December 2020

A New Family of Fault Tolerant Quantum Reed-Muller Codes

Harrison Beam Eggers
Clemson University, harrison.eggers@gmail.com

A New Family of Fault Tolerant Quantum Reed-Muller Codes

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Mathematics

---

by
Harrison Eggers
December 2020

Accepted by:
Dr. Felice Manganiello, Committee Chair
Dr. Shuhong Gao
Dr. Kevin James

Abstract

Fault tolerant quantum computation is a critical step in the development of practical quantum computers. Unfortunately, not every quantum error correcting code can be used for fault tolerant computation. Rengaswamy et. al. define CSS-T codes, which are CSS codes that admit the transversal application of the T gate, which is a key step in achieving fault tolerant computation. They then present a family of quantum Reed-Muller fault tolerant codes. Their family of codes admits a transversal T gate, but the asymptotic rate of the family is zero. We build on their work by reframing their CSS-T conditions using the concept of self-orthogonality. Using this framework, we define an alternative family of quantum Reed-Muller fault tolerant codes. Like the quantum Reed-Muller family found by Rengaswamy et. al., our family admits a transversal T gate, but also has a nonvanishing asymptotic rate.

We prove three key results in our search for a Reed-Muller CSS-T family with a nonvanishing rate. First, we show an equivalence between a code containing a self-dual subcode and the dual of that code being self-orthogonal. This allows us to more easily determine if a pair of codes define a CSS-T code. Next, we show that if C_1 and C_2 are both Reed-Muller codes that form a CSS-T code, C_1 must be self-orthogonal. This limits the rate of any family that is constructed solely from Reed-Muller codes. Lastly, we define a family of CSS-T codes by choosing $C_1 = \mathrm{RM}(r, 2r+1)$ and $C_2 = \mathrm{RM}(0, 2r+1)$ for some nonnegative integer r. We show that this family has an asymptotic rate of $\frac{1}{2}$, and show that it is the only possible CSS-T family constructed only from Reed-Muller codes where C_1 is self dual.

Acknowledgments

I want to thank my advisor, Dr. Felice Manganiello, for his guidance and support of my research. His insights and expertise proved invaluable. I particularly appreciated his availability in helping me work through hiccups in my research, which had a tendency to arise at the worst possible times. I also would like to thank Dr. Jessalyn Bolkema, our collaborator, for her perspective and clarity in our research meetings, as well as her willingness to provide guidance on this thesis. Her contributions were essential in every step of this process.

Table of Contents

Chapter 1

Introduction

In 1995, Peter Shor discovered an efficient quantum computing algorithm for factoring integers, demonstrating that quantum computers are capable of solving problems that classical computers cannot hope to solve. Since that time, despite many companies pouring millions of dollars into the development of quantum computing, quantum computers have yet to solve any practical problems that were significantly beyond the reach of classical computers. There are several reasons for this, but one of the biggest is the error rate of quantum computers. Because most quantum computers only have time to apply a few operations before their data becomes corrupt, efficient error correction is essential for quantum computing. Unlike classical computers, quantum computers experience a significant number of errors as the result of performing computational operations, rather than just while communicating or storing data. This has necessitated the development of quantum error correcting codes that allow quantum computers to manipulate data while that data is still encoded. This is referred to as *fault tolerant computation*, since the encoded data can be corrected for errors introduced as a result of the computation. Not all quantum error correcting codes allow for fault tolerant computation, so our focus is identifying and constructing codes which allow for fault tolerant computation.

1.1 Outline

In the remainder of this chapter, we will introduce some of the basic classical results we will use in later chapters. We define linear codes, enumerate some properties of linear codes, then introduce Reed-Muller codes. In Chapter 2, we define the basics of quantum computing. We first explain how to represent quantum information mathematically followed by the basics of manipulating quantum information. After introducing the basic tools of quantum computing, in Chapter 3 we discuss the challenges of quantum error correction, present our error model, and provide examples of a few simple quantum codes. We also prove that we can correct any error in our model if we can correct two specific types of errors. Building on this result, in Chapter 4, we focus on the stabilizer approach to quantum error correction and show that the stabilizer approach allows us to use classical codes to develop quantum codes through the CSS quantum code construction. In Chapter 5, we discuss fault tolerant quantum stabilizer codes and the definition of CSS-T codes, which are a class of CSS codes that allow fault tolerant computation. These results form the foundation for our work, which is presented in Chapter 6. We begin by building on the CSS-T definition to develop a simpler set of conditions for CSS-T codes. Using these conditions, we prove some general results about fault tolerant quantum Reed-Muller codes. We take use these results to describe a family of fault tolerant quantum codes which has nonvanishing rate. Finally, in Chapter 7, we conclude with a brief summary and suggestions for future work.

1.2 Classical Coding Background

The CSS-T construction that we use to define our family of fault tolerant quantum codes is based on classical codes, specifically binary linear block codes, so we first introduce some fundamentals of linear codes.

Definition 1.1. An $[\mathbf{n}, \mathbf{k}]$ **binary linear block code**, or simply an $[\mathbf{n}, \mathbf{k}]$ **linear code**, is a k dimensional subspace of $\mathbb{F}_2^n$.

We refer to k as the *dimension* of the code and n as the *length*. The process of encoding data is defined by a *generator matrix*, which is a full-rank $k \times n$ matrix that maps binary vectors of length k into *codewords*, that is elements of the code. There may be multiple generator matrices for a given code, but each generator matrix can only define one code. To correct errors using the

code, we often calculate a *syndrome* to identify which bits have an errors. The syndrome can be calculated by multiplying a vector by an $n-k \times n$ *parity check matrix*. If the vector is a codeword, the result of this multiplication is the zero vector, $\mathbf{0}$. Otherwise, the multiplication is nonzero for any vector not in the code. So we have that the kernel of the parity check matrix of a given code is exactly that code. Because codes can be defined by any full-rank matrix, the parity check matrix for a given code can act as a generator matrix for another code in its own right. We call the code whose generator matrix is the parity check matrix of a given code the *dual code* of that code. This generator matrix definition is not necessarily the easiest to use, so we define the dual code of a code in the following way:

Definition 1.2. The **dual code**, or **dual**, of a code C, denoted $C^{\perp}$, is defined as

$$C^{\perp} = \{v \in \mathbb{F}_2^n \mid v \cdot c = 0 \text{ for all } c \in C\},$$

where $v \cdot c$ is the normal dot product of two real vectors modulo 2.

Dual codes have some properties which can be determined based on the code of which they are the dual. The following proposition is especially useful.

Proposition 1.3. If C is an $[n, k]$ linear code, $C^{\perp}$ is an $[n, n-k]$ linear code.

Proof. Since the parity check matrix of a code is a generator matrix of the dual code, the dimension and length of the dual code is given by the dimensions of the parity-check matrix from C. The parity-check matrix for an $[n, k]$ code has dimensions $n-k \times n$, so the dual code is an $[n, n-k]$ linear code. □

We will see that dual codes play an important role in quantum error correction. One particularly interesting class of codes are the *self-dual* codes.

Definition 1.4. We call codes which are their own duals **self-dual** codes.

So C is a self-dual code if $C = C^{\perp}$. We also note here that any self-dual code must have dimension $\frac{n}{2}$. To understand the relationship between a code and its dual, consider the following example.

Example 1.5. Let C_1 and C_2 be defined by the following generator matrices:

$$G_{C_1} = \begin{pmatrix} 1 & 1 & 0 & 0 \end{pmatrix}, \quad G_{C_2} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

We see that C_1 has 2 codewords in it:

$$c_0 = (0,0,0,0)$$
$$c_1 = (1,1,0,0).$$

The dual code of C_1 has a 3×4 generator matrix, so we must find three linearly independent vectors to define the generator matrix of $C_1^{\perp}$. Note that for c_1, we have $(1,1,0,0) \cdot c_1 = 0$, $(0,0,1,0) \cdot c_1 = 0$, and $(0,0,0,1) \cdot c_1 = 0$. These vectors are linearly independent, so we conclude that a generator matrix for $C_1^{\perp}$ is

$$G_{C_1^{\perp}} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Performing a similar analysis on C_2, we begin by noting that C_2 has 4 codewords in it:

$$c_0 = (0,0,0,0)$$
$$c_1 = (1,1,0,0)$$
$$c_2 = (0,0,1,1)$$
$$c_3 = (1,1,1,1).$$

The dual code of C_2 has a 2×4 generator matrix, so we need only find two linearly independent vectors to define the generator matrix for $C_2^{\perp}$. Note that for every $c \in C_2$, $(1,1,0,0) \cdot c = 0$ and $(0,0,1,1) \cdot c = 0$. These two vectors are clearly linearly independent, so we conclude that a generator matrix for $C_2^{\perp}$ is

$$G_{C_2^{\perp}} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

Note that this is the same generator matrix as for C_2, so $C_2 = C_2^{\perp}$, and by definition we call C_2

self-dual.

Dual codes have several properties which will prove valuable later on.

Proposition 1.6. Let C be a linear code and $C^\perp$ its dual. C is the dual code of its dual code, that is $(C^\perp)^\perp = C$.

Proof. The parity check matrix of a dual code is the generator matrix of the original code. □

Proposition 1.7. Let C_1, C_2 be two linear codes. If $C_1 \subseteq C_2$, then $C_2^\perp \subseteq C_1^\perp$.

Proof. Let $c \in C_2^\perp$. Consider an arbitrary $c' \in C_1 \subseteq C_2$ note that $c \cdot c' = 0$. As this holds for any $c' \in C_1$, we conclude that $c \in C_1^\perp$, so $C_2^\perp \subseteq C_1^\perp$. □

When discussing codewords within a code, we often consider the distance between them.

Definition 1.8. The **distance** between two codewords c_1, c_2, denoted $d(c_1, c_2)$ is the number of entries where they differ in value.

We often consider the distance between a codeword and the **0** codeword, so we describe this as the weight

Definition 1.9. The **weight** of a codeword c, denoted $\mathrm{wt}(c)$ is the distance between c and the **0** codeword.

For binary codes, when the weight of a codeword is even, then we say that codeword is *even*. If every codeword in the code is even, we call that code an *even code*. Similarly we can extend the idea of distance to the entire code.

Definition 1.10. The **minimum distance** of a linear code is the minimum distance between any two distinct codewords in the code.

This property is very important to a code, since the number of simultaneous errors that the code can correct is given by $\left\lfloor \frac{d-1}{2} \right\rfloor$. As such, we often define linear codes in the following way.

Definition 1.11. An $[n, k]$ linear code with minimum distance d is called a $[\mathbf{n}, \mathbf{k}, \mathbf{d}]$ linear code.

When comparing two codes, it is often useful to compare how many bits of information they can carry for a given code length. This is referred to as the *rate*.

Definition 1.12. The **rate** of a $[n, k]$ linear code is given by $\frac{k}{n}$.

When comparing families, we often compare their *asymptotic rate*, which is the rate as n grows arbitrarily large.

Lastly, we introduce one specific family of codes that will be used in chapters 5 and 6. The Reed-Muller family of codes. The definition and properties of Reed-Muller codes follow the definition and properties given in [4]. We define this family of codes recursively.

Definition 1.13. Let $r, m \in \mathbb{N}$ with $r \leq m$. The **$\mathbf{r^{th}}$ order Reed-Muller code of length** 2^m, denoted $\mathrm{RM}(r, m)$, is defined as follows. If $r = m$, then we define $\mathrm{RM}(r, m) = \mathbb{F}_2^{2^m}$. If $r = 0$, we define $\mathrm{RM}(r, m) = \{\mathbf{0}, \mathbf{1}\}$, where $\mathbf{0}$ is the zero vector and $\mathbf{1}$ is the vector with ones in every position, both of length 2^m. Lastly, if $0 < r < m$, we define $\mathrm{RM}(r, m)$ recursively as

$$\mathrm{RM}(r, m) = \{(u, v \oplus u) \mid u \in \mathrm{RM}(r, m-1), v \in \mathrm{RM}(r-1, m-1)\},$$

where $\oplus$ represents element-wise addition modulo 2. If $r = m$, then the generator matrix is simply the $2^m \times 2^m$ identity matrix. If $0 < r < m$, the generator matrix is given by

$$G_{\mathrm{RM}(r,m)} = \begin{pmatrix} G_{\mathrm{RM}(r,m-1)} & G_{\mathrm{RM}(r,m-1)} \\ 0 & G_{\mathrm{RM}(r-1,m-1)} \end{pmatrix}.$$

And the generator matrix of $\mathrm{RM}(0, m)$ is defined as

$$G_{\mathrm{RM}(0,m)} = \begin{pmatrix} 1 & 1 & \dots & 1 \end{pmatrix},$$

which consists only of the codeword $\mathbf{1}$.

This family of codes are particularly popular because they are easy to analyze and implement. The following proposition describes how their parameters, r and m, relate to the length, dimension, and distance of the code.

Proposition 1.14. A $\mathrm{RM}(r, m)$ code is a $[2^m, \sum_{i=0}^{r} \binom{m}{i}, 2^{m-r}]$ linear code.

Proof. The length of the code was established in the definition, so we need only show the dimension and distance. We first show the dimension. Note that if $r = m$, we have that the dimension is 2^m, and we know $\sum_{i=0}^{m} \binom{m}{i} = 2^m$, so it holds in this case. This is also true for $m = 1$ by inspection. Now

assume that $\mathrm{RM}(i, m-1)$ has dimension $\sum_{j=0}^{i} \binom{m-1}{j}$. Since the dimension is given by the number of rows of the generator matrix, we have that $\dim(\mathrm{RM}(r,m)) = \dim(\mathrm{RM}(r, m-1)) + \dim(\mathrm{RM}(r-1, m-1)) = \sum_{i=0}^{r} \binom{m-1}{i} + \sum_{i=0}^{r-1} \binom{m-1}{i}$. Note the following properties of binomial coefficients:

$$\binom{m-1}{0} = \binom{m}{0} \quad \text{and} \quad \binom{m-1}{i-1} + \binom{m-1}{i} = \binom{m}{i}.$$

To use these properties, we pull out the first term of the first sum, then shift the index in the second sum, giving us

$$\sum_{i=0}^{r} \binom{m-1}{i} + \sum_{i=0}^{r-1} \binom{m-1}{i} = \binom{m-1}{0} + \sum_{i=1}^{r} \binom{m-1}{i} + \sum_{i=1}^{r} \binom{m-1}{i-1}.$$

Combining the two sums and using both of the stated properties of binomial coefficients, we see

$$\binom{m-1}{0} + \sum_{i=1}^{r} \binom{m-1}{i} + \sum_{i=1}^{r} \binom{m-1}{i-1} = \sum_{i=0}^{r} \binom{m}{i},$$

which is the dimension we wanted to prove.

We now show the distance. This is also easily seen for $m = 1$ and for $r = 0$ and $r = m$. Assume that $\mathrm{RM}(i, m-1)$ has minimum distance 2^{m-1-i} for all $0 \leq i < m$. If $0 < r < m$, then by the definition of the Reed-Muller code, $\mathrm{RM}(r, m)$ has two candidates for minimum distance, either twice the minimum distance of $\mathrm{RM}(r, m-1)$ or the minimum distance of $\mathrm{RM}(r-1, m-1)$. From our inductive hypothesis, we see that the former code has minimum distance 2^{m-1-r} and the latter code has minimum distance $2^{m-1-(r-1)} = 2^{m-r}$. Notice that twice the minimum distance of $\mathrm{RM}(r, m-1)$ gives us 2^{m-r} and the minimum distance of $\mathrm{RM}(r-1, m-1)$ is also 2^{m-r}, so the minimum distance of $\mathrm{RM}(r, m)$ must be 2^{m-r}. □

Reed-Muller codes have several properties, given in [4], that we will use when constructing our own families. The first property relates Reed-Muller codes which have the same length, but different degrees.

Proposition 1.15. For $0 \leq i \leq j \leq m$, we have $\mathrm{RM}(i, m) \subseteq \mathrm{RM}(j, m)$.

Proof. The proposition holds if $m = 1$ by direct computation and if $j = m$, since $\mathrm{RM}(m, m) = \mathbb{F}_2^{2^m}$. Assume inductively that $\mathrm{RM}(k, m-1) \subseteq \mathrm{RM}(l, m-1)$ for all $0 \leq k \leq l < m$. Let $0 < i \leq j < m$.

Then

$$\begin{aligned}\mathrm{RM}(i,m) &= \{(u, v \oplus u) \mid u \in \mathrm{RM}(i, m-1), v \in \mathrm{RM}(i-1, m-1)\} \\ &\subseteq \{(u, v \oplus u) \mid u \in \mathrm{RM}(j, m-1), v \in \mathrm{RM}(j-1, m-1)\} \\ &= \mathrm{RM}(j,m).\end{aligned}$$

So the proposition follows by induction if $0 < i$. If $i = 0$, we only need to show that the all-one vector of length 2^m is in $\mathrm{RM}(j,m)$ for $j < m$. Inductively assume the all-one vector of length 2^{m-1} is in $\mathrm{RM}(j, m-1)$. Then by Definition 1.13, we know the all-one vector of length 2^m is in $\mathrm{RM}(j,m)$ by choosing u to be the all-one vector of length 2^m and $v = \mathbf{0}$. □

The second important property states that the dual of a Reed-Muller code is another Reed-Muller code. Additionally, it shows the relationship between a Reed-Muller code and its dual.

Proposition 1.16. For $\mathrm{RM}(r,m)$, we have $\mathrm{RM}(r,m)^\perp = \mathrm{RM}(m-r-1, m)$.

Proof. We first make a slight extension of our definition to the Reed-Muller family by defining $\mathrm{RM}(-1,m) = \{\mathbf{0}\}$. Recall that $\mathrm{RM}(m,m) = \mathbb{F}_2^n$, so by this extension, we now have that $\mathrm{RM}(m,m)^\perp = \{\mathbf{0}\} = \mathrm{RM}(-1,m)$. We also have $\mathrm{RM}(-1,m)^\perp = \mathrm{RM}(m-(-1)-1,m) = \mathrm{RM}(m,m)$, so our proposition holds for the trivial cases. We see by direct computation that the proposition must hold for every r and m satisfying $-1 \le r \le m = 1$. Assume inductively that if $-1 \le i \le m-1$, then $\mathrm{RM}(i, m-1)^\perp = \mathrm{RM}((m-1)-i-1, m-1)$. let $0 \le r < m$. In order to prove equality, we first show that $\mathrm{RM}(m-r-1,m) \subseteq \mathrm{RM}(r,m)^\perp$. Let $x = (a, a+b) \in \mathrm{RM}(m-r-1,m)$ where $a \in \mathrm{RM}(m-r-1, m-1)$ and $b \in \mathrm{RM}(m-r-2, m-1)$, and let $y = (u, u+v) \in \mathrm{RM}(r,m)$ where $u \in \mathrm{RM}(r, m-1)$ and $v \in \mathrm{RM}(r-1, m-1)$. Note that $x \cdot y = 2a \cdot u + a \cdot v + b \cdot u + b \cdot v$, which can be simplified through direct use of our inductive hypothesis to $x \cdot y = b \cdot v$. From Proposition 1.15, we have $\mathrm{RM}(r-1, m-1) \subseteq \mathrm{RM}(r, m-1)$, so using the inductive hypothesis again, we conclude that $x \cdot y = \mathbf{0}$. Therefore, we conclude that $\mathrm{RM}(m-r-1,m) \subseteq \mathrm{RM}(r,m)^\perp$. Note that $\dim(\mathrm{RM}(r,m)) + \dim(\mathrm{RM}(m-r-1,m)) = \sum_{i=0}^{m} \binom{m}{i} = 2^m = \dim(\mathrm{RM}(r,m)) + \dim(\mathrm{RM}(r,m)^\perp)$. Subtracting the dimension of $\mathrm{RM}(r,m)$ from both sides, we conclude that $\dim(\mathrm{RM}(m-r-1,m)) = \dim(\mathrm{RM}(r,m)^\perp)$. Therefore, since we have that one code contains the other, $\mathrm{RM}(r,m)^\perp = \dim(\mathrm{RM}(m-r-1,m)$. □

Combining this result with the previous result, we see that every Reed-Muller code either

contains its dual or is contained within its dual. This structure makes Reed-Muller codes very convenient for us in later chapters.

Chapter 2

Quantum Computing

In classical computation, we represent data as a direct product of individual bits, where each bit is an element of $\mathbb{Z}_2$. If we want to modify classical data, we do so through a series of Boolean functions mapping $\mathbb{Z}_2^n$ to $\mathbb{Z}_2^n$. In quantum computation, we represent data as tensor products of qubit states, where each qubit state is a normalized (and nonzero) element of $\mathbb{C}^2$. If we want to modify quantum data, we do so through unitary transformations mapping $\mathbb{C}^{2^n}$ to $\mathbb{C}^{2^n}$. These essential differences in data storage and manipulation endow quantum computers with a more powerful tool set with which to solve problems.

2.1 Quantum Information

The basic building block of quantum computers is the quantum bit or *qubit*.

Definition 2.1. A **qubit** is a two-level quantum mechanical system.

A qubit is similar to a bit in that both are two-level systems. However, this is where the similarities end. A bit is a classical system, so can only be one level or the other. We usually represent the state of a bit as an element of $\mathbb{F}_2$. On the other hand, a qubit, being a quantum mechanical system, has properties that cannot be represented in the same way. The *state* of a qubit can be modeled as a normalized vector in $\mathbb{C}^2$ under the standard complex inner product. As the state of a qubit is a vector space over the complex numbers, the Hermitian transpose plays an important role in quantum computing and is denoted with a superscript $\dagger$.

Definition 2.2. If $X \in \mathbb{C}^{m\times n}$ for some $m, n \geq 1$, then the **Hermitian transpose** of X is $X^\dagger = \overline{X}^T$.

We often write these quantum states using the *bra-ket* or *Dirac* notation. A '*bra*', denoted $\langle x|$, represents a row vector and a '*ket*', denoted $|x\rangle$, represents a column vector. If $\vec{x} = [\alpha, \beta]^T \in \mathbb{C}^2$ is a column vector, then $|x\rangle = [\alpha, \beta]^T = \vec{x}$ and $\langle x| = [\overline{\alpha}, \overline{\beta}] = \vec{x}^\dagger$.

Definition 2.3. In quantum computing, we define the **standard** or **computational basis** of $\mathbb{C}^2$ to be

$$\left\{ |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}$$

Usually this basis is directly tied to a measurable state in the physical system such as the spin of an electron or the polarization of light. It is important to recognize that the zero ket, $|0\rangle$, is not the zero vector, but is instead a basis vector representing the zero state of our quantum state. Using this bra-ket notation, we can write an arbitrary quantum state $|\psi\rangle \in \mathbb{C}^2$ as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \alpha \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \beta \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

where $|\alpha|^2 + |\beta|^2 = 1$. In the state $|\psi\rangle$, α and β represent *probability amplitudes*. As might be expected by the term probability amplitude and the condition of normalization, the probability mass function of the qubit being observed in a given basis state is

$$p_{|\psi\rangle}(x) = \begin{cases} |\alpha|^2 & \text{if } x = |0\rangle \\ |\beta|^2 & \text{if } x = |1\rangle \end{cases}$$

Definition 2.4. We say a qubit is in **superposition** if it is in a quantum state consisting of a nontrivial linear combination of basis states.

One of the biggest differences between classical bits and qubits is measurement. Measurement and its effects have several different mathematical formulations, but for our purposes, this definition will suffice.

Definition 2.5. Measurement is the process of sampling the probability mass function defined by the state of a qubit.

When we measure a qubit in some state $|\psi\rangle$, we sample the probability mass function defined by that state: $p_{|\psi\rangle}$. Note that we are not able to directly determine α and β through a single measurement, although through repeated measurements of qubits in identical states, we can infer some of their properties. However, we cannot measure the same qubit repeatedly to sample the distribution. Due to quantum mechanical phenomena, when we measure a qubit, the qubit *collapses* to the basis state we measured, erasing all other states that disagree with the measurement. In other words, after measuring a qubit, we no longer experience superposition in the basis used to measure that qubit.

Example 2.6. Suppose we have a qubit in the state $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. If we measure the qubit to be in the basis state $|0\rangle$, then the qubit is no longer in the state $|\psi\rangle$, but is instead in the state $|0\rangle$.

As a result of the collapse of superposition, it can be difficult to determine exactly what the probability amplitudes of a given quantum state are. If you can create the state several times from scratch, then you can infer the probability amplitudes relative to each other. However, there is one aspect of the probability amplitudes that is impossible to measure.

Definition 2.7. For a quantum state $|\phi\rangle$ and constant $\alpha \in \mathbb{C}$, define $|\psi\rangle = \alpha|\phi\rangle$, with $|\alpha| = 1$. We call α the **global phase** of $|\psi\rangle$.

The global phase of a quantum state is totally undetectable. This means we can ignore it without losing any generality in our calculations. Furthermore, we can choose it to be whatever we want. As a matter of convention, we often choose the global phase such that the probability amplitude of the all-zeros state is a real number.

When we define operations on qubits, we will write the matrix representation with respect to the computational basis. However, this is not the only significant basis. Another basis that is important to be familiar with is the *Hadamard* basis.

Definition 2.8. In terms of the standard basis, the **Hadamard basis** consists of equal superposition between the two computational bases:

$$\left\{|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right\}.$$

This basis plays an important role in computation and error correction, so we will revisit it in the following sections.

As mentioned before, the state of an n qubit system can be described as the tensor product of n copies of $\mathbb{C}^2$. When we write states of multiple qubit systems, we we take the *Kronecker product* between of each of the basis states to define the basis for the multi-qubit system.

Definition 2.9. Let $A \in \mathbb{C}^{n\times m}$ be an n by m matrix and let $B \in \mathbb{C}^{p\times q}$ be a p by q matrix. We define the **Kronecker product** of A and B to be the $np \times mq$ matrix given by:

$$A \otimes B = \begin{pmatrix} a_{1,1}B & a_{1,2}B & \dots & a_{1,m}B \\ a_{2,1}B & a_{2,2}B & \dots & a_{2,m}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1}B & a_{n,2}B & \dots & a_{n,m}B \end{pmatrix}$$

When writing multi-qubit states, we often drop the $\otimes$ or even move everything into the same ket for simplicity. Using the latter notation, we see the computational basis for a two qubit system is

$$\left\{ |0\rangle \otimes |0\rangle = |00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix},\ |0\rangle \otimes |1\rangle = |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix},\ |1\rangle \otimes |0\rangle = |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix},\ |1\rangle \otimes |1\rangle = |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right\}.$$

So for $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ and $|\phi\rangle = \gamma|0\rangle + \delta|1\rangle$, we see the following representations are equivalent:

$$|\psi\rangle \otimes |\phi\rangle = |\psi\rangle|\phi\rangle = |\psi\phi\rangle = \alpha\gamma|00\rangle + \alpha\delta|01\rangle + \beta\gamma|10\rangle + \beta\delta|11\rangle$$

As with the single qubit case, the probabilities of measuring each of these basis states is given by the square of the modulus of each of the coefficients. This gives us that the joint probability mass

function of $|\psi\rangle \otimes |\phi\rangle$ is

$$p_{|\psi\rangle\otimes|\phi\rangle}(x,y) = \begin{cases} |\alpha\gamma|^2 & \text{if } x = |0\rangle, y = |0\rangle \\ |\alpha\delta|^2 & \text{if } x = |0\rangle, y = |1\rangle \\ |\beta\gamma|^2 & \text{if } x = |1\rangle, y = |0\rangle \\ |\beta\delta|^2 & \text{if } x = |1\rangle, y = |1\rangle \end{cases}$$

If we measure both qubits, then the final state collapses to just a single two-qubit basis state. However, if we only measure a single qubit, then only that qubit collapses, and the resulting multi-qubit state drops all of the states that contradict that measurement.

Example 2.10. To understand the effect of measurement on a multi-qubit quantum state, consider the three-qubit state

$$|\psi\rangle = \frac{1}{\sqrt{5}}\left(|000\rangle + |001\rangle + |010\rangle + |110\rangle + |111\rangle\right).$$

Suppose we measure a 0 on the last qubit. The states $|001\rangle$ and $|111\rangle$ no longer exist, since they contradict our measured reality. In this case, after measurement, our system is in the state

$$|\psi'\rangle = \frac{1}{\sqrt{3}}(|000\rangle + |010\rangle + |110\rangle)$$

Suppose instead that we measure a 1 on that qubit. Now the states $|000\rangle$, $|010\rangle$ and $|110\rangle$ no longer exist, so our system is in the state

$$|\psi''\rangle = \frac{1}{\sqrt{2}}(|001\rangle + |111\rangle).$$

The last major concept we introduce is that of entanglement.

Definition 2.11. We say a collection of n qubits are **entangled** if their states cannot be written as a tensor product of individual, independent qubit states.

Example 2.12. Consider the Bell state

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

If we could write this as a tensor product of individual qubits, then we could write

$$|\psi\rangle \otimes |\phi\rangle = \alpha\gamma|00\rangle + \alpha\delta|01\rangle + \beta\gamma|10\rangle + \beta\delta|11\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

for some $\alpha, \beta, \gamma, \delta \in \mathbb{C}$ such that $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ and $|\phi\rangle = \gamma|0\rangle + \delta|1\rangle$. By equating the coefficients on each basis state, we have the following system of equations.

$$\alpha\gamma = \frac{1}{\sqrt{2}} \tag{2.1}$$

$$\alpha\delta = 0 \tag{2.2}$$

$$\beta\gamma = 0 \tag{2.3}$$

$$\beta\delta = \frac{1}{\sqrt{2}}. \tag{2.4}$$

Attempting to solve by replacement, we note from (2.1) that $\alpha = \frac{1}{\gamma\sqrt{2}}$. Replacing this α into (2.2), we see $\delta = 0$. However, this forces (2.4) to be $\beta\delta = 0 \neq \frac{1}{\sqrt{2}}$, so our system is inconsistent. Thus, there are no $|\psi\rangle$ and $|\phi\rangle$ such that $|\psi\rangle \otimes |\phi\rangle = |\Phi^+\rangle$. So we conclude that $|\Phi^+\rangle$ represents an entangled state.

2.2 Quantum Operations

In classical computation, the fundamental operations take the form of functions called gates which map the states of one or two bits to one bit. Some common gates include NOT, AND, OR, and XOR. If a set of gates allows us to create any possible binary function, we call it *functionally complete*. There are many different functionally complete sets of gates, but the most common set is the singleton set consisting only of the NAND gate which is simply a single gate consisting of an AND gate followed by a NOT gate.

In quantum computation, the simplest operations take the form of unitary operations which map one, two, or sometimes three qubits to the same number of qubits with which we started. In the spirit of classical computation, we often refer to these unitary operations as *quantum gates*, or just *gates*. If a set of quantum gates can approximate any arbitrary unitary operation with a finite number of gates from that set, we call the set *universal*.

2.2.1 Common Gates

Definition 2.13. An operation, $U : \mathbb{C}^{2^n} \to \mathbb{C}^{2^n}$, is said to be **unitary** if $UU^\dagger = U^\dagger U = I_{2^n}$, where I_{2^n} is the identity operation. We will denote the set of all unitary operations on n qubits as $\mathbb{U}^n$

The most basic set of quantum gates is the *Pauli gates*. These gates play an important role in nearly every quantum algorithm and are especially important in quantum error correction. They consist of the following operations:

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix},$$

where i is the imaginary constant. These gates have an important property: each of these operations is its own inverse, that is $X^2 = Z^2 = Y^2 = I^2 = I$. We will prove later that any matrix in $\mathbb{C}^{2\times 2}$ can be written as a complex linear combination of these four gates. We will leverage this fact when we discuss error correction to expand the class of errors we can correct significantly. We also note that the Pauli gates form the foundation for the *Pauli group* on one qubit:

Definition 2.14. The **Pauli group**, $\mathcal{P}$, is defined as the subgroup of unitary matrices generated by the Pauli rotation matrices.

$$\mathcal{P} = \langle X, Z, Y \rangle = \{\pm I, \pm iI, \pm X, \pm iX, \pm Z, \pm iZ, \pm Y, \pm iY\}.$$

We can extend this idea to n qubits, by considering n Kronecker products of elements from the Pauli group on one qubit

$$\mathcal{P}^n = \left\{ \bigotimes_{i=1}^{n} P_i \;\middle|\; P_i \in \mathcal{P} \right\} \subseteq \mathbb{U}^n.$$

Because of the algebraic structure of the Pauli group, we note that for each $p \in \mathcal{P}^n$, we have $p^2 = I_{2^n}$. Thus, every element from the Pauli group on n qubits has eigenvalues of only ± 1. This result makes the Pauli group on n qubits a convenient set of operations to work with.

Based on the definition of Pauli group, we can derive another set of important gates: the Clifford Group.

Definition 2.15. The **Clifford Group** on n qubits is the normalizer of the Pauli group on n qubits.

$$\text{Cliff}_n = \left\{ U \in \mathbb{U}^n \;\middle|\; UPU^\dagger \in \mathcal{P}^n \text{ for all } P \in \mathcal{P}^n \right\} \subseteq \mathbb{U}^n$$

We will see that the Clifford group contains many very important gates, but not all gates. Aside from the Pauli gates, the Hadamard (H), Controlled NOT (CX), and Phase (S) are three of the most popular Clifford gates. Their matrix representations are

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad CX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}.$$

The Clifford gates are exactly the gates that can, with a couple other limitations, be simulated efficiently on a classical computer. We will discuss this in greater detail in Chapter 5, but the primary takeaway is that Clifford gates are easy to use, but limited in function. Fortunately, it turns out that these are the only gates required to perform error correction with stabilizer codes, so their study is still worthwhile.

Not all quantum gates are Clifford gates. The T gate (T), the Toffoli gate (CCX), and the $\sqrt{\text{SWAP}}$ gate are three common non-Clifford gates. Their matrix representations are

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}, \quad CCX = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \quad \sqrt{\text{SWAP}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2}(1+i) & \frac{1}{2}(1-i) & 0 \\ 0 & \frac{1}{2}(1-i) & \frac{1}{2}(1+i) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Because the Clifford gates can be simulated classically, these gates play an important role in allowing quantum algorithms to make the most of their quantum advantage.

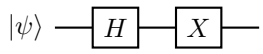
As discussed above, there are many universal sets of quantum gates. Some sets are easier than others to implement on different quantum computing architectures, and others are eas-

ier to use to design quantum algorithms. For our purposes, we will consider the universal set $\{X, Z, Y, H, CX, S, T\}$. We will see later that this set simplifies the problem of developing fault tolerant stabilizer codes significantly.

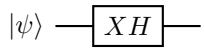
2.2.2 Representation

Taking inspiration from the logic circuit used in classical computing, quantum computing uses a similar circuit notation. We use wires to indicate which qubit is being acted upon, and we use boxes overlaid on the wire to indicate the operation.

Example 2.16. If you have some qubit in quantum state $|\psi\rangle$ and want to apply a Hadamard transformation followed by a Pauli X gate, you could describe it in matrix notation as $XH|\psi\rangle$ or in the circuit notation as



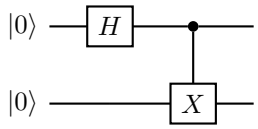
If you wanted to consider this operation as a single gate that applied XH in one operation, you could describe the circuit as



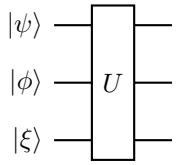
These circuits are equivalent in the sense that they describe the same operation on the qubit in the state $|\psi\rangle$.

Some gates require more than one input qubit. Sometimes the extra qubit is a control qubit, as is the case with the Controlled Not or Toffoli gates, other times it may be best interpreted as an input, as in the case of a $\sqrt{\text{SWAP}}$ gate. In the case of a control qubit, we use a dot on the wire of the control qubit and a line drawn down to the gate being controlled, as the following example shows.

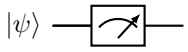
Example 2.17. Suppose we have two qubits both in the $|0\rangle$ state initially and wanted to entangle them to create a Bell state: $|00\rangle + |11\rangle$. To create this state, you apply a Hadamard gate to one qubit and then use that qubit as a control for a control not gate operating on the other qubit. The circuit to perform this operation can be described as follows.



In the case where none of the inputs really make sense as controls, we just draw a larger box around all the input wires, as demonstrated in the circuit below, which applies an arbitrary unitary operation to all three qubits.



When we denote the measuring of a qubit, we use the following meter symbol:



There are more gates used to design quantum algorithms than we discussed here, but these are the critical gates for our purposes. With this background, we can now move into a discussion on the necessity and challenges of quantum error correction.

Chapter 3

Quantum Coding

In classical coding theory, to correct bit-flip errors, we encode some number of data bits into a larger number of code bits. There are many approaches to perform this encoding, but a popular class of classical error correcting codes are linear block codes, which map data bits to codewords using a generator matrix and corrects the error using a parity-check matrix to determine which bits have flipped. In quantum coding theory, we have to correct a larger class of errors, but we find that we can use an analogous approach. However, because we are correcting different kinds of errors, we cannot use classical codes directly. Furthermore, qubits have to be treated a little bit more carefully than their classical counterpart, which presents certain challenges that must be overcome in order to correct quantum errors.

3.1 Challenges

Quantum mechanics presents certain challenges to quantum computing. Firstly, quantum computers experience significantly more errors than their classical counterparts, so any codes we develop must have minimal processing overhead. Furthermore, measurements destroy quantum superposition, so we have to be able to correct errors without reading the qubits themselves. This is a significant departure from classical coding theory, where we can read the bits and identify which one has an error. Additionally, the no-cloning theorem prevents us from duplicating our data in a classical sense, since we cannot duplicate arbitrary quantum states without destroying the original. Lastly, the set of errors we must correct is much larger than the bit-flips of the classical case. So we

need different codes to correct this larger set of errors.

3.1.1 Error Rates

In classical computing, we only need to really be concerned with data corruption when communicating and storing data, and these data rates are low. One study, [7], estimated that classical DRAM experiences an error rate of $1.476 \times 10^{-14} \frac{\text{errors}}{\text{bit} \cdot \text{second}}$. The error rates of qubits are much higher than the rates of classical bits. A study from 2019, [11], measured the error rates of an IBM quantum computer to be approximately approximately $10^4 \frac{\text{errors}}{\text{qubit} \cdot \text{second}}$. In quantum computing, we have to correct communication and storage errors too, but we also must correct errors introduced by operating on the qubits. The same IBM study also found that single qubit operations caused errors at a rate of approximately 0.5%. In light of these error rates, quantum codes must be easily implemented and allow us to modify encoded qubits while they are still encoded.

3.1.2 Measurements

We discussed in the previous chapter how measuring a qubit collapses its superposition and alters any qubits entangled with our measured qubit. This concept alone breaks any hope of using classical codes directly. In classical linear block codes, for example, we read the bits, then multiply the bits by the parity check matrix to generate an error syndrome that tells us where the errors occurred. In the quantum computing paradigm, this approach breaks down on the first step. In order to correct errors, we have to invent a new method for checking parity that does not require observing the qubits first to determine what state they are in.

3.1.3 No-Cloning Theorem

The No-Cloning Theorem presents another challenge to quantum coding theorists: we cannot duplicate data. So we cannot bypass the measurement problem by duplicating a state, measuring the new one, then correcting the old one. Furthermore, we cannot generate repetition codes in the same way we do with classical data.

Theorem 3.1 (No-Cloning Theorem)**.** There is no unitary operation U and initial quantum state $|e\rangle$ such that $U|\psi\rangle|e\rangle = |\psi\rangle|\psi\rangle$ for every quantum state $|\psi\rangle$,.

Proof. Suppose such a unitary U and initial state $|e\rangle$ do exist. Then for any two arbitrary quantum states $|\psi\rangle$,$|\phi\rangle$, we have

$$U\frac{1}{\sqrt{2}}(|\psi\rangle + |\phi\rangle) \otimes |e\rangle = \frac{1}{\sqrt{2}}(|\psi\rangle + |\phi\rangle) \otimes \frac{1}{\sqrt{2}}(|\psi\rangle + |\phi\rangle) = \frac{1}{2}(|\phi\rangle|\phi\rangle + |\phi\rangle|\psi\rangle + |\phi\rangle|\psi\rangle + |\phi\rangle|\phi\rangle).$$

However, since U is linear, we must also have

$$U\frac{1}{\sqrt{2}}(|\psi\rangle + |\phi\rangle) \otimes |e\rangle = \frac{1}{\sqrt{2}}(U|\psi\rangle|e\rangle + U|\phi\rangle|e\rangle) = \frac{1}{\sqrt{2}}(|\psi\rangle|\psi\rangle + |\phi\rangle|\phi\rangle).$$

But we see that for arbitrary states, $\frac{1}{2}(|\phi\rangle|\phi\rangle + |\phi\rangle|\psi\rangle + |\phi\rangle|\psi\rangle + |\phi\rangle|\phi\rangle) \neq \frac{1}{\sqrt{2}}(|\psi\rangle|\psi\rangle + |\phi\rangle|\phi\rangle)$. Therefore, there cannot be any U or $|e\rangle$ that allows us to duplicate an arbitrary quantum state. □

3.1.4 Error Model

For our purposes, we will concern ourselves with single-qubit linear errors, that is, errors that can be represented as $E \in \mathbb{C}^{2\times 2}$. Linear block codes are designed to correct bit-flips, but in quantum error correction, we must correct infinitely many different errors, some of which are not necessarily invertible. Our error class includes any unitary operation that was unintentionally applied to a single qubit, but it also includes non unitary operations, as this example shows.

Example 3.2. Let $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ be the state of our qubit. Suppose that this qubit decoheres due to the environment, such that after our error, the qubit is in the basis state $|0\rangle$. This error can be described by the matrix

$$E = \frac{1}{\alpha + \beta}\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}.$$

So this error is an element of $\mathbb{C}^{2\times 2}$. Note that

$$EE^\dagger = \frac{1}{\alpha + \beta}\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}\frac{1}{\overline{\alpha + \beta}}\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} = \frac{1}{|\alpha + \beta|^2}\begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix} \neq I_2.$$

So in this case our error is not unitary, but is an error we will be able to correct.

3.2 Error Correction

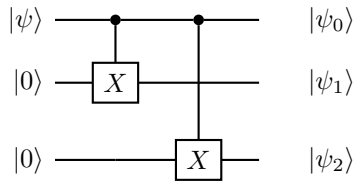
Despite these challenges, it turns out that quantum error correction is actually feasible. The basic approach is similar to classical computing. To protect our qubits against errors, we encode our data qubits, often referred to as *logical qubits*, into a larger number of qubits, often referred to as *physical qubits*. The specific method for this encoding, though, is often very different to

Definition 3.3. An $[\![\mathbf{n}, \mathbf{k}]\!]$ **quantum code** is a 2^k dimensional subspace of a 2^n dimensional Hilbert space

Essentially, a quantum code code defines the relationship between the states of n physical qubits and the states of k logical qubits. We refer to physical qubits states that are in the code as *code states*. Since we cannot measure our physical qubits directly without destroying our code, we must add other qubits that we can measure to tell us what and where the errors are. These extra qubits are referred to as *ancillary qubits*.

3.2.1 Specific Errors

To illustrate how the challenges facing quantum coding affect quantum error correcting codes, we begin by providing examples of codes which correct specific Pauli errors. We begin with a code that can correct a single X error. Consider the quantum code that maps a qubit in the logical qubit $|\psi\rangle$ to three physical qubits via the following circuit.



This maps the an arbitrary state $|\psi\rangle$ to the following encoded state:

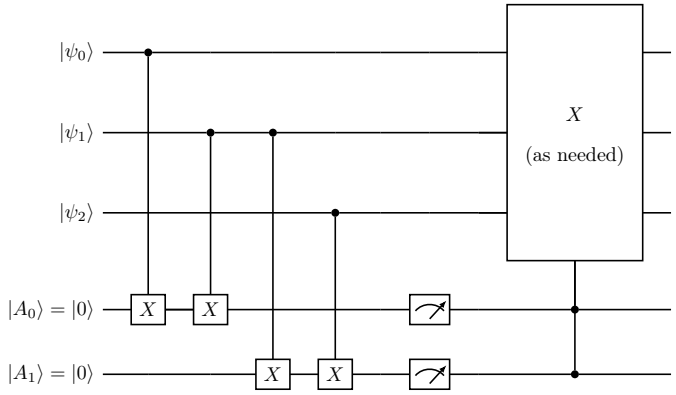
$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \mapsto |\psi_0\psi_1\psi_2\rangle = \alpha|000\rangle + \beta|111\rangle.$$

Note that this circuit does not clone $|\psi\rangle$, since the cloned state $|\psi\psi\psi\rangle$ would be

$$|\psi\psi\psi\rangle = (\alpha|0\rangle+\beta|1\rangle)^{\otimes 3} = \alpha^3|000\rangle+\alpha^2\beta(|001\rangle+|010\rangle+|100\rangle)+\alpha\beta^2(|011\rangle+|110\rangle+|101\rangle)+\beta^3|111\rangle,$$

which is clearly a different state than our encoded state. Thus, we have not violated the No-Cloning theorem.

To correct an X error on this code, we cannot simply measure the qubits directly and figure out which one was flipped, because that would destroy the data contained in $|\psi_0\psi_1\psi_2\rangle$. Instead we measure their syndrome via the following circuit.



Depending on the measurements of $|A_0\rangle$ and $|A_1\rangle$, we can identify which qubit had the error, since the error will cause a mismatch between one qubit and another qubit in the code. The syndrome can be interpreted according to the following table.

$\|A_0\rangle$	1	1	0	0
$\|A_1\rangle$	1	0	1	0
Error Location	2	1	3	No Error

Table 3.1: Syndrome Decoding

Note that, despite finding the error, we cannot deduce which state ψ is in, so this operation preserves the superposition and allows us to apply the fix to the correct qubit without destroying our original data.

Example 3.4 (X Error Correction)**.** Suppose we encode our logical qubit that is in the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ via the encoding above, and that there is an error on the middle qubit. So our physical qubits are in the state.

$$|\psi_0\psi_1\psi_2\rangle = \alpha|010\rangle + \beta|101\rangle$$

After the first CNOT, the ancillary qubits are in the state

$$|A_0A_1\rangle = \alpha|00\rangle + \beta|10\rangle,$$

because the probability amplitude of the $|1\rangle$ state in the first code qubit is β. Because the first and second code qubits are entangled, when applying the second CNOT, the probability amplitude of a flip given that the first qubit is in the state $|0\rangle$ is 1, and the probability amplitude of a flip given the first qubit is in the state $|1\rangle$ is 0. So the only basis state represented is $|10\rangle$. Recall we can ignore the global phase, so after the second CNOT, the qubit is in the following state

$$|A_0A_1\rangle = |10\rangle$$

After the third CNOT, the second ancillary qubit flip when the second code qubit is $|1\rangle$, so we have

$$|A_0A_1\rangle = \beta|10\rangle + \alpha|11\rangle$$

Using the same process as for the second CNOT, after the fourth CNOT, we have

$$|A_0A_1\rangle = |11\rangle$$

When we measure this syndrome, we get $|11\rangle$ with probability 1. This syndrome, according to the table 3.1, corresponds to an X error in the second qubit. Applying the fix to the second qubit gives us the following state:

$$|\psi_0\psi_1\psi_2\rangle = \alpha|000\rangle + \beta|111\rangle,$$

so we have restored the encoded state to a valid codeword.

We can modify this code to correct Z errors as well. A Z error, also known as a phase error, maps

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \mapsto Z|\psi\rangle = \alpha|0\rangle - \beta|1\rangle$$

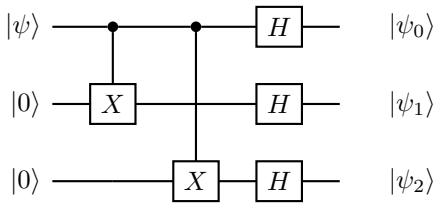
in the computational basis. If we instead first perform a Hadamard transformation on $|\psi\rangle$, we see

$$H|\psi\rangle = \alpha|+\rangle + \beta|-\rangle \mapsto ZH|\psi\rangle = \alpha|-\rangle + \beta|+\rangle$$

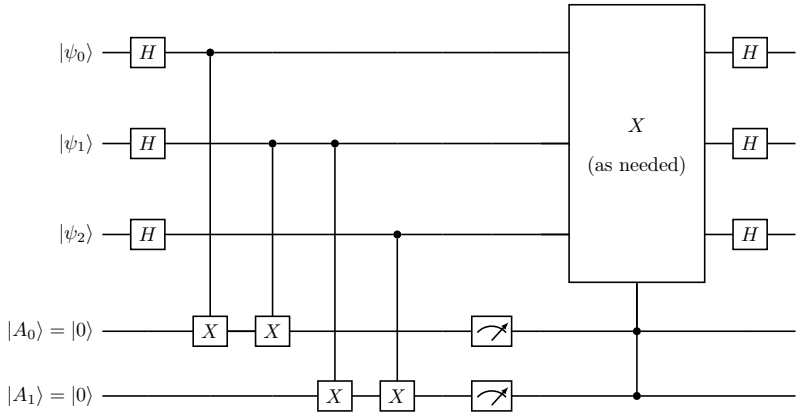
This looks similar to the X error, which we already know how to solve. However in this case, we need to map the starting state in the computational basis to the triplicate state in the Hadamard basis because this error flips a bit in that basis.

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \mapsto |\psi_0\psi_1\psi_2\rangle = \alpha|{+}{+}{+}\rangle + \beta|{-}{-}{-}\rangle$$

To achieve this, we use the following circuit



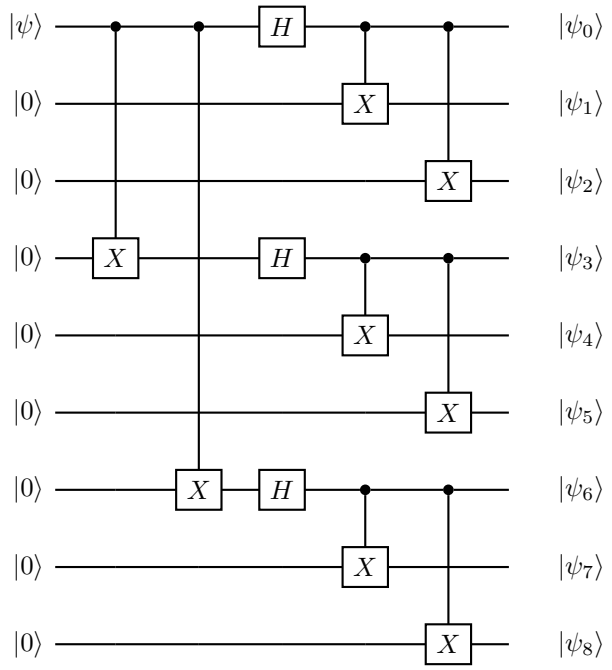
To measure a syndrome in this code, the simplest approach is to apply Hadamard gates to convert back to the computational basis, then take the same syndrome measurement that we used in the X error example followed by applying the necessary corrections. To leave the code where we started, we apply Hadamard gates at the end of the process to prepare to catch another phase error.



Note that this is not necessarily the most efficient or effective error correction routine. It simply demonstrates that error correction of phase errors is possible. The best circuit for syndrome measurement will likely depend on the physical construction of the quantum computer.

We can can combine these codes in such a way that we can correct both an X and a Z error. Developed by Peter Shor in 1995, the Shor code involves encoding a qubit in a three qubit Z

code, followed by encoding each of those qubits into a 3 qubit X code [8]. The encoding circuit is shown below.



In this encoding we map an arbitrary logical state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ to the following encoded state:

$$|\psi\rangle \mapsto |\psi_0\psi_1\psi_2\psi_3\psi_4\psi_5\psi_6\psi_7\psi_8\rangle = \frac{\alpha}{2\sqrt{2}}(|000\rangle + |111\rangle)^{\otimes 3} + \frac{\beta}{2\sqrt{2}}(|000\rangle - |111\rangle)^{\otimes 3},$$

where $|\phi\rangle^{\otimes 3} = |\phi\phi\phi\rangle$. As we might expect from its construction, this code is capable of correcting a single X error and a single Z error concurrently. Furthermore, the Shor code can correct a single Y error. Note that

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

The correction of Y depends on the fact that a Y error essentially consists of an X error and a Z error occurring simultaneously. The following example may help illustrate how the Shor code works to correct Y errors.

Example 3.5 (Correcting a Y error)**.** Suppose we encode a logical qubit in the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ using the Shor code, and that there is a Y error on the fifth qubit, that is the state $|\psi_4\rangle$. Our errored

state would be

$$\frac{\alpha}{2\sqrt{2}}(|000\rangle+|111\rangle)(i|010\rangle-i|101\rangle)(|000\rangle+|111\rangle)+\frac{\beta}{2\sqrt{2}}(|000\rangle-|111\rangle)(i|010\rangle+i|101\rangle)(|000\rangle-|111\rangle).$$

Factoring out the i's from both terms, we recognize that i is a global phase, so we can drop it altogether, giving us the following state:

$$\frac{\alpha}{2\sqrt{2}}(|000\rangle+|111\rangle)(|010\rangle-|101\rangle)(|000\rangle+|111\rangle)+\frac{\beta}{2\sqrt{2}}(|000\rangle-|111\rangle)(|010\rangle+|101\rangle)(|000\rangle-|111\rangle).$$

We note that in the second set of triplets, there is an X error, since $|010\rangle$ and $|101\rangle$ are not codewords in the three qubit X code. Furthermore, we see that there are no X errors in the first or third set of triplets, so we need not correct those blocks. Using the X code approach on the block of bits containing the error, that is $|\psi_3\psi_4\psi_5\rangle$, we correct the error using the approach as shown in example 3.4, giving us the state

$$\frac{\alpha}{2\sqrt{2}}(|000\rangle+|111\rangle)(|000\rangle-|111\rangle)(|000\rangle+|111\rangle)+\frac{\beta}{2\sqrt{2}}(|000\rangle-|111\rangle)(|000\rangle+|111\rangle)(|000\rangle-|111\rangle).$$

Now we see that the entire second set of triplets is out of phase with the other two sets of triplets. To correct this error, we consider the Z code on the first qubit of each set of triplets, that is

$$|\psi_0\psi_3\psi_6\rangle=\alpha|+-+\rangle+\beta|-+-\rangle.$$

The Z code can correct this error by applying a Z gate to $|\psi_3\rangle$. This gives us the overall state of

$$\frac{\alpha}{2\sqrt{2}}(|000\rangle+|111\rangle)(|000\rangle+|111\rangle)(|000\rangle+|111\rangle)+\frac{\beta}{2\sqrt{2}}(|000\rangle-|111\rangle)(|000\rangle-|111\rangle)(|000\rangle-|111\rangle).$$

We see that this is the original encoded state, and of course a valid codeword in the code.

From this example, we see that the Shor code can not only correct X and Z errors, but can also correct Y errors. Thus, the Shor code can correct any single Pauli error in our physical qubits.

3.2.2 General Errors

Now that we see that we can correct Pauli errors in single qubits, we explain the special role that the Pauli gates in correcting arbitrary errors.

Lemma 3.6. The Pauli gates, along with the identity operation, form a basis for $\mathbb{C}^{2\times 2}$.

Proof. We begin by noting that $\dim(\mathbb{C}^{2\times 2}) = 4$, and since our set has four elements, we need only show that they are linearly independent. Let $\alpha_1, \alpha_2, \alpha_3, \alpha_4 \in \mathbb{C}$ and consider

$$\alpha_1 I + \alpha_2 X + \alpha_3 Y + \alpha_4 Z = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

This gives us the following system of equations

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & -i & 0 \\ 0 & 1 & i & 0 \\ 1 & 0 & 0 & -1 \end{pmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Row reducing, we have

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

So $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 0$. Therefore, $\{I, X, Y, Z\}$ forms a basis for $\mathbb{C}^{2\times 2}$. □

Since our error model consists of any operator in $\mathbb{C}^{2\times 2}$ and the Pauli gates form a basis for this space, we can decompose any error as a linear combination of Pauli gates. This fact allows us the prove that we can correct any error simply by measuring a syndrome and correcting the associated Pauli error.

Theorem 3.7. If we can independently correct any Pauli error on a single qubit by measuring a syndrome calculated on ancillary qubits, we can correct any arbitrary error in $\mathbb{C}^{2\times 2}$.

Proof. Let $|\psi\rangle$ be the qubit in the code that experiences the error, and $|A\rangle$ be the ancillary qubits used to measure that error, and $E \in \mathbb{C}^{2\times 2}$ be the error. Recall that $E = \alpha_1 I + \alpha_2 X + \alpha_3 Z + \alpha_4 Y$

with $\alpha_1, \alpha_2, \alpha_3, \alpha_4 \in \mathbb{C}$, since the Pauli gates form a basis for $\mathbb{C}^{2\times 2}$. Applying the error to a $|\psi\rangle$, our quantum state is

$$(E|\psi\rangle)|A\rangle = (\alpha_1 I|\psi\rangle + \alpha_2 X|\psi\rangle + \alpha_3 Z|\psi\rangle + \alpha_4 Y|\psi\rangle)|A\rangle.$$

After calculating the syndrome using the ancillary qubits, we have

$$(E|\psi\rangle)|A\rangle = \alpha_1 I|\psi\rangle|A_I\rangle + \alpha_2 X|\psi\rangle|A_X\rangle + \alpha_3 Z|\psi\rangle|A_Z\rangle + \alpha_4 Y|\psi\rangle|A_Y\rangle.$$

Now when we measure the ancillary qubits, the superposition collapses to a single error syndrome, leaving the qubits in the state

$$E|\psi\rangle|A_B\rangle = B|\psi\rangle|A_B\rangle,$$

where $B \in \{I, X, Y, Z\}$. Thus, simply by measuring the ancillary qubits, we have reduced our error to a single Pauli error. We can now apply the fix for the Pauli error indicated by syndrome given by the measured ancilla, and we have corrected the error, E. □

This theorem means that the Shor code can correct any error in $\mathbb{C}^{2\times 2}$, which is surprising given its simple construction. Furthermore, it shows that any code that can correct both at least one X error and at least one Z error can correct any error in our model. This leads us into the definition of one of the most popular classes of quantum error correcting codes, the *stabilizer code*.

Chapter 4

Stabilizer Codes

Stabilizer codes are to quantum error correction what linear block codes are to classical error correction. Stabilizer codes provide a simple framework for correcting errors in a set of qubits. We can take this comparison further by using classical linear codes to generate stabilizer codes with certain properties. This allows us to bypass some of the oddities of quantum information and leverage the well-known theory behind classical codes.

4.1 Definition

Before we define what a quantum stabilizer code is, we first recall what a stabilizer is.

Definition 4.1. Let H be a group with some action on a set S. The **stabilizer** of an element $s \in S$, denoted H_s, is defined as the subgroup of H that fixes s. That is,

$$H_s = \{h \in H \mid hs = s\}.$$

Now using the definition of a stabilizer, we can now define the stabilizer code.

Definition 4.2. A **stabilizer code** on n qubits is defined by choosing a subgroup of the Pauli group on n qubits, $P \subseteq \mathcal{P}^n$. We then define the set of codewords in our stabilizer code to be the quantum states which are stabilized by this subgroup.

$$V_P = \left\{ |\psi\rangle \in \mathbb{C}^{2^n} \;\middle|\; p|\psi\rangle = |\psi\rangle \text{ for all } p \in P \right\}.$$

For a stabilizer code, we typically choose the stabilizer first out of elements of the Pauli group, then figure out what it stabilizes to find our codewords. This may feel a little bit backwards from the definition of a stabilizer, but we will see that it simplifies the process of finding a code tremendously. However, we can simplify this definition a little bit more through the next result.

Proposition 4.3. Let G be a set of generators for the subgroup $P \subseteq \mathcal{P}^n$, and let $V_P \subseteq \mathbb{C}^{2^n}$ be the set of quantum states stabilized by this subgroup P. If $|\psi\rangle$ is a quantum state such that $g|\psi\rangle = |\psi\rangle$ for all $g \in G$, then $|\psi\rangle \in V_P$.

Proof. Suppose $|\psi\rangle$ is a quantum state such that $g|\psi\rangle = |\psi\rangle$ for all $g \in G$ and let $p \in P$ be an arbitrary element. Consider $p|\psi\rangle$. Since $p \in P$, we can write $p = \prod_{i=1}^{k} g_i$. So we see $p|\psi\rangle = \prod_{i=1}^{k} g_i|\psi\rangle$. Since each g_i stabilizes $|\psi\rangle$ we have $\prod_{i=1}^{k} g_i|\psi\rangle = |\psi\rangle$. Thus, we see that $p|\psi\rangle = |\psi\rangle$, therefore $|\psi\rangle \in V_P$. □

With this result, we can define the code by only considering the quantum states stabilized by some generating set $G = \{g_1, g_2, ..., g_l\}$, such that $\langle g_1, g_2, ..., g_l\rangle = P \subseteq \mathcal{P}^n$. We could define P using any generating set we want, but we will see that there is value in choosing generating set to be *minimal*.

Definition 4.4. Let H be a group and G be a generating set of H. We say G is a **minimal generating set**, or is **minimal**, if every proper subset of G generates a strict subgroup of H.

If we choose our generators to be the minimal generating set, then we can relate the cardinality of that set to the dimension of the vector space formed by the stabilized quantum states.

Proposition 4.5. Let $P \subseteq \mathcal{P}^n$ be a subgroup of the Pauli group on n qubits, G be a minimal generating set for P. If V_p is the set of all qubit states stabilized by every element of G, then V_p forms a vector space of dimension $2^{n-|G|}$.

This proof draws heavily from proofs in [3] and [1].

Proof. Let $|\psi\rangle, |\phi\rangle \in V_p$ be quantum states and $\alpha, \beta \in \mathbb{C}$ satisfying $|\alpha|^2 + |\beta|^2 = 1$ and $g \in G$. Note that $g(\alpha|\psi\rangle + \beta|\phi\rangle) = \alpha g|\psi\rangle + \beta g|\phi\rangle = \alpha|\psi\rangle + \beta|\phi\rangle$. Therefore, V_P is a vector subspace of $\mathbb{C}^{2^n}$.

To show that the dimension of the vector space is $2^{n-|G|}$, we proceed by induction on $|G|$. Consider the case where the stabilizer is generated by a single, nontrivial element $g_1 \neq I_{2^n}$.

Since $g_1 \in \mathcal{P}^n$, all of its eigenvalues must either be 1 or -1. Furthermore, since $g_1 = \bigotimes_{i=1}^{n} p_i$ is nontrivial, at least one of the $p_i \in \mathcal{P}$ gates is in the set $\{X, Z, Y\}$. The trace of a Kronecker product of operators is the same as the product of the trace of each operator, so $\mathrm{Tr}(g_1) = \prod_{i=1}^{n} \mathrm{Tr}(p_i)$. However, $\mathrm{Tr}(X) = \mathrm{Tr}(Y) = \mathrm{Tr}(Z) = 0$, and since there is at least one $p_i \in \{X, Y, Z\}$, we conclude $\mathrm{Tr}(g_1) = 0$. The trace of an operator is the sum of its eigenvalues, and since the only possible eigenvalues of g_1 are ± 1, we conclude that the number of eigenvectors with eigenvalue 1 is the same as the number of eigenvectors with eigenvalue -1, so these two subspaces divide the vector space of all possible computational states in half. The vectors stabilized by g_1 are the vectors corresponding to the eigenvalue 1. The dimension of the original vector space is 2^n, so the vector space V_p must have dimension $2^{n-1} = 2^{n-|G|}$.

Now suppose that for minimal generating sets of size $l-1$, the $+1$ eigenspace has dimension $2^{n-(l-1)}$ and consider a minimal generating set $G = \{g_1, g_2, ..., g_l\}$. From our assumption, we have $G' = \{g_1, g_2, ..., g_{l-1}\}$ defines a vector space of dimension $2^{n-(l-1)}$. Let I be the identity element in $\mathbb{U}^n$. Note that $\frac{1}{2}(I + g_i)$ is the projector onto the $+1$ eigenspace of the generator g_i, so $\mathrm{Proj}_{G'} = \prod_{i=1}^{l-1} \frac{I+g_i}{2}$ projects onto the $+1$ eigenspace of G'. Note that $\mathrm{Tr}(\mathrm{Proj}_{G'} g_l) = 0$, since G is minimal. Therefore, following the same argument as in the base case, we conclude that the $+1$ eigenspace of G has dimension exactly half of the dimension of the $+1$ eigenspace of G'. So V_P has dimension $2^{n-(l-1)+1} = 2^{n-l} = 2^{n-|G|}$. Therefore, by induction, we conclude that for n qubits and any minimal generating set G, the dimension of V_P is always $2^{n-|G|}$. □

Now that we know the dimension of V_P, we have the following result.

Proposition 4.6. Let G be a minimal set of generators of a subgroup $P \subseteq \mathcal{P}^n$ such that $|G| = n-k$ for some $k \in \mathbb{N}$. A stabilizer code defined by G forms an $[\![n, k]\!]$ quantum code.

Proof. Since $|G| = n - k$, the dimension of vector space V_P is $2^{n-(n-k)} = 2^k$, so this space is isomorphic to the space consisting of k logical qubits. So this code is an $[\![n, k]\!]$ quantum stabilizer code. □

Given that we can describe classical codes and quantum codes using analogous definitions of length and dimension, we naturally look for an analogous definition for the classical concept of distance. Fortunately, for stabilizer codes such an analog exists, but, like classical computing, we first need to define the weight of an error.

Definition 4.7. The **weight** of an element in $\mathcal{P}^n$ is given by the number of non-identity elements in its tensor representation.

Now, we can discuss the concept of distance.

Definition 4.8. The **minimum distance**, or simply **distance**, of a stabilizer code is the minimum weight of the errors which the stabilizer code cannot correct.

Using this definition of distance, we can see that it parallels the classical concept of distance. As it turns out, from [3], we know that in order to guarantee the correction of t concurrent errors using a stabilizer code, the code must have distance $d \geq 2t + 1$. So now, we introduce the analogous definition in its entirety.

Definition 4.9. We say a stabilizer code is an $[\![\mathbf{n}, \mathbf{k}, \mathbf{d}]\!]$ **quantum code** if it uses n physical qubits to encode k logical qubits with minimum distance d.

Recall from our discussion of the Shor code that $Y \in \mathcal{P}$ can be written as $Y = iXZ$, so the Y operation is simply the sequential application of Z followed by X and some global phase factor. We can leverage this fact to introduce a simplified notation for the generators of a stabilizer code. Consider an $[\![n, k]\!]$ stabilizer code defined by the generators G. Each generator is the tensor product of some number of single-qubit Pauli gates with perhaps a global phase. Instead of defining each generator, g_i by a string of n, one-qubit Pauli gates, we can instead denote it as a string in $\mathbb{Z}^{2n}$, where the first n qubits have a 1 in each position where g_i has an X or a Y gate and a 0 in all other positions and the second set of n qubits as a 1 in each position where g_i has a Z or a Y gate and a 0 in all other positions. We will often organize these strings into a matrix where each string is a row.

Example 4.10. Consider the $[\![5, 1, 3]\!]$ stabilizer code defined by the generators

$$g_1 = X \otimes Z \otimes Z \otimes X \otimes I$$

$$g_2 = I \otimes X \otimes Z \otimes Z \otimes X$$

$$g_3 = X \otimes I \otimes X \otimes Z \otimes Z$$

$$g_4 = Z \otimes X \otimes I \otimes X \otimes Z$$

Using this new notation, we would write

$$\begin{array}{l} g_1 = \\ g_2 = \\ g_3 = \\ g_4 = \end{array} \left(\begin{array}{ccccc|ccccc} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right).$$

This 5 qubit code is the shortest code that is capable of correcting a single arbitrary error. To measure the error syndrome with a stabilizer code, initialize to $|0\rangle$ one ancillary qubit for each generator. First apply a Hadamard gate to each of the ancillary qubits, then apply a CNOT to the ancillary controlled on the qubit in each position with a X in the corresponding generator, then apply a Hadamard gate to the ancillary qubits. Next, apply a CNOT gate to the ancillary controlled on the qubit in each position with an Z in the corresponding generator. Finally measure each ancillary qubit to determine the syndrome of the error. [3]

As it turns out, we can describe the X,Z, and Shor codes in this stabilizer formalism as well.

Example 4.11. We begin with the X code. The X code encodes 1 logical qubit into 3 physical qubits, so our generator set must have $3 - 1 = 2$ linearly independent elements. For an arbitrary encoded state, we see that the codewords take the form

$$|\psi_0\psi_1\psi_2\rangle = \alpha|000\rangle + \beta|111\rangle.$$

Note that $Z \otimes Z \otimes I \in \mathcal{P}^3$ stabilizes this state. None of the gates in this stabilizer change the $|000\rangle$ basis state, and the two Z gates flip the phase of $|111\rangle$ by -1 twice, once through the first bit and once through the second. So we see that ultimately this state remains the same after the application of this element. Therefore, it stabilizes the state and we have one element of our generating set. We can choose $I \otimes Z \otimes Z$ using the same argument to show it stabilizes the state. Since there are only two generators that are not identical or trivial, we conclude the set is minimal. Represented in the binary vector format, the X code can be expressed as

$$\left(\begin{array}{ccc|ccc} 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right).$$

The Z code is similar. It will have two generators in its minimal generating set. An arbitrary encoded state has the form

$$|\psi_0\psi_1\psi_2\rangle = \alpha|+++\rangle+\beta|---\rangle = \frac{\alpha}{2\sqrt{2}}(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)+\frac{\beta}{2\sqrt{2}}(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)(|0\rangle-|1\rangle).$$

Note that for this code, $X \otimes X \otimes I \in \mathcal{P}^3$ stabilizes the state. After the application of this operator, we have

$$(X \otimes X \otimes I)|\psi_0\psi_1\psi_2\rangle = \frac{\alpha}{2\sqrt{2}}(|1\rangle + |0\rangle)(|1\rangle + |0\rangle)(|0\rangle + |1\rangle) + \frac{\beta}{2\sqrt{2}}(|1\rangle - |0\rangle)(|1\rangle - |0\rangle)(|0\rangle - |1\rangle).$$

We note that nothing has really changed on the terms multiplied by α. For the terms multiplied by β, we factor out -1 from the first qubit and the second qubit. These two factors of -1 cancel, and we have the original state

$$(X \otimes X \otimes I)|\psi_0\psi_1\psi_2\rangle = \frac{\alpha}{2\sqrt{2}}(|0\rangle + |1\rangle)(|0\rangle + |1\rangle)(|0\rangle + |1\rangle) + \frac{\beta}{2\sqrt{2}}(|0\rangle - |1\rangle)(|0\rangle - |1\rangle)(|0\rangle - |1\rangle).$$

So we conclude that this operator stabilizes this state. Using a similar approach, we find that $I \otimes X \otimes X$ stabilizes the state. These two states are neither identical nor trivial, so we have that they form the minimal generating set for our stabilizer. Writing this in the binary string form, we have

$$\left(\begin{array}{ccc|ccc} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{array}\right).$$

The last example is the Shor Code. There are $9 - 1 = 8$ generators required to represent this code. Using what we know about the codes from which the Shor code is constructed, we can define these stabilizers without proving anything about the underlying codewords. Recall that the Shor code consists of three separate X correction codes tacked together, so there are six generators with two Z's and the rest I representing the X code on each triplet of qubits. The remaining two generators account for the overarching Z code. This code is first defined on the bits 0,3, and 6, but each of those bits are expanded into three more bits for the X code, so the stabilizer of the Z code expands with them. Where the initial Z would have generators $X_0I_1I_2X_3I_4I_5I_6I_7I_8$ and $I_0I_1I_2X_3I_4I_5X_6I_7I_8$, after encoding the three X codes, these two generators also expand to be $X_0X_1X_2X_3X_4X_5I_6I_7I_8$ and $I_0I_1I_2X_3X_4X_5X_6X_7X_8$. Using the binary string form, the Shor code can be described as

$$\left(\begin{array}{ccccccccc|ccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right).$$

This simple alternative representation turns out to be a powerful tool in creating and analyzing stabilizer codes. Furthermore, organizing these rows into a matrix is more than a convenient notation. There is a useful relationship between the generating set and its matrix representation. This result and subsequent proof can be found in [3].

Proposition 4.12. The generating set is minimal if and only if the rows of its matrix representation are linearly independent.

Proof. We prove the contrapositive. Note first that g_i^2 must equal I for all i. Observe that addition modulus 2 in the row vector representation corresponds to multiplication of group elements. Thus the rows of the check matrix are linearly dependent if and only if the product of some generators is equal to the identity, up to some overall multiplicative factor. However, $-I \notin S$, so the multiplicative factor must be 1, and the last condition corresponds to the condition that $g_j = g_j^{-1} = \prod_{i \neq j} g_i^{a_i}$, so we can remove g_j from the generating set without reducing its span. Therefore, our original set of generators is not minimal. □

Given this result, we can now utilize our linear algebra tools to analyze the generators of any subgroup of the Pauli group on n qubits. Furthermore, this correspondence between generators and matrices provides the foundation for a special family of stabilizer codes: CSS codes.

4.2 CSS Codes

A special class of stabilizer codes are the Calderbank-Shor-Steane (CSS) codes. They were published by Calderbank and Shor in [2] and independently by Steane in [9], both in 1996. CSS

codes provide the connection between linear block codes and quantum stabilizer codes promised at the beginning of this chapter.

Definition 4.13. To create a **CSS code**, we begin with any two classical linear codes satisfying $C_2 \subseteq C_1$. Define the generators of our stabilizer code based on the parity check matrix of C_1, H_{C_1} and the dual of C_2, $H_{C_2^\perp}$ using the binary string representation:

$$G = \left(\begin{array}{c|c} H_{C_2^\perp} & 0 \\ 0 & H_{C_1} \end{array}\right).$$

The key feature of CSS stabilizer codes is that the generators defining the code contain either X or Z terms, but not both. This simplifies the analysis of these codes, since there are disjoint subset of generators used to detect and correct each basic Pauli error. There are elements in the stabilizer that contain both X and Z terms, but we can ignore those when calculating syndromes, since we only need generators.

Proposition 4.14. Let C_1 be an $[n, k_1, d_1]$ classical linear code and let $C_2 \subseteq C_1$ be a $[n, k_2, d_2]$ linear code. Define $d_2^\perp$ to be the minimum distance of the dual of C_2. The CSS code defined by C_1 and C_2 is an $[\![n, k_1 - k_2, d]\!]$ quantum code with $d \geq \min(d_1^\perp, d_2)$.

Proof. The dimension of a stabilizer code is $n - |G|$ where G consists of the rows of the binary string representation. The parity check matrix of C_1 has $n - k_1$ rows, and the parity check matrix of $C_2^\perp$ has $n - (n - k_2) = k_2$ rows in it. So the dimension of the CSS code is $n - (n - k_1 + k_2) = k_1 - k_2$. Since the X and Z stabilizers are disjoint, we can consider each in turn. Note that the Z stabilizers, defined by C_1, can correct $\left\lfloor \frac{d_1 - 1}{2} \right\rfloor$ X errors and the X stabilizers, defined by $C_2^\perp$, can correct $\left\lfloor \frac{d_2^\perp - 1}{2} \right\rfloor$ Z errors. Since, in order to correct a Y error, we need to correct both an X and a Z error, the number of errors we can correct simultaneously must be $\left\lfloor \frac{\min(d_1, d_2^\perp) - 1}{2} \right\rfloor$. If a stabilizer code can correct t errors, the distance of that code is $d > 2t + 1$. So the distance of a CSS code is $d > 2 \left\lfloor \frac{\min(d_1, d_2^\perp) - 1}{2} \right\rfloor + 1 = \left\lfloor \min(d_1, d_2^\perp) \right\rfloor$. Since distance is always an integer, we have $d \geq \min(d_1, d_2^\perp)$. □

With this definition of CSS codes, we now have a tool by which we can construct quantum codes using only classical coding theory. We note immediately that the X, Z, and Shor codes are all CSS codes because they all have a minimal generating set which can be written without

elements containing both X and Z generators. CSS codes are not necessarily optimal however. The shortest CSS code which can correct a single error is the Steane code, a seven-qubit code, which was discovered by Andrew Steane in 1996 [10]. However, the five-qubit code that we introduced above is the smallest stabilizer code that can correct a single error.

Example 4.15. To construct the CSS code, the Steane code defines C_1 to be a [7,4,3] Hamming code and $C_2 = C_1^{\perp}$. The parity check matrix for C_1 is identical to that of $C_2^{\perp}$, which gives us the following generator matrix for the stabilizer code:

$$\left(\begin{array}{ccccccc|ccccccc} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{array}\right).$$

Chapter 5

Fault Tolerant Stabilizer Codes

Because quantum computers experience errors as a result of performing operations on their qubits, it is impossible to reliably decode a qubit before performing an operation, then recode afterwards. Instead, quantum computers must operate on its logical qubits by manipulating the underlying physical qubits. The Gottesman-Knill theorem shows that we can apply Clifford gates to our stabilizer-encoded logical qubits, but in order to achieve universal quantum computation, we need to be able to apply at least one non-Clifford gate. As it turns out, only certain stabilizer codes allow us to apply a logical non-Clifford gate by operating on the physical qubits.

5.1 Gottesman-Knill Theorem

The Gottesman-Knill Theorem is a significant result based on this stabilizer representation for quantum states. This theorem can be found in [3].

Theorem 5.1 (Gottesman-Knill)**.** Suppose a quantum computation is performed which involves only the following elements: state preparations in the computational basis, Clifford gates, measurements of observables in the Pauli group, and classical controls conditioned on those measurements. Such a computation may be simulated on a classical computer in polynomial time.

This theorem shows that there is a large class of quantum computations that can be performed efficiently on a classical computer. Furthermore, it means that in order for a quantum computer to be able to outperform a classical computer, it needs to either prepare states that are

not in the computational basis or be able to implement a quantum operation not in the Clifford group. One promising method for preparing non-computational basis states is called *magic state distillation*. This involves approximating a state that is not attainable using Clifford gates on computational states, then using that state to perform non-Clifford operations. However, our interest is in implementing non-Clifford quantum operations logically without the need of a magic state. Often quantum computers can perform, on physical qubits, certain non-Clifford operations as easily as Clifford operations. By developing a process for these non-Clifford physical operations to apply a non-Clifford logical operation, we can make the most of the physical capabilities of a given quantum computer. For some computer architectures, the T gate is fairly easy to implement. Furthermore, it is simpler to analyze than many other non-Clifford gates, so this gate will be the focus of our efforts to find quantum codes for fault tolerant computing.

5.2 CSS-T Codes

As the Gottesman-Knill Theorem indicates, in order to achieve universal fault tolerant computation, we must be able to implement a non-Clifford gate on the encoded qubits by manipulating the physical qubits. Unfortunately, not every stabilizer code allows us to do this.

Definition 5.2. We say a quantum code **admits** a physical operation if the application of that physical operation maps each code state to a valid code state.

Note that this is similar to the definition of a stabilizer. However, for an operation to be admitted, it does not have to map each code state to itself, but rather it must map each code state to some, potentially different, code state. From this, we see that the condition for an operation to be a admitted is weaker than the condition for it to be in the stabilizer. A quantum stabilizer code must admit every element of its stabilizer, but not every admitted operation is in the stabilizer.

In [6], Rengaswamy et. al. outline the requirements for a stabilizer code to admit the application of T gates. The class of codes that they found are stabilizer codes which can be used to define a code that allows fault tolerant universal quantum computation using the T gate. Their work demonstrates the requirements for a code to admit the T gate applied to any subset of physical qubits. However, not all subsets are equally easy to use for universal computation.

Definition 5.3. We say a quantum operation is **transversal** if it involves the application of the

same gate to each physical qubit.

Transversal operations are easiest to implement and help limit the propagation of errors. They are also relatively easy to analyze. In light of this, we will focus our attention to the codes which admit the application of a transversal T operation. Before we introduce the specific conditions given in [6] for fault tolerant stabilizer codes, we must first introduce a few other classical coding concepts.

Definition 5.4. We say a codeword $c \in C$ is **supported** on a binary vector $s \in \mathbb{F}_2^n$ if $c \times s = c$, where $\times$ is the bit-wise multiplication operation.

When considering whether a codeword is supported on a binary vector s, it can be helpful to consider s as a filter which blocks all codewords that have 1's in any position that s has a 0, and passes all the other codewords. With this in mind, consider the following example.

Example 5.5. Let $s = (1,1,1,0,0,0)$, $c_1 = (1,1,0,0,0,0)$, $c_2 = (0,0,1,1,0,0)$, and $c_3 = (0,0,0,0,1,1)$. Considering the filter interpretation, we see that c_1 is supported on s because it isn't blocked by the 0's in the fourth, fifth, and sixth positions. However, c_2 gets blocked by the 0 in the fourth position, and c_3 gets blocked by the 0's in the fifth and sixth position, so neither c_2 or c_3 are supported by s. More formally, we can observe that $c_1 \times s = (1,1,0,0,0,0) = c_1$, so c_1 is supported on s. Now for c_2, we have $c_2 \times s = (0,0,1,0,0,0) \neq c_2$, and for c_3, we see $c_3 \times s = (0,0,0,0,0,0) \neq c_3$, so c_2 and c_3 are not supported on s.

We can extend this definition for the support of a single codeword on a vector to an entire code over a vector.

Definition 5.6. For some $[n,k]$ linear code C and binary string $s \in \mathbb{F}_2^n$, we consider the **restriction of C to (the support of)** s, denoted $C|_s$, to be the code created by removing from C all codewords which are not supported on s then shortening the code by removing the indices corresponding to zeros in s from the remaining codewords.

Informally, the restriction of a code, C, to the support of some vector is the code created by filtering out all of the unsupported codewords in C then removing the columns from the generator matrix of the new code that are guaranteed by s to be 0 for every codeword. To understand how the restriction of a code modifies a code, consider the following example.

Example 5.7. Let $s \in \mathbb{F}_2^6$ be defined to be $s = (1, 1, 1, 0, 1, 0)$. Define the linear code C by the generator matrix

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

This gives us the following codewords in C.

$$\begin{aligned} c_0 &= 0\ 0\ 0\ 0\ 0\ 0 \\ c_1 &= 1\ 1\ 0\ 0\ 0\ 0 \\ c_2 &= 1\ 1\ 1\ 1\ 0\ 1 \\ c_3 &= 1\ 1\ 0\ 1\ 1\ 1 \\ c_4 &= 1\ 1\ 1\ 0\ 1\ 0 \\ c_5 &= 0\ 0\ 1\ 1\ 0\ 1 \\ c_6 &= 0\ 0\ 1\ 0\ 1\ 0 \\ c_7 &= 0\ 0\ 0\ 1\ 1\ 1 \end{aligned}$$

By inspecting each codeword, we see that c_0, c_1, c_4, and c_6 are the only codewords supported by s. Removing the codewords that are not supported gives us the following codewords in $C' \subseteq C$.

$$\begin{aligned} c_0 &= 0\ 0\ 0\ 0\ 0\ 0 \\ c_1 &= 1\ 1\ 0\ 0\ 0\ 0 \\ c_4 &= 1\ 1\ 1\ 0\ 1\ 0 \\ c_6 &= 0\ 0\ 1\ 0\ 1\ 0 \end{aligned}$$

For the final step of the restriction of C to s, we remove the 4^{th} and 6^{th} indices to find the four codewords of $C|_s$.

$$\begin{aligned} c_0 &= 0\ 0\ 0\ 0 \\ c_1 &= 1\ 1\ 0\ 0 \\ c_2 &= 1\ 1\ 1\ 1 \\ c_3 &= 0\ 0\ 1\ 1 \end{aligned}$$

So a generator matrix of $C|_s$ is

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

Using these classical definitions, we can now define CSS-T codes, which are a subset of CSS codes satisfying an additional requirement. In [5], Rengaswamy et. al. introduced the class of CSS-T codes and prove that CSS-T codes admit the transversal application of the T gate to the physical qubits.

Definition 5.8. A **CSS-T code** is a CSS code defined by $C_2 \subseteq C_1$ such that C_2 is even and for every $x \in C_2$, $C_1^{\perp}|_x$ contains a self-dual code.

This definition provides a connection between classical linear codes and fault tolerant stabilizer codes, allowing us to leverage the theory of linear codes to discover and construct fault tolerant stabilizer codes. In the next chapter, we will define a new family of fault tolerant stabilizer codes using this definition.

In [6], Rengaswamy et. al. also define a more general condition for stabilizer codes that admit non-transversal applications of the T gate. However, the transversal application is most promising for fault tolerant computation and the easiest to work with, so we will focus on this particular case. Furthermore, Rengaswamy et. al. show that for any non-degenerate stabilizer code that admits a transversal T, there is a CSS-T code of the same length with a larger dimension and greater distance that also admits a transversal T. Since CSS-T codes have these properties, we focus on them specifically.

In [5], Rengaswamy presents a family of quantum Reed-Muller codes which forms a CSS-T family. His construction begins by setting m and r such that $\frac{m-1}{3} < r \leq \frac{m}{3}$. He defines his CSS code using $C_1 = RM(r, m)$ and $C_2 = RM(r-1, m)$. This family constructs a $[\![2^m, \binom{m}{r}, 2^r]\!]$ quantum code. For every code in this family the transversal application of the T gate results in the logical application of the T gate to each logical qubit. The family has asymptotically growing distance, but asymptotically vanishing rate.

Chapter 6

Self-Orthogonality and CSS-T codes

The CSS-T definition does not lend itself to finding suitable classical codes easily, so we sought out an alternative condition that was sufficient to satisfy the condition given in [5]. Our research honed in on what it meant to contain a self-dual code and whether there were alternative conditions that would guarantee that a CSS code was a CSS-T code. As it turns out, by considering self-orthogonal codes, we can define an alternative equivalent condition for a CSS code to be CSS-T.

6.1 Sufficient CSS-T Condition

We begin our discussion on the requirements to contain a self-dual code with a slightly weakened version of self-duality.

Definition 6.1. A classical linear code C is said to be **self-orthogonal** if C is contained in its dual. That is, C is self-orthogonal if $C \subseteq C^{\perp}$.

Consider the following example of a simple self-orthogonal code.

Example 6.2. Let C be the linear code defined by the generator matrix

$$G_C = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

The dual code, $C^{\perp}$, would have generator matrix

$$G_{C^{\perp}} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Note that the top two rows of $G_C^{\perp}$ make the generator matrix G_C, so clearly $C \subseteq C^{\perp}$. Thus, C is self-orthogonal.

Before we proceed to our larger results, we discuss one property of self-orthogonal codes.

Lemma 6.3. Let C be a self-orthogonal code. Then it is an even code.

Proof. Since $C \subseteq C^{\perp}$, then, for all $c \in C$, $c^T c \equiv 0 \pmod 2$. This implies that every $c \in C$ has even weight. □

Using the concept of self-orthogonality, we can prove a strong result regarding the containment of self-dual codes.

Theorem 6.4. Let n be an even number and $k \geq \frac{n}{2}$. An $[n, k]$ binary linear code C contains a self-dual code if and only if $C^{\perp}$ is self-orthogonal.

Proof. We begin by proving the forward direction. Let C be an $[n, k]$ linear code that contains a self-dual code $D = D^{\perp}$. Since $D \subseteq C$, we see $C^{\perp} \subseteq D^{\perp} = D$. Combining these two relationships, we see $C^{\perp} \subseteq D \subseteq C$. Thus, $C^{\perp}$ is self-orthogonal.

We now prove the backwards implication. Suppose that $C^{\perp}$ is self-orthogonal. We first show a specific case: $\dim(C) = \frac{n}{2}$. The dimension of the dual is $\dim(C^{\perp}) = n - \dim(C) = n - \frac{n}{2} = \frac{n}{2}$, and from our assumption, we know that $C^{\perp} \subseteq C$. Since the dimension of both C and $C^{\perp}$ are the same, we see $C = C^{\perp}$. Since C is self-dual, it contains a self-dual code, namely itself.
Now we consider the case where $\dim(C) = k > \frac{n}{2}$. Define $S = \{\overline{C} \mid C^{\perp} \subseteq \overline{C} \subseteq C, \dim(\overline{C}) = \frac{n}{2}\}$ to be the set of subcodes of C that have the dimension necessary to be self-dual codes. We now prove a couple properties of S. The first property we prove is that if $\overline{C} \in S$ we know $\overline{C}^{\perp} \in S$. Suppose that $\overline{C} \in S$. From Proposition 1.7 we know $C^{\perp} \subseteq \overline{C} \subseteq C$ implies that $C^{\perp} \subseteq \overline{C}^{\perp} \subseteq (C^{\perp})^{\perp} = C$ and $\dim(\overline{C}^{\perp}) = n - \dim(\overline{C}) = n - \frac{n}{2} = \frac{n}{2}$, so $\overline{C}^{\perp} \in S$. So, if $\overline{C} \in S$ we know $\overline{C}^{\perp} \in S$. The

second property is that the cardinality of S is odd. To calculate the cardinality of S, we will use the Gaussian Binomial Coefficient. The Gaussian Binomial Coefficient counts how many subspaces of a given dimension are contained in a space of some known, larger dimension. However, this method does not allow us to directly account for the only the subspaces that happen to contain a common subspace. To count only those codes which also contain $C^\perp$, we consider the quotient spaces $C^\perp/C^\perp = \{0\} \subseteq \overline{C}/C^\perp \subseteq C/C^\perp$. Note that for every $\overline{C} \in S$, there is exactly one $\overline{C}/C^\perp \subseteq C/C^\perp$, so by counting the number of subspaces of the form $\overline{C}/C^\perp$ are contained in $C/C^\perp$, we find the cardinality of S. The dimension of $\overline{C}/C^\perp$ is $\dim(\overline{C}) - \dim(C^\perp) = \frac{n}{2} - (n-k) = k - \frac{n}{2}$, and the dimension of $C/C^\perp$ is $\dim(C) - \dim(C^\perp) = k - (n-k) = 2k - n$. Note that since $k > \frac{n}{2}$, this is a positive number. Since we know the dimension of each of these spaces, we can now use the formula for the Gaussian Binomial Coefficient to find the cardinality of S.

$$|S| = \begin{bmatrix} 2k-n \\ k-\frac{n}{2} \end{bmatrix}_2 = \prod_{i=0}^{k-\frac{n}{2}-1} \frac{2^{2k-n-i}-1}{2^{i+1}-1}.$$

Now note that $2^{2k-n-i} - 1$ and $2^{i+1} - 1$ are both odd for all $i \in \mathbb{N}$. Therefore, $|S|$ is odd. If there were no self-dual codes in S, then each code must pair off with exactly one other code, its dual. Since the dual-code equivalence relation partitions the set, a lack of self-dual codes in S would force S to be even. However, we showed above that S is odd, so we know that there must be at least one self-dual code in S. Therefore, there exists a self-dual subcode in C.

Having proven implication in both directions, we conclude that for an even n and $k \geq \frac{n}{2}$, a $[n, k]$ binary linear code C contains a self-dual code if and only if $C^\perp$ is a self-orthogonal code. $\square$

This theorem provides an equivalence between the dual of a code being self-orthogonal and that code containing a self-dual subcode. In the context of CSS-T codes, this means that for each $s \in C_2$, instead of attempting to construct a self-dual subcode in $C_1^\perp|_s$, we only need to check if $C_1^\perp|_s^\perp$ is self-orthogonal, which usually easier to prove.

6.2 CSS-T Codes with Nonvanishing Rate

In [5], Rengaswamy et. al. present a family of CSS-T codes defined using classical Reed-Muller codes. Their family has good distance properties, but a vanishing rate. In this section, we explore alternative constructions using Reed-Muller codes for both C_1 and C_2. Our objective is to

define a family of CSS-T codes that has a nonvanishing rate using Reed-Muller codes. We begin our search with a proposition that restricts possible choices for C_1 significantly.

Proposition 6.5. If $C_1 = \mathrm{RM}(r_1, m)$ and $C_2 = \mathrm{RM}(r_2, m)$ with $r_2 \leq r_1$ define a CSS-T code, then $C_1 \subseteq C_1^\perp$.

Proof. Since C_1 is a Reed-Muller code, $C_1^\perp = \mathrm{RM}(m - r_1 - 1, m)$ is also a Reed-Muller code. Either $r_1 \leq m - r_1 - 1$ or $m - r_1 - 1 < r_1$. Assume for the sake of contradiction that $m - r_1 - 1 < r_1$, then we have the strict containment $C_1^\perp \subset C_1$ by Proposition 1.15. Let $x = (1, 1, ..., 1)$ be a vector of length 2^m with a one in every position. By Proposition 1.15, we also get $\mathrm{RM}(0, m) \subseteq \mathrm{RM}(r_2, m) = C_2$, so $x = (1, 1, ..., 1) \in \mathrm{RM}(0, m) \subseteq C_2$ because $\mathrm{RM}(0, m)$ is the repetition code of length 2^m. Since x supports every codeword in $C_1^\perp$, we see $C_1^\perp|_x = C_1^\perp = \mathrm{RM}(m - r_1 - 1, m)$ and and we see $C_1^\perp|_x^\perp = C_1 = \mathrm{RM}(r_1, m)$. If $C_1^\perp|_x$ contained a self-dual subcode, then by Theorem 6.4, $C_1^\perp|_x^\perp \subseteq C_1^\perp|_x$, which would mean $r_1 \leq m - r_1 - 1$. However, we assumed that $m - r_1 - 1 < r_1$, so this is a contradiction. Therefore, $r_1 \leq m - r_1 - 1$, which by Proposition 1.15 implies that $C_1 = \mathrm{RM}(r_1, m) \subseteq \mathrm{RM}(m - r_1 - 1, m) = C_1^\perp$. □

This result reduces the number possible CSS-T families significantly. It also places an upper bound on the rate of any CSS-T code defined using Reed-Muller codes by limiting the dimension of C_1. The dimension of the CSS-T code is given by the dimension of C_1 minus the dimension of C_2. So, in our search for a family with a nonvanishing rate, we want to maximize the dimension of C_1 as much as possible. Of the remaining possible codes, we see that $C_1 = C_1^\perp$ gives the largest dimension possible for C_1. As it turns out, there is a family of CSS-T codes constructed of Reed-Muller codes that has a nonvanishing rate.

Theorem 6.6. Consider a family CSS-T codes constructed using Reed-Muller codes for both C_1 and C_2. If $C_1 = C_1^\perp$ for every CSS-T code in this family, then the codes in this family are defined by $C_1 = \mathrm{RM}(r, 2r + 1)$ and $C_2 = \mathrm{RM}(0, 2r + 1)$ for any nonnegative integer r.

Proof. Let $C_2 \subseteq C_1$ be Reed-Muller codes defining a CSS-T code in this family. Since C_1 is self-dual, we have $C_1 = \mathrm{RM}(r, 2r + 1)$ for some nonnegative integer r. Since $C_2 \subseteq C_1$ is a Reed-Muller code, we must have $C_2 = \mathrm{RM}(r_2, 2r + 1)$ for some $r_2 \leq r$. We now consider a couple cases on r. If $r = 0$, then $C_1 = C_2 = C_1^\perp = C_2^\perp$ are all the repetition code of length 2. So C_2 is an even code. Also, $C_1^\perp|_x = C_1^\perp = C_1$ if $x = (1, 1)$, the only nonzero codeword in C_2. So for every $x \in C_2$, we see that $C_1^\perp|_x$ contains a self-dual subcode. So our proposition holds in the case $r = 0$.

We now prove the proposition for $r > 0$. Let r be some positive integer. We now consider a couple cases on r_2. Suppose that $C_2 = \mathrm{RM}(r_2, 2r+1)$ with $r_2 \geq 1$. Let $x = (0, 0, ..., 0, 1, 1, ..., 1)$ be the codeword consisting of 2^{2r} zeroes followed by 2^{2r} ones. We claim that x is contained in C_2. To see this, consider the generator matrix of $\mathrm{RM}(1, 2r+1)$:

$$G_{\mathrm{RM}(1,2r+1)} = \begin{pmatrix} G_{\mathrm{RM}(1,2r)} & G_{\mathrm{RM}(1,2r)} \\ \mathbf{0} & G_{\mathrm{RM}(0,2r)} \end{pmatrix} = \begin{pmatrix} G_{\mathrm{RM}(1,2r)} & G_{\mathrm{RM}(1,2r)} \\ \mathbf{0} & \mathbf{1} \end{pmatrix}$$

where $\mathbf{1}$ is the codeword of length 2^{2r} with 1's in every position. Note that x is the bottom row of this generator matrix. So we see that $x \in \mathrm{RM}(1, 2r+1)$.From Proposition 1.15, since $r_2 \geq 1$, we know that $\mathrm{RM}(1, 2r+1) \subseteq \mathrm{RM}(r_2, 2r+1) = C_2$, so we have that $x \in C_2$. Now consider $C_1^{\perp}|_x$. The generator matrix of $C_1^{\perp} = \mathrm{RM}(r, 2r+1)$ is given by

$$G_{\mathrm{RM}(r,2r+1)} = \begin{pmatrix} G_{\mathrm{RM}(r,2r)} & G_{\mathrm{RM}(r,2r)} \\ \mathbf{0} & G_{\mathrm{RM}(r-1,2r)} \end{pmatrix}.$$

Note that the codewords of $C_1^{\perp}$ supported on x are exactly the codewords that are generated by only the lower row of the generator matrix. After puncturing $C_1^{\perp}$ at the indices where x has zeros, we see $C_1^{\perp}|_x = \mathrm{RM}(r-1, 2r)$. Note that this means $C_1^{\perp}|_x^{\perp} = \mathrm{RM}(2r-(r-1)-1, 2r) = \mathrm{RM}(r, 2r)$. Recall that Proposition 1.15 says that a Reed-Muller $\mathrm{RM}(r, 2r)$ contains every Reed-Muller code $\mathrm{RM}(r', 2r)$ with $r' \leq r$. So we have the strict containment: $C_1^{\perp}|_x \subset C_1^{\perp}|_x^{\perp}$. Notice that this implies that $C_1^{\perp}|_x^{\perp}$ is not self-orthogonal. Hence, by Theorem 6.4, we see that $C_1^{\perp}|_x$ cannot contain a self-dual subcode. Thus, if $r_2 \geq 1$, C_1 and C_2 cannot form a CSS-T code because $C_1^{\perp}|_x$ does not contain a self-dual subcode for every $x \in C_2$. Therefore, r_2 cannot be greater than or equal to 1.

We now show that $r_2 = 0$ allows us to define a CSS-T code. If $r_2 = 0$, then $C_2 = \mathrm{RM}(0, 2r+1)$, the repetition code of length 2^{2r+1}. Since the length of the code is even, C_2 is an even code. Let $x \in C_2$ be the only nonzero codeword. Since x is the codeword consisting of only ones, every vector in $\mathbb{F}_2^{2^{2r+1}}$ is supported in x, so we see $C_1^{\perp}|_x = C_1^{\perp}$. Since these codes are equal, their duals are equal: $C_1 = C_1^{\perp}|_x^{\perp}$. Recall that C_1 is self-dual, so we have $C_1^{\perp}|_x = C_1^{\perp} = C_1 = C_1^{\perp}|_x^{\perp}$. Hence, $C_1^{\perp}|_x$ contains a self-dual subcode: itself. Thus, $r_2 = 0$ does define a CSS-T code.

Therefore, the only value of r_2 that allows $C_1 = \mathrm{RM}(r, 2r+1), C_2 = \mathrm{RM}(r_2, 2r+1)$ to define a CSS-T code is $r_2 = 0$. □

We now consider the asymptotic rate and distance of this family.

Proposition 6.7. Consider the family of CSS-T codes given by $C_1 = \mathrm{RM}(r, 2r+1)$ and $C_2 = \mathrm{RM}(0, 2r+1)$ for any nonnegative integer r. The codes in this family are $[\![2^{2r+1}, 2^{2r}-1, 2]\!]$ CSS-T quantum codes, and the family has asymptotic rate $\frac{1}{2}$.

Proof. The distance of a CSS code is the minimum of the distance of C_1 and $C_2^\perp$. From Proposition 1.14, the distance of C_1 is $d_1 = 2^{2r+1-r} = 2^{r+1}$. We have $C_2^\perp = \mathrm{RM}(2r+1-0-1, 2r+1) = \mathrm{RM}(2r+1-1, 2r+1)$, so by the same proposition, the distance of $C_2^\perp$ is $d_2^\perp = 2^{2r+1-(2r+1-1)} = 2^1 = 2$. For any nonnegative r, the minimum of these two is 2, so we conclude that the distance of a CSS code in this family is 2.

The dimension of a CSS code is given by the difference of the dimension of C_1 and the dimension of C_2. Since C_1 is self-dual, its dimension is half of its length. So the dimension of C_1 is $k_1 = \frac{1}{2}2^{2r+1} = 2^{2r}$. Notice that C_2 is the repetition code, which always has dimension 1. Thus, the dimension of a code in this family for a given r is $k = 2^{2r} - 1$.

The rate of this family is given by $\frac{k}{n} = \frac{2^{2r}-1}{2^{2r+1}} = \frac{2^{2r}}{2^{2r+1}} - \frac{1}{2^{2r+1}} = \frac{1}{2} - \frac{1}{2^{2r+1}}$. As r tends toward infinity, we note that the rate of this code tends towards $\frac{1}{2}$. Thus, the asymptotic rate of this family is $\frac{1}{2}$. □

The distance of the codes in this family is lackluster, but we see that the family does have a nonvanishing rate. Furthermore, since C_1 has the largest possible dimension for CSS-T codes and C_2 has the smallest possible dimension, we note that this family has the highest rate of any CSS-T code defined using Reed-Muller codes.

Chapter 7

Conclusion

Quantum computing promises to allow us to solve problems that are unapproachable on classical computers. However, the error rates of quantum computers is a major limitation of their usefulness. Because errors are often introduced when we try to manipulate qubits, fault tolerant quantum codes are essential for practical quantum computers. In [6], Rengaswamy et. al. provide a general condition for fault tolerant stabilizer codes and a condition specific to CSS codes. They use their results to define a family of quantum Reed-Muller fault tolerant codes. Their family of codes admits a transversal T gate, but the asymptotic rate of the family is zero. Our results build on their specific condition for CSS codes to define an equivalent set of conditions for CSS-T using the concept of self-orthogonality. We then show that if we choose C_1 and C_2 to be Reed-Muller codes that define a CSS-T code, then C_1 must be self-orthogonal. This restriction prompts us to investigate the case where C_1 is the largest possible code that satisfies $C_1 \subseteq C_1^{\perp}$. The investigation results in an alternative family of quantum Reed-Muller CSS-T codes where $C_1 = \mathrm{RM}(r, 2r+1) = C_1^{\perp}$ and $C_2 = \mathrm{RM}(0, 2r+1)$. Like the quantum Reed-Muller family from [6], our family also admits a transversal T gate, but has a nonvanishing asymptotic rate. In the course of constructing this family, we also prove that this is the only possible family of CSS-T codes where C_1 and C_2 are Reed-Muller codes and C_1 is self-dual.

Regarding potential next steps, we see two avenues for future research. First, we could seek alternative CSS-T families with better distance properties. It may be possible to define a family somewhere between our family and the family given in [6], and this deserves further investigation.

Another direction for research may be to partially leave the Reed-Muller family. We found that requiring C_1 and C_2 to both be Reed-Muller codes was very restrictive. We expect that considering C_2 to be not Reed-Muller will open up many possibilities for interesting families that have nonvanishing rates and good distances.

Bibliography

[1] D. Bacon. Cse599d lecture 18 notes. (accessed: 11.08.2020).

[2] A. R. Calderbank and Peter W. Shor. Good quantum error-correcting codes exist. *Phys. Rev. A*, 54:1098–1105, Aug 1996.

[3] W. Huffman and V. Pless. *Fundamentals of Error Correcting Codes*. Cambridge University Press, 2003.

[4] M. Nielsen and I. Chuang. *Quantum Computing and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2011.

[5] N. Rengaswamy, R. Calderbank, M. Newman, and H. D. Pfister. Classical coding problem from transversal t gates. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 1891–1896, 2020.

[6] N. Rengaswamy, R. Calderbank, M. Newman, and H. D. Pfister. On optimality of css codes for transversal t. *IEEE Journal on Selected Areas in Information Theory*, 1(2):499–514, 2020.

[7] Bianca Schroeder, Eduardo Pinheiro, and Wolf-Dietrich Weber. Dram errors in the wild: A large-scale field study. In *SIGMETRICS*, 2009.

[8] Peter W. Shor. Scheme for reducing decoherence in quantum computer memory. *Phys. Rev. A*, 52:R2493–R2496, Oct 1995.

[9] A. M. Steane. Error correcting codes in quantum theory. *Phys. Rev. Lett.*, 77:793–797, Jul 1996.

[10] A. M. Steane. Multiple-particle interference and quantum error correction. *Proc. R. Soc. Lond. A.*, 452:2551–2577, Nov 1996.

[11] Swamit S. Tannu and Moinuddin K. Qureshi. Not all qubits are created equal: A case for variability-aware policies for nisq-era quantum computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '19, page 987–999, New York, NY, USA, 2019. Association for Computing Machinery.