phys. Rev. Lett. **86**, 5188 (2001).

[7] D. DiVincenzo, *Quantum Computation*, Science **270** p. 255-261, (1995).

[8] R. Raussendorf, D. Browne and H. J. Briegel, *Measurement-based Quantum Computation on Cluster State*, Phys. Rev. A **68**, 022312 (2003).

[9] C.H.Bennett and D. DiVincenzo, *Quantum Information and Computation*, Nature **404**, p. 247-255, (2000).

[10] M. Hirvensalo, *Quantum Computing (Second Edition)*, (Springer-Verlag, Berlin & Heidelberg, 2004)

[11] A. Barenco, C. H. Bennett, R. Cleve, D. DiVincenzo et al., *Elementary Gates for Quantum Computation*, Phys. Rev. A **52**, 3457 (1995).

[12] D. Browne and H. Briegel, *One-way Quantum Computation*, e-print:arXiv:quant-ph/0603226, (2006).

[13] P. Walther, K.J. Resch, T. Rudolph et al., *Experimental One-way Quantum Computing*, Nature **434**, p.169-176, (2005)

[14] D. Browne, Lecture: *One-way Quantum Computation*, Department of Materials, University of Oxford, (Gregynog, 2006)

[15] T.D. Ladd, F. Jelezko, R. Laflamme, Y. Nakamura et al., *Quantum Computers*, Nature **464**, p. 45-53, (2010).

8 Acknowledgements

My sincere gratitude goes to Dr. Markus Müller, whose enthusiasm for quantum computing and patience in teaching are truly admirable. Without his persistent generous guidance and tremendous help, this dissertation would not be here.
I would also like to thank Dr. Ed Bennett, for providing me the guidance to take on this project.

9 Appendices

9.1 Mathematica Simulation of MBQC

MBQC Simulation

Definitions

Pauli Eigenstates

These are the ket states, where k0 = $|0\rangle$, k1 = $|1\rangle$, kxp = $|+\rangle$ and so on.

```
k0 = {{1}, {0}}; k1 = {{0}, {1}};
kxp = 1 / Sqrt[2] (k0 + k1); kxm = 1 / Sqrt[2] (k0 - k1);
kyp = 1 / Sqrt[2] {{1}, {I}}; kym = 1 / Sqrt[2] {{1}, {-I}};
```

These are bra states of the kets above

```
b0 = Transpose[k0]; b1 = Transpose[k1]; bxp = Transpose[kxp];
bxm = Transpose[kxm];  byp = Transpose[kyp];
bym = Transpose[kym]; tP[x_] := ConjugateTranspose[x];
```

Matrices

Standard gates: Identity, Z, X, Y and zero matrix to build different gates later. The function mP is simply to ease the process of (not having to type 'KroneckerProduct' all the time) modulus product of 2 qubits.

```
Id = {{1, 0}, {0, 1}}; Z = {{1, 0}, {0, -1}}; Y = {{0, -I}, {I, 0}};
X = {{0, 1}, {1, 0}}; zer0 = {{0, 0}, {0, 0}};
mP[x_, y_] := KroneckerProduct[x, y]; mP3[x_, y_, z_] := KroneckerProduct[x, y, z];
```

Unitary Gates

Single Qubit Gates: Hadamard, Rotational gate which takes α as angle and 'gate' as any of the Pauli gates.

```
Had = (1 / Sqrt[2] (Z + X)); Rot[gate_, α_] := (Cos[α / 2] Id) - (I Sin[ α / 2] gate);
 Ph[d_] := {{Exp[d I], 0}, {0, Exp[d I]}};
RotExp[gate_] := MatrixExp[-I α gate];
```

Two Qubit Gates

Creates any two qubit control gate according input gate 'G' (defined in Matrices section).

```
ctrl2[G_] := ArrayFlatten[{{Id, zer0}, {zer0, G}}];
```

$$\frac{1+i}{\sqrt{2}}\,\text{KroneckerProduct}[\text{Rot}[Z, \text{Pi}/2], \text{Id}].\text{ctrl2}[\text{Rot}[Z, \text{Pi}]]$$

```
ctrl2[Ph[Pi / 2].Rot[Z, Pi]]
```

```
{{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, -1}}
```

```
{{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, -1}}
```

```
ctrl2[Rot[Z, Pi]]
```

```
{{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, -i, 0}, {0, 0, 0, i}}
```

Projector & Measurement

proj function simply creates a projector operator and outputState could give the output after measurement. prob measures the probability that state would be measured. stochasticOuput allows one to produce random ouput after measurement.

```
proj[stateProjector_] := stateProjector.ConjugateTranspose[stateProjector];
(*stateProjector is the eigenstate you want to measure*)

prob[prjState_, inState_] := Norm[proj[prjState].inState]^2;
(*prjState is the projected measurement state,
inState is the input or measured state*)

outputState[projector_, inputState_] :=
  (proj[projector].inputState) / Norm[proj[projector].inputState];
(*projector is well the projected measurement and inputState is well..
 input*)

stochasticOutput[prjState1_, prjState2_, inState_] :=
 (*If[prob[prjState1,inState]>=prob[prjState2,inState],*)
 If[prob[prjState1, inState] > RandomReal[1],
  outputState[prjState1, inState], outputState[prjState2, inState]]
  (*,If[prob[prjState2,inState]>RandomReal[1],
  outputState[prjState2,inState],outputState[prjState1,inState]]]*)
(*stochasticOutput takes two projected states or eigenvalues
 of that basis then calculates the probability and uses
 random number generator to make the output random*)
```

Just to prove stochasticOutput works

```
Bell = Sqrt[3] / 2 mP[k0, k0] + 1 / 2 mP[k1, k1]
co = Range[1000];
For[n = 1, n <= 1000, n++,
 co[[n]] = stochasticOutput[mP[k1, Id], mP[k0, Id], Bell]]
NumberOfState1Measured = Count[co, {{1}, {0}, {0}, {0}}]
NumberOfState2Measured = Count[co, {{0}, {0}, {0}, {1}}]
```

$$\left\{\left\{\frac{\sqrt{3}}{2}\right\}, \{0\}, \{0\}, \left\{\frac{1}{2}\right\}\right\}$$

```
764
```

```
236
```

Example 1 : Measuring Bell States

Creating Bell State

To create a 1/Sqrt[2] $\left(\left|00\right\rangle + \left|11\right\rangle\right)$ Bell state, we could start with 2 $\left|0\right\rangle$ qubit, transforming one of them with a hadamard gate, then entangling with a CNOT

```
Psi0 = ctrl2[X].mP[(Had.k0), k0]
```

$$\left\{\left\{\frac{1}{\sqrt{2}}\right\}, \{0\}, \{0\}, \left\{\frac{1}{\sqrt{2}}\right\}\right\}$$

Projector to two qubits Hilbert Space is the tensor product of projected Eigenstate with the 4x4 Identity Matrix

Probability of first Qubit in Ket[0] and output state

```
prob[mP[k0, Id], Psi0]
```

$$\frac{1}{2}$$

```
outputState[mP[k0, Id], Psi0]
```

```
{{1}, {0}, {0}, {0}}
```

Probability of first Qubit in Ket[1] and output state

```
prob[mP[k1, Id], Psi0]
```

$$\frac{1}{2}$$

```
psif = outputState[mP[k1, Id], Psi0]
```

```
{{0}, {0}, {0}, {1}}
```

Probability of second qubit in Ket[+] and output state

```
prob[mP[Id, kxp], Psi0]
```

$$\frac{1}{2}$$

```
outputState[mP[Id, kxp], Psi0]
```

$$\left\{\left\{\frac{1}{2}\right\}, \left\{\frac{1}{2}\right\}, \left\{\frac{1}{2}\right\}, \left\{\frac{1}{2}\right\}\right\}$$

```
prob[mP[Id, k0], psif]
```

0

Constructing Graph States

Simple 2 - Qubit Graph State

```
gs2 = ctrl2[Z].mP[kxp, kxp];
gsr2 = mP[RotExp[Z], Id].gs2
outputState[mP[(Had.kxp), Id], gs2]
stochasticOutput[mP[kxp, Id], mP[kxm, Id], gs2]
Rot[Y, Pi / 2].Rot[Z, Pi].k0
```

$$\left\{\left\{\frac{e^{-i\alpha}}{2}\right\}, \left\{\frac{e^{-i\alpha}}{2}\right\}, \left\{\frac{e^{i\alpha}}{2}\right\}, \left\{-\frac{e^{i\alpha}}{2}\right\}\right\}$$

$$\left\{\left\{\frac{1}{\sqrt{2}}\right\}, \left\{\frac{1}{\sqrt{2}}\right\}, \{0\}, \{0\}\right\}$$

$$\left\{\{0\}, \left\{\frac{1}{\sqrt{2}}\right\}, \{0\}, \left\{-\frac{1}{\sqrt{2}}\right\}\right\}$$

$$\left\{\left\{-\frac{i}{\sqrt{2}}\right\}, \left\{-\frac{i}{\sqrt{2}}\right\}\right\}$$

4 - Qubit Linear GraphState Construction

Start with $|0000\rangle$ state

```
q4linear = KroneckerProduct[k0, k0, k0, k0]
```

$$\{\{1\}, \{0\}, \{0\}, \{0\}, \{0\}, \{0\}, \{0\}, \{0\}, \{0\}, \{0\}, \{0\}, \{0\}, \{0\}, \{0\}, \{0\}, \{0\}\}$$

Apply Hadamard To All and apply CZ between all

```
q4Linear = KroneckerProduct[Had, Had, Had, Had].q4linear; \.10

q4LinEntg = KroneckerProduct[ctrl2[Z], Id, Id].KroneckerProduct[Id, ctrl2[Z], Id].
  KroneckerProduct[Id, Id, ctrl2[Z]].q4Linear

q4LinEntg2 = 1 / 2 (KroneckerProduct[kxp, k0, kxp, k0]) +
  1 / 2 (KroneckerProduct[kxp, k0, kxm, k1]) +
  1 / 2 (KroneckerProduct[kxm, k1, kxm, k0]) +
  1 / 2 (KroneckerProduct[kxm, k1, kxp, k1])
\
```

$$\left\{\left\{\frac{1}{4}\right\},\left\{\frac{1}{4}\right\},\left\{\frac{1}{4}\right\},\left\{-\frac{1}{4}\right\},\left\{\frac{1}{4}\right\},\left\{\frac{1}{4}\right\},\left\{-\frac{1}{4}\right\},\right.$$
$$\left.\left\{\frac{1}{4}\right\},\left\{\frac{1}{4}\right\},\left\{\frac{1}{4}\right\},\left\{\frac{1}{4}\right\},\left\{-\frac{1}{4}\right\},\left\{-\frac{1}{4}\right\},\left\{-\frac{1}{4}\right\},\left\{\frac{1}{4}\right\},\left\{-\frac{1}{4}\right\}\right\}$$

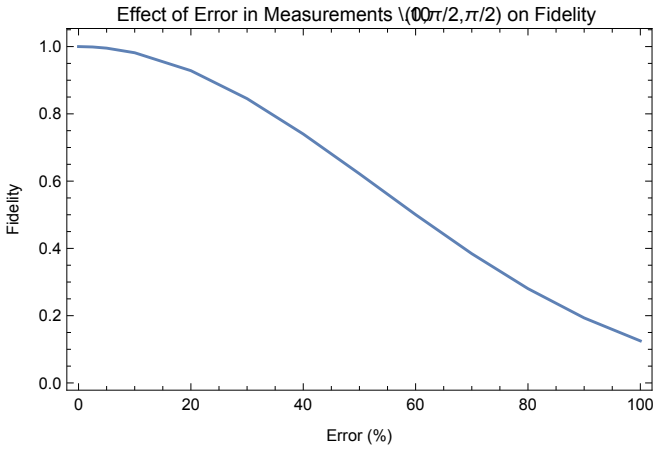
$$\left\{\left\{\frac{1}{4}\right\},\left\{\frac{1}{4}\right\},\left\{\frac{1}{4}\right\},\left\{-\frac{1}{4}\right\},\left\{\frac{1}{4}\right\},\left\{\frac{1}{4}\right\},\left\{-\frac{1}{4}\right\},\right.$$
$$\left.\left\{\frac{1}{4}\right\},\left\{\frac{1}{4}\right\},\left\{\frac{1}{4}\right\},\left\{\frac{1}{4}\right\},\left\{-\frac{1}{4}\right\},\left\{-\frac{1}{4}\right\},\left\{-\frac{1}{4}\right\},\left\{\frac{1}{4}\right\},\left\{-\frac{1}{4}\right\}\right\}$$

```
outputStateEntg[projector_, inputState_] :=
  (projector.inputState) / Norm[projector.inputState];

probEntg[prS_, inState_] := Norm[prS.inState]^2;


ϵ = 0.11;

m1 = outputStateEntg[
    KroneckerProduct[proj[Rot[Z, -Pi * (1 + ϵ) ].kxp], Id, Id, Id], q4LinEntg2];

m2 = outputStateEntg[
    KroneckerProduct[Id, proj[Rot[Z, -Pi / 2 * (1 + ϵ) ].kxp], Id, Id], m1];

m3 = outputStateEntg[
    KroneckerProduct[Id, Id, proj[Rot[Z, -Pi * (1 + ϵ) ].kxp], Id], m2];

psic = KroneckerProduct[Rot[Z, -Pi].kxp, Rot[Z, -Pi / 2].kxp,
    Rot[Z, -Pi].kxp, Had.Rot[Z, Pi].Rot[X, Pi / 2].Rot[Z, Pi].kxp];

Abs[ConjugateTranspose[psic].m3]^2 // N
{{0.890335}}
```

Fidelity and Errors in Measurement

```
Err = {0, 0.025, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1};
Fid = Range[13];

α1 = 0;
β1 = -Pi / 2;
γ1 = -Pi / 2;

psic1 = KroneckerProduct[Rot[Z, α1].kxp, Rot[Z, β 1].kxp,
    Rot[Z, γ1].kxp, Had.Rot[Z, -γ1].Rot[X, -β1].Rot[Z, -α1].kxp];
For[k = 1, k <= 13, k++, (ϵ = Err[[k]];
  Fid[[k]] = Abs[ConjugateTranspose[psic1].outputStateEntg[
        KroneckerProduct[proj[Rot[Z, α1 * (1 + ϵ) ].kxp], Id, Id, Id],
        outputStateEntg[KroneckerProduct[Id, proj[Rot[Z, β1 * (1 + ϵ) ].kxp],
          Id, Id], outputStateEntg[KroneckerProduct[Id, Id,
           proj[Rot[Z, γ1 * (1 + ϵ) ].kxp], Id], q4LinEntg2]]]]^2 // N)]
Fid = Flatten[Fid];
data = Transpose[{Err * 100, Fid}];

TableForm[data,
 TableHeadings → {None, {"Error in Measurements(%)", "Fidelity"}}]

ListLinePlot[data, Axes → None,
 Frame → True, FrameLabel → {"Error (%)", "Fidelity"},
 PlotLabel → "Effect of Error in Measurements \(α1,0,π/2,π/2) on Fidelity "]
```

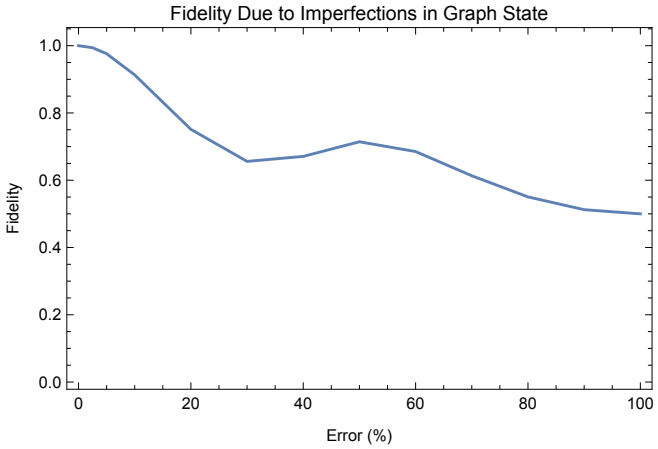
Error in Measurements(%)	Fidelity
0	1.
2.5	0.998844
5.	0.995383
10.	0.981646
20.	0.928367
30.	0.845258
40.	0.740011
50.	0.621859
60.	0.500363
70.	0.384233
80.	0.280379
90.	0.193318
100	0.125



Fidelity and Imperfections in Graph State

```
Err1 = {0, 0.025, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1};
Fid1 = Range[13];

α2 = 0;
β2 = 0;
γ2 = 0;

psic2 = KroneckerProduct[Rot[Z, α2].kxp, Rot[Z, β2].kxp,
    Rot[Z, γ2].kxp, Had.Rot[Z, -γ2].Rot[X, -β2].Rot[Z, -α2].kxp];
For[k = 1, k <= 13, k++, (ϵ = Err1[[k]];
  Fid1[[k]] = Abs[ConjugateTranspose[psic2].
        (outputStateEntg[KroneckerProduct[proj[Rot[Z, α2 ].kxp], Id, Id, Id],
          outputStateEntg[KroneckerProduct[Id, proj[Rot[Z, β2 ].kxp], Id, Id],
           outputStateEntg[KroneckerProduct[Id, Id, proj[Rot[Z, γ2 ].kxp], Id],
            KroneckerProduct[
              ctrl2[Ph[Pi / 2 * (1 + ϵ)].Rot[Z, Pi * (1 + ϵ)]], Id, Id].
             KroneckerProduct[Id, ctrl2[Ph[Pi / 2 * (1 + ϵ)].Rot[
                 Z, Pi * (1 + ϵ)]], Id].
             KroneckerProduct[Id, Id, ctrl2[Ph[Pi / 2 * (1 + ϵ)].Rot[
                 Z, Pi * (1 + ϵ)]]].q4Linear ]]])]^2 // N)]
Fid1 = Flatten[Fid1];
data1 = Transpose[{Err1 * 100, Fid1}];

TableForm[data1, TableHeadings → {None, {"Imperfections (%)", "Fidelity"}}]
ListLinePlot[data1, Axes → None,
 Frame → True, FrameLabel → {"Error (%)", "Fidelity"},
 PlotLabel → "Fidelity Due to Imperfections in Graph State"]
```

Imperfections (%)	Fidelity
0	1.
2.5	0.993882
5.	0.976116
10.	0.913031
20.	0.751238
30.	0.655934
40.	0.67084
50.	0.714286
60.	0.685219
70.	0.613246
80.	0.550412
90.	0.512421
100	0.5



Adaptive Measurements for a Single Qubit Rotation

```
stochasticOutputEntg[Mp1_, Mp2_, state_] :=
 If[probEntg[Mp1, state] > RandomReal[1],
  outputStateEntg[Mp1, state],
  outputStateEntg[Mp2, state]]


corrFactor[currState_, reqProj_] :=
  If[probEntg[reqProj, currState] == 1, 0,
   1];

(*Angles to be measured*)
α = -Pi;
β = -Pi / 2;
γ = -Pi;

p0 = KroneckerProduct[proj[Rot[Z, α].kxp], Id, Id, Id];
p1 = KroneckerProduct[proj[Rot[Z, α].kxm], Id, Id, Id];
(*Projectors of the measurement*)

mR1 = stochasticOutputEntg[p0, p1, q4LinEntg2]

b1 = corrFactor[mR1, p0]
(*Binary measurement outcome,
by checking whether the probability of the outcome state
  and the required projector == 1, if it is b =0, if not b=1 *)

c1 = (-1)^(b1)
(*Adaptive value for angle*)

p2 = KroneckerProduct[Id, proj[Rot[Z, (c1 * (β))].kxp], Id, Id];
p3 = KroneckerProduct[Id, proj[Rot[Z, (c1 * (β))].kxm], Id, Id];

mR2 = stochasticOutputEntg[p2, p3, mR1]

b2 = corrFactor[mR2, p2]

c2 = (-1)^(b2)

p4 = KroneckerProduct[Id, Id, proj[Rot[Z, (c2 * (γ))].kxp], Id];
p5 = KroneckerProduct[Id, Id, proj[Rot[Z, (c2 * (γ))].kxm], Id];

mR3 = stochasticOutputEntg[p4, p5, mR2]

b3 = corrFactor[mR3, p4]

c3 = (-1)^(b3)

psicheck = KroneckerProduct[
  Rot[Z, α].If[b1 == 0, Id, Z].kxp, Rot[Z, (c1 * (β))].If[b2 == 0, Id, Z].kxp,
  Rot[Z, (c2 * (γ))].If[b3 == 0, Id, Z].kxp, If[b3 == 0, Id, X].If[b2 == 0, Id, Z].
   If[b1 == 0, Id, X].Had.Rot[Z, -γ].Rot[X, -β].Rot[Z, -α].kxp]

Abs[ConjugateTranspose[psicheck].(mR3)]^2
(*Check if outcome is the same as expected, if {{1}} means its correct*)
```

$$\left\{\left\{\frac{1}{2\sqrt{2}}\right\}, \left\{\frac{1}{2\sqrt{2}}\right\}, \left\{\frac{1}{2\sqrt{2}}\right\}, \left\{-\frac{1}{2\sqrt{2}}\right\}, \{0\}, \{0\}, \{0\}, \{0\}, \left\{\frac{1}{2\sqrt{2}}\right\}, \left\{\frac{1}{2\sqrt{2}}\right\}, \left\{\frac{1}{2\sqrt{2}}\right\}, \left\{-\frac{1}{2\sqrt{2}}\right\}, \{0\}, \{0\}, \{0\}, \{0\}\right\}$$

$$1$$

$$-1$$

$$\left\{\left\{\frac{1}{4}\right\}, \left\{\frac{1}{4}\right\}, \left\{\frac{1}{4}\right\}, \left\{-\frac{1}{4}\right\}, \left\{-\frac{i}{4}\right\}, \left\{-\frac{i}{4}\right\}, \left\{-\frac{i}{4}\right\}, \left\{\frac{i}{4}\right\}, \left\{\frac{1}{4}\right\}, \left\{\frac{1}{4}\right\}, \left\{\frac{1}{4}\right\}, \left\{-\frac{1}{4}\right\}, \left\{-\frac{i}{4}\right\}, \left\{-\frac{i}{4}\right\}, \left\{-\frac{i}{4}\right\}, \left\{\frac{i}{4}\right\}\right\}$$

$$1$$

$$-1$$

$$\left\{\left\{\frac{1}{2\sqrt{2}}\right\}, \{0\}, \left\{\frac{1}{2\sqrt{2}}\right\}, \{0\}, \left\{-\frac{i}{2\sqrt{2}}\right\}, \{0\}, \left\{-\frac{i}{2\sqrt{2}}\right\}, \{0\}, \left\{\frac{1}{2\sqrt{2}}\right\}, \{0\}, \left\{\frac{1}{2\sqrt{2}}\right\}, \{0\}, \left\{-\frac{i}{2\sqrt{2}}\right\}, \{0\}, \left\{-\frac{i}{2\sqrt{2}}\right\}, \{0\}\right\}$$

$$1$$

$$-1$$

$$\left\{\left\{\frac{1}{2\sqrt{2}}\right\}, \{0\}, \left\{\frac{1}{2\sqrt{2}}\right\}, \{0\}, \left\{-\frac{i}{2\sqrt{2}}\right\}, \{0\}, \left\{-\frac{i}{2\sqrt{2}}\right\}, \{0\}, \left\{\frac{1}{2\sqrt{2}}\right\}, \{0\}, \left\{\frac{1}{2\sqrt{2}}\right\}, \{0\}, \left\{-\frac{i}{2\sqrt{2}}\right\}, \{0\}, \left\{-\frac{i}{2\sqrt{2}}\right\}, \{0\}\right\}$$

$$\{\{1\}\}$$

9.2 Tables of Values for Graphs

Error in Measurements (%)	Fidelity
0	1.
2.5	0.993589
5.	0.975084
10.	0.907704
20.	0.702848
30.	0.489437
40.	0.315239
50.	0.182138
60.	0.0865776
70.	0.0297814
80.	0.00591776
90.	0.000346071
100	0.

Figure 19: Table of values for graph in Fig.15.

Error in Measurements (%)	Fidelity
0	1.
2.5	0.998844
5.	0.995383
10.	0.981646
20.	0.928367
30.	0.845258
40.	0.740011
50.	0.621859
60.	0.500363
70.	0.384233
80.	0.280379
90.	0.193318
100	0.125

Figure 20: Table of values for graph in Fig.16.

Imperfections (%)	Fidelity
0	1.
2.5	0.993872
5.	0.975968
10.	0.910841
20.	0.723607
30.	0.548117
40.	0.419821
50.	0.333333
60.	0.276393
70.	0.239488
80.	0.216542
90.	0.203994
100	Indeterminate

Figure 21: Table of values for graph in Fig.18.