COMPARING THE PERFORMANCE OF DIFFERENT QUANTUM ERROR CORRECTING CODES

An Undergraduate Research Thesis

by

MARSHALL B. PICKETT

Submitted to the Engineering Honors program at
Texas A&M University
in partial fulfillment of the requirements for the designation as an

ACE SCHOLAR

Approved by Research Advisor: Dr. Andreas Klappenecker

May 2019

Major: Computer Science

TABLE OF CONTENTS

ABSTRACT

Comparing the Performance of Different Quantum Error Correcting Codes

Marshall B. Pickett
Department of Computer Science
Texas A&M University

Research Advisor: Dr. Andreas Klappenecker
Department of Computer Science
Texas A&M University

This thesis discusses Quantum Error Correction and why it is essential for the development of quantum computation. The introduction will cover the basics of quantum computing and classical error correction. In the second section we will show why Quantum Error Correction is necessary and how different types of error correcting codes help detect and prevent errors. The experiments section will compare the real world performance of different types of quantum error correcting codes using a quantum simulator and prevent the results of the experiments. The final section will discuss the conclusions our the research.

ACKNOWLEDGMENTS

I'd like to thank Dr. Klappenecker for his guidance and reassurance on this project. His suggestions, knowledge, and discussions helped me throughout the semester as I was learning new material and trying to select a topic of research.

I would also like to thank Andrew Nemec for discussing with me and helping me understand many unfamiliar topics prevalent in quantum error correction.

NOMENCLATURE

QC	Qantum Computing
QEC	Quantum Error Correction
EC	Error Correction
CSS	Calderbank, Shor, Steane

LIST OF FIGURES

FIGURE Page

1. INTRODUCTION

In this section we present the background information of quantum computing (QC) and classical error correction necessary to understand the topic of Quantum Error Correction (QEC).

Ever since the dawn of the Information Age, the speed and complexity of computer has increased at an incredible pace. For the past 40+ years, Moore's law has been shown to be fairly accurate. It says the number of transistors in integrated circuit chips doubles every two years. One of the main reasons the semiconductor industry has been able to follow such a trend is because of their ability to make smaller and smaller transistors. This allows more transistors to be packed into a relatively constant die size. However, there are some problems that are beginning to surface with classical computer.

One problem is that electrical components cannot become smaller and smaller indefinitely. At some point there is a physical limit to how small a transistor can become and still preform reliably. When the size of the transistors become smaller enough to be measured on the atomic scale, quantum effects start to interfere with the performance and function of the transistors. We can see that the pace in which in transistor size has decreased has slowed in the past decade. The physical limits of silicone as starting to be reached, which is starting to limit the speed of the processors. To further increase the speed, there will need to be other engineering solutions.

Another issue with classical computers is that there are entire classes of problems that cannot be solved on classical computer no matter how fast or powerful the computer is. There are problems with small inputs that would take today's most powerful supercomputer longer than the age of the universe to solve. Even if the computer's speed was increased by a factor of 1000, these types of problems would still take too long to solve

on a classical computer. This is because of the fundamental architecture of a classical computer.

To overcome these issues, there will need to be a paradigm shift in the way we view and process information. This is where using quantum mechanics to process information comes in. Richard Feynman made a conjecture in 1982 that a quantum computer which used quantum mechanics inherently might be more powerful than a classical computer [1]]. And indeed it has been shown that quantum computation provide a measureable speed up over classical computation in certain applications. For example, an algorithm was discovered by Peter Shor to factor numbers on a quantum computer in polynomial time [2]. Also there is Grover's algorithm, which can find an object in an unsorted database in $O(\sqrt{N})$ time on a quantum computer.

With classical computation approaching its physical limit in terms of transistor speed and size, it's important to take advantage of quantum's superiority to satisfy society's ever increasing computational needs. However there is a major challenge that must be overcome before utilizing the full power of a quantum computer. Unlike classical computers, which process information in a pretty reliable manner, current quantum computer are noisy and there is a relativity high probability that errors are introduced into the quantum computation. Without some way to minimize the errors or combat the errors through correction, we will not be able to reliably use quantum computation to produce any meaningful results. This is why quantum error correcting techniques are so important.

1.1 Background

Bits and Qubits

Even with the increase in speed and complexity, computers have been operating with the same basic principle: manipulating and encoding bits of information into useful computation results.

Before discussing quantum error correction, we first introduce the basics of quantum information processing. The fundamental unit of information for classical computing is the binary digit or bit. A bit can only have one of two values, typically represented as 0 or 1. The two values can be interpreted as logical values (true/false, yes/no, on/off). In quantum computing, a qubit or quantum bit is the basic unit of quantum information. It is analogous to the classical bit in that it can also be either a 0 or 1. However, because of quantum mechanics, a qubit can also exist in a complex superposition of the two states. Dirac notation is used to represent the two basis states, denoted as $|0\rangle$ and $|1\rangle$. The arbitrary state of an individual qubit , $|\psi\rangle$, can be expressed as,

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

where $|0\rangle$ and $|1\rangle$ are the two orthonormal basis states of the qubit and α and β are complex numbers. We also have the condition,

$$\alpha^2 + \beta^2 = 1 \tag{1.2}$$

which corresponds to the probability of observing a 0 or 1 when measuring that qubit. α^2 is the probabilty of observing a 0 and β^2 is the probability of observing a 1. The state can also be represented more compactly as a column vector $\left(\begin{smallmatrix}\alpha\\ \beta\end{smallmatrix}\right)$ [3].

Quantum Gates

From the principles of quantum mechanics, only linear operators describe evolutions of closed quantum systems. These linear operators are quantum gates and they are used to manipulate the state of a qubit. Quantum gate operations are represented by unitary operations. To follow the laws physics, all operations on quantum states are reversible and hence unitary. The unitary operation can be represented by a 2x2 matrix. Four important

gates are the Pauli operators, $\sigma_x, \sigma_y, \sigma_z$, and the identity operation, σ_i.

$$\sigma_i = I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \sigma_x = X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \sigma_y = Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \sigma_z = Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{1.3}$$

These operators are important because any single qubit operation, G, can be expressed as the linear combination of these operators.

$$G = c_I\sigma_I + c_x\sigma_x + c_y\sigma_y + c_z\sigma_z \tag{1.4}$$

Another important gate, that has no classical analog is the Hadamard Gate, H.

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{1.5}$$

The Hadamard Gate is very important because it is used to create entanglement between qubits, which is an essential property that gives quantum computation its power.

Another important gate is the controlled not or CNOT gate. The CNOT gate operates on a quantum register consisting of 2 qubits. The CNOT gate flips the second qubit (the target qubit) if and only if the first qubit (the control qubit is $|1\rangle$. The CNOT gate can be represented by the matrix:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{1.6}$$

Quantum Registers

Similar to classical registers, a quantum register is built by combining several qubits. This is equivalent mathematically to the tensor product of two-dimensional vector spaces. A quantum register of n qubits can represent any normalized complex linear combination of the 2^n orthogonal basis states. These vector spaces with inner products are called Hilbert spaces. We can describe them more compactly with a binary vector and the bra-ket notation.

$$|b_1\rangle \otimes \cdots \otimes |b_n\rangle = |b_1 \cdots b_n\rangle = |\mathbf{b}\rangle \text{ where } b_i \in \{0, 1\}$$

The $\otimes$ operator is called the tensor product. The tensor product is a way of putting together vector spaces to form larger vector spaces. What makes quantum registers different from classical registers, and what gives them the power over classical registers is entanglement. Consider the state of a 2-qubit system that could be described as

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

This is the famous Bell state in which two qubits are entangled. Both qubits are in a superposition of $|0\rangle$ or $|1\rangle$, but are connected in a manner which cannot be directly implemented classically. Because of entanglement, measurement of one qubit with result in either a 0 or a 1 and will cause the other qubit to collapse and become the same value. When measured, the qubits must be either 00 or 11. There is 0% probabilty that you will observe 01 or 10.

1.2 Introduction to Error Correction

In classical error correction, we want to encode our information in a way that protects against error that could occur within our communication channel. Many communication channels are subject to noise, therefore errors may be introduced to a message transmitted

from the sender to the receiver. If we were to send a message over a noisy channel and an error was introduced into our message (e.g. a bit somewhere was flipped 0->1), then we want to be able to reconstruct the original message.

Review of Classical Error Correction

It was Richard Hamming in 1947 that is credited with the modern development of error correction codes. The essential point of error correcting codes is to use a set of code words from $\{0,1\}^n$ that represent some meaningful information (i.e. codewords) is subset of the entire $\{0,1\}^n$ data space. Some form of redundancy is built into the message to form an error correcting code. The redundancy allows the receiver to detect a limited number of errors that may occur anywhere in the message, and often to correct these errors without retransmission.

Below is a diagram of a classical communication channel model.

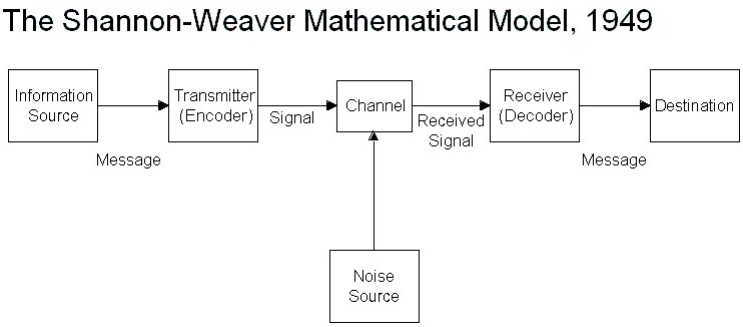

Figure 1.1: Shannon-Weaver model of communication illistrating how messages are sent over a noisy channel

Three Bit repetition code

A simple example on an error correcting code is the three bit repetition code, denoted as [3,1]. A triplet of zeros, 000, represents a logical 0, and a triplet of ones, 111, represents

a logical 1. The error correction works by taking a majority vote, so in this case the code can correct up to 1 error that happens on any bit. For example, if 000 was sent, but 001 was received, then an error in the third bit has occurred. However, by majority vote, 001 is decoded back into 000, which was the original message.

Seven Bit Hamming code

A more non-trivial example is the 7-bit Hamming code which relies on encoding and decoding matrices to detect and correct errors in a transmitted message. A successful error correcting scheme utilizes the decoding portion to identify if and what type of error occurred when a message passed through the noisy channel. The encoding portion can be formalized as an $n \times r$ matrix G, that takes a message m of length r and converts it into a codeword of length n, where the codewords make up the span of the columns of G. The corresponding matrix for the 7-bit Hamming code is

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

which encodes a 4-bit message as a 7-bit codeword. This code or encoding scheme has a maximum distance, d, of 3 since each of the rows in G differ in at most 3 spots, which means it can correct at most 1 error. The equation relating distance to the number

of correctable errors is

$$\text{maximum number of correctable errors} = \lfloor \frac{1}{2}(d-1) \rfloor \tag{1.7}$$

Along with the encoding matrix G, we also define a parity check matrix H, so that

$$GH^T = 0 \tag{1.8}$$

In the case of the of 7-bit Hamming code, the H corresponding to G would be

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

H is used to check if any error has occurred in the codeword because if received message, $x = c + e$, where c is a codeword and e is a 1-bit error, then

$$xH^T = (c+e)H^T = cH^T + eH^T = 0 + eH^T = eH^T \tag{1.9}$$

eH^T uniquely identifies the error and is known as the *error syndrome* [4].

1.3 Quantum Error Correction

Error correction is needed in quantum computation because today's computers are very noisy. Without protecting against errors, sustained computation would not be possible because even after a short amount of time errors would be introduced into our computation and propagate further making the output indistinguishable. Quantum Error correction is essential for building a fault tolerant quantum computer. Although many techniques and research from classical error correction can be applied to Quantum Error Correction, there

are several obstacles to formulating a theory of quantum error correction. One of the main challenges is the no-cloning theorem, which says that it is impossible create an identical copy of an arbitrary unknown quantum state. In this case, the repetition code cannot be applied to a quantum code. Another major challenge is that errors are continuous. Unlike classical errors in which the bit is either flipped or not, continuous errors can occur on a qubit, in which a qubit is partially flipped or the phase is partially changed.

Many researchers did not believe quantum error correction would be possible, but in 1995 Peter Shor discovered a method of storing the information of one qubit into an entangled state of nine qubits, and published his nine qubit code [citation]. He showed that there are ways to overcomes the challenges previously mentioned. As shown above, any operation (including an error operation) on a single qubit can be represented by a linear combination of the $I, X, Y,$ and Z operators. Using this fact, you can correct any arbitrary error that has occured with a σ_x gate and a σ_z gate.

Three Qubit Flip Code

In a similar way to the classical 3-bit repitition code discussed above, we use the same strategy to help protect against quantum errors. While the classical repition code works because classical bits can be measured and repeated, this stops being the case in the quantum channel, due to the no-cloning theorem. It is no longer possible to repeat a single qubit three times, so to overcome this, the three-qubit bit flip code is used, which uses entanglement and syndrome measurements. The encoding consists of the following mappings $|0\rangle \mapsto |0_L\rangle \equiv |000\rangle$ and $|1\rangle \mapsto |1_L\rangle \equiv |111\rangle$. The state $|\psi\rangle$ will be mapped to

$$\alpha |0\rangle + \beta |1\rangle \mapsto \alpha |000\rangle + \beta |111\rangle \tag{1.10}$$

This mapping can be realized for example using two CNOT gates, entangling the system with two ancillary qubits initialized in the state $|0\rangle$. The encoded state is then passed

through the noisy channel. Here is the circuit diagram depicting the encoding, correcting, and decoding of a three qubit code [5].

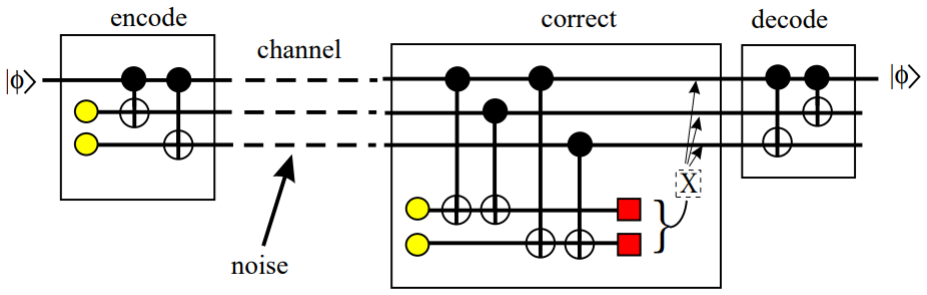

Figure 1.2: Simple example illustrating the principles of quantum error correction

After the message has passed through the channel, the correction phase uses ancilla qubits to measure the parity of the encoded qubits. Conducting measurements on the ancilla qubits is necessary because we don't want to collapse the superposition of the encoded qubits by measuring the encoded qubits. This simple code will help protect against a single bit flip errors that could occur on any of the three qubits. This makes it equivalent to the original classical repetition code for error correction. If we assume that at most one X error occured in the noisy channel, the detection phase will be able to detect apply the correct σ_x gate to the qubit and reverse the error. As an example, assume an X error happened on the 1st qubit in the noisy channel. Applying an X_1 error to the encoded state results in the mapping

$$\alpha\,|000\rangle + \beta\,|111\rangle \xrightarrow{X_1} \alpha\,|100\rangle + \beta\,|011\rangle.$$

In the correction phase, ancilla qubits, that are initiallized to $|0\rangle$, are used to measure the parity of the three encoded qubits. There are four possible outcomes from the ancilla measurements, also called the error syndrome measurement.

$|00\rangle$ signifies there is no error. No σ_x gate is needed.

$|01\rangle$ signifies there was an error 1st qubit, so a σ_x gate is applied to that qubit to fix the error.

$|10\rangle$ signifies there was an error 2nd qubit, so a σ_x gate is applied to that qubit to fix the error.

$|11\rangle$ signifies there was an error 3rd qubit, so a σ_x gate is applied to that qubit to fix the error.

This encoding scheme can correct X error, but it cannot correct Z errors. To see that this doesn't work for phase errors, note that applying Z_2 error results in the mapping

$\alpha|000\rangle + \beta|111\rangle \xrightarrow{Z_2} \alpha|000\rangle - \beta|111\rangle$.

This is a valid encoding of the state $\alpha|0\rangle - \beta|1\rangle$, so the error is undetectable. To have an effective error correcting code, we also need to be able to correct the Z errors. This is accomplished with the Hadamard gate, because of the property $HX = ZH$. This shows that X and Z errors are orthogonal to each other. This means the Hadamard matrix will change bases and turn bit errors into phase errors, and vice versa. Here is a simple illustration to show that an X gate applied in the Hadamard basis is equal to the Z gate.

$$HX = ZH \rightarrow HXH = Z \tag{1.11}$$

$$\sigma_x^H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} = \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix} = \sigma_z \tag{1.12}$$

The code that protects against phase errors is the follow mapping:

$H|0\rangle \mapsto H^{\otimes 3}|000\rangle$

$H|1\rangle \mapsto H^{\otimes 3}|111\rangle$

This expands to:

$|0_L\rangle \mapsto \frac{1}{\sqrt{8}}(|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle + |111\rangle)$

$|1_L\rangle \mapsto \frac{1}{\sqrt{8}}(|000\rangle - |001\rangle - |010\rangle + |011\rangle - |100\rangle + |101\rangle + |110\rangle - |111\rangle)$

Now we can combine both encoding schemes by concatenating our bit flip code with our phase flip code to protect against both types of errors. This is the 9-qubit code that Peter Shor wrote about in his 1995 paper [6].

Shor Nine Qubit Code

Peter Shor's Code encodes one logical qubit into nine physical qubits. It can protect against both bit flip errors and phase flip errors. The 9-qubit Shor code is given by the following mapping:

$|0\rangle \mapsto |0\rangle_L \equiv \frac{1}{\sqrt{8}}(|000000000\rangle + |000000111\rangle + |000111000\rangle + |000111111\rangle + |111000000\rangle + |111000111\rangle + |111111000\rangle + |111111111\rangle)$

$|1\rangle \mapsto |1\rangle_L \equiv \frac{1}{\sqrt{8}}(|000000000\rangle - |000000111\rangle - |000111000\rangle + |000111111\rangle - |111000000\rangle + |111000111\rangle + |111111000\rangle - |111111111\rangle)$

Here is a circuit diagram of the Shor code

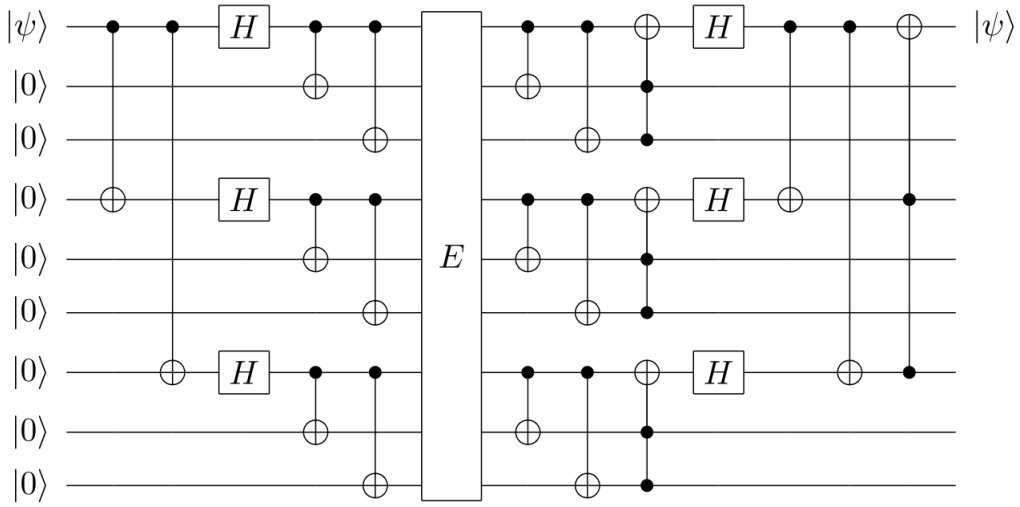

Figure 1.3: Circuit diagram for the Shor Nine Qubit Code

We showed that the code can correct bit flip errors (σ_x) and phase flip errors (σ_z), but it can also correct σ_y errors (up to a global phase) because σ_y can be obtained from a

product of σ_x and σ_z. Therefore, this code can correct any arbitrary error on one qubit. The quantum circuit works by splitting the qubits up into groups of three. Each group of three is treated as one qubit (like the repetition example) to correct for bit flips. Within each group of three, a control qubit goes through a Hadamard gate to change it's basis into the $|+\rangle, |-\rangle$ basis. This will protect the each group of three from phase errors.

After Peter Shor released his code and showed it was possible to protect against errors, this sparked interest to research other quantum errors codes. Many researchers started discovered different classes or families of codes that had different properties. One important class of codes are the Calderbank-Shor-Steane (CSS) codes. In 1996, Andrew Steane introduced a new CSS quantum error correcting code that is based on the classical [7,4,3] Hamming code.

Steane Seven Qubit Code

This section introduces the Steane 7-qubit code which uses seven physical qubits to encode one logical qubit. It was first introduced in 1996 by Andrew Steane [7]. To understand this code, we revisit the example of the classical error correction. Recall that in classical error, we have an encoding matrix G and a parity check matrix H satisfying $GH^T = 0$, with rank(G) + rank$(H) = n$. A message, m, is encoded to a codeword, c, by $mG = c$. If there is an error, e, then are codeword becomes $c + e$. We can extract the error syndrome by $(c+e)H^T = eH^T$. With the error syndrome obtained we can apply the corrective measure needed to change our codeback back to c.

For this quantum code, we will use the encoding matrix from our classical error correction example and we will devide the codewords into two sets, C_1 and C_1', given by

$$C_1 = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

and

$C_1' = C_1 + 1111111.$

The logical bits are defined as follows:

$|0\rangle_L = \frac{1}{\sqrt{8}} \sum_{v \in C_1} |v\rangle$

$|1\rangle_L = \frac{1}{\sqrt{8}} \sum_{w \in C_1'} |w\rangle$

Note that this corrects bit flip errors. To correct for phase flip errors we can use the Hadamard gate.

$H^{\otimes 7} |0\rangle_L = \frac{1}{\sqrt{2}}(|0\rangle_L + |1\rangle_L)$

$H^{\otimes 7} |1\rangle_L = \frac{1}{\sqrt{2}}(|0\rangle_L - |1\rangle_L)$

This Steane code is an example of a perfect CSS (Calderbank-Shor-Steane) code. It uses the classical binary self-dual [7,4,3] Hamming code to correct for qubit flip errors (X errors) and the dual of the Hamming code, the [7,3,3] code, to correct for phase flip errors (Z errors). The Steane code is able to correct arbitrary single qubit errors.

Figure 1.4 is the circuit diagram for the Steane code.

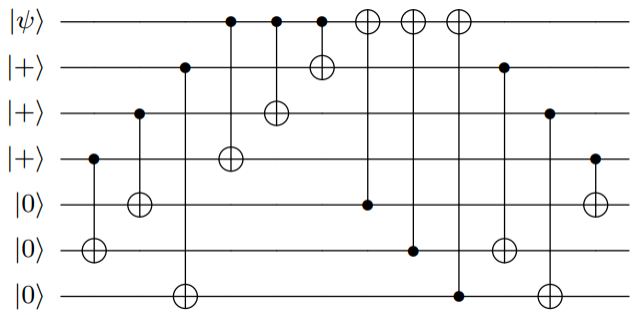

Figure 1.4: Circuit diagram for the Steane Seven Qubit Code

The next chapter will discuss how these codes can be used in practice and will cover different experiments that will measure their performance in real world scenarios.

2. METHODS

In this chapter we will present the environment used to simulate and test the different quantum error correcting codes.

2.1 Alfred Quantum Simulator

The Alfred Quantum Simulator is a quantum simulator that is written in C that can model the states of a quantum system. It keeps track of the state of a system using an array containing the 2^n complex coefficients of the basis states. Since there is not a native data type for complex numbers, each complex number is defined by a struct, which contains the real and imaginary parts.

There are several important functions within the simulator. One is the measurement function. Once a qubit is measured, if it was in a super position of $|0\rangle$ and $|1\rangle$, it was collapse down from the super position into either $|0\rangle$ or $|1\rangle$ based on the probabilities. Recall from (1.1), the coefficients govern the probability of observing a $|0\rangle$ or $|1\rangle$. After measurement, the quantum state is updated. Another important function is the apply gate function. This function allows you to apply any quantum gate to a specified qubit. The function also allows you to attach control points to other qubits. In that case the quantum gate is only applied if the current quantum state meets the control conditions.

Experiments are written in a simplified scripting language to express the different actions that can be performed to a quantum system, such as defining and applying different quantum gates and measuring different qubits. With this simulator, we can have precise control over the type of errors and the location the errors are introduced into the system. We will be able to model how errors are introduced and see how the state is corrected. For our experiments, we will be analyzing the Shor [[9,1,3]] code and the Steane [[7,1,3]] code.

2.2 Types of Errors

Before we discuss the results of the simulation experiments, let's discuss the nature of errors that occur in a quantum system and show how error correction can be effective. Recall from equation (1.3) that any single qubit operation can be expressed as a linear combination of the Pauli operations. Because an error on a qubit can be viewed as an operation, any error can be described in terms of the Pauli operations. The types of errors that can occur on a qubit are:

I - no error

X - bit flip error

Z - phase flip error

Y - bit and phase flip error

In actuality there are an infinite number of different errors that can occur on a qubit because the error can be a continuous linear combination of the four errors described above. Every general error E can be described as a superposition of this discrete set:

$$E = e_1 I + e_2 X + e_3 Z + e_4 Y \tag{2.1}$$

Because there could be an infinite number of different errors, one might think we would need to be able to correct each different error, which seems like an impossible task. However, because of the equation above, we actually only need to be able to correct X, Y, and Z errors. This is accomplished with syndrome measurements which are described in the next section.

2.3 Error Syndrome Measurements

Syndrome measurements were briefly discussed in the Three Qubit Flip Code section, however they are crucial for fault tolerant computing. Syndrome measurements use ancilla

qubits to perform a multi-qubit measurement that does not disturb the quantum information in the encoded state but reveals information about any errors that could have occured. The purpose of the error syndrome measurement is to take an arbitrary error (which maybe in a super position of the basis errors) that occurred on a qubit and measure in the basis of the discrete error set. The measurement then "forces" the error to collapse down into one of the four basis errors. After the measurement, we know exactly which qubit was affected and which error has occured so we can apply the correct Pauli operation to correct the error. The syndrome measurement tells us information about the error that has happened, but nothing about the value that is stored in the encoded qubits. Otherwise knowing the value would collapse any quantum superposition of the logical qubit with any other qubits in the quantum computer.

2.4 Types of Error Channels

Depolarization Channel

The depolarization channel is a model for noise in quantum systems. It can be described as saying that, with probability p an error occurs, while with probabilty $1-p$ there is no error. In the computational basis ($|0\rangle, |1\rangle$), the error can be any one of the three Pauli errors, X, Z, or Y. If an error occurs, each X, Z, or Y is equally likely to occur.

In this way we can simulate the noise in our experiements. During each quantum gate operation, the gate will have a small probability of introducing an error on the qubit after the gate has performed it's operation. If an error occurs, then either an X, Z, or Y error will occur on the qubit with equal probablity [8].

Dephasing Channel

The next type of channel is the dephasing channel, also called the phase-damping channel. In this type of channel, the qubits have a small probability of interacting with the environment and being kicked up into a higher energy level. Realistically, this might

be caused by some sort of heavy particle interacting with the system. Mathematically, we can represent this by saying σ_Z is applied with probability $p/2$ and nothing happens with probability $(1 - p/2)$ [8].

Amplitude Damping Channel

The amplitude damping channel models the dynamics of a system due to energy dissipation. In some cases a qubit might spontaneously emit a photon, causing the qubit to drop energy levels. We can represent this by saying with some probability p, a state of $|1\rangle$ would get bumped down to a state $|0\rangle$, and with probability $(1 - p)$ no change would happen [8].

3. EXPERIMENTS AND RESULTS

In this chapter we will present the different experiments that were run on the Alfred simulator and how each experiment was set up and configured in the simulator. At the end of the section we will present the results and observations from the simulations.

3.1 Experiments

3.1.1 Alfred Simulator Setup

In this section, different experiments will be discusses to show the performance of each error correcting code. To run experiments in the Alfred simulator, commands are written out in a script file. The simulator runs through the script line by line and executes each command. For the purpose of our experiments, each script file is broken down into different sections.

The first section is where we set up the environment of the quantum system. In this section we define our different quantum gates that we will be using, define how many qubits are in our system, and what the starting state is for the system. After the starting state has been defined we can now the gate function to execute different gates on qubits to manipulate the system.

The following sections in the script file follow the typical communication procedures when transmitting information over a noisy channel. There is the encoding section, where 1 logical qubit is encoded into several of the physical qubits. After that comes the transmission section where we simulate tranmitting the information over a noisy channel. This is the section where an error is most likely to be introduced into the system.

After the transmission section is the error detection and correction section. This is where is we use extra ancilla qubits to make error syndrome measurements. With those

error syndrome measurements, we are apply to apply the appropriate corrective procedure to reverse the effect of the error.

The final section, which is usually optional in real world scenarios, is the decoding section. The encoded qubits are decoded back into the original starting state.

Most of the experiments will look at what sort of errors can occur in the transmission section as well look into what happens when the gates themselves have some probability to introduce an error.

3.1.2 Types of Experiements

The first experiment was simulating the depolarization channel. The probability, p, of a gate introducing an error was varied from 0.02 to 0.005. Each run of the simulator was categorized as a success or failure depending on if the logical bits of the corrected state matched the initial state. To acquire an accurate success rate percentage, each experiment was run 100 times.

In the next experiment, p was held constant at 0.005 while the type and location of the error was changed. The purpose of this was to try and see if certain errors on certain qubits affected the success rate more than others.

While comparing success rates between the [[9,1,3]] and [[7,1,3]] code, the results did not match our intuiton, so a new [[7,1,3]] circuit was tested using double the amount of ancilla qubits. Instead of four control nots per ancilla qubit, there are two to try and minimize corrupted ancilla qubits. Then classical logic is used (gates that do not produce any sort of error) to apply an X gate to the correct logical qubit.

The next couple of experiments will showcase a different type of channel. Simulating a dephasing channel, with each gate, there is a small probability of introducing a phase error. We can model this by applying the σ_Z operation or by applying a small rotation

gate, $R_\phi = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}$, where ϕ is the phase shift. For each error that occured, ϕ could be a random number between 0 and $\pi/4$.

The final type of experiments tried to simulate the amplitude damping channel.

3.1.3 Set up for Shor Nine Qubit Code

The Shor [[9,1,3]] code uses 9 physical qubits to encode 1 logical qubit and 8 ancilla qubits to measure and correct any errors for a total of 17 qubits. The quantum gates that are used in this experient are the Hadamard gate, the X gate, and the controlled not gate.

Qubit 8 is the initial state that will be encoded into 9 qubits. Qubits 0 through 8 are the encoded qubits. Qubits 9 and 10 are the ancilla qubits used to correct Z errors that could occur on qubits 0, 1, or 2. Qubits 11 and 12 are the ancilla qubits used to correct Z errors that could occur on qubits 3, 4, or 5. Qubits 13 and 14 are the ancilla qubits used to correct Z errors that could occur on qubits 6, 7, or 8. Qubits 15 and 16 are the ancilla qubits used to correct X errors that could occur on any of the eight logical qubits.

Figure 3.1 shows a diagram of the decoding circuit of the Shor code. The logical and ancilla qubits are labeled.

3.1.4 Set up for Steane Seven Qubit Code

The Steane [[7,1,3]] code uses 7 physical qubits to encode 1 logical qubit and 6 ancilla qubits to measure and correct any errors for a total of 13 qubits. The quantum gates that are used in this experient are the Hadamard gate, the X gate, and the controlled not gate.

Qubit 6 is the initial state that will be encoded into 7 qubits. Qubits label 0 through 6 are the encoded qubits. Qubits 8, 9, and 10 are ancilla qubits that protect againt X errors. Qubits 11, 12, 13 are ancilla qubits that protect againt Z errors. Figure 3.2 is the diagram for the encoding circuit with the qubits labeled.

To detect and correct errors, the three ancilla qubits are configured in such a way as to

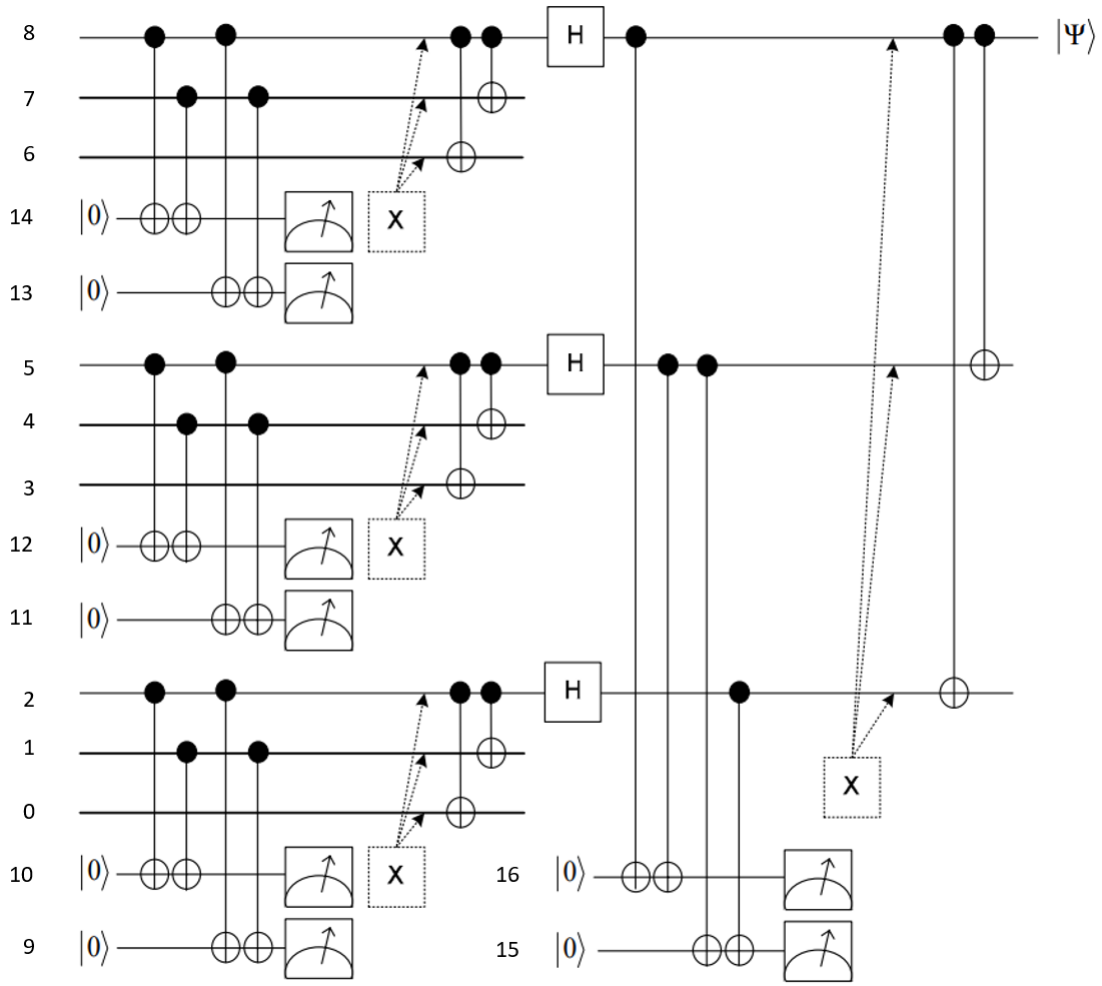

Figure 3.1: Decoding and error correcting circuit of the Shor code

enumerate all possible positions where an error can occur. Figure 3.3 shows diagram for the ancilla bits.

As explained before the Hadamard gate can change the basis of the state so that Z errors can be detected and corrected. When in the Hadamard basis, Z errors act as X errors, so we can use the same error correction circuit as shown in figure 3.3. Figure 3.4 shows the full error correction circuit.

As noted previously, during experimentation, it was noticed that the success rate of the [[7,1,3]] code wasn't outperforming the [[9,1,3]] code so a modified version of the [[7,1,3]]

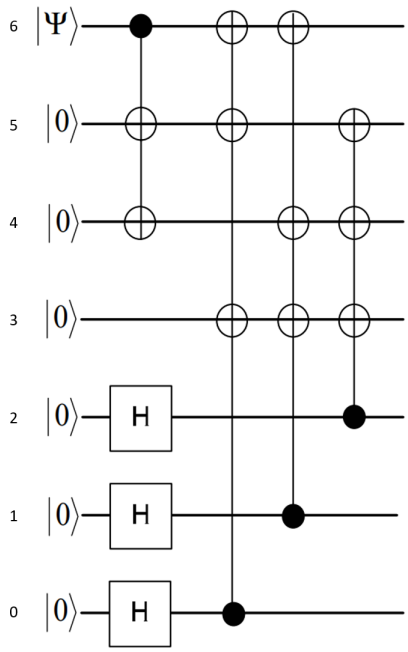

Figure 3.2: Steane encoding circuit

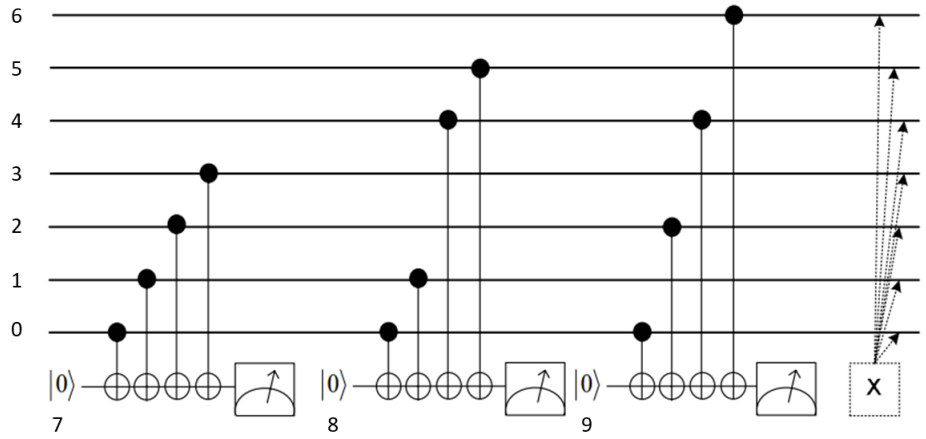

Figure 3.3: Steane error syndrome circuit

circuit was used with double the amount ancilla qubits. The same set of experiments was performed on this new circuit.

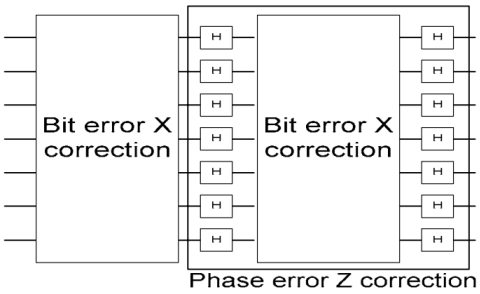

Figure 3.4: Full error correcting circuit for [[7,1,3]] Steane code

3.2 Results

3.2.1 Shor Nine Qubit Code

The results of the first experiment match our intuition where the success rate increases with a decrease in probability, p. Figure 3.5 shows the success rate of a Z error on the 6th qubit.

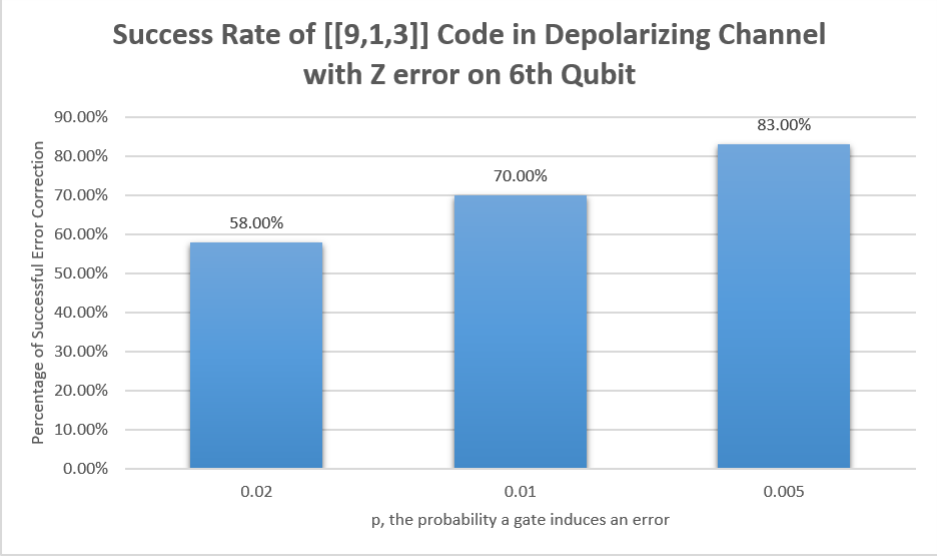

Figure 3.5: Success rate for [[9,1,3]] code in depolarizing channel

The results of the next experiment show the success rate of different type and location

of errors. From the simulations we did, it looks like X errors had a higher success rate than the Z errors, with an average success rate for X errors at and an average success rate for Z errors at.

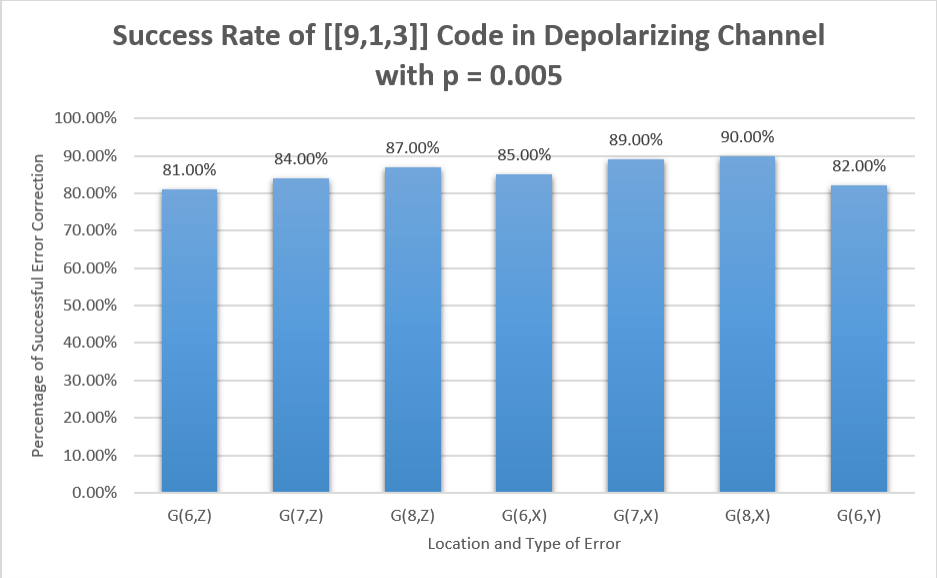

Figure 3.6: Success rate for [[9,1,3]] code in depoalizing channe with different type and locations of errors

The next couple of experiments show the success rate for the different types of channels. Figure 3.7 shows the success rate for a dephasing channel and 3.8 shows the success rate for a amplitude damping channel.

3.2.2 Steane Seven Qubit Code

Similar to the results of the [[9,1,3]] code, we can see the success rate of the [[7,1,3]] code increase when p is decreased. Figure 3.9 shows the results with an X error on the 5th qubit. Figure 3.10 shows the success rate for the [[7,1,3]] code for different types and location of errors. The next two figures show the experiments simulating different quantum channels. Figure 3.11 shows the dephasing channel and figure 3.12 shows the

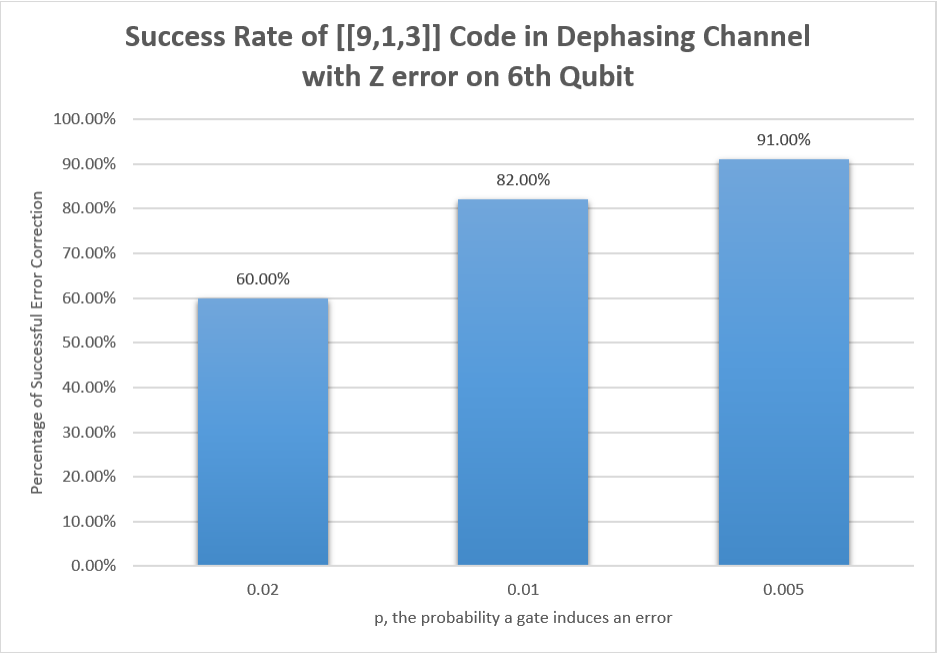

Figure 3.7: Success rate for [[9,1,3]] code in dephasing channel

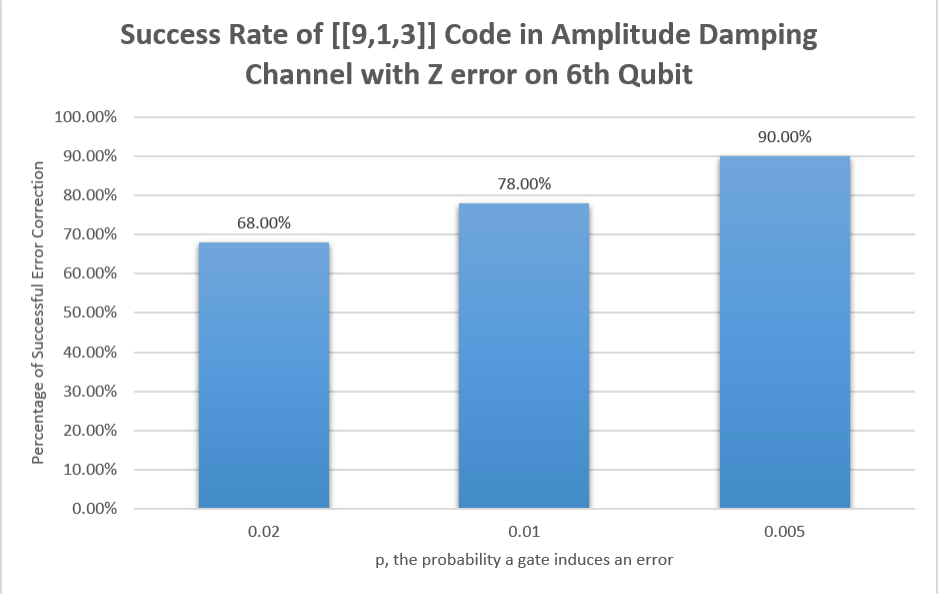

Figure 3.8: Success rate for [[9,1,3]] code in amplitude damping channel

amplitude damping channel. As discussed previously a second [[7,1,3]] circuit with double the amount of ancilla qubits was tested to see if the success rates were changed.

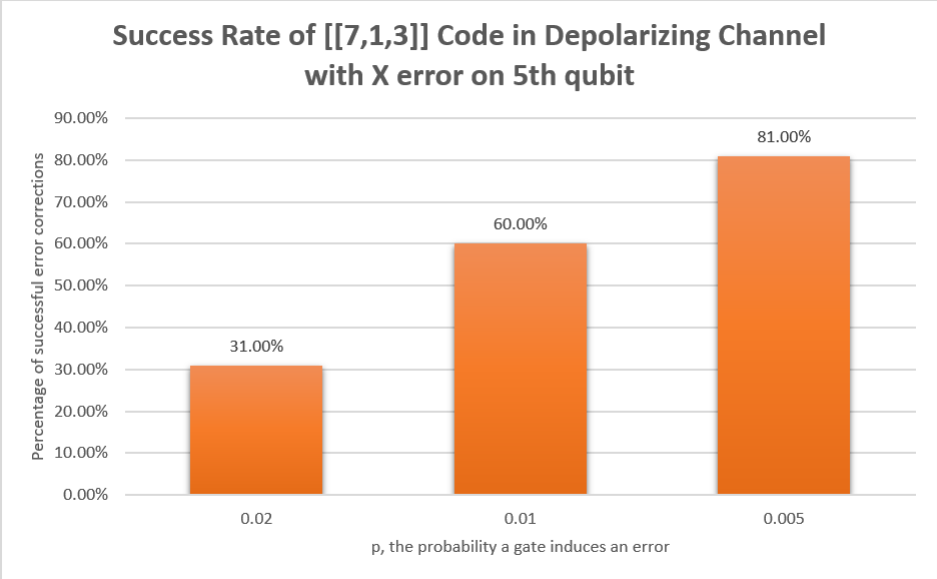

Figure 3.9: Success rate for [[7,1,3]] code in depolarizing channel

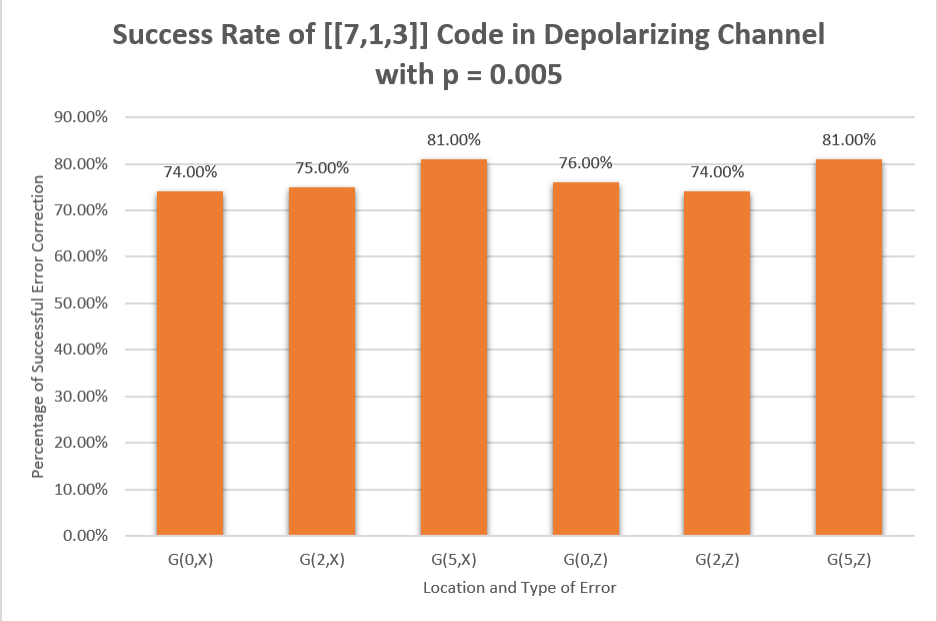

Figure 3.10: Success rate for [[7,1,3]] code in depoalizing channe with different type and locations of errors

Figure 3.13 compares the results between the two circuits.

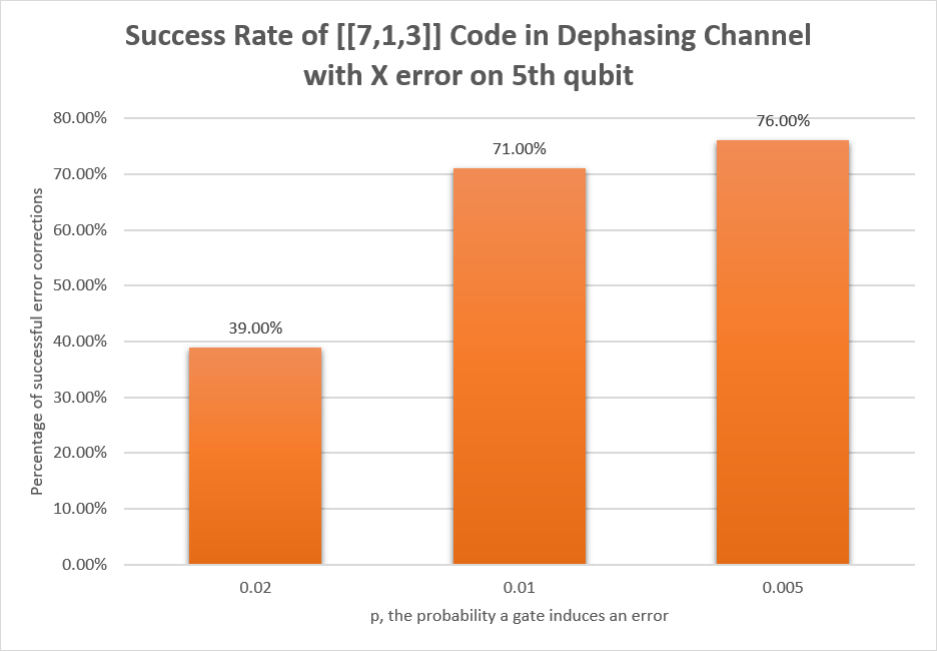

Figure 3.11: Success rate for [[7,1,3]] code in dephasing channel

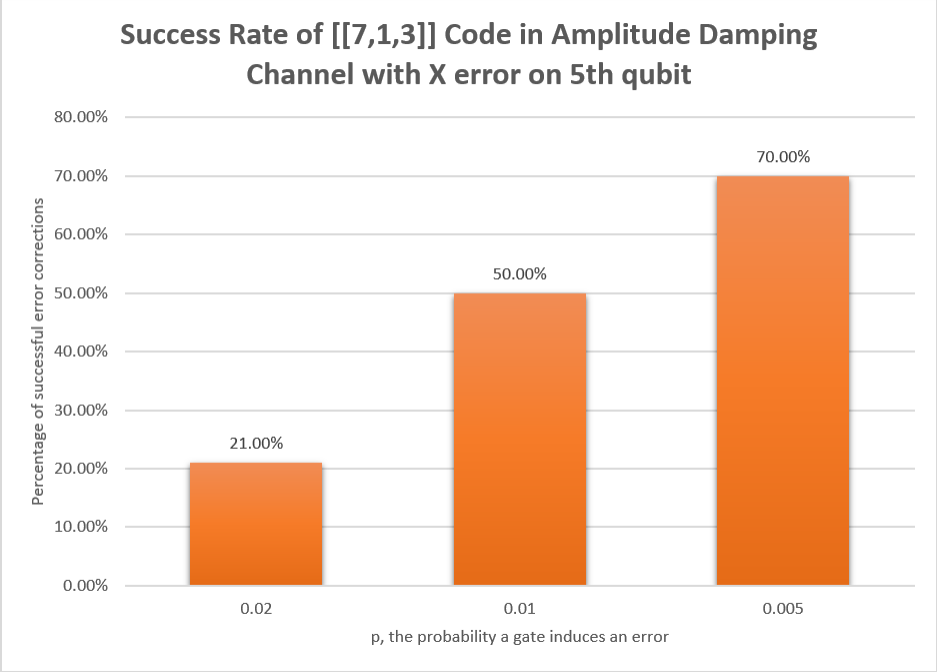

Figure 3.12: Success rate for [[7,1,3]] code in amplitude damping channel

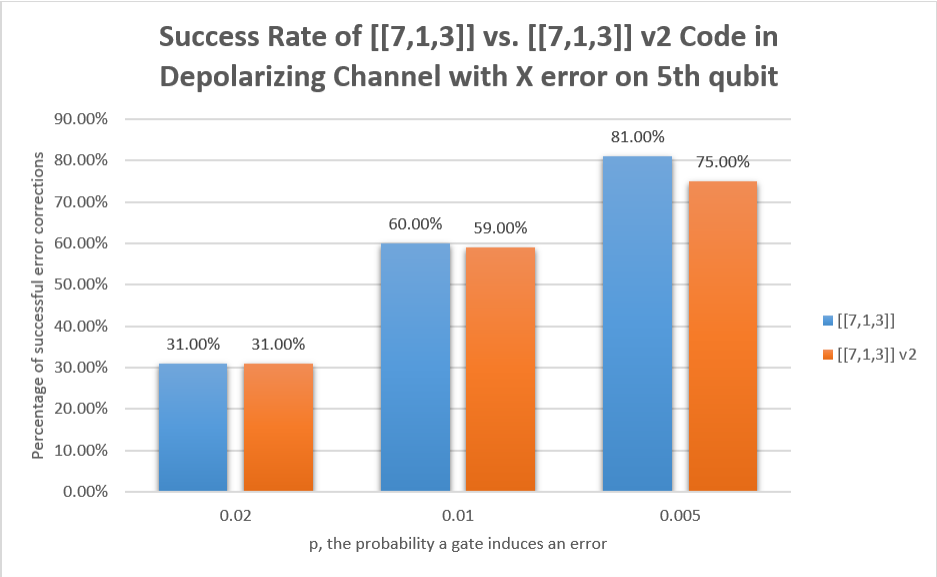

Figure 3.13: Success rate for [[7,1,3]] vs [[7,1,3]] v2 code in depolarizing channel

3.3 Observations

Unsurprisingly, we see that in both codes the success rate increases as p decreases. The relationship between the success rate and p seems to be nonlinear, probably logarithmic. The success rate seems to increase linearly however the change in p is not linear, it is halved each time.

For the [[9,1,3]] code it seemed like the type and position of the error did not have a big effect on the success rate. Figure 3.6 shows that most of the errors are in the 80% range. With the highest at 90%, the lowest at 81%, and the average at 85.4%, this gives a range of about $\pm$ 5%, which is most likely within the margin of error for the simulations. If the margin of error is smaller than $\pm$ 1%, then we can see there is a slight correlation between the location and the success rate. It appears the success rate between location 6, 7, and 8 increases a couple percent for each location respectively. One possible reason that location is the highest is because it has two separate syndrome measurements associated with it. So perhaps each measurement will help increase the success rate.

Similar results are displayed in the [[7,1,3]] code. From figure 3.10, the location and type of error do not seem to matter too much, as they are all around the upper 70%. The lowest is 74%, the highest is 81%, and the average is 76.8%. This gives a range of about $\pm$ 3%, which is most likely within the margin of error for the simulations. The type of error does not seem to matter because accounting for the location, the difference in success rate for the type is only 1%. However, the location of the error could possibly affect the success rate. We can see that at location 5, there is a higher success rate by about 6% (for both errors) than compared to locations 0 and 2.

When analyzing the different quantum channels, we can see which channels provide higher success rates for different p's. Unsurprisingly, the depolarizing channel shows the worst performance. For both the [[9,1,3]] code and the [[7,1,3]] code, the depolarizing channel has the lowest success rate for all the values of p. This is to be expected according to the theoretical results, because the depolarizing channel takes a quantum state to a maximally mixed state. The next best performing channel according to our experiments was the dephasing channel. This is most likely because there is only one type of error that is occurring and often times a phase error will not affect the bit value of the qubit. The best performing channel was the amplitude damping channel. This is most likely because the error essentially only affects half the qubits, the qubits that are in the $|1\rangle$ state.

4. SUMMARY AND CONCLUSIONS

This chapter will summarize the results of this research and discuss the implications on the field of quantum computing. Then we will discuss some improvements and the future direction of this research.

4.1 Discussion of Results

In this research, we explored how error correcting codes can help a quantum system mitigate the effects of errors. We exposed two different error correcting codes, the [[9,1,3]] Shor code and the [[7,1,3]] Steane code, to different types of errors and documented the changes in the quantum state. With the comparison between the [[9,1,3]] code and the [[7,1,3]] code, we can get a better understanding of how and why these codes might be implemented in a real quantum computer.

Overall most the results from the experiments line up with the theoretical results. I believe the experiments produced interesting results that can give a little bit of insight into how errors can affect a quantum state. One interesting result was the [[9,1,3]] code seemed to offer better resistance to errors compared to the [[7,1,3]] code. This is most likely attributed to using more ancilla qubits on the [[9,1,3]]. Each ancilla qubit interacts with fewer qubits in the [[9,1,3]], which provides better protection on the ancillas by two seperate methods. One, fewer interactions with the ancilla qubit bit reduce the probabilty of corrupting the ancilla qubit itself and two, with fewer qubits governing the logic of the ancilla qubit, there is a smaller probability of being a logical error. This suggests that it is best to use ancilla qubits on small local areas. Another result that was interesting is that even with gates that have a 0.5 % chance of introducing an error, the overall success rate is only 80 %, and this is only with 1 logical qubit. I suspect with more qubits actually performing some calculations would have a much lower success rate. The results suggest that

there will need to be considerable effort in making sure gate operations are near perfect.

4.2 Improvements and Future Work

There are several areas where we can make improvements to this research. One improvement I would make to the experiments is to modify the simulator scripts to be able to run many and be able to read the results. As it stands each experiment runs the simulator only once. So to gather the results for the each experiment, the simulator had to be run manually 100 times and the results had to be manually tabulated. A much more efficient method to gather the results would be to programmatically run the simulator 1000s of times for the same experiment. This would also give a much more accurate success rate for a given experiment.

Another area for improvement would be to conduct the experiments on an actual quantum computer. There are real quantum computers that are open to the public to run experiments on. One example is the IBM quantum devices. Today, IBM has several real quantum devices and simulators available for use through the cloud. These devices are accessed and used through Qiskit, and open source quantum software development kit, and IBM Q Experience, which offers a virtual interface for coding a quantum computer.

Another area that needs improvement would be to build some circuits that actually compute something meaningful, even if the computation is simple. These circuits would need to incorporate more than one qubit to do so and there are a lot of challenges to do so. Classical quantum simulators are limited to around 25-30 qubits because of the computation and memory requirements for a large amount of qubits. Each qubit added doubles the space required to store the state information. In order to protect against errors, each logical qubit takes 7 – 9 physical qubits plus extra ancilla qubits. This limited number of qubits would be hard to do computations with.

Because of the conclusion showing that the [[9,1,3]] code provides better protection

on the ancilla qubits, I believe our future work would focus on finding and testing Low Density Parity Check codes. LDPC codes by definition have sparse parity check matrices. This means that for each data qubit there are fewer ancilla check qubits. There have been several publishings into construction techniques of quantum LDPC codes such as [9] and [10], so further research looks promising.

REFERENCES

[1] R. P. Feynman, "Simulating physics with computers," *International journal of theoretical physics*, vol. 21, no. 6, pp. 467–488, 1982.

[2] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proceedings 35th annual symposium on foundations of computer science*, pp. 124–134, Ieee, 1994.

[3] S. J. Devitt, W. J. Munro, and K. Nemoto, "Quantum error correction for beginners," *Reports on Progress in Physics*, vol. 76, no. 7, p. 076001, 2013.

[4] A. Wei, "Peter shor on quantum error correction." https://windowsontheory.org/2018/12/07/quantum-error-correction/. Accessed: 2019-03-15.

[5] R. Baumann, "Quantum error correction (qec)," 2003.

[6] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Physical review A*, vol. 52, no. 4, p. R2493, 1995.

[7] A. Steane, "Multiple-particle interference and quantum error correction," *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 452, no. 1954, pp. 2551–2577, 1996.

[8] J. Preskill, "Quantum information chapter 3." University Lecture Notes, October 2018. URL: http://www.theory.caltech.edu/people/preskill/ph219/chap3_15.pdf.

[9] M. S. Postol, "A proposed quantum low density parity check code," *arXiv preprint quant-ph/0108131*, 2001.

[10] T. Camara, H. Ollivier, and J.-P. Tillich, “Constructions and performance of classes of quantum ldpc codes,” *arXiv preprint quant-ph/0502086*, 2005.