Universidade do Minho
Escola de Engenharia
Departamento de Informática

José Eduardo Moreira Barros Pereira

Quantum Error-Correcting Codes

November 2021

Universidade do Minho
Escola de Engenharia
Departamento de Informática

José Eduardo Moreira Barros Pereira

Quantum Error-Correcting Codes

Master dissertation
Master Degree in Physics Engineering

Dissertation supervised by
José Carlos Bacelar Ferreira Junqueira de Almeida
José Pedro Miranda Mourão Patrício

November 2021

STATEMENT OF INTEGRITY

ACKNOWLEDGEMENTS

To keep it short and sweet I would like to thank both my supervisors Prof.José Bacelar and Prof.Pedro Patrício, my parents and sister and my colleagues and friends. Without their help, this journey would have been so much more difficult. Wish you all the best, forever.

ABSTRACT

Quantum computing is a new and exciting field of research that, using the properties of quantum mechanics, has the potential to be a disruptive technology, being able to perform certain computations faster than any classical computer, such as Shor's factorization algorithm and Grover's algorithm. Although there are several quantum computer with different underlying technologies, one of the main challenges of quantum computation is the occurence of errors, destroying the information and making computation impossible. Errors may have several different sources namely, thermal noise, faulty gates or incorrect measurements. The present dissertation aims to study and employ methods for reducing the effects of errors during quantum computation and correct them using Stabilizer Codes, which are a very powerful tool to produce circuit encoding networks that can, in theory, protect quantum systems from errors during transmission. A proof of concept algorithm was implemented using Qiskit, a Python based program development language for the IBM Q machines, and tested on both simulators and real systems. The algorithm is capable of, given any stabilizer in standard form, generate the circuit encoding network. Due to technological limitations associated with current quantum computers the results obtained in ibmq_guadalupe fail to show the efficacy of Stabilizer Codes.

Keywords— Quantum Computing, Quantum Error Correction, Stabilizer Codes, IBMQ

RESUMO

A computação quântica é uma área de investigação recente que, usando as propriedades da mecânica quântica, tem o potencial de ser uma tecnologia disruptiva, sendo capaz de realizar alguns tipos de computação de forma mais rápida do que qualquer outro computador clássico atual, tais como, o algoritmo de fatorização de Shor e o algoritmo de procura de Grover. Apesar de já existirem vários computadores quânticos com tecnologias de diferentes modos de operação, um dos principais desafios que a computação quântica enfrenta é a existência de erros, destruindo a informação presente e impossibilitando a computação. Os erros podem ser de várias fontes, nomeadamente, ruído térmico, operações deficientes ou medidas incorrectas. Esta dissertação tem como objectivo estudar e aplicar métodos para reduzir os efeitos dos erros durante a computação quântica e corrigi-los usando códigos estabilizadores, que são uma ferramenta poderosa para produzir circuitos que podem, em teoria, proteger sistemas quânticos de erros ocorridos durante a transmissão. Foi implementado um algoritmo usando Qiskit, uma linguagem à base de Python usada para desenvolver programas nas máquinas da IBM, que foi testado em simuladores e em sistemas físicos. O algoritmo é capaz de, dado um estabilizador na sua forma *standard*, gerar o circuito codificador. Devido a limitações da tecnologia associadas aos atuais computadores quânticos, os resultados obtidos na máquina ibmq_guadalupe não demonstram a eficácia dos códigos estabilizadores.

Palavras-chave— Computação Quântica, Correcção de Erros Quânticos, Códigos Estabilizadores, IBMQ

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

In recent years there's been an increasing interest in using the properties of quantum systems to perform computation. Shor's work Shor (1995b) proved that a quantum computer not only could be used to perform complex calculations but also could, in principle, compute certain types of problems, namely prime factorization Shor (1994), exponentially faster than classic computers. Quantum computation also seems to be usefull in the areas of quantum criptography Bennett et al. (1992), quantum internet Kimble (2008), quantum metrology Giovannetti et al. (2004).

The biggest limiting factor of quantum computation appears to be the difficulty of eliminating errors in the data caused by inaccuracy and decoherence Unruh (1995), regardless of the underlying technology used, be it semiconductor qubits Chatterjee et al. (2021), photonic qubits Adami and Cerf (1998) or superconducting qubits Kelly et al. (2015). In the storage and transmission of data, errors can be corrected by using error-correcting codes. However, unlike classical bits, quantum bits cannot be cloned Wootters and Zurek (1982), therefore redundancy, i.e., copies of the same qubit, cannot be used to detect and correct if an error occured. This means that new methods need to be created to make sure the information stored in the quantum bits is reliable. Such methods have been developed by Shor Shor (1995b) to create a code that protects one qubit of information from all possible errors by transforming one qubit of information into an encoded state containing 9 qubits. Smaller codes have since been discovered, with the smallest perfect code containing only 5 qubits to encode 1 qubit of information Laflamme et al. (1996). Although powerfull, this codes were not derived from pure methodology, unlike Stabilizer Codes Gottesman (1997) which can be derived to accommodate any situation.

The main objective of this thesis is to study classical error-correcting codes and their quantum counterpart and to implement these quantum algorithms on one of IBM's Quantum System One machines to try to replicate their performance on a real machine.

Chapter 2 will consist of classical information theory by Shannon. Chapter 3 provides a review of classical error correcting codes. Chapter 4 details the basic notions of quantum computing like qubits, gates and quantum circuits. Chapter 5 details the basic properties of a quantum code and define the Stabilizer formalism. Chapter 6 provides the results for the encoding of the 5 qubit error correction stabilizer code using Qiskit and further discussion. Finally, Chapter 7 summarizes the dissertation and guidelines for future work are proposed to extend the solutions presented throughout this dissertation.

2

CLASSICAL INFORMATION THEORY

The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point (Shannon, 1948). This message is correlated to some system containing a set of different messages. The system must be designed to operate for each possible selection, not just the one which will actually be chosen since this is unknown at the time of design. A general communications system is shown in Fig.1 and can be divided in 5 parts:

- **Source** (usually refered to as Alice): produces a message or sequence of messages to be communicated to the receiving terminal;
- **Transmitter**: operates on the message in some way to produce a signal suitable for transmission over the channel;
- **Channel**: *medium* used to transmit the signal from transmitter to receiver.
- **Receiver**: ordinarily performs the inverse operation of that done by the transmitter, reconstructing the message from the signal;
- **Destination** (usually refered to as Bob): person for whom the message is intended.

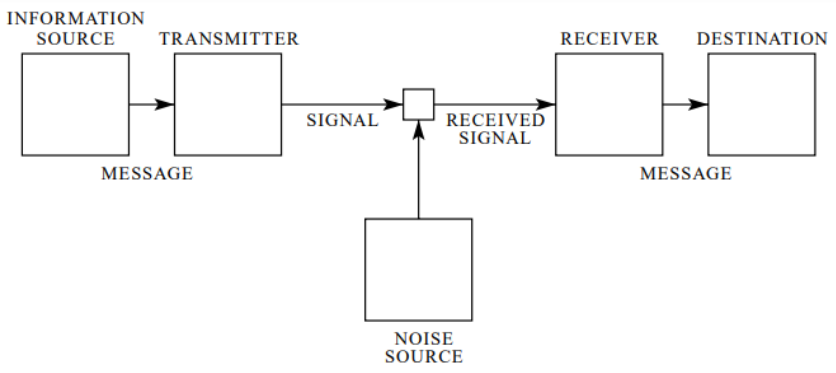


Figure 1: General communication system

In a more mathematical approach, a source can be seen as an ordered pair $\mathcal{S} = (S, P)$, where $S = \{x_1, \dots, x_n\}$ is a finite set, known as the source alphabet, and P is a probability distribution on S. We denote the probability of x_i by p_i. The probability distribution gives us a notion of uncertainty, meaning if $p_i = 1 \implies p_j = 0, \forall j \neq i$. In other words, if only one word is to be sent then we have no uncertainty and also no information from the source. On the other hand, if all words have the same probability of being sent, $p_i = \frac{1}{n}, \forall i = 1, \dots, n$, then the uncertainty is maximal and so is the information. This being said, it is important to quantify the amount of information associated with a source. This is done using the entropy function H given by:

$$H(p_1, \ldots, p_n) = -\sum_{i=1}^{n} p_i log(p_i) \tag{1}$$

Perhaps the simplest example regarding the entropy function is one where the source has two possible symbols with probabilities p and $q = 1 - p$. Its entropy function is given by $H = -(plog(p) + qlog(q))$ and is plotted in Fig.2. As previously stated, the maximum entropy is achieved when all possible messages are equiprobable, $p = q = \frac{1}{2}$.

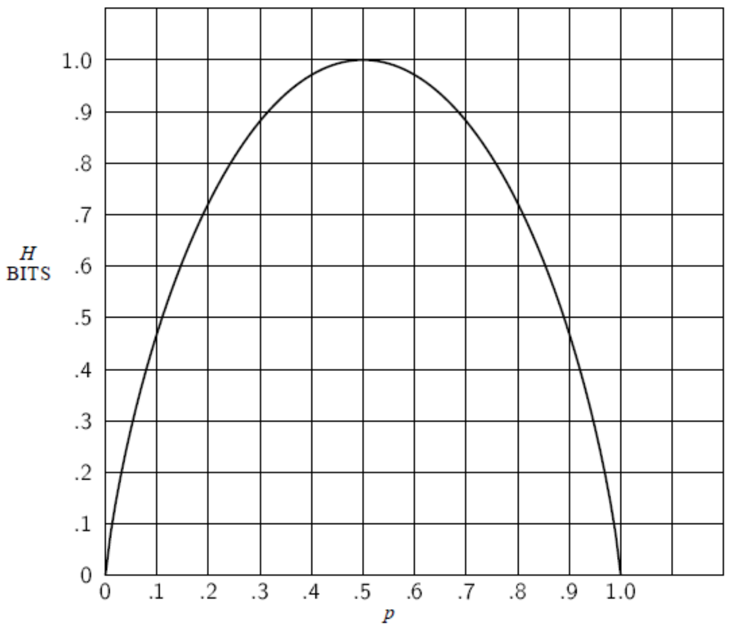


Figure 2: Entropy in the case of two possibilities p and $1 - p$

2.1 NOISELESS CODING THEOREM

Most communications sytems contain some amount of redundancy, meaning not all characters in the alphabet $\mathcal{S}$ are required to read a given message accurately. The Noiseless Coding theorem states that by clever encoding, we can use this redundancy to reduce the average codeword length to a minimum value of the entropy, regardless the nature of the source symbols.

Example 1 *Consider a source which produces a sequence of letters chosen independently from* $\mathcal{S} = (A, B, C, D)$ *with* $P = (\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8})$. *The entropy, H, is given by:*

$$\begin{aligned} H &= -\frac{1}{2}log\frac{1}{2} + \frac{1}{4}log\frac{1}{4} + \frac{2}{8}log\frac{1}{8} \\ &= \frac{7}{4} \end{aligned}$$

Thus we can encode each codeword in $\mathcal{S}$ into binary digits with an average of $\frac{7}{4}$ binary digits per symbol. A common encoding for such a system would be $A = 00; B = 01; C = 10; D = 11$ where the average codeword length is 2 bits. However, according to the Noiseless Coding Theorem, a more efficient coding can be achieved. The encoding that can achieve maximum entropy is:

$$A = 0; B = 10; C = 110; D = 111$$

2.2 NOISY CODING

In the case where the message is subject to noise during transmission the Noisy Coding Theorem serves to quantify the redundancy needed to incorporate into the message in order to correct all of the errors introduced by noise. Consider a binary noisy communication channel has an error probablity of q, that is, each bit has probability q of being swaped (0 to 1 and vice-versa). The average number of wrong bits sent in any given message of length N_0 is therefore qN_0. The number of possible ways in which qN_0 errors can be distributed among the N_0 message bits is:

$$E = \frac{N_0!}{(qN_0)!(N_0 - qN_0)!} \tag{2}$$

In order to correct the errors, Bob needs only to know the positions in which they occured, flipping them. He can do so through a correction channel as seen in fig.4, needing at least $log(E)$ bits.

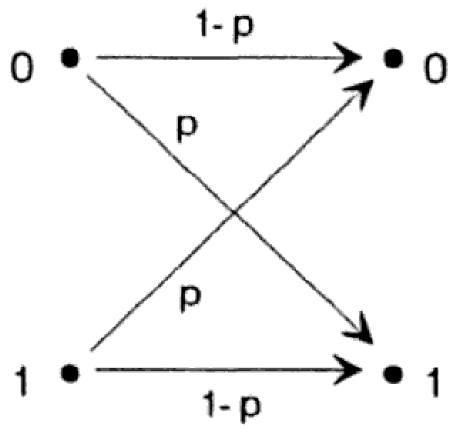


Figure 3: Communication in a noisy channel (from Roman (1992))

Using Stirling approximation Dutka (1991) gives:

$$logE \approx N_0[-qlogq - (1-q)log(1-q)] = N_0 H(q) \tag{3}$$

where $H(q)$ is the entropy associated with the probability of a single bit error. It follows that at least $N_0[1 + H(q)]$ bits are required in the combined original signal and correction channels if the corrupted message received by Bob is to be corrected. This result assumes no errors occured in the correction channel during the transmission. If this channel is itself noisy, a second correction on

$qN_oH(q)$ bits is required using a minimum of $N_0H^2(q)$, following a third, fourth and so on. The total number of required bits is:

$$N = \frac{N_0}{1 - H(q)} \tag{4}$$

Summarizing, it is possible to faithfully enconde 2^{N_0} messages using $N = \frac{N_0}{1-H(q)}$ bits for error correction. This result is Shannon's noisy-channel coding theorem for the binary channel.

Encoding different messages as bit strings, or *codewords*, must be made such that the messages are distinguishable after passing through the noisy channel. Examples of efficient coding schemes are shown in section 3.

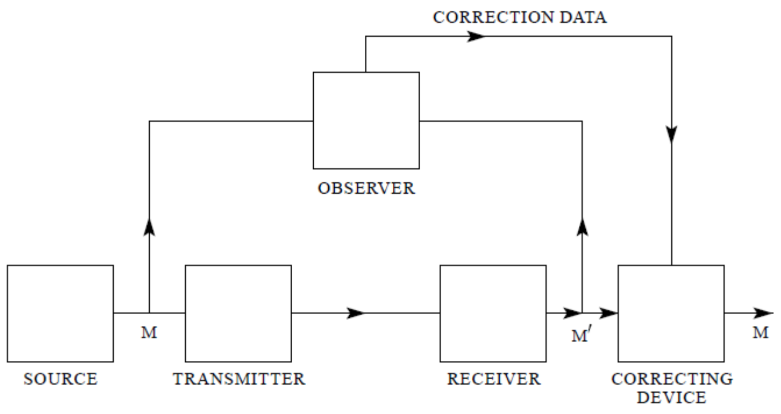


Figure 4: Communication system with correction

3

CLASSICAL ERROR-CORRECTING CODES

A code is a set of messages called codewords that can be transmitted between two parties. An error-correcting code is a code for which it is sometimes possible to detect and correct errors that occur during transmission of the codewords. Some applications of error-correcting codes include correction of errors that occur in information transmitted via Internet, data stored in a computer, etc. A binary code is a non-empty subset of $\mathbb{Z}_2^n$. It is linear if it forms a subspace of $\mathbb{Z}_2^n$. It's usually described by the parameters $[n, k]$, wherein n refers to the length of the codewords and k to the dimension of the vector space. To correct a received vector it's only necessary to use the nearest neighbor policy, i.e., assume the fewest possible number of errors, and correct the received vector to the codeword from which it differs in the fewest positions. This method is limited, for there is not always a unique codeword that corrects a received vector.

Definition 1 *Let $\mathcal{C}$ be a code in $\mathbb{Z}_2^n$. For any vectors $x, y \in \mathcal{C}$, the Hamming distance $d(x, y)$ is defined as:*

$$d(x,y) = \sum_{i=1}^{n} |x_i - y_i|, \tag{5}$$

where the sumation is considered over $\mathbb{Z}$. The smallest Hamming distance between any two distinct codewords in a code $\mathcal{C}$ is called the minimum distance of $\mathcal{C}$, denoted $d(\mathcal{C})$ or simply d.

Definition 2 *For $x \in \mathbb{Z}_2^n$ and $r \in \mathbb{Z}^+$, let*

$$S_r = \{y \in \mathbb{Z}_2^n | d(x,y) \leq r\} \tag{6}$$

$S_r(x)$ is called the ball of radius r centered in x.

Let $\mathcal{C}$ be a code with minimum distance d, and let t be the largest integer such that $t < \frac{d}{2}$. Then $S_t(x) \cap S_t(y) = \emptyset, \forall x, y \in \mathcal{C}, x \neq y$. Furthermore, if z is a received vector in $\mathbb{Z}_2^n$ with $d(u, z) \leq t$ for some $u \in \mathcal{C}$ then $z \in S_t(u)$ and $z \notin S_t(v), \forall v \in \mathcal{C} \setminus \{u\}$. That is, if a received codeword $z \in \mathbb{Z}_2^n$ differs from a codeword $u \in \mathcal{C}$ in t or fewer positions, then every other codeword in $\mathcal{C}$ will differ from z in more than t positions. Thus, the nearest neighbor policy will always allow t or fewer errors to be corrected in the code. The code $\mathcal{C}$ is said to be *t-error correcting.*

The Hamming bound is an upper bound for the number of codewords in a code Hoffman et al. (1991) of length n and distance $d = 2t + 1$.

Theorem 3.0.1 (Hamming Bound) *Suppose $\mathcal{C}$ is a t-error correcting code in $\mathbb{Z}_2^n$. Then:*

$$|C| \cdot \sum_{i=0}^{t} \binom{n}{i} \leq 2^n, \tag{7}$$

where $|C|$ denotes the number of codewords.

A code $\mathcal{C}$ is said to be perfect if $|C| \cdot \sum_{i=0}^{t} \binom{n}{i} = 2^n$, meaning every vector in $\mathbb{Z}_2^n$ is correctable.

Definition 3 *Let $\mathcal{C}$ be an $[n, k]$ linear code. A matrix H with the property that $Hc^T = 0 \iff c \in \mathcal{C}$ is called the parity check matrix for $\mathcal{C}$.*

Theorem 3.0.2 *Let $\mathcal{H}$ be a parity check matrix for a linear code $\mathcal{C}$. Then $\mathcal{C}$ has distance d if and only if any set of $d-1$ columns of $\mathcal{H}$ is linearly independent, and at least one set of d columns of $\mathcal{H}$ is linearly dependent.*

3.1 HAMMING CODES

Hamming Codes are a class of perfect, linear, easy to decode, 1-error correcting codes. In order to construct a Hamming code $\mathcal{C}$ with parameters $[n, k]$, first we construct the parity check matrix $\mathcal{H}$ consisting of all nonzero vectors of length k.
The rows of a generator matrix $\mathcal{G}$ are just a basis for the null space of $\mathcal{H}$. Lastly, the codewords in $\mathcal{C}$ are formed by multiplying all binary vectors of length k by $\mathcal{G}$. The resulting code is a $[n, k]$ linear code with 2^n codewords capable of correcting one error.

Example 2 *Consider the construction of the $[7, 4]$ Hamming code. The parity check matrix for this code is given by:*

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

Solving the equation $Hx = 0$ yields a generator matrix G. In this case G is:

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

Finally, to construct the codewords in $\mathcal{C}$ we multiply all k-length binary words, ranging from (0000) to (1111), by $\mathcal{G}$. For instance, the binary word (1000) is coded in $\mathcal{C}$ as (1110000) = (1000)·G.

Because the columns in H are the binary representations of the numbers up to $2^k - 1$, excluding 0, any two columns are distinct and the minimum number of linearly dependent columns is 3.

Therefore, by Theorem.3.0.2, a Hamming code has distance $d = 3$. For $n = 2^k - 1$ and $d = 2t + 1 = 3$ (so $t = 1$),

$$|\mathcal{C}| \cdot \sum_{i=0}^{t} \binom{n}{i} = |\mathcal{C}| \cdot$$

This proves that all Hamming codes are perfect, one error correcting codes.

Hamming codes present a very simple method for detecting errors in received vectors that occur from codewords in linear codes constructed using generator matrices. For a linear code $\mathcal{C}$ with parity check matrix $\mathcal{H}$, $Hc^t = 0 \iff c \in \mathcal{C}$. The problem of correcting errors in a received vector is also very simple for the Hamming Code. First we consider that any received vector r can be decomposed as $r = c + e$, where $c \in \mathcal{C}$ and e refers to the error vector. Hence $Hr^t = Hc^t + He^t = He^t$. Because Hamming codes are one-error correcting and perfect, the only type of error vector, e, we should consider are the vectors e_i that contain a one in the i^{th} position and the rest is all zeros. If $r \notin C \iff Hr^t \neq 0$, meaning Hr^t is equal to one of the columns of H. Suppose it's equal to the j^{th} column. This means that the error in r is e_j. Note that since the j^{th} column in H is the binary expression of the number j, then the error in r is e_j.

3.2 BCH CODES

Hamming codes are only one-error-correcting. If more than one error occurs during the transmission of a Haming codeword, the received vector will not be corrected to the sent codewod. BCH codes are linear codes and can be constructed to be multiple-error-correcting. These codes present several features that make them very important, namely:

- good error-correcting properties when the length is not too big;
- relatively easy encoding and correction scheme;
- provide a good foundation upon which to base other families of codes

Furthermore they are quite extensive, meaning, for any positive integers r and t with $t \leq 2^{r-1} - 1$, there is a BCH code of length $n = 2^r - 1$ which is t-error correcting and has dimension $k > n - rt$.

Definition 4 *$\mathcal{C}$ is a cyclic code if $(a_0 a_1 \ldots a_{n-1}) \in \mathcal{C} \iff (a_1 \ldots a_{n-1} a_0) \in \mathcal{C}$*

It is convenient to represent cyclic codes in terms of polynomials. A polynomial of degree n over $\mathbb{K}$, where $\mathbb{K}$ is a field, is a polynomial $a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$, where the coefficients $a_0, \ldots, a_n, a_n \neq 0$, are elements of $\mathbb{K}$. The set of all polynomials over $\mathbb{K}$ is denoted by $\mathbb{K}[x]$ and forms a ring. Elements of $\mathbb{K}[x]$ will be denoted by $f(x), g(x), p(x)$ and so forth. The polynomial $f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$ of degree at most $n-1$ over $\mathbb{K}$ may be regarded as the codeword $v = a_0 a_1 a_2 \ldots a_{n-1}$ of length n in $\mathbb{K}^n$. Thus a code $\mathcal{C}$ of length n can be represented as a set of polynomials over $\mathbb{K}$ of degree at most $n-1$.

Lemma 3.2.1 *Let $\mathcal{C}$ be a cyclic code and let $v \in \mathcal{C}$. Then for any polynomial $a(x), c(x) = a(x)v(x) mod(1 + x^n)$ is a codeword in $\mathcal{C}$.*

We define the generator polynomial of a linear cyclic code $\mathcal{C}$ to be the unique monic polynomial of minimum degree in $\mathcal{C}$.

Theorem 3.2.2 *Let $\mathcal{C}$ be a cyclic code of length n and let $g(x)$ be the generator polynomial. If $n - k = degree(g(x))$ then:*

- *$\mathcal{C}$ has dimension k;*
- *The codewords corresponding to $g(x), xg(x), \ldots, x^{k-1}g(x)$ are a basis for $\mathcal{C}$;*
- *$c(x) \in \mathcal{C} \iff c(x) = a(x)g(x) \exists a(x) : degree(a(x)) < k$*

Theorem 3.2.3 *$g(x)$ is the generator polynomial for a linear cyclic code of length n if and only if $g(x)$ divides $1 + x^n$*

Corollary 3.2.3.1 *The generator polynomial $g(x)$ for the smallest cyclic code of length n containing the polynomial $v(x)$ is the greatest common divisor of $v(x)$ and $1 + x^n$.*

The simplest generator matrix for a linear cyclic code is the matrix in which the rows are the codewords corresponding to the generator polynomial multiplied by the first $k-1$ powers of x.

$$\mathcal{G} = \begin{pmatrix} g(x) \\ xg(x) \\ x^2g(x) \\ \dots \\ x^{k-1}g(x) \end{pmatrix} \tag{8}$$

Let $\mathcal{C}$ be a linear cyclic code $[n,k]$ with generator polynomial $g(x)$. Encoding a message consists simply of polynomial multiplication; that is, the message polynomial $a(x)$ is encoded as $a(x)g(x) = c(x)$, resulting in the codeword polynomial $c(x)$. Instead of storing the entire $k \times n$ generator matrix, it is only necessary to store the generator polynomial, which is a significant improvement in terms of the complexity of encoding. Performing the inverse operation (decoding) is achieved by dividing the received codeword $c(x)$ by $g(x)$, yielding the original message polynomial $a(x)$.

To construct a linear cyclic code $[n,k]$, one must find a factor of $1+x^n$ having degree $n-k$. The fact that every generator must divide $1+x^n$ allows for the discovery of all linear cyclic codes of a given length n.

A polynomial $f(x)$ in $\mathbb{K}[x]$ of degree at least one is irreducible if it has degree one or is not the product of two polnomials in $\mathbb{K}[x]$, both of which having a degree at least one. An irreducible polynomial over $\mathbb{K}$ of degree $n, n > 1$ is said to be primitive if it is not a divisor of $1+x^m$ for any $m < 2^n - 1$. Using a primitive polynomial to construct $GF(2^r)$ makes computing in the field much easier than using a non-primitive irreducible polynomial. To see this, let $\beta \in K^n$ represent the word corresponding to $x \mod h(x)$, where $h(x)$ is a primitive polynomial of degree n. Then $\beta^i \iff x^i \mod h(x)$. Note that $1 = x^m \mod h(x) \implies 0 = 1 + x^m \mod h(x)$ and thus that h(x) divides $1+x^m$. Since $h(x)$ is primitive, it does not divide $1+x^m$ for $m < 2^n - 1$ and thus $\beta^m \neq 1$ for $m < 2^n - 1$. Since $\beta^j = \beta^i$ for $j \neq i \iff \beta^i = \beta^{j-i}\beta^i \implies \beta^{j-i} = 1$, we conclude that $K^n \setminus \{0\} = \{\beta^i | i = 0, 1, \dots, 2^n - 2\}$. In conclusion, every non-zero word in K^n can be represented by some power of β.

Definition 5 *An element $\alpha \in GF(2^r)$ is primitive if $a^m \neq 1$ for $1 \leq m < 2^r - 1$*

An element $\alpha \in GF(2^r)$ is said to be a root of a polynomial $p(x) \in F[x]$ if $p(\alpha) = 0$.

For any element $\alpha \in GF(2^r)$, we define the minimal polynomial of α as the polynomial in $\mathbb{K}[x]$ of smallest degree having α as a root, denoted $m_\alpha(x)$.

Theorem 3.2.4 *Let $\alpha \in GF(2^r)$. Let $m_\alpha(x)$ be a minimal polynomial of α. Then:*

- *m_α is irreducible over $\mathbb{K}$;*
- *if $f(x)$ is any polynomial over $\mathbb{K}$ such that $f(\alpha) = 0$, then $m_\alpha(x)$ is a factor of α*
- *the minimal polynomial is unique;*
- *the minimal polynomial $m_\alpha(x)$ is a factor of $1 + x^{2^r-1}$*

Theorem 3.2.5 *Let $\alpha \in GF(2^r)$ with minimal polynomial $m_\alpha(x)$, then $\{\alpha, \alpha^2, \alpha^4, \dots, \alpha^{2^r-1}\}$ is the set of all the roots of $m_\alpha(x)$. The degree($m_\alpha(x)$) is l.c.m$\{\alpha, \alpha^2, \alpha^4, \dots, \alpha^{2^r-1}\}$.*

To construct a BCH code of length n, we begin by letting $f(x) = x^n - 1 \in \mathbb{Z}_2[x]$. Then the ring $R = \mathbb{Z}_2[x]/(f(x))$ can be represented by all polynomial of $\mathbb{Z}_2[x]$ of degree less than n. The generator polynomial, $g(x)$, is defined as :

$$g(x) = lcm\{m_1(x), m_2(x), m_3(x), \ldots, m_{2t}(x)\} \tag{9}$$

as the least common multiple of the minimal polynomials $m_i(x)$ in $\mathbb{Z}_2[x]$.

Some operations like encoding and decoding require computations using polynomials, so an important fact about polynomials in $\mathbb{Z}_2[x]$ is that:

$$\Big(\sum_{i=1}^{r} x_i\Big)^2 = \sum_{i=1}^{r} x_i^2 \tag{10}$$

since all cross terms will contain a multiple of 2 so they disappear ($2 \mod 2 = 0$).

Theorem 3.2.6 *Let C be a BCH code that results from a primitive polynomial of degree n by considering the first s powers of α, and suppose $c(x) \in Z_2[x]$ has degree less than $2^n - 1$. Then $c(x) \in C \iff c(\alpha^i) = 0$ for $i = 1, \ldots, s$.*

Theorem 3.2.7 *Let C be a BCH code that results from considering the first $2t$ powers of α. Then C is t-error correcting.*

Correction in BCH

Let $\mathcal{C}$ be a BCH code that results from a primitive polynomial of degree n by considering the first $2t$ powers of α. Suppose $c(x) \in \mathcal{C}$ is transmitted and we receive the polynomial $r(x) \neq c(X) \in Z_2[x]$ of degree less than $2^n - 1$. Then $r(x) = c(x) + e(x)$ for some non zero error polynomial $e(x) \in Z_2[x]$ of degree less than $2^n - 1$. Theorem 3.2.6 implies that $r(\alpha^i) = e(\alpha^i)$ for $i = 1, \dots, 2t$. The values of $r(\alpha^i)$ are called the syndromes of $r(x)$. Supposing

$$e(x) = x^{m_1} + x^{m_2} + \cdots + x^{m_p}$$

for some integer error position $m_1 < m_2 < \cdots < m_p$ with $p \leq t$ and $m_p < 2^n - 1$. To find these error positions, we begin by computing the first $2t$ syndromes of $r(x)$, denoted $r_1, r_2, \dots, r_{2t}$.

$$\begin{aligned}
r_1 &= r(\alpha) = e(\alpha) = \alpha^{m_1} + \alpha^{m_2} + \ldots + \alpha^{m_p} \\
r_2 &= r(\alpha^2) = e(\alpha^2) = (\alpha^2)^{m_1} + (\alpha^2)^{m_2} + \ldots + (\alpha^2)^{m_p} \\
&\vdots \\
r_{2t} &= r(\alpha^{2t}) = e(\alpha) = (\alpha^{2t})^{m_1} + (\alpha^{2t})^{m_2} + \ldots + (\alpha^{2t})^{m_p}
\end{aligned}$$

The error locator polynomial, defined as:

$$\begin{aligned}
E(z) &= (z - \alpha^{m_1})(z - \alpha^{m_2}) \ldots (z - \alpha^{m_p}) \\
&= z^p + \sigma_1 z^{p-1} + \ldots + \sigma_p
\end{aligned}$$

is essential to error corretion. Its roots show the error positions in $r(x)$. To find the roots, we must find the coefficients $\sigma_1, \sigma_2, \dots, \sigma_p$ of $E(z)$. These coefficients are the elementary symmetric functions in $\alpha^{m_1}, \alpha^{m_2}, \dots, \alpha^{m_p}$, meaning:

$$\begin{aligned}
\sigma_1 &= \sum_{i=1}^{p} \alpha^{m_i} \\
\sigma_2 &= \sum_{i,j=1}^{p} \alpha^{m_i} \alpha^{m_j} \\
&\vdots \\
\sigma_p &= \alpha^{m_1} \ldots \alpha^{m_p}
\end{aligned}$$

Evaluating $E(\alpha^{m_j})$ for all $1 \leq j \leq p$ and multiplying each result by $(\alpha^{m_j})^i$ for any $1 \leq i \leq p$, since $E(\alpha^{m_j}) = 0$ for all $1 \leq j \leq p$, yields the following system of equations for $1 \leq i \leq p$:

$$\begin{aligned}
0 &= (\alpha^{m_1})^i[(\alpha^{m_1})^p + \sigma_1(\alpha^{m_1})^{p-1} + \ldots + \sigma_p] \\
0 &= (\alpha^{m_2})^i[(\alpha^{m_2})^p + \sigma_1(\alpha^{m_2})^{p-1} + \ldots + \sigma_p] \\
&\vdots \\
0 &= (\alpha^{m_p})^i[(\alpha^{m_p})^p + \sigma_1(\alpha^{m_p})^{p-1} + \ldots + \sigma_p]
\end{aligned}$$

By distributing the $(\alpha^{m_j})^i$ in the preceding equation and summing the results, we obtain the following equation for $1 \leq i \leq p$:

$$0 = r_{i+p} + \sigma_1 r_{i+p-1} + \sigma_2 r_{i+p-2} + \ldots + \sigma_p r_i$$

Since this holds for $1 \leq i \leq p$, this yields a system of p linear equations in the p unknown $\sigma_1, \sigma_2, \ldots, \sigma_p$ that are equivalent to the following matrix equation.

$$\begin{bmatrix} r_1 & \ldots & r_p \\ \vdots & & \vdots \\ r_p & \ldots & r_{2p-1} \end{bmatrix} \begin{bmatrix} \sigma_p \\ \vdots \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} r_{p+1} \\ \vdots \\ r_{2p} \end{bmatrix} \tag{11}$$

If the $p \times p$ coefficient matrix is nonsingular, then we can solve uniquely for $\sigma_1, \ldots, \sigma_p$, we can then form the error locator polynomial $E(z)$ and determine $\alpha^{m_1}, \ldots, \alpha^{m_p}$ by trial and error as the roots of $E(z)$. This reveals the error positions $m_1, \ldots, m_p$ in $r(x)$.
Since the number of errors in a received polynomial $r(x)$ is not known before attempting to correct it, in a t-error correction BCH code it is assumed that the received polynomial contains a maximum of t errors and using the first $2t$ syndromes of $r(x)$. If it does not contain exactly t errors, then the matrix in 11 will be singular. In this case, we can simply reduce the number of assumed errors to $t-1$ and repeat the error correction procedure using only the first $2t-2$ syndromes of $r(x)$. As long as produced matrix in 11 is singular, this procedure can be repeated, each time reducing the number of assumed error until the coefficient matrix in 11 is nonsingular. If the error is in $r(x)$, then the coefficient matrix will be nonsingular for any number of assumed errors between 1 and t.
To consolidate everything said about BCH codes, their encoding and decoding, as well as error correcting, an example is presented.

Example 3 *Let $f(x) = x^{15} - 1$, and choose the primitive polynomial $p(x) = x^4 + x + 1$. Then for the element $\alpha = x$ in the field $Z_2[x]/((p(x))$ of order 16 we list the field elements that correspond to the first 15 powers of α in the following table:*

Let C be the BCH code that results from considering the first six powers of α. To determine the generator polynomial $g(x)$ for C, we must find the minimal polynomials $m_1(x), m_2(x), \ldots, m_6(x)$. Since $p(x)$ is primitive and $\alpha = x$ then $p(\alpha) = 0$. From Theorem ..... it follows $p(\alpha^2) = p(\alpha)^2 = 0$ and $p(\alpha^4) = p(\alpha)^4 = 0$. Thus, $m_1(x) = m_2(x) = m_4(x) = p(x)$. From Theorem .... $f(x)$ can be factorized in the following manner:

$$x^{15} - 1 = (x+1)(x^2+x+1)(x^4+x+1)(x^4+x^3+1)(x^4+x^3+x^2+x+1).$$

Power	Field Element
α^1	α
α^2	α^2
α^3	α^3
α^4	$\alpha+1$
α^5	$\alpha^2+\alpha$
α^6	$\alpha^3+\alpha^2$
α^7	$\alpha^3+\alpha+1$
α^8	α^2+1
α^9	$\alpha^3+\alpha$
α^{10}	$\alpha^2+\alpha+1$
α^{11}	$\alpha^3+\alpha^2+\alpha$
α^{12}	$\alpha^3+\alpha^2+\alpha+1$
α^{13}	$\alpha^3+\alpha^2+1$
α^{14}	α^3+1
α^{15}	1

Table 1: Field Elements

By substituting α^3 and α^5 into each of these irreducible factors, we conclude that $m_3(x) = x^4+x^3+x^2+x+1$ and $m_5(x) = x^2+x+1$. Furthermore, $m_3(\alpha^6) = m_3(\alpha^3)^2 = 0 \implies m_6(x) = m_3(x)$. Thus, $g(x) = m_1(x)m_3(x)m_5(x) = x^{10}+x^8+x^5+x^4+x^2+x+1$. The code that results from this generator polynomial is a [15,5] BCH code capable of correcting up to 3 errors. Supposing a codeword in C was trasmitted and the received vector $r(x) = (101111110010000)$. Because $g(x)$ does not divide $r(x)$, $r(x) \notin C$. Since C is 3-error correcting, to correct $r(x)$ it is necessary to compute the first six syndromes of $r(x)$.Using the table of powers of α and the corresponding field elements in Table 1, the syndromes are computed as follows:

$$\begin{aligned} r_1 &= r(\alpha) \\ &= 1+\alpha^2+\alpha^3+\alpha^4+\alpha^5+\alpha^6+\alpha^7+\alpha^{10} \\ &= \dots \\ &= \alpha^3 \end{aligned}$$

$$\begin{aligned} r_3 &= r(\alpha^3) \\ &= 1+\alpha^6+\alpha^9+\alpha^{12}+\alpha^{15}+\alpha^{18}+\alpha^{21}+\alpha^{30} \\ &= \dots \\ &= \alpha^6 \end{aligned}$$

$$\begin{aligned} r_5 &= r(\alpha^5) \\ &= 1+\alpha^{10}+\alpha^{15}+\alpha^{20}+\alpha^{25}+\alpha^{30}+\alpha^{35}+\alpha^{50} \\ &= \dots \\ &= \alpha^{10} \end{aligned}$$

From Theorem.. the remaining syndromes are computed in the following manner:

$$\begin{aligned} r_2 &= r(\alpha^2) = (r(\alpha))^2 = (\alpha^3)^2 = \alpha^6 \\ r_4 &= r(\alpha^4) = (r(\alpha))^4 = (\alpha^3)^4 = \alpha^{12} \\ r_6 &= r(\alpha^6) = (r(\alpha^3))^2 = (\alpha^6)^2 = \alpha^{12} \end{aligned}$$

Assuming $r(x)$ contains three errors, we must find $\sigma_1, \sigma_2, \sigma_3$ that satisfy the following equation:

$$\begin{bmatrix} \alpha^3 & \alpha^6 & \alpha^6 \\ \alpha^6 & \alpha^6 & \alpha^{12} \\ \alpha^6 & \alpha^{12} & \alpha^{10} \end{bmatrix} \begin{bmatrix} \sigma_3 \\ \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} \alpha^{12} \\ \alpha^{10} \\ \alpha^{12} \end{bmatrix}$$

The determinant of the 3×3 coefficient matrix is α^{12}. Therefore, this matrix is nonsingular and $r(x)$ contains exactly 3 errors. We can use Cramer's Rule to determine σ_1, σ_2, and σ_3.

$$\begin{aligned} \begin{vmatrix} \alpha^{12} & \alpha^6 & \alpha^6 \\ \alpha^{10} & \alpha^6 & \alpha^{12} \\ \alpha^{12} & \alpha^{12} & \alpha^{10} \end{vmatrix} &= \alpha^{28} + \alpha^{30} + \alpha^{28} + \alpha^{24} + \alpha^{36} + \alpha^{26} \\ &= \dots \\ &= \alpha^{14} \end{aligned}$$

Solving the matrix and using Cramer's rule yields:

$$\begin{aligned} \sigma_1 &= \frac{1}{\alpha^{12}} = \alpha^3 \\ \sigma_2 &= \frac{\alpha^{10}}{\alpha^{12}} = \alpha^{13} \\ \sigma_3 &= \frac{\alpha^{14}}{\alpha^{12}} = \alpha^2 \end{aligned}$$

The resulting error locator polynomial is $E(z) = z^3 + \alpha^3 z^2 + \alpha^{13} z + \alpha^2$. By evaluating $E(z)$ at sucessive powers of α, we can find that the roots of $E(z)$ are $1, \alpha^5$ and α^{12}. Hence, the error in $r(x)$ is $e(x) = 1 + x^5 + x^{12}$. Thus, we correct $r(x)$ to the following codeword $c(x)$.

$$c(x) = r(x) + e(x) = x^2 + x^3 + x^4 + x^6 + x^7 + x^{10} + x^{12}$$

4

QUANTUM THEORY

As stated in Chapter 2, information and probabilities are closely related. This relation is even more obvious in the case of quantum mechanics. Quantum mechanics is a set of laws and ideas used for describing phenomena of atomic scales Dirac (1930).
The state of a quantum system (spin of an electron, polarization of a photon, etc.) is completely specified by its state vector, the ket $|\psi\rangle$. If $|\psi_1\rangle$ and $|\psi_2\rangle$ are possible states then their superpositions

$$|\psi\rangle = a_1 |\psi_1\rangle + a_2 |\psi_2\rangle \tag{12}$$

is also a state of the system, where a_1 and a_2 are complex numbers.
The bra $\langle\psi|$ provides an equivalent representation of the state in the form

$$\langle\psi| = a_1^* \langle\psi_1| + a_2^* \langle\psi_2| \tag{13}$$

where a_1^* and a_2^* are the complex conjugates of a_1 and a_2 respectively.
Two states $\langle\psi|$ and $\langle\phi|$ are said to be orthogonal if their inner product, defined as $\langle\psi|\phi\rangle$ is zero. The inner product of a state with itself is real and strictly positive, $\langle\psi|\psi\rangle \geq 1$. A state is said to be normalized if its inner product is equal to unity, $\langle\psi|\psi\rangle = 1$.
If $|\psi\rangle$ is normalized, then $|a_1|^2 + |a_2|^2 = 1$ and $|a_1|^2$ and $|a_2|^2$ represent the probabilities that the initial state $|\psi\rangle$, given a measure, collapses to $|\psi_1\rangle$ and $|\psi_2\rangle$, respectively. More generally, given n possible states $|\psi_n\rangle$

$$|\psi\rangle = \sum_n a_n |\psi_n\rangle \tag{14}$$

If $|\psi\rangle$ is normalized and the states $|\psi_n\rangle$ are orthonormal, then

$$\sum_n |a_n|^2 = 1 \tag{15}$$

Definition 6 *A linear operator between vector spaces V and W is defined to be any function $A : V \to W$ which is linear in its inputs,*

$$A\Big(\sum_i a_i |\psi_i\rangle\Big) = \sum_i a_i A |\psi_i\rangle \tag{16}$$

An operator $\hat{B}^\dagger$ is said to be the Hermitian conjugate of an operator $\hat{B}$ if, for any pair $|\psi\rangle$ and $|\phi\rangle$,

$$\langle\psi|\hat{B}^\dagger|\phi\rangle = \langle\phi|\hat{B}|\psi\rangle^* \tag{17}$$

The Hermitian conjugate has the following properties:

$$(\hat{B}^\dagger)^\dagger = \hat{B} \tag{18}$$

$$(\hat{B}+\hat{C})^\dagger = \hat{B}^\dagger + \hat{C}^\dagger \tag{19}$$

$$(\hat{B}\hat{C})^\dagger = \hat{C}^\dagger\hat{B}^\dagger \tag{20}$$

$$(\lambda\hat{B})^\dagger = \lambda^*\hat{B}^\dagger \tag{21}$$

where $\hat{C}$ represents another arbitrary operator and λ is any complex number. Any operator $\hat{A}$ such that $\hat{A}^\dagger = \hat{A}$ is said to be an Hermitian operator. Hermitian operators are important in Quantum Mechanics because they relate to observable quantities, also named observables, such as spin or momentum. The eigenvalues λ_n of an Hermitian operator $\hat{A}$ satisfy the eigenvalue equation

$$\hat{A}\left|\Psi_n\right\rangle = \lambda_n \left|\Psi_n\right\rangle \tag{22}$$

where the $\left|\Psi_n\right\rangle$ are the eigenstates. The conjugate equation with λ_n replaced by λ_m is

$$\left\langle\Psi_m\right|\hat{A}^\dagger = \left\langle\Psi_m\right|\hat{A} = \lambda_m^*\left\langle\Psi_m\right| \tag{23}$$

Multiplying both sides by $\left|\Psi_n\right\rangle$ gives

$$\langle\Psi_m|\hat{A}|\Psi_n\rangle = \lambda_m^*\langle\Psi_m|\Psi_n\rangle \tag{24}$$

Similarly, multiplying by $\left\langle\Psi_m\right|$ in eq.22 gives

$$\langle\Psi_m|\hat{A}|\Psi_n\rangle = \lambda_n\langle\Psi_m|\Psi_n\rangle \tag{25}$$

Subtracting eq.24 from eq.23 gives

$$(\lambda_m^* - \lambda_n)\langle\Psi_m|\Psi_n\rangle = 0 \tag{26}$$

so if $m = n \implies \lambda_n^* - \lambda_n = 0$ and the eigenvalues must be real. If, however, $\lambda_m \neq \lambda_n$ then the states $\left|\Psi_m\right\rangle$ and $\left|\Psi_n\right\rangle$ are orthogonal. Hermitian operators have real eigeinvalues associated with orthormal eigenstates.
An important property of operators is that they do not, in general, commute. This means that the order in which operators are applied to a given state matters and, in general, $\hat{A}\hat{B}\left|\psi\right\rangle \neq \hat{B}\hat{A}\left|\psi\right\rangle$. The commutator of $\hat{A}$ and $\hat{B}$ is defined to be

$$[\hat{A},\hat{B}] = \hat{A}\hat{B} - \hat{B}\hat{A} \tag{27}$$

If $[\hat{A}, \hat{B}] = 0$ then $\hat{A}$ and $\hat{B}$ are said to commute.
The anticommutator is defined to be

$$\{\hat{A}, \hat{B}\} = \hat{A}\hat{B} + \hat{B}\hat{A} \tag{28}$$

which is Hermitian if $\hat{A}$ and $\hat{B}$ are Hermitian.
The uncertainties associated with the observables A and B, ΔA and ΔB respectively, for any given state are bounded by the uncertainty principle Heisenberg (1927):

$$\Delta A \Delta B \geq \frac{1}{2} |\langle [\hat{A}, \hat{B}] \rangle| \tag{29}$$

Definition 7 *The outer product of two normalized states $|\phi_1\rangle$ and $|\phi_2\rangle$ is the operator $|\phi_1\rangle \langle\phi_2|$. This outer product is Hermitian if and only $|\phi_1\rangle = |\phi_2\rangle$.*

$$|\phi_1\rangle \langle\phi_2|\psi\rangle = \langle\phi_2|\psi\rangle \, |\phi_1\rangle \tag{30}$$

The evolution of a state $|\psi(t)\rangle$ is governed by the Schrödinger equation

$$i\hbar \frac{d}{dt} |\psi(t)\rangle = \hat{H} |\psi(t)\rangle \tag{31}$$

where $\hat{H}$ is the Hamiltonian. The formal solution of the Schrödinger equation is

$$|\psi(t)\rangle = \hat{U}(t) |\psi(0)\rangle \tag{32}$$

where $\hat{U}$ is a unitary operator, for which $\hat{U}^\dagger = \hat{U}^{-1}$ so that $\hat{U}^\dagger \hat{U} = \hat{I} = \hat{U}\hat{U}^\dagger$. The evolution operator $\hat{U}(t)$ itself satisfies the Schrödinger equation

$$i\hbar \frac{d}{dt} \hat{U}(t) = \hat{H}\hat{U}(t) \tag{33}$$

Time evolution of a quantum state is associated with the action of a unitary operator. This evolution is equivalent to information processing, and information extraction from the system is done with the use of measurements.

Furthermore, inner product is preserved troughout the action of unitary operators to the system. If a unitary operator $\hat{U}$ transforms a state $|\psi\rangle$ into a new state $|\psi'\rangle = \hat{U} |\psi\rangle$ then

$$\langle\phi'|\psi'\rangle = \langle\phi|\hat{U}^\dagger \hat{U}|\psi\rangle = \langle\phi|\psi\rangle \tag{34}$$

It is often helpful to break a complicated unitary transformation into a sequence of simpler ones. A unitary operator produced by an operator $\hat{U}$ will be equivalent to a sequence on n unitary operators $\hat{U}_1, \hat{U}_2, \ldots, \hat{U}_n$ if $\hat{U} = \hat{U}_n, \ldots, \hat{U}_2, \hat{U}_1$. When applying to a state $|\psi\rangle$

$$\hat{U} |\psi\rangle = \hat{U}_n \ldots \hat{U}_2 \hat{U}_1 |\psi\rangle \tag{35}$$

The order of operators is important because the operators $\hat{U}_i$ do not necessarily mutually commute.

The tensor product, denoted $\otimes$, is a mathematical operation that merges vector spaces together to form larger vector spaces.

Definition 8 *Suppose V and W are vector spaces of dimension m and n, respectively, Then $V \otimes W$ is an $m \times n$ dimensional vector space. The elements of $V \otimes W$ are linear combinations of tensor products $|v\rangle \otimes |w\rangle, |v\rangle \in V, |w\rangle \in W$.*[1] *The tensor product has the following properties:*

- *For an arbitrary scalar z and elements $|v\rangle \in V, |w\rangle \in W$,*

$$z(|v\rangle \otimes |w\rangle) = z\,|v\rangle \otimes |w\rangle = |v\rangle \otimes (z\,|w\rangle)$$

- *For arbitrary $|v_1\rangle, |v_2\rangle \in V, |w\rangle \in W$*

$$(|v_1\rangle + |v_2\rangle) \otimes |w\rangle = |v_1\rangle \otimes |w\rangle + |v_2\rangle \otimes |w\rangle$$

- *For arbitrary $|v\rangle \in V, |w_1\rangle, |w_2\rangle \in W$*

$$|v\rangle \otimes (|w_1\rangle + |w_2\rangle) = |v\rangle \otimes |w_1\rangle + |v\rangle \otimes |w_2\rangle$$

Definition 9 *The density operator language provides a convenient means for describing quantum systems whose state is not completely known. The density operator, also known as density matrix, for the system is defined by the equation:*

$$\rho \equiv \sum_i p_i\,|\psi_i\rangle\langle\psi_i|$$

The evolution of the density operator is described by the equation:

$$\rho = \sum_i p_i\,|\psi_i\rangle\langle\psi_i| \xrightarrow{U} \sum_i p_i U\,|\psi_i\rangle\langle\psi_i|\,U^\dagger = U\rho U^\dagger$$

Definition 10 *Fidelity is a measure of distance between quantum states, meaning it can be used as an indirect measure of the sucess of transmission of quantum states. The fidelity of states ρ and σ is defined to be*

$$F(\rho,\sigma) \equiv tr\sqrt{\rho^{\frac{1}{2}}\sigma\rho^{\frac{1}{2}}}$$

The fidelity of a pure state $|\psi\rangle$ and an arbitrary state ρ is given by

$$F(|\psi\rangle,\rho) = \mathrm{Tr}\left\{\sqrt{\langle\psi|\rho|\psi\rangle\,|\psi\rangle\langle\psi|}\right\} = \sqrt{\langle\psi|\rho|\psi\rangle}$$

That is, the fidelity is equal to the square root of the overlap between $|\psi\rangle$ and ρ.

A quantum state whose state $|\psi\rangle$ is known exactly is said to be in a pure state. In this case the density operator is simply $\rho = |\psi\rangle\langle\psi|$. Otherwise, ρ is in a mixed state; it is said to be a mixture of the different pure states in its ensemble for ρ. A pure state satisfies $tr(\rho^2) = 1$, while a mixed state satisfies $tr(\rho^2) < 1$.

A quantum bit, usually called qubit, is the basic unit of quantum information and its represented by a quantum system with two orthogonal states, labeled $|0\rangle$ and $|1\rangle$. Because of the nature of

1 For convenience, $|v\rangle \otimes |w\rangle$ may be written as $|vw\rangle, |v\rangle|w\rangle$ or $|v,w\rangle$.

quantum mechanics, the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ consisting of a superposition Dirac (1981) is also a valid state for the qubit, where α and β are complex numbers and $|\alpha|^2 + |\beta|^2 = 1$. The state $|\psi\rangle$ can also be written as the column vector

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \tag{36}$$

Altough it may seem that a quantum qubit is able to store an infinite amount of information that is not correct. Holevo's bound A.S.Holevo (1973) establishes an upper bound to the amount of information that can be known about a quantum state. Holevo's bound proves that the amount of classical information that can be retrieved from a qubit can be only up to 1 classical bit.

4.1 QUANTUM COMPUTATION

4.1.1 *Pauli Matrices*

All changes occuring to an one qubit quantum state can be described by the following unitary operators, also called Pauli operators:

$$\begin{aligned} \hat{I} &= |0\rangle\langle 0| + |1\rangle\langle 1| \\ X = \hat{\sigma}_x &= |0\rangle\langle 1| + |1\rangle\langle 0| \\ Y = \hat{\sigma}_y &= i(|1\rangle\langle 0| - |0\rangle\langle 1|) \\ Z = \hat{\sigma}_z &= |0\rangle\langle 0| - |1\rangle\langle 1| \end{aligned}$$

These correspond, respectively, to the identity operator and to the x,y, and z-components of the angular momentum, in units of $\hbar/2$. Their matrix form equivalent are given by

$$\hat{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{37}$$

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{38}$$

$$Y = i\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} = iXZ \tag{39}$$

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{40}$$

Because quantum computation is interested in manipulating and changing the state of quantum bits it's important to understand the effects of the Pauli Matrices on simple qubits.

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle$$

$$X|1\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle$$

$$Z|0\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle$$

$$Z|1\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \end{pmatrix} = -|1\rangle$$

From the equations, one concludes that X has the effect of bit-flipping, i.e., transforming a quantum state in its orthogonal state, acting as a NOT gate in classical computation; operator Z has the effect of phase-flipping.

The Pauli operators, excluding $\hat{I}$, have the following properties

$$\begin{aligned} [\sigma_i, \sigma_j] &= \sigma_i\sigma_j - \sigma_j\sigma_i = 2\sigma_k \\ \{\sigma_i, \sigma_j\} &= \sigma_i\sigma_j + \sigma_j\sigma_i = 2\delta_{i,j}\hat{I}, \quad \delta_{i,j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

The first equation tells us that choosing any two dimensions of spin, their commutator yields the remaining dimension. The anticommutator of two different spin components is zero.

4.1.2 *Hadamard and CNOT*

Another essential matrix to perform quantum computation is the Hadamard matrix

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{41}$$

It has the effect of trasforming a qubit from the computational basis $|0\rangle$ or $|1\rangle$, to a superposition of the two:

$$H|0\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1\\ 0\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\ 1\end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle$$

$$H|1\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}0\\ 1\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\ -1\end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle$$

Figure.5 denotes the circuit representation of all the gates mentioned above.

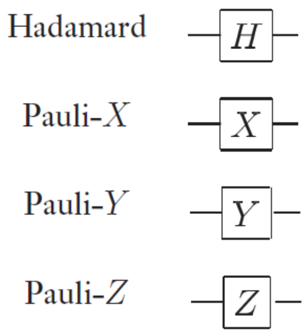


Figure 5: Single qubit gates

Perhaps the most useful controlled operation is the controlled-NOT, or simply CNOT. It's a quantum gate with two input qubits, known as control qubit $|c\rangle$ and target qubit $|t\rangle$. The effect of this gate is to take a quantum system of the form $|c\rangle|t\rangle \rightarrow |c\rangle|t \oplus c\rangle$; that is, if the control qubit is $|1\rangle$ then the target qubit is flipped, otherwise the target qubit is left alone. The matrix representation of CNOT is

$$CNOT = \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix} \tag{42}$$

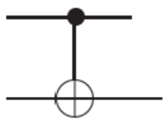


Figure 6: Circuit representation for the CNOT. The top line represents the control qubit, the bottom line the target qubit. (from Nielsen and Chuang (2002))

Quantum computation employs the usage of multiple qubits. A state $|\psi\rangle$ composed of n qubits, all prepared in the state $|0\rangle$, is written as:

$$|\psi\rangle = |0\rangle \otimes |0\rangle \otimes \cdots \otimes |0\rangle$$ [2]

that is, the tensor product of n $|0\rangle$ kets. Single-qubit operations can still be performed with ease. For example, applying the unitary operator $\hat{U}$ to the m^{th} qubit of the state $|\psi\rangle$ can be achieved in the following manner

$$\underbrace{\hat{I} \otimes \hat{I} \otimes \ldots}_{\text{m-1 terms}} \otimes \hat{U} \otimes \underbrace{\hat{I} \otimes \hat{I} \otimes \ldots \hat{I}}_{\text{n-m terms}} |\psi\rangle = \underbrace{|0\rangle \otimes |0\rangle \otimes \ldots}_{\text{m-1 terms}} \otimes (\hat{U}\,|0\rangle) \otimes \underbrace{\cdots \otimes |0\rangle}_{\text{n-m terms}}$$

4.1.3 *Bloch Sphere*

Another helpful representation of a qubit is to imagine the qubits states as points on the surface of a sphere of unit radius, the Bloch sphere. Opposite points represent a pair of mutually orthogonal states. The north and south poles correspond to the states $|0\rangle$ and $|1\rangle$. A qubit state

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + \exp\{i\varphi\}\sin\left(\frac{\theta}{2}\right)|1\rangle \tag{43}$$

corresponds to a point with spherical polar coordinates θ and φ. Any single-qubit unitary operator can be written in the form

$$\hat{U} = \exp\left\{i\alpha\hat{I} + i\beta\vec{a}\cdot\hat{\vec{\sigma}}\right\}$$

where α and β are real constants, $\vec{a}$ is a unitary vector and $\hat{\vec{\sigma}}$ is the vector operator $(\sigma_x, \sigma_y, \sigma_z)$.

2 It is not always necessary to use the tensor product symbol $\otimes$. For example, the state $|0\rangle \otimes |0\rangle \otimes |0\rangle$ can be written as $|000\rangle$ when there is no danger of misunderstanding. The same can't be applied to operators. For example, $\hat{\sigma_x} \otimes \hat{\sigma_y} \otimes \hat{\sigma_z}$ is an operator applied to a 3-qubit state while $\hat{\sigma_x}\hat{\sigma_y}\hat{\sigma_z}$ denotes the three Pauli operators acting on the same qubit. Furthermore, throughout this thesis an operator can be written both as $\hat{I}$ and I.

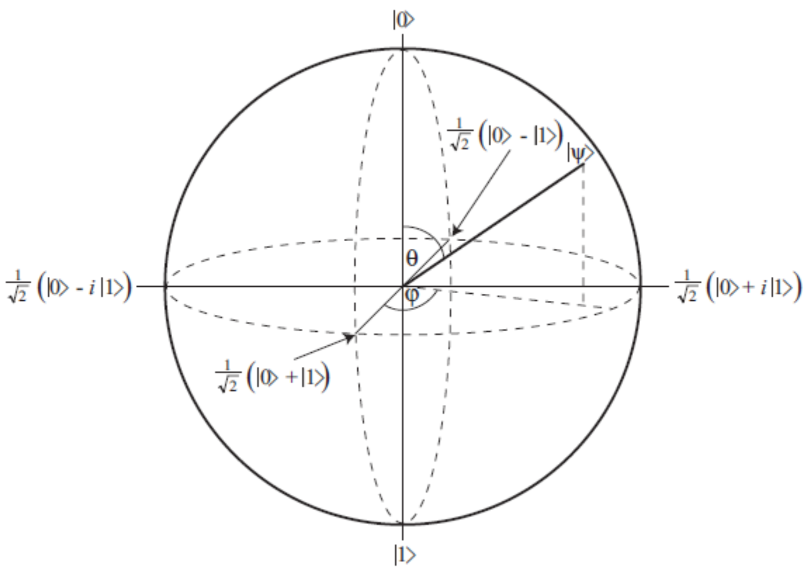

$|0\rangle$
$\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$
$|\psi\rangle$
θ
φ
$\frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle)$
$\frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$
$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$
$|1\rangle$

Figure 7: Bloch sphere (from Barnett (2009))

5

QUANTUM ERROR-CORRECTING CODES

A noisy quantum channel can be a regular communications channel which we expect to preserve at least some degree of quantum coherence, or it can be the passage of time as a set of qubits interacts with its environment, or it can be the result of operating with a noisy gate on some qubits in a quantum computer Gottesman (1997).
The channel applies a superoperator to the input density matrix. We can diagonalize this superoperator and write it as the direct sum of operators acting directly on the pure input states. If a code can correct any of the possible operators, it can correct the full superoperator.

5.1 PROPERTIES OF A QUANTUM CODE

Similarly to classic codes, quantum codes are linear. A code to encode k qubits in n qubits will have 2^k basis codeword. The encodings $|\psi_i\rangle$ of the original 2^k basis states form a basis for the space of codewords. When a coherent error occurs, the code states are altered by some linear transformation T:

$$|\psi_i\rangle \rightarrow T\,|\psi_i\rangle \tag{44}$$

An error-correction process can be modeled by a unitary linear transformation that entangles the erroneous states $T\,|\psi_i\rangle$ with an *ancilla* $|A\rangle$ and transforms the combination to corrected state:

$$(T\,|\psi_i\rangle) \otimes |A\rangle \rightarrow |\psi_i\rangle\,|A_T\rangle \tag{45}$$

At this point, the *ancilla* qubit can be measured in order to restore it to its original state without disturbing the states $|\psi_i\rangle$. This process will correct the error even if the original state is a superposition of the basis states:

$$\left(T\sum_{i=1}^{2^k} c_i\,|\psi_i\rangle\right) \otimes |A\rangle \rightarrow \left(\sum_{i=1}^{2^k} c_i\,|\psi_i\rangle\right) \otimes |A_T\rangle$$

Definition 11 *Let G be the group generated by all* $3n$ *Pauli matrices. This group G has the following properties:*

- $M^2 = \pm 1, \quad M \in G$
- $[A, B] = 0$ *or* $\{A, B\} = 0, \quad A, B \in G$

The codewords of the quantum error-correcting code span a subspace T of the Hilbert space.

In order for the code to correct two errors E_a and E_b, it is necessary to be able to distiguish error E_a acting on a basis codeword $|\psi_i\rangle$ from error E_b acting on a different basis codeword $|\psi_j\rangle$. This can be achieved if $E_a |\psi_i\rangle$ is orthogonal to $E_b |\psi_j\rangle$

$$\langle \psi_i | E_a^\dagger E_b | \psi_j \rangle = 0 \tag{46}$$

when $i \neq j$ for correctable errors E_a and E_b. However eq.46 is insufficient to guarantee a code will work as a quantum error-correcting code. When a measurement is made, no information about the actual state of the code can be obtained. To obtain information about a superposition of the basis states means the superposition is broken. Information is obtained when measuring $\langle \psi_i | E_a^\dagger E_b | \psi_i \rangle$. Therefore this quantity must be the same for all the basis codewords

$$\langle \psi_i | E_a^\dagger E_b | \psi_i \rangle = \langle \psi_j | E_a^\dagger E_b | \psi_j \rangle \tag{47}$$

Combining eq.46 and eq.47 yields

$$\langle \psi_i | E_a^\dagger E_b | \psi_j \rangle = C_{ab}\delta_{ij} \tag{48}$$

Now suppose $E \in G$ and $\exists M \in S : \{E, M\} = 0$. Then $\forall |\psi\rangle, |\phi\rangle \in T$

$$\langle \phi | E | \psi \rangle = \langle \phi | EM | \psi \rangle = -\langle \phi | ME | \psi \rangle = -\langle \phi | E | \psi \rangle$$

so $\langle \phi | E | \psi \rangle = 0$. This implies that for two errors E and F, $E|\psi\rangle$ and $F|\phi\rangle$ are orthogonal for all $|\psi\rangle, |\phi\rangle \in T$ whenever $F^\dagger E$ anticommutes with everything in S.

It is assumed that errors occur independently on different qubits, with equal probability of being a σ_x, σ_y or σ_z error.

5.2 Shor's 9-qubit code

In this code we wish to protect 1 bit of information from all possible errors that can occur. To do this we build 1 logical qubit, consisting of 9 qubits, in the following manner

$$\begin{aligned} |0\rangle &\to |\bar{0}\rangle = (|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle \\ |1\rangle &\to |\bar{1}\rangle = (|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle) \end{aligned}$$

The data is no longer stored in a single qubit but in nine instead. To understand error detection in this code first we need to understand the effect of the operator $\sigma_z \otimes \sigma_z$ in a given state $|\psi\rangle = |\psi_0\rangle \otimes |\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle \equiv |\psi_0\psi_1 \ldots \psi_n\rangle$.

Definition 12 *Let σ_{z_i} be the operator σ_z applied to the i^{th} qubit of a multiple particle system.*

Applying the operator $\sigma_{z_i} \otimes \sigma_{z_j}, 0 \leq i \leq n, 0 \leq j \leq n, i \neq j$, to the state $|\psi\rangle$ yields the following eigenvalue $\varepsilon_{i,j}$

$$\varepsilon_{i,j} = \begin{cases} +1 & \text{if } |\psi_i\rangle = |\psi_j\rangle \\ -1 & \text{if } |\psi_i\rangle \neq |\psi_j\rangle \end{cases}$$

Therefore, measuring the eigenvalue of this operator $\sigma_{z_i} \otimes \sigma_{z_j}$ is equivalent to comparing the value of two qubits without actually measuring them, since measurement would destroy the superposition. Using both $\sigma_{z_1} \otimes \sigma_{z_2}$ and $\sigma_{z_1} \otimes \sigma_{z_3}$ operator on $|\bar{0}\rangle$ informs us if an error occurred in the first block of three qubits and where said error lies. If the first two qubits are the same, then $\varepsilon = +1$; otherwise, $\varepsilon = -1$. Assuming the first qubit was flipped, then by comparing the first two qubits, we find they are different ($\varepsilon_{1,2} = -1$), which is not allowed for any valid codeword in the code. Therefore we know an error occured, and furthermore, it flipped either the first or second qubit. Comparing the first and third qubits we again find they are different ($\varepsilon_{1,3} = -1$) . This means the error occured in the first qubit and correction can be achieved simply by flipping the erroneous qubit.
Similarly, to detect a sign error, we compare the signs of the first and second blocks and the first and third blocks of three qubits. This is equivalent to measuring the eigenvalues of $\sigma_{x_1}\sigma_{x_2}\sigma_{x_3}\sigma_{x_4}\sigma_{x_5}\sigma_{x_6}$ and $\sigma_{x_1}\sigma_{x_2}\sigma_{x_3}\sigma_{x_7}\sigma_{x_8}\sigma_{x_9}$. If the signs agree, the eigenvalues will be $+1$; if they disagree, the eigenvalues will be -1. It is also possible to have both a bit flip and a sign flip on the same qubit. However, by going through both processes described above, it is possible to fix first the bit flip then the sign flip. The set of operators that fix $|\bar{0}\rangle$ and $|\bar{1}\rangle$ form a group S, called the stabilizer of the code, and M_i are called the generators of this group. In order to totally correct the code, we must measure the eigenvalues of a total of eight operators, therefore, this code has 8 generators.

M_1	σ_z	σ_z	I	I	I	I	I	I	I
M_2	σ_z	I	σ_z	I	I	I	I	I	I
M_3	I	I	I	σ_z	σ_z	I	I	I	I
M_4	I	I	I	σ_z	I	σ_z	I	I	I
M_5	I	I	I	I	I	I	σ_z	σ_z	I
M_6	I	I	I	I	I	I	σ_z	I	σ_z
M_7	σ_x	σ_x	σ_x	σ_x	σ_x	σ_x	I	I	I
M_8	σ_x	σ_x	σ_x	I	I	I	σ_x	σ_x	σ_x

Table 2: The stabilizer for Shor's 9-qubit code

If more than one one qubit of a nine-tuple decoheres, the encoding scheme does not work Shor (1995a). If each qubit decoheres with a probability of p, then the probability that k qubits do not decohere is $(1-p)^k$. The probability that a t-error correcting code works, that is, that less than t errors occcur is given by

$$\sum_{i=0}^{t} \binom{N}{N-i} (1-p)^{N-i} p^i,$$

where N represents the length of the code, in this case $N = 9$ and $t = 1$. For this code, the probability that less than two errors occur is

$$\begin{aligned} p_{\text{errors}<2} &= \binom{9}{9}(1-p)^9 + \binom{9}{8}(1-p)^8 \cdot p \\ &= (1-p)^9 + 9p(1-p)^8 \\ &= (1-p)^8(1-p+9p) \\ &= (1-p)^8(1+8p) \end{aligned}$$

The probability that at least two qubit in any particular nine-tuple decohere is $1-(1+8p)\times(1-p)^8 \approx 36p^2$, and the probability that $9k$ qubits can be decoded to give the original quantum state is approximately $(1-36p^2)^k$. For a probability of decoherence less than $\frac{1}{36} \approx 3\%$ this encoding provides an improved storage method for quantum-coherent states of large number of qubits.

5.3 STABILIZER CODING

Definition 13 *The stabilizer S of a subspace T is defined as* $S = \{M \in G : M|\psi\rangle = |\psi\rangle \, \forall \, |\psi\rangle \in T\}$

The stabilizer always forms a group. Since the generators must commute with each other, the stabilizer is an Abelian group.

In general. if $M \in S, \{M, E\} = 0$ and $|\psi\rangle \in T$, then

$$ME|\psi\rangle = -EM|\psi\rangle = -M|\psi\rangle \tag{49}$$

Definition 14 *Let $\mathcal{G}$ be a group. The normalizer of $\mathcal{G}$, denoted $N(\mathcal{G})$ is defined to be:*

$$N(\mathcal{G}) = \{U : U\mathcal{G}U^\dagger = \mathcal{G}\},$$

5.3.1 $\overline{X}$ *and* $\overline{Z}$ *operators*

Since the elements of $N(S)$ move codewords around within T, they have a natural interpretation as encoded operations on the codewords. Since S fixes T, actually only $N(S)/S$ will act on T nontrivially. If we pick a basis for T consisting of eigenvectors of n commuting elements of $N(S)$, we get an automorphism $N(S)/S \implies G_k$. $N(S)/S$ can therefore be generated by i (which we will by and large ignore) and $2k$ equivalence classes, which I will write $\overline{X_i}$ and $\overline{Z_i}$ $(i = 1 \ldots k)$, where $\overline{X_i}$ maps to σ_{x_i} in G_k and $\overline{Z_i}$ maps to σ_{z_i} in G_k. They are encoded σ_x and σ_z operators for the code. If $k = 1$, I will write $\overline{X}_1 = \overline{X}$ and $\overline{Z_1} = \overline{Z}$. The $\overline{X}$ and $\overline{Z}$ operators satisfy:

$$[\overline{X}_i, \overline{X}_j] = 0 \tag{50}$$

$$[\overline{Z}_i, \overline{Z}_j] = 0 \tag{51}$$

$$[\overline{X}_i, \overline{Z}_j] = 0 \quad (i \neq j) \tag{52}$$

$$\{\overline{X}_i, \overline{X}_j\} = 0 \tag{53}$$

Suppose we have an $\overline{X}$ which in this language is written $(u|v) = (u_1 u_2 u_3 | v_1 v_2 v_3)$, where u_1 and v_1 are r-dimensional vectors, u_2 and v_2 are $(n-k-r)$-dimensional vectors, and u_3 and v_3 are k-dimensional vectors. However, elements of $N(S)$ are equivalent up to multiplication by elements of S. Therefore, we can also perform eliminations on $\overline{X}$ to force $u_1 = 0$ and $v_2 = 0$. Then, because $\overline{X}$ is in $N(S)$, we must satisfy:

$$\begin{pmatrix} I & A_1 & A_2 & B & C_1 & C_2 \\ 0 & 0 & 0 & D & I & E \end{pmatrix} \begin{pmatrix} v_1^T \\ 0 \\ v_3^T \\ 0 \\ u_2^T \\ u_3^T \end{pmatrix} = \begin{pmatrix} v_1^T + A_2 v_3^T + C_1 u_2^T + C_2 u_3^T \\ u_2^T + E u_3^T \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{54}$$

Suppose we want to choose a complete set of k $\overline{X}$ operators. We can combine their vectors into two $k \times n$ matrices $(0 U_2 U_3 | V_1 0 V_3)$. We want them to commute with each other, so $U_3 V_3^T + V_3 U_3^T = 0$. Suppose we pick $U_3 = I$. Then we can take $V_3 = 0$, and by equation 54, $U_2 = E^T$ and $V_1 = E^T C_1^T + C_2^T$. The rest of the construction will assume that this choice has actually been made. Another choice of U_3 and V_3 will require us to perform some operation on the unencoded data to compensate. We can also pick a complete set of k $\overline{Z}$ operators, which act on the code as encoded σ_z operators. They are uniquely defined (up to multiplication by S, as usual) given the $\overline{X}$ operators. Given the properties stated in 50, we can bring a $\overline{Z}_i$ operator into the standard form $(0 U_2' U_3' | V_1' 0 V_3')$. Then

$$U_3' V_3^T + V_3' U_3^T = I \tag{55}$$

When $U_3 = I$ and $V_3 = 0$, $V_3' = I$. Since equation 54 holds for the $\overline{Z}$ operators too, $U_2' = U_3' = 0$ and $V_1' = A_2^T$.

5.3.2 *Encoding and Decoding Stabilizer Codes*

In order to actually encode states using a quantum code, we need to decide which states will act as basis states for the coding space. In order to do this, it is convenient to use language of binary vector spaces Cleve and Gottesman (1997).

Definition 15 *Let $M_1, \ldots, M_{n-k}$ be the stabilizer generators. To represent them in a binary vector spaces, we simply write the stabilizer as a pair of $(n-k) \times n$ binary matrices, the rows corresponding to the different generators and the columns to the different qubits.*
The left block has a 1 whenever the generator has a σ_x in the appropriate place. The same for the right block whenever the generator has a σ_z.

In order to produce codewords, the stabilizer in matrix form must be converted to the standard form. This conversion will involve two types of operations:

- Replacing a generator M_i with $M_i M_j$ for some generator $M_j, j \neq i$. Since the stabilizer is a group, it is unchanged by this operation. The corresponding effect on the binary matrices is to add row j to row i.

- Rearranging qubits in the code, which corresponds to reordering the columns of the matrix.

Combining these two operations, we can perform Gaussian elimination, putting the code in the form:

$$\left(\begin{array}{cc|cc} I & A & B & C \\ 0 & 0 & D & E \end{array}\right)$$

Another Gaussian elimination on E yields:

$$\left(\begin{array}{ccc|ccc} I & A_1 & A_2 & B & C_1 & C_2 \\ 0 & 0 & 0 & D_1 & I & E_2 \\ 0 & 0 & 0 & D_2 & 0 & 0 \end{array}\right)$$

We can always put the code into the standard form:

$$\left(\begin{array}{ccc|ccc} I & A_1 & A_2 & B & C_1 & C_2 \\ 0 & 0 & 0 & D & I & E \end{array}\right)$$

5.3.3 *Networks for Encoding and Decoding*

Given a stabilizer in standard form along with the $\overline{X}$ operators in standard form, it is straightforward to produce a network to encode the corresponding code. The operation of encoding a stabilizer code can be written as:

$$\begin{aligned} |c_1 \dots c_k\rangle \to \Big(\sum_{M\in S} M\Big)\overline{X}_1^{c_1} \dots \overline{X}_k^{c_k} |0\dots 0\rangle \\ = (I+M_1)\dots(I+M_{n-k})\overline{X}_1^{c_1} \dots \overline{X}_k^{c_k} |0\dots 0\rangle \end{aligned} \tag{56}$$

where M_i are the generators of the stabilizer, and $\overline{X}_i$ are the encoded σ_x operators of the k encoded qubits.

Because in standard form the first r generators M_i have a 1, corresponding to a σ_x, on the i^{th} qubit, they can be written as follows:

$$M_i \equiv X_i \implies I + M_i = I + X_i = H \tag{57}$$

Therefore, applying the first r generators to each *pivot* qubit can be written as a simple Hadamard transform 41. Next, we apply each generator M_i conditioned on the i^{th} qubit. We can do this because the control qubit has not been the target of a previous operation, therefore it hasn't been changed. This is the importance of putting the stabilizer matrix and the operators $\overline{X}$ and $\overline{Z}$ in the standard form.

We can decode a code by performing the above process in reverse. However, if we want to measure the i^{th} encoded qubit without decoding, we can measure the eigenvalue of $\overline{Z}_i$. If the eigenvalue is $+1$, the i^{th} encoded qubit is $|0\rangle$; if it is -1, the i^{th} encoded qubit is $|1\rangle$. In standard form, $\overline{Z}_i$ is the tensor product of σ_z, meaning it will have eigenvalue $\varepsilon = (-1)^P$, where P is the parity of the qubits acted on by $\overline{Z}_i$. For the 5 qubit code example the decoding scheme can be achieved by the circuit in Fig.9.

Afterwards, applying the $\overline{X}$ operator conditioned on the ancilla qubit, the system will be reset to the $|\overline{0}\rangle$ state. The measurement of the ancilla qubit will determine the message sent.

In general, the total number of two-qubit operations is bounded by

$$k(n-k-r)+r(n-1) \leq (k+r)(n-k) \leq n(n-k)$$

and the number of one-qubit operations is bounded by r.

6

THE 5 QUBIT CODE

In this chapter I will study in detail the 5 qubit stabilizer code. As the name suggests, it requires 5 qubits to produce a logical qubit and therefore protect one qubit from errors. This code is cyclic (i.e.,the stabilizer and codewords are invariant under cyclic permutations of the qubits). It has distance 3 and is nondegenerate.
The encoding of the code using Qiskit is shown in Appendix A. Qiskit is an open-source Software Developing Kit for working with quantum computers at the level of pulses, circuits, and application modules. Together with IBM Quantum, one can code and test any quantum circuit on a simulator (IBM Qasm Simulator) or on a IBM Quantum System.
The stabilizer in its standard form as well as the $\overline{X}$ and $\overline{Z}$ operators are shown in Table 3. Note that the codewords to be sent can be written very simply as follows:[1]

$$|\overline{0}\rangle = \sum_{M \in S} M\,|00000\rangle \tag{58}$$

$$|\overline{1}\rangle = \overline{X}\,|\overline{0}\rangle \tag{59}$$

M_1	1	0	0	0	1	1	1	0	1	1
M_2	0	1	0	0	1	0	0	1	1	0
M_3	0	0	1	0	1	1	1	0	0	0
M_4	0	0	0	1	1	1	0	1	1	1
$\overline{X}$	0	0	0	0	1	1	0	0	1	0
$\overline{Z}$	0	0	0	0	0	1	1	1	1	1

Table 3: The stabilizer fot the five-qubit code in standard form.

Using the rules in Chapter.5.3.3 for producing an encoding network yields the circuit in Fig.8. This network leaves the qubits in the $|\overline{0}\rangle$ state and as Eq.59 suggests, merely applying the $\overline{X}$ operator at the end yields the $|\overline{1}\rangle$ state.

In my code I assumed a standard transmission like the one shown in Fig.1 with a trasmitter that encodes and sends the message, a buffer zone meant to represent the channel of transmission of the message where a possible error would emerge and a receiver that decodes and reads the message. The decoding network is done by performing the network in Fig.8 in reverse. Furthermore, I only

1 Because each codeword has 16 terms I will not write them down extensively. However, note that the $|\overline{0}\rangle$ is a superposition of all 5-qubit states with even parity and the $|\overline{1}\rangle$ is a superposition with odd parity.

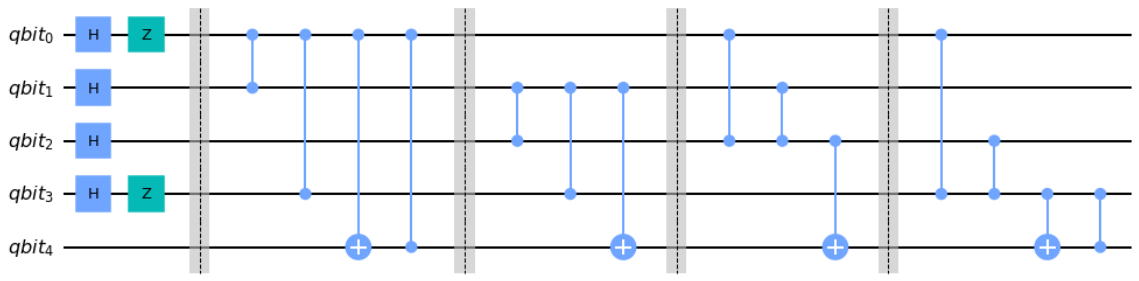


Figure 8: Encoding network for the 5 qubit code

considered bit flip errors during transmission.

Although it would be possible to read the output of this encoding without decoding using the circuit in Fig.9 and checking the value of the ancilla qubit, without the decoding there is no guarantee that the received qubit is correct because there is no error correction. Even if no error occured during encoding and transmission this set up can induce errors when measuring, due to it not being fault-tolerant Gottesman (1998)Shor (1996). I tried exploring fault-tolerant computing but came to the conclusion that the current architecture of Qiskit does not allow it. More specifically, when using faul-tolerant gates it's sometimes necessary for an operation in the middle of a circuit to be conditioned on the measured state of an ancilla which is not possible using Qiskit because all measurements of qubits are made at the end of the computation.

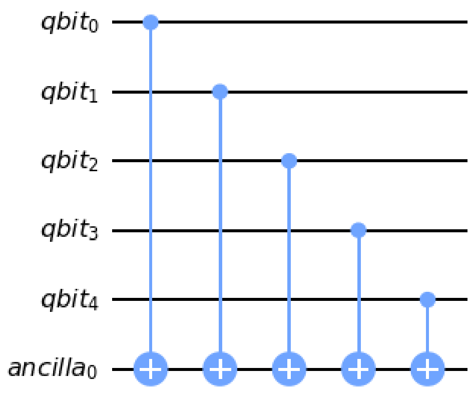
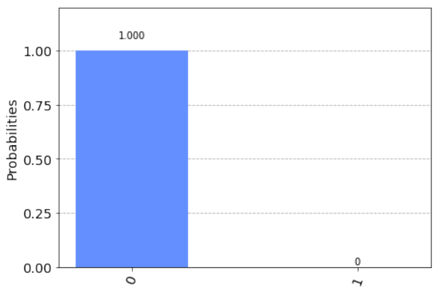


Figure 9: Reading the output

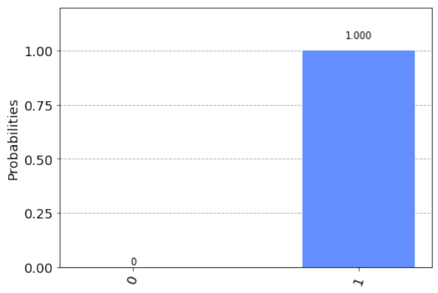
6.1 RESULTS

6.1.1 *Qasm Simulator*

In this section I present the results of the algorithm when run of *ibmq_qasm_simulator* which is meant to simbolize the ideal quantum computer, error free. Because of that, during the transmission step, I introduced a bit flip to study the behaviour of the stabilizer code and to evaluate its viability in a different scenario.

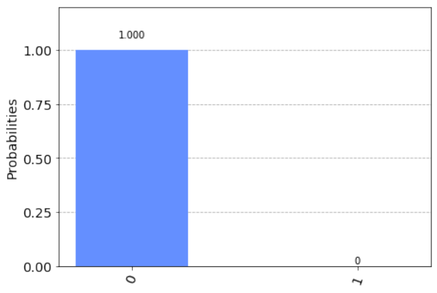


(a) Qubit sent $|0\rangle$

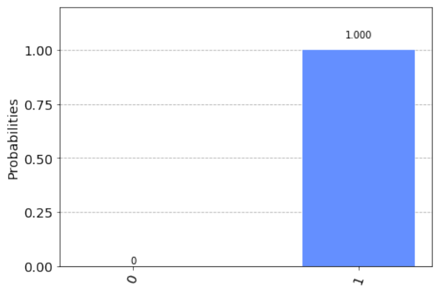


(b) Qubit sent $|1\rangle$

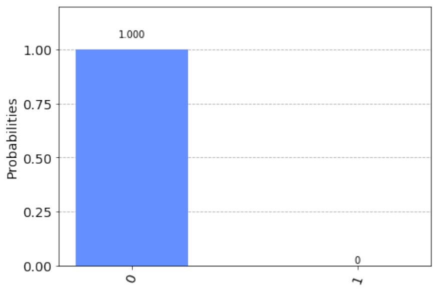
Figure 10: Output read after transmission, no error introduced



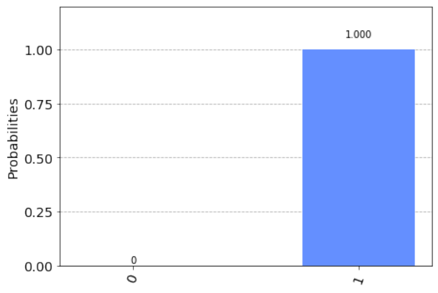
(a) Sent $|0\rangle$, error on $\mathrm{qbit}[0]$



(b) Sent $|1\rangle$, error on $\mathrm{qbit}[0]$

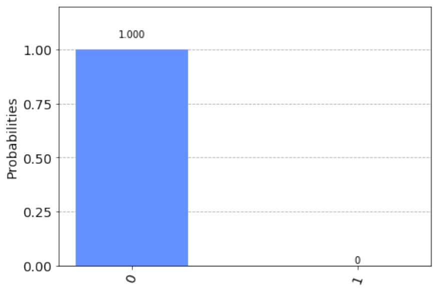


(c) Sent $|0\rangle$, error on $\mathrm{qbit}[1]$

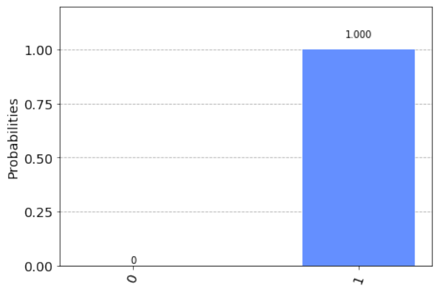


(d) Sent $|1\rangle$, error on $\mathrm{qbit}[1]$

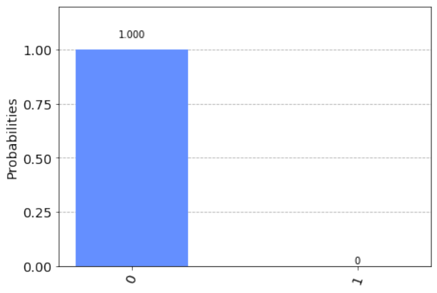
Figure 11: Results after transmission with errors on $\mathrm{qbit}[0]$ and $\mathrm{qbit}[1]$



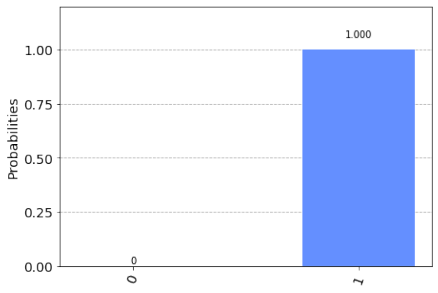
(a) Sent $|0\rangle$, error on $\mathrm{qbit}[2]$



(b) Sent $|1\rangle$, error on $\mathrm{qbit}[2]$



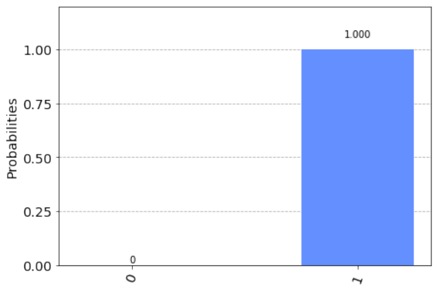
(c) Sent $|0\rangle$, error on $\mathrm{qbit}[3]$



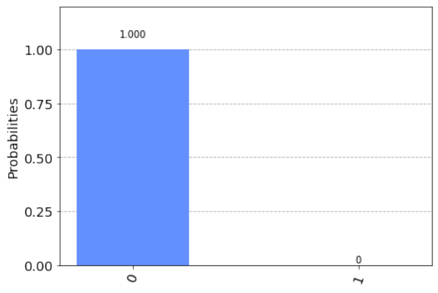
(d) Sent $|1\rangle$, error on $\mathrm{qbit}[3]$

Figure 12: Results after transmission with errors on $\mathrm{qbit}[2]$ and $\mathrm{qbit}[3]$

As shown up to this point the results on the simulator are in perfect accordance with the expected. The problem seems to arise when the $\mathrm{qbit}[4]$ suffers a bit flip. In this case, the system does not seem able to correct the problem and the result is the inverse of the input, as shown in Fig.13. I was not able to identify the reason for this incongruence between the theoretical work and the implementation.



(a) Sent $|0\rangle$, error on $\mathrm{qbit}[4]$



(b) Sent $|1\rangle$, error on $\mathrm{qbit}[4]$

Figure 13: Results after transmission with errors on $\mathrm{qbit}[4]$

6.1.2 *IBM Quantum System*

In this section I will show the results obtained by running the circuit to send a $|\bar{0}\rangle$ in the 16 qubit *imbq_guadalupe* machine IBM (2021).
To attempt to execute this algorithm on a real quantum machine, several things have to be considered. For instance, the circuits have to be mapped onto the real machine, which has limited connectivity among physical qubits, as shown in Fig.14. This restriction implies that more gates have to be added in order to swap logical qubits among physical qubits, such that non connected qubits can still be entangled.

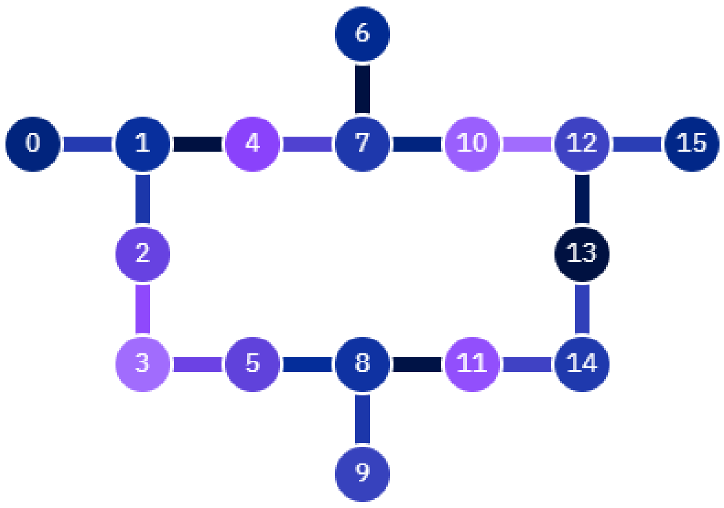


Figure 14: Connectivity of the qubits in the IBMQ Guadalupe.

The results shown in Fig.15 cannot be distinguished from pure noise, meaning all possible states are equiprobable and from Chapter 2 we conclude that no information is contained. The cause for this is that more than one error occurs during the computation and the sources for such errors can be decoherence of the qubits, noisy gates or noisy measurements. This result is not entirely surprising. Quantum computation is a new and demanding field of research that requires state of the art quantum processors with cryogenic components, control electronics, and classical computing technology. If a computation is too complicated, meaning with a large quantum gate count, current systems may not be able to perform the computation before errors build up in the circuit and render the output useless. Further research in building better gates and increasing the decoherence times of qubits is required.

Although this results are disappointing, the theoretical background for quantum error correcting codes is laid down and as long as companies like Google and IBM keep innovating and building better quantum computers, the hardware will eventually catch up. Quantum computation being such a disruptive technology, I believe at some point quantum computers together with powerfull error correction codes will be able to perform large and complex computation faster than classical

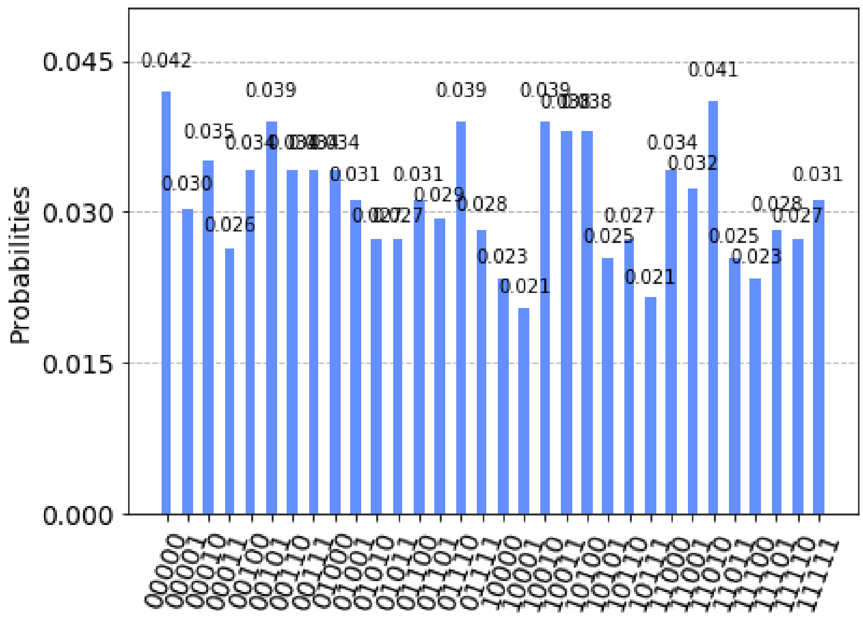


Figure 15: Output of the system after computation in the IBMQ Guadalupe

computers in some areas. After that, consumers, industries and businesses will significantly alter the way that they operate, sweeping away classical computers and replacing them forever with quantum computers.

7 CONCLUSION

The dissertation aimed to study quantum error correcting stabilizer codes and to present an algorithm that simulates a simple data transmission scenario with error correction. The algorithm has been evaluated both via simulation of a quantum machine, indicative of future quantum computing capabilities, and on a real quantum computer. We obtained good results in the simulator, unlike in the real quantum computer where the results were pure noise.

7.1 PROSPECT FOR FUTURE WORK

Recently, IBM announced that mid-circuit measurements will be possible, making fault-tolerant computation possible, although in the short term, due to requiring an increased gate count and circuit complexity, may be shown to not improve the error correction significantly. Having said that, better quantum gates, increased decoherence times and better qubit connectivity will improve quantum computation significantly, allowing for bigger and more complex computations.

The algorithm could also be tested using a better simulator. The naive approach of hard coding the errors into the transmission could be replaced by a simulator that includes a noise evolution operator and with that, better understand the limitations of the encoding. For instance, if the noisy simulation indicates that, similarly to a real machine, a specific error is more likely to appear, then a code with better ajusted properties could be used instead.

Lastly, stabilizer coding is a mathematical tool and, as such, disregards the nature of the qubits. Different error correcting methods that take into account the nature of the qubit may prove to be useful.

BIBLIOGRAPHY

Ibm quantum, 2021. URL https://quantum-computing.ibm.com.

C. Adami and N. J. Cerf. Quantum computation with linear optics. In *1st NASA Conference on Quantum Computing and Quantum Communications*, 2 1998.

A.S.Holevo. Bounds for the quantity of information transmitted by a quantum communication channel. *Probl. Peredachi Inf.*, 9:3–11, 1973.

Stephen Barnett. *Quantum information*, volume 16. Oxford University Press, 2009.

Charles H. Bennett, François Bessette, Gilles Brassard, Louis Salvail, and John Smolin. Experimental quantum cryptography. *Journal of Cryptology*, 5(1):3–28, Jan 1992. ISSN 1432-1378. doi: 10.1007/BF00191318. URL https://doi.org/10.1007/BF00191318.

Anasua Chatterjee, Paul Stevenson, Silvano De Franceschi, Andrea Morello, Nathalie P. de Leon, and Ferdinand Kuemmeth. Semiconductor qubits in practice. *Nature Reviews Physics*, 3(3):157–177, Mar 2021. ISSN 2522-5820. doi: 10.1038/s42254-021-00283-9. URL https://doi.org/10.1038/s42254-021-00283-9.

Richard Cleve and Daniel Gottesman. Efficient computations of encodings for quantum error correction. *Physical Review A*, 56(1):76–82, Jul 1997. ISSN 1094-1622. doi: 10.1103/physreva.56.76. URL http://dx.doi.org/10.1103/PhysRevA.56.76.

P. A. M. Dirac. *The Principles of Quantum Mechanics*. Clarendon Press, 1930.

Paul Adrien Maurice Dirac. *The principles of quantum mechanics*. Number 27. Oxford university press, 1981.

Jacques Dutka. The early history of the factorial function. *Archive for History of Exact Sciences*, 43 (3):225–249, Sep 1991. ISSN 1432-0657. doi: 10.1007/BF00389433. URL https://doi.org/10.1007/BF00389433.

Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum-enhanced measurements: Beating the standard quantum limit. *Science*, 306(5700):1330–1336, 2004. ISSN 0036-8075. doi: 10.1126/science.1104149. URL https://science.sciencemag.org/content/306/5700/1330.

Daniel Gottesman. Stabilizer codes and quantum error correction, phd thesis, 1997.

Daniel Gottesman. Theory of fault-tolerant quantum computation. *Physical Review A*, 57(1):127–137, Jan 1998. ISSN 1094-1622. doi: 10.1103/physreva.57.127. URL http://dx.doi.org/10.1103/PhysRevA.57.127.

W. Heisenberg. Über den anschaulichen inhalt der quantentheoretischen kinematik und mechanik. *Zeitschrift für Physik*, 43(3):172–198, Mar 1927. ISSN 0044-3328. doi: 10.1007/BF01397280. URL https://doi.org/10.1007/BF01397280.

D. G. Hoffman, Wal, D. A. Leonard, C. C. Lidner, K. T. Phelps, and C. A. Rodger. *Coding Theory: The Essentials*. Marcel Dekker, Inc., USA, 1991. ISBN 0824786114.

J. Kelly, R. Barends, A. G. Fowler, A. Megrant, E. Jeffrey, T. C. White, D. Sank, J. Y. Mutus, B. Campbell, Yu Chen, and et al. State preservation by repetitive error detection in a superconducting quantum circuit. *Nature*, 519(7541):66–69, Mar 2015. ISSN 1476-4687. doi: 10.1038/nature14270. URL http://dx.doi.org/10.1038/nature14270.

H. J. Kimble. The quantum internet. *Nature*, 453(7198):1023–1030, Jun 2008. ISSN 1476-4687. doi: 10.1038/nature07127. URL http://dx.doi.org/10.1038/nature07127.

Raymond Laflamme, Cesar Miquel, Juan Pablo Paz, and Wojciech Hubert Zurek. Perfect quantum error correcting code. *Phys. Rev. Lett.*, 77:198–201, Jul 1996. doi: 10.1103/PhysRevLett.77.198. URL https://link.aps.org/doi/10.1103/PhysRevLett.77.198.

Michael A Nielsen and Isaac Chuang. *Quantum computation and quantum information*. Cambridge Univ. Press., 2002.

Steven Roman. *Coding and information theory*, volume 134. Springer Science & Business Media, 1992.

C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3): 379–423, July 1948. ISSN 0005-8580. doi: 10.1002/j.1538-7305.1948.tb01338.x.

Peter W. Shor. Scheme for reducing decoherence in quantum computer memory. *Phys. Rev. A*, 52: R2493–R2496, Oct 1995a. doi: 10.1103/PhysRevA.52.R2493. URL https://link.aps.org/doi/10.1103/PhysRevA.52.R2493.

Peter W. Shor. Scheme for reducing decoherence in quantum computer memory. *Phys. Rev. A*, 52: R2493–R2496, Oct 1995b. doi: 10.1103/PhysRevA.52.R2493. URL https://link.aps.org/doi/10.1103/PhysRevA.52.R2493.

P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994. doi: 10.1109/SFCS.1994.365700.

P.W. Shor. Fault-tolerant quantum computation. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 56–65, 1996. doi: 10.1109/SFCS.1996.548464.

W. G. Unruh. Maintaining coherence in quantum computers. *Physical Review A*, 51(2):992–997, Feb 1995. ISSN 1094-1622. doi: 10.1103/physreva.51.992. URL http://dx.doi.org/10.1103/PhysRevA.51.992.

W. K. Wootters and W. H. Zurek. A single quantum cannot be cloned. *Nature*, 299(5886):802–803, Oct 1982. ISSN 1476-4687. doi: 10.1038/299802a0. URL https://doi.org/10.1038/299802a0.

A

QISKIT IMPLEMENTATION

A.1 THE ALGORITHM

```
from qiskit import *
#[n,k,d] quantum code

n = 5
k = 1
r = n-k

M1 = [[1,0,0,0,1],[1,1,0,1,1]]
M2 = [[0,1,0,0,1],[0,0,1,1,0]]
M3 = [[0,0,1,0,1],[1,1,0,0,0]]
M4 = [[0,0,0,1,1],[1,0,1,1,1]]

M = [M1,M2,M3,M4]

X = [[0,0,0,0,1],[1,0,0,1,0]]

Z = [[0,0,0,0,0],[1,1,1,1,1]]

def sendMessage(message,qc,qbits,output,errorbit):

    #encoding into the T space
    for i in range(r):
        qc.h(qbits[i])
        if M[i][1][i] == 1:
            qc.z(qbits[i])


    #Generators
    for i in range(r):
        for j in range(n):
            #Mx
            if M[i][0][j] and i != j:
                qc.cx(qbits[i],qbits[j])
            #Mz
            if M[i][1][j] and i != j:
                qc.cz(qbits[i],qbits[j])
        qc.barrier()


    #Encoding the message
    if (message == 1):
        for i in range(n):
            if X[0][i] == 1:
                qc.x(qbits[i])
            if X[1][i] == 1:
                qc.z(qbits[i])




    #Random Error during transmission
    qc.x(qbits[errorbit])


    #Decoding
    #Generators in reverse order
    for i in range(r-1,-1,-1):
        for j in range(n):
            #Mx
            if M[i][0][j] and i != j:
                qc.cx(qbits[i],qbits[j])
            #Mz
            if M[i][1][j] and i != j:
                qc.cz(qbits[i],qbits[j])
        qc.barrier()




    #decoding out of the T space
    for i in range(r):
        if M[i][1][i] == 1:
            qc.z(qbits[i])
        qc.h(qbits[i])


    for i in range(n):
        qc.measure(qbits[i],output[i])
```

A.2 READING THE RESULTS

The function countsToResult(), together with the function parity(), allows the user to transform the 5 qubit output that he receives and read its parity in order to know the message sent.

```
def parity(s):
    n = 0

    for i in range(len(s)):
        n = n + int(s[i])

    return(n%2 == 0)

def countsToResult(l):
    l = list(l)
    output = –
        '0' : 0,
        '1' : 0
    ”

    for i in range(len(l)):
        if(parity(l[i][0])):
            output['0'] += l[i][1]
        else:
            output['1'] += l[i][1]

    return output
```

```
#Both the message and the errorbit would change accordingly
message = 0
errorbit = 0

qbits = QuantumRegister(n,"qbit")

output = ClassicalRegister(n,"output")

qc = QuantumCircuit(qbits,output)

sendMessage(message,qc,qbits,output,errorbit)
```

A.2.1 *Qasm Simulator*

```
# Use qasm˙simulator
backend˙sim = BasicAer.get˙backend('qasm˙simulator')

# Execute the circuit on the qasm simulator.
job˙sim = execute(qc, backend˙sim, shots=1000)

# Grab the results from the job.
result˙sim = job˙sim.result()

counts = result˙sim.get˙counts(qc)

from qiskit.tools.visualization import plot˙histogram

out = countsToResult(counts.items())
plot˙histogram(output)
```

A.2.2 *IBMQ Guadalupe*

```
backend = provider.get˙backend('ibmq˙guadalupe')
qobj = assemble(transpile(qc, backend=backend), backend=backend,shots = 1024)
job = backend.run(qobj)

results = job.result()
counts = results.get˙counts()
plot˙histogram(counts)
```