

# Optimal Quaternary Hermitian Linear Complementary Dual Codes for Entanglement-Assisted Quantum Error Correction

*Maysara Al Jumaily*

Submitted in partial fulfilment  
of the requirements for the degree of

Master of Science

Department of Computer Science  
Brock University  
St. Catharines, Ontario

---

# Abstract

The objective of this thesis is to find suboptimal and optimal parameters from classical codes and import them into entanglement-assisted quantum codes. The thesis begins by introducing classical error correction, followed by a detailed introduction to quantum computing. Topics that are discussed in the introduction include qubits, quantum phenomena, such as superposition and entanglement, and quantum gates/circuits. The thesis then reviews the basics of quantum error correction and provides Shor's code to reinforce the reader's understanding. Subsequently, the formalism of stabilizer codes is thoroughly examined. We then explain the generalized concept of stabilizer codes which is entanglement-assisted quantum codes. They do not require generators to satisfy the commutativity property. Rather, they utilize the usage of ebits to resolve the anti-commutativity constraint.

Next, the thesis explains quaternary field and then the Java program implemented to find the optimal parameters. Lastly, the thesis concludes with presenting the parameters of the new codes that were obtained throughout the research. We have found the suboptimal largest distance for quaternary hermitian linear complementary dual codes that can be imported as entanglement-assisted quantum error correction for parameters  $[22, 9, 9 \text{ or } 10]_4$ ,  $[22, 12, 7 \text{ or } 8]_4$ ,  $[23, 8, 11 \text{ or } 12]_4$ ,  $[23, 10, 9 \text{ or } 10]_4$ ,  $[23, 13, 7 \text{ or } 8]_4$ ,  $[24, 10, 10 \text{ or } 11]_4$ ,  $[24, 11, 9 \text{ or } 10]_4$ ,  $[24, 14, 7 \text{ or } 8]_4$ ,  $[25, 12, 9 \text{ or } 10]_4$ ,  $[25, 13, 8 \text{ or } 9]_4$ , as well as the optimal largest distance for  $[17, 11, 5]_4$  and  $[17, 13, 3]_4$ .

---

# Acknowledgements

This work would not have been possible without the guidance of my supervisor, Dr. Sheridan Houghten, from Brock University. She has been supportive since the start of my graduate studies. Dr. Houghten gave me all the assistance that I ever wished for and the space whenever I needed. She is the person who introduced me to the field of Coding Theory and I will forever be grateful. I am thankful for having Dr. Ke Qiu from Brock University, Dr. Michael Winter from Brock University and Dr. Daniel Ashlock from University of Guelph for their time and feedback regarding the progress of my thesis.

I would like to also thank Dr. Mark Wilde from Louisiana State University who helped me when I reached out for him, and Dr. Ching-Yi Lai from National Chiao Tung University invested significant amount of time answering my questions regarding Entanglement-Assisted Quantum Error Correction. He was determined that I understand the material very well and I wouldn't acquire the knowledge I currently have without his valuable feedback. Furthermore, Dr. Alastair Kay from Royal Holloway University of London was generous enough to update the Quantikz package upon my request.

An important person that is close to my heart is Dr. Babak Farzad who we have unfortunately lost in 2018. I had my first ever undergraduate lecture with him, and he taught me the fundamentals of computer science. Dr. Farzad, I am eternally in your debt, and you will always be a part of my academic success.

A special thanks goes to Brock University for kindly granting me scholarships and bursaries throughout my studies.

Lastly but certainly not least, I would like to thank my family members for all the support throughout the years. I would like to thank my beautiful mother who ensured to prepare *every* single meal I took with me to the university. I thank my father for giving me the mental motivation and making sure I have anything I ask for. I appreciate the time and effort my youngest brother put into proofreading my thesis. I also admire the support and motivation provided by my eldest brother and sister throughout these years. All my family members were there for me when I was at my lowest and without them, I wouldn't have succeeded. I thank God for blessing me with all these people who helped me throughout my journey.

---

# Contents

List of Tables

List of Figures

List of Symbols

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Classical Error Correction</b>	<b>5</b>
2.1	Encoding and Decoding of Linear Codes . . . . .	8
2.1.1	Encoding Process . . . . .	9
2.1.2	Decoding Process . . . . .	10
2.1.3	Self-(Dual/Orthogonal) and Equivalent Codes . . . . .	12
2.2	Weight enumerator . . . . .	14
<b>3</b>	<b>Introduction to Quantum Computing</b>	<b>16</b>
3.1	Hilbert Space . . . . .	16
3.2	Mathematical Preliminaries . . . . .	18
3.3	An Overview of a Qubit . . . . .	20
3.4	Global Phase versus Relative Phase . . . . .	25
3.5	Quantum Phenomena . . . . .	27
3.6	Quantum Gates . . . . .	28
3.7	Introduction to Quantum Circuits . . . . .	33
<b>4</b>	<b>Introduction to Quantum Error Correction</b>	<b>41</b>
4.1	Digitalizing Environment Noise on a Qubit . . . . .	41
4.2	Three-Qubit Quantum Repetition Code . . . . .	43
4.2.1	Bit-flip Correction Code . . . . .	43
4.2.2	Phase-flip Correction Code . . . . .	47

---

4.2.3	Shor's Code . . . . .	51
<b>5</b>	<b>Introduction to Stabilizer Codes</b>	<b>55</b>
5.1	Repetition Bit-flip Code Revisited . . . . .	55
5.2	Stabilizer Codes . . . . .	57
5.2.1	Syndrome Extraction Procedure . . . . .	61
5.3	Criteria for Quantum Error Correction . . . . .	61
5.3.1	Examples of Stabilizer Codes . . . . .	62
<b>6</b>	<b>Entanglement-Assisted QECC</b>	<b>64</b>
6.1	The Encoding/Decoding Algorithm . . . . .	70
<b>7</b>	<b>Introduction to Quaternary Fields and Codes</b>	<b>74</b>
<b>8</b>	<b>Programming Implementation of Thesis</b>	<b>78</b>
8.1	Construction Method for Finding Optimal Hermitian Linear Complementary Codes . . . . .	78
8.1.1	Comparing our Modified Method to the Original Procedure . . . . .	79
8.2	The Programming Implementation of the Thesis . . . . .	80
8.2.1	Vector Generator . . . . .	80
8.2.2	Storing Linear Combinations . . . . .	80
8.2.3	Determining Whether if a Matrix is Invertible . . . . .	81
8.2.4	The Weight of a Quaternary Vector . . . . .	81
8.2.5	The Core Engine of the Program . . . . .	82
8.2.6	Verifying Results with MATLAB . . . . .	84
<b>9</b>	<b>Previous work and new Results</b>	<b>86</b>
9.1	New Suboptimal and Optimal Parameters . . . . .	88
<b>10</b>	<b>Thesis Conclusion and Future Work</b>	<b>96</b>
10.1	Summary of Contribution . . . . .	96
10.2	Future Work . . . . .	97
	<b>Bibliography</b>	<b>97</b>
	<b>Appendices</b>	<b>103</b>
A.1	Repetition Code-bit Flip Circuit Using CZ Gates . . . . .	103

# List of Tables

2.1	All of the $2^k = 2^4 = 16$ codewords of the $[7, 4, 3]$ Hamming code with their weights . . . . .	10
2.2	All possible coset leaders and their syndromes for the $[7, 4, 3]$ Hamming code . . . . .	12
3.1	All single-qubit gates associated with their circuit representation . . .	31
3.2	The truth table for <b>CNOT</b> gate . . . . .	32
3.3	The truth table for <b>CZ</b> gate . . . . .	32
3.4	The truth table for <b>SWAP</b> gate . . . . .	33
3.5	The truth table for anti- <b>CNOT</b> . . . . .	36
3.6	An equivalent representation of the truth table for <b>CZ</b> gate . . . . .	38
4.1	Showing possible syndromes and with the error caused on the $ \psi\rangle$ and the operation needed to correct it . . . . .	46
4.2	All possible syndromes for Shor's Code with their effect on the encoded qubit $ \psi\rangle = \alpha \overline{0}\rangle + \beta \overline{1}\rangle$ and the operation needed to correct it . . . . .	53
5.1	Showing possible syndromes for quantum repetition code as eigenvalues, with the error caused on the $ \psi\rangle$ and the operation needed to correct it . . . . .	57
5.2	The commutation relation of Pauli elements with themselves . . . . .	58
6.1	The symplectic product of two binary vectors . . . . .	68
7.1	The addition table (left), multiplication (middle) and conjugation (right) of elements in $\mathbb{F}_4$ . . . . .	74
7.2	All of the $4^k = 4^2 = 16$ codewords of the $[6, 2, 4]_4$ code . . . . .	75
9.2	Suboptimal and optimal parameters we have found. The notation $\frac{x}{y}$ is the lower bound ( $x$ ) and upper bound $y$ (both inclusive) of the largest distance of the code . . . . .	89

9.3	The updated table for optimal values for $n \leq 25$ all along with the results found in this thesis in bold. The values with an asterisk denotes the accurate largest distances . . . . .	95
-----	--	----

# List of Figures

2.1	A diagram illustrating the journey a message takes when being transmitted through an erroneous medium . . . . .	6
2.2	The geometric representation of sphere packing where two codewords (blue points) $c_i$ and $c_j$ exactly $d = 2t + 1$ distance apart. The $t$ represents the Hamming distance between the codeword at the center of the sphere and other vectors (green and red points). The vectors in red are not correctable whereas the green vectors are . . . . .	8
2.3	We can see that having a distance less than $2t + 1$ yields to ambiguities. In the left portion of the figure, we have a vector that lies exactly at the intersection of two spheres. Moreover, the right side shows two vectors that can be mapped to two different codewords. In both cases, there are vectors that cannot be uniquely decoded . . . . .	8
3.1	The visualization of a bit in a 2D plane . . . . .	22
3.2	The Bloch sphere representation with the equatorial plane passing through the origin. The red vector is the qubit state $ \psi\rangle$ . . . . .	22
3.3	The two main states, $ 0\rangle$ on the left side and $ 1\rangle$ on the right side . . .	23
3.4	The four common states other than $ 0\rangle$ and $ 1\rangle$ . . . . .	24
3.5	CNOT gate circuit representation . . . . .	31
3.6	CZ gate circuit representation . . . . .	32
3.7	SWAP gate circuit representation . . . . .	33
3.8	A quantum circuit that flips the first and last qubits. It starts with $ \psi\rangle =  0\rangle \otimes  0\rangle \otimes  0\rangle =  000\rangle$ and outputs $ 101\rangle$ after applying the gates . . . . .	34
3.9	The left circuit contains the I gate for completeness whereas the right circuit doesn't because it is assumed to be implied . . . . .	34
3.10	Creating a customized gate which has three inputs and three. The matrix of $U$ is $U = X \otimes X \otimes X$ . . . . .	34
3.11	Using the customized gate $U$ created in Figure 3.10 . . . . .	35



3.12	The order of applying gates is done following a left-to-right fashion . . . . .	35
3.13	The order of applying gates on each qubit is done in a left-to-right fashion . . . . .	35
3.14	$Q_1$ (the anticontrol qubit) will perform an <b>XNOR</b> operation on $Q_2$ (the target qubit) with $Q_1$ if $Q_1$ is $ 0\rangle$ or a superposition of $ 0\rangle$ . . . . .	36
3.15	A qubit in superposition acting as the control qubit in a <b>CNOT</b> gate . . . . .	37
3.16	A qubit in superposition acting as the control qubit in a <b>CZ</b> gate . . . . .	38
3.17	Equivalent <b>CZ</b> circuits with two control qubits and one target qubit . . . . .	39
3.18	The <b>SWAP</b> gate represented as three <b>CNOT</b> gates . . . . .	40
4.1	Repetition code single bit-flip correction circuit using <b>CNOT</b> gates to find the syndromes . . . . .	44
4.2	The phase-flip circuit correction for the three-qubit repetition code using <b>CNOT</b> gates for syndrome correction . . . . .	49
4.3	The phase-flip circuit correction for the three-qubit repetition code using <b>CZ</b> gates for syndrome correction . . . . .	50
4.4	Nine-qubit Shor Code circuit which corrects at most one of $\mathcal{E} \in \{X, Y, Z\}$ . . . . .	54
A.1.1	Repetition code single bit-flip correction circuit using <b>CZ</b> gates to find the syndromes . . . . .	103

# List of Symbols

## Coding Theory

$\mathbf{C}$	A classical linear code.
$\mathbf{C}^\perp$	The dual code of some classical linear code $\mathbf{C}$ .
$[n, k, d]$	A classical linear code with $n$ as length, $k$ as dimension and $d$ as distance.
$[n, k, d]_q$	A quaternary classical linear code with $n$ as length, $k$ as dimension and $d$ as distance.
$\llbracket n, k, d \rrbracket$	A quantum code with $n$ as length, $k$ as dimension and $d$ as distance with a field of $q$ elements.
$\llbracket n, k, d; c \rrbracket$	An entanglement-assisted quantum code with $n$ as length, $k$ as dimension, $d$ as distance and $c$ ebits.

## Functions

$\text{DIST}(v_1, v_2)$	The Hamming distance between two vectors $v_1$ and $v_2$ of equal length.
$\text{WT}(v)$	The weight of the vector $v$ .
$\text{DIM}(\mathbf{C})$	The dimension of the code $\mathbf{C}$ .
$\text{SYN}(v)$	The syndrome of the vector $v$ .
$\text{RANK } G$	The rank of the matrix $G$ .
$\text{DET}(G)$	The determinant of the matrix $G$ .
$\text{W}(x)_{\mathbf{C}}$	The weight enumerator of the code $\mathbf{C}$ .

## Mathematics

$i$	The complex number $\sqrt{-1}$ .
$x^\dagger$	Complex conjugate of some element.
$\alpha$	Probability of $ 0\rangle$ evaluating to 0.
$\beta$	Probability of $ 1\rangle$ evaluating to 1.
$\mathbb{Z}$	Whole integers: $\dots, -3, -2, -1, 0, 1, 2, 3, \dots$
$\mathbb{Z}^+$	Positive whole integers: $1, 2, 3, \dots$
$\mathbb{R}$	Real numbers.
$\mathbb{C}$	Complex numbers of the form of $a + bi$ where $a, b \in \mathbb{R}$ and $i = \sqrt{-1}$ .
$\mathbb{F}$	Finite (Galois) field.
$\mathbb{F}_q^n$	Finite (Galois) field with vectors of length $n$ constructed from some alphabets of size $q$ .
$\mathbb{F}_4$	Quaternary field with the element $\{0, 1, \omega, \bar{\omega}\}$ .
$\otimes$	Tensor (or Kronecker) product applied on two matrices/column vectors.
$M^{\otimes n}$	$n$ -fold tensor product applied on $M$ , <i>i.e.</i> , $\underbrace{M \otimes M \otimes \dots \otimes M}_{n \text{ times}}$ .
$\oplus$	Logical xor that is applied on binary vectors.
$\odot$	Symplectic product that is applied on binary vectors.
$[A]$	The set of equivalent classes of some element $A$ .
$\mathcal{P}^n$	A closed Group of $n$ tensor product of Pauli elements.
$\star$	The multiplication operation used to form a commutative for any $A, B \in [\mathcal{P}^n]$ .
$[A, B]$	Commuting Pauli operators $A, B \in \mathcal{P}^n$ such that $AB - BA = 0$ .
$\{A, B\}$	Anticommuting Pauli operators $A, B \in \mathcal{P}^n$ such that $AB = -BA$ .



The two-operator pair that stabilizes an ebit. The left side is applied on the sender's part and the right is applied on the receiver's side.

### Physics/Linear Algebra

$|\Phi^+\rangle$  An ebit that is shared between the sender Alice and the receiver Bob in the form of:  $|\Phi^+\rangle = \frac{|00\rangle^{AB} + |11\rangle^{AB}}{\sqrt{2}}$


$\langle \cdot |$  Called "bra", a compact form of a row vector in dual Hilbert space.


$\mathcal{H}$  Hilbert space.

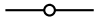
$|\cdot\rangle$  Called "ket", a compact form of a column vector in Hilbert space.


$\langle a|b\rangle$  Inner product in Hilbert space which equates to  $(\sum_{k=1}^n a_k^* b_k) \in \mathbb{C}$ .

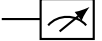
### Quantum Circuits


Classical wire An ordinary classical wire: 



Quantum wire A quantum wire allowing quantum phenomena to occur: 


Anti-control Applies on qubit in  $|0\rangle$  or a superposition of  $|0\rangle$ : 


Control Applies on qubit in  $|1\rangle$  or a superposition of  $|1\rangle$ : 


Measurement Circuit diagram for measuring a qubit: 


I Pauli- $I$  with the following matrix and gate:  $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ , 

X Pauli- $X$  with the following matrix and gate:  $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ ,  or 

Y Pauli- $Y$  with the following matrix and gate:  $\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$ , 

Z Pauli- $Z$  with the following matrix and gate:  $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ , 

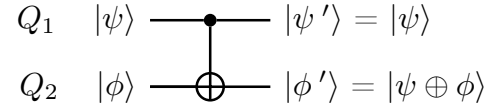
H Hadamard and its matrix and gate:  $\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ , 

P Phase with its matrix and gate:  $\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$ , 

U Some unitary operator gate.

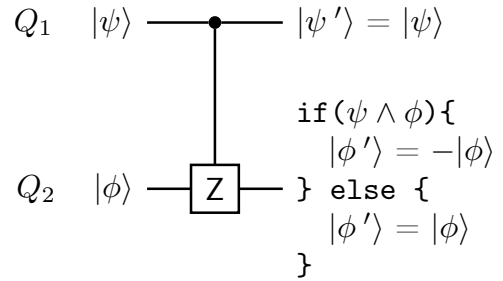
CNOT Controlled-not with the following matrix and gate:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$



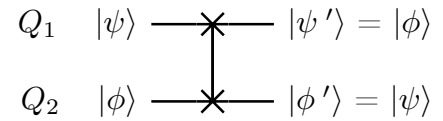
CZ Controlled-Z with the following matrix and gate:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$



SWAP Swap gate with the following matrix and gate:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



# Chapter 1

## Introduction

In 1948, Claude Shannon published the pioneer paper “A Mathematical Theory of Communication” of information theory and coding theory. It showed that data can be encoded before transmission over a noisy channel as well as then decoded and corrected to some degree of accuracy [29, pg. 1, 52]. Later, in 1950, Richard Hamming published “Error Correcting and Error Correcting Codes” which established the first error correcting code [26]. It is now known as the Hamming Code and can correct a single error.

In 1965, Gordon Moore, the co-founder of Intel Corporation, published a paper describing a pattern regarding the number of components (consisting of transistors, resistors, capacitors, and inductors) inside of a chip with respect to time [44]. The paper believed that as techniques and technology evolves, the number of components inside the same chip size doubles roughly every two years and that the pattern would continue. Later, that trend became known as Moore’s Law. After some years, the era of miniaturization permitted the production of personal computers [41] as well as the development of the Internet.

The Internet has revolutionized the way the world communicate. Essentially, the Internet is a collection of links that allows packets to tunnel through arriving from a sender to a receiver. These packets contain the data the receiver requested from the sender. During the transmission phase, noise is generated by an electrical impulse which can affect the data being transmitted. The data, which is a string of 0’s and 1’s, can be arbitrarily inverted from  $0 \rightarrow 1$  or  $1 \rightarrow 0$  due to noise in the transmission channel. To ensure the transferred and received data are the same, a mechanism needs to be established to preserve the consistency of data transmission. Error-Correcting Codes, from Coding Theory, help to ensure the transmission of data arrive to the receiver without alteration. This is done by adding redundancy bits to the data at the sender’s

side during the encoding phase. At the receiver's end, a decoder is used to decode the data and check for consistency. In the case where data is corrupted, the original data can be retrieved with the assistance of the redundancy bits added previously. For many decades, Moore's law continued through its predicted pattern until recently. Today's best microprocessors are occupied densely with billions of transistors with 7 nanometer gap between transistors [61]. Dealing with such nanoscopic scale allows for strange quantum effects to occur such as quantum tunnelling which marks the end of Moore's law. Quantum tunnelling is the behaviour where a particle can randomly penetrate through a physical barrier and land on the other side [43, pg xix]. Put in other words, the electrons passing through transistors can jump from one place to another which can yield to inaccurate computation and even corrupt the data. Quantum Mechanics is the framework for describing the behaviour of matter and light in meticulous detail on an atomic level [49, pg 1]. The world at an atomic level behaves unexpectedly, hence, we will be discussing the quantum phenomena of superposition and entanglement (which will be important concepts throughout the thesis) as well as decoherence which is the essence of quantum error correction. In quantum mechanics, particles are described by a *wave function* [54, chap 2]. A wave function is a mathematical description of the quantum state within its quantum system [21]. A quantum system is any collection of physical objects that is represented by a wave function [16, pg 173]. Quantum computing is idea of developing a computer which uses these quantum phenomena. In this thesis, a quantum system will be the collection of a qubit(s) whereas the quantum state will be represented as a vector(s). A qubit in a quantum computer is the equivalent of a bit in a classical one and will be further discussed in Chapter 3. In quantum computing, a quantum register of size  $n$  is a collection of  $n$  qubits [18]. Decoherence is the loss of information from a system in superposition into the environment [54, pg 63]. Superposition is a non-existence classical phenomenon where a qubit can be a classical 0 or classical 1 or in a combined 0/1 state (in superposition). As long as the qubit is not being observed or measured, the state will continue in superposition. Once the qubit is measured, the qubit will have to collapse (with some probability) to the classical state 0 or classical state 1. Hence, performing a measurement on the qubit will cause it to lose information and destroy the superposition. Quantum Entanglement is a term used by Schrödinger which expresses that a quantum state of one particle depends on some details of the quantum state of the other particle by interacting with each other [16, pg 173]. In other words, the wave function of two entangled qubits cannot be described independently of each other. Both qubits are required in order to describe the system. Furthermore,

when two particles (or qubits) are entangled, they will be correlated in a peculiar fashion. The measurement/alternating of one particle would instantaneously affect the other particle, regardless the spatial distance separating them throughout the universe. Einstein described this behaviour as “spooky action at a distance” [00, pg 122]. In the recent times, entanglement is being studied comprehensively from the perspective of quantum computation and quantum communication [55]. Quantum computation requires the creation and maintenance of highly complex quantum states which is a complicated mission due to decoherence. Noise and decoherence are some of the biggest obstacles to overcome in order to successfully construct a quantum computer [07, pg 133]. A quantum system will interact with its surroundings (environment) which will cause the quantum system to entangle with its environment, hence, reducing the entanglement within the system itself [55]. Such interaction will cause errors and our quantum information needs to be protected against them [07, pg 133]. Quantum error correction is the process of encoding quantum states into qubits so that errors or decoherence in a small number of individual qubits will have little or no effect on the encoded data [14]. The field of quantum computing gained lots of attention when Peter Shor published “Algorithms for Quantum Computation: Discrete Logarithms and Factoring” in 1994 [53] to which is now known as Shor’s algorithm. The paper describes a polynomial-time algorithm for prime factorization and discrete logarithms on a quantum computer. This will break public-key cryptography schemes such as RSA encryption algorithm, ElGamal encryption, Digital Signature Algorithm and Diffie-Hellman key exchange as their security depends on the difficulty of computing prime factorization and the discrete logarithm problem. Fortunately, post-quantum cryptography, a new branch of cryptographic algorithms has been developed to ensure secure cryptosystems can be used reliably against quantum computer attacks. Later, Shor published yet another powerful paper titled “Scheme for reducing decoherence in quantum computer memory”, which proposed a quantum error correction scheme that corrects a quantum error on a single qubit using nine qubits. This is known as Shor’s Code. Gottesman in 1997 established the formalism of stabilizer codes which allow us to import classical dual codes and transform them to quantum codes [23]. In 2006, Brun, Devetak and Hsieh established the formalism of entanglement-assisted quantum error correction [12] that uses entanglement to give the ability to important non-dual containing codes. The objective of this thesis is to find optimal entanglement-assisted quantum error correcting codes.

This thesis consists of ten chapters. The second chapter will introduce the essence of Coding Theory. We will examine linear binary codes and see how we can represent a



code using a generator matrix. Then, we discuss the process of encoding and decoding of linear codes. Next, the thesis explains self-dual codes, equivalent codes and the weight enumerator of a linear code. The third chapter focuses on the introduction to quantum computing starting with the definition of Hilbert space and Kronecker (tensor) product. Next, we rigorously define a qubit followed by a quantum system of  $n$  qubits. Quantum phenomena such as superposition and entanglement will be discussed afterwards. We then continue with quantum gates and finish the chapter with quantum circuits. The fourth chapter is dedicated to quantum error correction. We discuss how errors on a single qubit can be broken down to a specific form. We then introduce the three-qubit code and give a circuit that correct a bit-flip and another circuit to correct a phase-flip. We end the chapter with Shor's code which protects a qubit against a bit-flip or a phase-flip or a combination of both. The fifth chapter is centered around stabilizer codes. It will show how we can use the idea of commuting generators to detect errors. The sixth chapter is a generalized version of stabilizer codes that allows the generators to anticommute. It uses ebits to resolve the anticommutativity dilemma. The idea is to start with an anticommuting generators and find a new set of extended generators that commute. This is achieved with the usage of ebits. The seventh chapter gives an introduction to the quaternary field as well as Hermitian linear complementary dual codes. The latter will be the focus of this thesis. The next chapter, chapter eight, discusses the programming implementation and the usage of Bariess algorithm for finding the determinant of a quaternary matrix. Next, will discuss the new optimal and suboptimal quaternary hermitian linear complementary dual codes found. Lastly, we will give our thoughts for future work.

## Chapter 2

# Classical Error Correction

The essence of electronic communication is the ability to send and receive data through a physical medium such as a fiber optic cable or wirelessly such as WiFi. During the transmission phase, noise generated from an electrical impulse or environmental factors such as rain, cold, etc. can affect the data being transmitted. The data, which is a string of 0's and 1's, can be arbitrarily changed from  $0 \rightarrow 1$  or  $1 \rightarrow 0$  due to the noise encountered. Shannon's work shows that data can first be encoded before transmission then decoded at the receiver side to correct errors that might occur, to a certain degree [29, pg. 1, 52]. The objective of Coding Theory is to ensure the message received is the one that was sent. The operation of sending a message is [29, chap 1]:

- 1) A sender starts with a plain message to send. They use an encoder to encode the message. The output from the encoder is called a *codeword*.
- 2) The codeword is sent through a channel which is subject to noise that can cause errors.
- 3) The corrupted codeword arrives at the receiver's end.
- 4) The receiver uses a correction scheme to correct the data.
- 5) Once the corrupted codeword is corrected, a decoder is used which outputs the plain message sent by the sender.

An illustration of the entire operation is found in [Figure 2.1](#). The responsibility of an encoder is to transform plain text to a codeword. A *codeword* is a vector, or a word, that belongs to a code. A *code* is a set  $\mathcal{C}$  of codewords. A *linear* code  $\mathcal{C}$  satisfies the condition that any linear combination of codewords in  $\mathcal{C}$  is also a codeword in  $\mathcal{C}$ . Furthermore, any linear code can be described by a generator matrix  $G$ , where its rows are linearly independent and form a basis for the linear code<sup>1</sup>. All codewords in a linear code  $\mathcal{C}$  are generated by linear combinations of the rows in  $G$ .

---

<sup>1</sup>The all-zero vector cannot be a row in  $G$ .

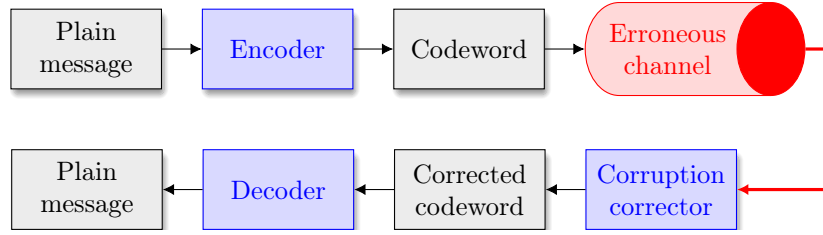


Figure 2.1: A diagram illustrating the journey a message takes when being transmitted through an erroneous medium

**Definition 1.** A linear code  $\mathcal{C}$  is described as  $[n, k, d]$ -code, and is a subset of  $\mathbb{F}_q^n$  vector space that encodes  $k$  logical bits into  $n$  physical bits where

$q$  is the number of symbols ( $q = 2$  is binary and  $q = 4$  is quaternary) and must be a prime or a power of prime.

$n$  is the length of each vector/codeword and  $n \geq k$ .

$k$  is the dimension of the vector space formed by the codewords in  $\mathcal{C}$ , which generates  $q^k$  codewords.

$d$  is the minimum hamming distance between any pair of arbitrary but distinct codewords in  $\mathcal{C}$ .

**Definition 2.** The weight function,  $WT(\cdot)$ , returns the number of nonzero symbols in a vector  $\mathbf{v} \in \mathbb{F}_q^n$ .

**Definition 3.** The Hamming distance is the number of positions in which two distinct codewords (equal in length) differ. It is defined as  $DIST(\mathbf{v}_1, \mathbf{v}_2)$  where  $\mathbf{v}_1$  and  $\mathbf{v}_2$  are vectors in  $\mathbb{F}_q^n$ . The Hamming distance also satisfies the following properties [29, sec 1.4]

- 1)  $DIST(\mathbf{v}_1, \mathbf{v}_2) \geq 0 \quad \forall \mathbf{v}_1, \mathbf{v}_2 \in \mathbb{F}_q^n$  (non-negativity).
- 2)  $DIST(\mathbf{v}_1, \mathbf{v}_2) = 0$  if and only if  $\mathbf{v}_1 = \mathbf{v}_2$  (identity of indiscernibles).
- 3)  $DIST(\mathbf{v}_1, \mathbf{v}_2) = DIST(\mathbf{v}_2, \mathbf{v}_1) \quad \forall \mathbf{v}_1, \mathbf{v}_2 \in \mathbb{F}_q^n$  (symmetry).
- 4)  $DIST(\mathbf{v}_1, \mathbf{v}_3) \leq DIST(\mathbf{v}_1, \mathbf{v}_2) + DIST(\mathbf{v}_2, \mathbf{v}_3) \quad \forall \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \in \mathbb{F}_q^n$  (triangle inequality).

**Definition 4.** The minimum distance of a code  $\mathcal{C}$  is the minimum Hamming distance between any two distinct codewords in  $\mathcal{C}$ .

**Theorem 1.** Let  $\mathbf{v}_1$  and  $\mathbf{v}_2$  be two vectors in  $\mathbb{F}_q^n$ , then

$$DIST(\mathbf{v}_1, \mathbf{v}_2) = WT(\mathbf{v}_1 - \mathbf{v}_2).$$

*Proof of Theorem 1.* Let  $\mathbf{v}_1$  and  $\mathbf{v}_2$  are two distinct vectors in  $\mathbb{F}_q^n$  and  $\text{DIST}(\mathbf{v}_1, \mathbf{v}_2)$  is  $x$ . Then  $\mathbf{v}' = \mathbf{v}_1 - \mathbf{v}_2$  will have  $x$  non-zero symbols, or equivalently,

$$\text{WT}(\mathbf{v}') = x = \text{WT}(\mathbf{v}_1 - \mathbf{v}_2) = \text{DIST}(\mathbf{v}_1, \mathbf{v}_2).$$

■

A generator matrix  $G$  is said to be in *standard form* if

$$G = [I_{k \times k} \mid P_{k \times (n-k)}]_{k \times n}, \quad (2.1)$$

where  $I$  is the identity matrix and  $P$  is a  $k \times (n - k)$  matrix. Another important matrix to consider is the parity check matrix. Assuming the generator matrix is in standard form, the parity check matrix  $H$  is given as

$$H = [-P_{(n-k) \times k}^T \mid I_{(n-k) \times (n-k)}]_{(n-k) \times n}, \quad (2.2)$$

where  $P^T$  is the transpose of  $P$ . The relationship between the generator matrix and parity check matrix is that

$$\begin{aligned} HG^T &= [0]_{(n-k) \times k} \\ GH^T &= [0]_{k \times (n-k)}, \end{aligned}$$

which results in the null matrix.

**Theorem 2.** An  $[n, k, d]$ -code can correct up to and including  $t$  errors if  $d \geq 2t + 1$  (proof can be found in [06, thm 2.2]).

**Lemma 1.** A linear code  $\mathcal{C}$  will correct at most  $t = \lfloor \frac{d-1}{2} \rfloor$  errors.

In Lemma 1, we used the floor function in the case where  $t$  is an odd number, dividing by 2 gives us  $2m + \frac{1}{2}$ , for some positive integer  $m$ . Since Hamming weight and Hamming distance are integers, we truncate the remaining  $\frac{1}{2}$ . We can see this geometrically in Figure 2.2. In sphere packing, all spheres are disjoint from each other within the space. A sphere contains a codeword at the center and all other vectors in green will be mapped to that codeword during the correction phase. Vectors in red cannot be corrected to the codewords in the center of the spheres because they are closer to that codeword than any other. In the example, that each ring represents one error from the codeword pinned in the center. The illustration shows that in this

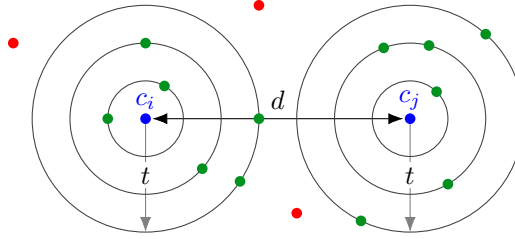


Figure 2.2: The geometric representation of sphere packing where two codewords (blue points)  $c_i$  and  $c_j$  exactly  $d = 2t + 1$  distance apart. The  $t$  represents the Hamming distance between the codeword at the center of the sphere and other vectors (green and red points). The vectors in red are not correctable whereas the green vectors are

example, we can correct at most three errors, hence,  $t = 3$ . Furthermore, the distance  $d$  from  $c_i$  to  $c_j$  is  $2t + 1 = 7$ . Consider Figure 2.3 where  $d = 2t$  and  $d = 2t - 1$ . The

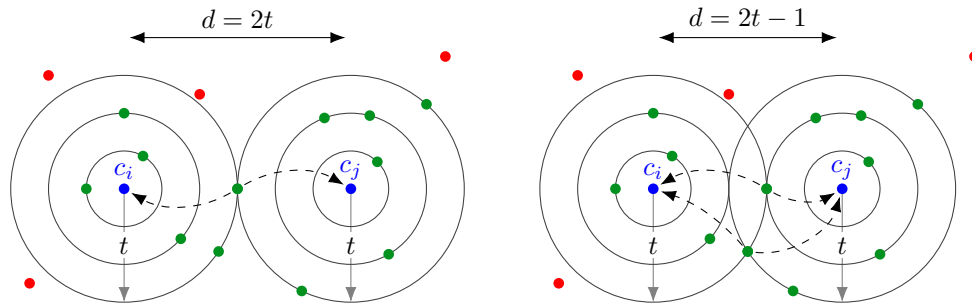


Figure 2.3: We can see that having a distance less than  $2t + 1$  yields to ambiguities. In the left portion of the figure, we have a vector that lies exactly at the intersection of two spheres. Moreover, the right side shows two vectors that can be mapped to two different codewords. In both cases, there are vectors that cannot be uniquely decoded

simplest of linear codes is the  $[3, 1, 3]$  repetition code that encodes one logical bit into three physical bits. It allows the transmission of the messages 0 or 1 and is assumed to encounter a single bit flip at most. The encoder will encode  $0 \rightarrow \bar{0} = 000$  and  $1 \rightarrow \bar{1} = 111$ . The total number of vectors of length 3 is  $2^n = 2^3 = 8$  and there are  $2^k = 2^1 = 2$  codewords, namely, 000 and 111. We correct received codewords 001, 010 and 100 to 000 and correct 011, 101 and 110 to 111. The quantum version of this code will be discussed extensively in Section 4.2.1.

## 2.1 Encoding and Decoding of Linear Codes

In this section, we will elaborate further on how the encoding and decoding processes are performed. We will use the  $[7, 4, 3]$  Hamming code [25] as an example to illustrate practically.

### 2.1.1 Encoding Process

Let  $\mathbf{C}$  be an  $[n, k, d]$  code over  $\mathbb{F}_q^n$  with a generator matrix  $G$  in standard form and a parity check matrix  $H$ . The code  $\mathbf{C}$  will have  $q^k$  possible codewords which yields to a one-to-one correspondence with  $q^k$  different messages. Let

$$\mathbf{x} = [x_1 \ x_2 \ \dots \ x_k]_{1 \times k}$$

be a message to be encoded. The codeword  $\mathbf{c}$  associated with  $\mathbf{x}$  is  $\mathbf{c} = \mathbf{x}G$ , which is a  $1 \times n$  vector. Using this method, the codeword  $\mathbf{c}$  generated and the logical (information bits) it holds are shown in Equation 2.3.

$$\mathbf{c} = \left[ \overbrace{c_1 = x_1 \quad c_2 = x_2 \quad \dots \quad c_k = x_k}^{\text{Information bits } \mathbf{x}} \quad \overbrace{c_{k+1} \quad \dots \quad c_{n-1} \quad c_n}^{\text{Redundancy/parity check bits}} \right] \quad (2.3)$$

The redundancy bits are added to assist in the recovery process in the case where errors might occur during transmission phase. Another standard way to do encoding is by using the parity check matrix  $H$ . We know that any codeword  $\mathbf{c}^T$  multiplied by  $H$  (*i.e.*,  $H\mathbf{c}^T$ ) gives us the all-zero vector. We follow the same pattern found in Equation 2.3. We can use the parity check matrix, more precisely, the submatrix  $P^T$  in Equation 2.2 to extract the redundancy/parity check symbols,

$$P^T \mathbf{x}^T = [c_{k+1} \ c_{k+2} \ \dots \ c_{n-1} \ c_n]^T. \quad (2.4)$$

Consider the example of a  $[7, 4, 3]$  Hamming code which encodes four logical bits into seven physical bits and corrects  $t = \lfloor \frac{d-1}{2} \rfloor = \lfloor \frac{3-1}{2} \rfloor = 1$  error<sup>2</sup>. It is given with the following generator matrix and corresponding parity check matrix:

$$G = \left[ \begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array} \right] \begin{matrix} \textcircled{1} \\ \textcircled{2} \\ \textcircled{3} \\ \textcircled{4} \end{matrix}, \quad H = \left[ \begin{array}{cccc|ccc} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array} \right]$$

The all-zero vector in a linear code will always be a valid codeword. The list of all codewords with their weights are shown in Table 2.1. For example, suppose the message  $\mathbf{x} = [0 \ 1 \ 1 \ 0]$  needs to be sent over a channel. The first way of encoding is

<sup>2</sup>In general, binary Hamming codes are given in the form of  $[2^a - 1, 2^a - a - 1, 3]$  for some  $a \geq 2$

#	Linear combinations	Codeword	Weight
00	None	0000000	0
01	① in $G$	1000011	3
02	② in $G$	0100101	3
03	① + ②	1100110	4
04	③ in $G$	0010110	3
05	① + ③	1010101	4
06	② + ③	0110011	4
07	① + ② + ③	1110000	3
08	④ in $G$	0001111	4
09	① + ④	1001100	3
10	② + ④	0101010	3
11	① + ② + ④	1101001	4
12	③ + ④	0011001	3
13	① + ③ + ④	1011010	4
14	② + ③ + ④	0111100	4
15	① + ② + ③ + ④	1111111	7

Table 2.1: All of the  $2^k = 2^4 = 16$  codewords of the  $[7, 4, 3]$  Hamming code with their weights

to have  $\mathbf{c} = \mathbf{x}G = [0 \ 1 \ 1 \ 0 \ 0 \ 1 \ 1]$ . The second way is  $P^T x^T = [0 \ 1 \ 1]$ . Hence, the codeword  $\mathbf{c} = [0 \ 1 \ 1 \ 0 \ 0 \ 1 \ 1]$  will be sent over.

### 2.1.2 Decoding Process

This subsection is primarily based on [29, sec. 1.11.2]. The decoding process takes care of receiving a (possible) erroneous vector  $\mathbf{y} \in \mathbb{F}_q^n$  and mapping it to the original codeword  $\mathbf{c}$ . Asymptotically speaking, decoding is more difficult than encoding for large codes. We will introduce two decoding methods: maximum likelihood decoding and syndrome decoding.

Maximum likelihood decoding is a probabilistic method that maps an erroneous codeword  $\mathbf{y}$  that was received after transmission to a codeword. This is done by comparing  $\mathbf{y}$  to all the codewords and mapping it to the codeword with the highest probability, which is the codeword closest to  $\mathbf{y}$ . We can use the concept of sphere as shown in Figure 2.2. A sphere  $S$  with radius  $t$  centered around some codeword  $\mathbf{c} \in \mathbb{F}_q^n$  is given as

$$S_t(\mathbf{c}) = \{\mathbf{v} \in \mathbb{F}_q^n \mid \text{DIST}(\mathbf{c}, \mathbf{v}) \leq t\}, \quad (2.5)$$

which are the vectors that are at Hamming distance less than or equal to  $t$  from  $\mathbf{c}$ .

The total possible vectors in a sphere  $S_t(\mathbf{c})$  is

$$S = \binom{n}{0}(q-1)^0 + \binom{n}{1}(q-1)^1 + \cdots + \binom{n}{t-1}(q-1)^{t-1} + \binom{n}{t}(q-1)^t, \quad (2.6)$$

where  $q$  is the cardinality of the alphabet in the code. For the first term, we have the codeword itself. For the second term, we are choosing all the vectors that differs by one position from the codeword and multiply by the possible different symbols that position can take. The last term is choosing possible vectors that differ from the codeword by at most  $t$  positions. Each different position can have  $q-1$  different symbols. Since we have  $2^k$  different codewords, the total possible combinations  $S_{all}$  of vectors satisfying all disjoint spheres is

$$S_{all} = 2^k \cdot \left( \binom{n}{0}(q-1)^0 + \binom{n}{1}(q-1)^1 + \cdots + \binom{n}{t-1}(q-1)^{t-1} + \binom{n}{t}(q-1)^t \right). \quad (2.7)$$

Another way to decode is by using Syndrome decoding. Let  $\mathbf{C}$  be an  $[n, k, d]$  code over  $\mathbb{F}_q^n$  which corrects at most  $t = \lfloor \frac{d-1}{2} \rfloor$  errors with parity check matrix  $H$ . Assume the codeword  $\mathbf{c} \in \mathbf{C}$  was sent through an erroneous channel and was received as  $\mathbf{y} \in \mathbb{F}_q^n$ . Suppose that  $\mathbf{e} \in \mathbb{F}_q^n$  is the error encountered, *i.e.*,  $\mathbf{y} = \mathbf{c} + \mathbf{e}$ . Using the parity check matrix  $H$ , we know any codeword  $\mathbf{y}$  received without errors gives  $H\mathbf{y}^T = [0 \ 0 \ \dots \ 0 \ 0]$ . In the case where  $\mathbf{y}$  encountered an error  $\mathbf{e}$ , then

$$H\mathbf{y}^T = H(\mathbf{c} + \mathbf{e})^T = H\mathbf{c}^T + H\mathbf{e}^T = [0 \ 0 \ \dots \ 0 \ 0] + H\mathbf{e}^T = H\mathbf{e}^T. \quad (2.8)$$

In the case where  $x$  errors occur, then  $\text{WT}(\mathbf{e}) = x$ . Moreover, this suggests that if the codeword length is  $n$  and can correct  $t = 1$  errors at most, then there are  $\binom{n}{t} = \binom{n}{1} = n$  coset leaders. In the case where we are in  $\mathbb{F}_q^n$  and the code  $\mathbf{C}$  corrects  $t$  errors at most, then we will have the same number of coset leaders as given in Equation 2.6. With that said, the idea is to then construct a lookup table where each of the coset leader and its *syndrome*, given as

$$\text{SYN}(\mathbf{y}) = H\mathbf{y}^T. \quad (2.9)$$

Let us continue the  $[7, 4, 3]$  Hamming code example we introduced in the previous section. Recall that we wanted to send the message  $\mathbf{x} = [0 \ 1 \ 1 \ 0]$  over the channel. We encoded it to  $\mathbf{c} = [0 \ 1 \ 1 \ 0 \ 0 \ 1 \ 1]$  and we sent  $\mathbf{c}$  over. This code will correct at most  $t = \lfloor \frac{d-1}{2} \rfloor = \lfloor \frac{3-1}{2} \rfloor = 1$  error. Suppose that through the transmission an error occurred and we received  $\mathbf{y} = [0 \ 1 \ 0 \ 0 \ 0 \ 1 \ 1]$ . Our lookup table will include all possible errors that can occur and corrected by the code which is given by Table 2.2.



We can compute the syndrome of the error we received:

#	Coset Leader ( $\mathbf{e}$ )	Syndrome ( $H\mathbf{y}^T$ )
00	0000000	000
01	1000000	011
02	0100000	101
03	0010000	110
04	0001000	111
05	0000100	100
06	0000010	010
07	0000001	001

Table 2.2: All possible coset leaders and their syndromes for the  $[7, 4, 3]$  Hamming code

$$\text{SYN}(\mathbf{y}) = H\mathbf{y}^T = H \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}^T = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}^T. \quad (2.10)$$

Using Table 2.2, we conclude that  $\mathbf{e} = [0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0]$  is the error occurred. We can simply xor  $\mathbf{e}$  with  $\mathbf{y}$  to get the original codeword:

$$\mathbf{c} = \mathbf{y} \oplus \mathbf{e} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}. \quad (2.11)$$

### 2.1.3 Self-(Dual/Orthogonal) and Equivalent Codes

Let  $\mathcal{C}$  be a linear code over  $\mathbb{F}_q^n$  with a generator matrix  $G$ . Since all rows in  $G$  are linearly independent and span the entire space of  $\mathcal{C}$ , they form a basis for  $\mathcal{C}$ . The subspace of  $\mathcal{C}$  has a dual subspace.

**Definition 5.** Let  $\mathcal{C}^\perp$  be the dual subspace of  $\mathcal{C}$  given as

$$\mathcal{C}^\perp = \{ \mathbf{x} \in \mathbb{F}_q^n \mid \mathbf{x} \cdot \mathbf{c} = 0 \ \forall \mathbf{c} \in \mathcal{C} \}, \quad (2.12)$$

where  $\mathbf{x} \cdot \mathbf{c} = \sum_{i=1}^n x_i c_i \forall \mathbf{x} = [x_1 \ x_2 \ \dots \ x_n]$  and  $\mathbf{y} = [y_1 \ y_2 \ \dots \ y_n]$  in  $\mathbb{F}_q^n$ .

**Definition 6.** A binary linear code  $\mathcal{C}$  is said to be self-orthogonal if  $\mathcal{C} \subseteq \mathcal{C}^\perp$ .

The  $[7, 4, 3]$  Hamming code is self-orthogonal. Let  $\mathcal{C}$  be the set of words that is

generated by  $H$  and  $C^\perp$  be the set of words that is generated by  $G$ , then

$$\begin{aligned} C &= \{0000000, 0111100, 1011010, 1100110, \\ &\quad 1101001, 1010101, 0110011, 0001111\} \\ C^\perp &= \{0000000, \underline{1000011}, \underline{0100101}, 1100110, \\ &\quad \underline{0010110}, 1010101, 0110011, \underline{1110000}, \\ &\quad 0001111, \underline{1001100}, \underline{0101010}, 1101001, \\ &\quad \underline{0011001}, 1011010, 0111100, \underline{1111111}\}. \end{aligned}$$

**Lemma 2.** A  $[n, k, d]$  binary linear code  $C$  is said to be self-dual if  $C = C^\perp$ . The dual code  $C^\perp$  will be a  $[n, n - k, d]$  binary linear code implies that  $\text{DIM}(C) = \text{DIM}(C^\perp)$  deducing that  $k = n - k$ .

An example of a self-dual code is the  $[8, 4, 4]$  extended Hamming code with the following generator matrix and parity check matrix:

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}, \quad H = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The generator and parity check matrices of  $[8, 4, 4]$  code generates the following codewords:

$$\begin{aligned} C = C^\perp &= \{00000000, 10000111, 01001011, 11001100, \\ &\quad 00101101, 10101010, 01100110, 11100001, \\ &\quad 00011110, 10011001, 01010101, 11010010, \\ &\quad 00110011, 10110100, 01111000, 11111111\}. \end{aligned}$$

**Theorem 3.** A self-dual linear code  $C$  will have  $G$  as the generator matrix and  $H$  as the parity check matrix.  $C^\perp$  will have  $H$  as the generator matrix and  $G$  as the parity check matrix.

*Proof of Theorem 3.* By definition,  $C^\perp$  is the set that contains all codewords  $\mathbf{y}$  such that  $\mathbf{y} \cdot \mathbf{c} = 0$  for all  $\mathbf{c} \in C$ . Given that  $C = C^\perp$ , then  $\mathbf{c} \cdot \mathbf{c} = 0 \quad \forall \mathbf{c} \in C$ . Furthermore, let  $\mathbf{x}_{1 \times k}$  and  $\mathbf{x}'_{1 \times (n-k)}$ <sup>3</sup> be two non-encoded messages which will be encoded by  $C$  and

<sup>3</sup>The dimension of  $\mathbf{x}'$  is  $1 \times (n - k)$  and not  $1 \times k$  since  $\mathbf{x}'$  will be transposed and multiplied by the  $H$  and the dimensions need to match.

$\mathbf{C}^\perp$ , respectively. It is sufficient to show that  $H\mathbf{c}^T = \mathbf{0}$  for all codewords  $\mathbf{c} \in \mathbf{C}$  and  $\mathbf{c}'^T G = \mathbf{0}$  for all  $\mathbf{c}' \in \mathbf{C}^\perp$ . The message  $\mathbf{x}$  is encoded as  $\mathbf{x}G$  implying that  $\mathbf{c} = \mathbf{x}G$ . The message  $\mathbf{x}'$  is encoded as  $\mathbf{x}'H$  implying that  $\mathbf{c}' = \mathbf{x}'H$ . We now need to show that  $\mathbf{c}$  and  $\mathbf{c}'^T$  are orthogonal:

$$\begin{aligned} \mathbf{c} \cdot \mathbf{c}'^T &= (\mathbf{x}G) \cdot (\mathbf{x}'H)^T \\ &= (\mathbf{x}G) \cdot (H^T \mathbf{x}'^T) \\ &= \mathbf{x}(GH^T) \mathbf{x}'^T \\ &= \mathbf{x}_{1 \times k} \begin{bmatrix} 0 \end{bmatrix}_{k \times (n-k)} \mathbf{x}'^T_{(n-k) \times 1} \\ &= 0. \end{aligned}$$

■

Self-dual and self-orthogonal classical codes can be transformed into quantum stabilizer codes. Further on that in [Chapter 5](#). Throughout this chapter, we assumed that all generator matrices of linear codes are in standard form, as shown in [Equation 2.1](#). However, generator matrices do not have to be in standard form. We can apply Gaussian elimination to produce a matrix  $G$  in the same form as [Equation 2.1](#).

Moreover, we can have two different codes that are equivalent to one another.

**Definition 7.** Let  $\mathbf{C}_1$  and  $\mathbf{C}_2$  be two binary linear codes. The two codes are said to be equivalent if there exists some permutation of coordinates which maps  $\mathbf{C}_1$  to  $\mathbf{C}_2$  [[29](#), sec 1.6].

A permutation matrix, which is a square binary matrix with each row and column containing a single 1 and 0's everywhere else, can be used to permute the coordinates.

**Definition 8.** Let  $\mathbf{C}_1$  and  $\mathbf{C}_2$  be two binary linear codes with  $G_1$  and  $G_2$  as generator matrices, respectively. Let  $M$  be a permutation matrix such that  $G_1 \cdot M = G_2$ , then  $\mathbf{C}_1$  and  $\mathbf{C}_2$  are permutation equivalent.

## 2.2 Weight enumerator

The *weight enumerator* is a polynomial function which contains the Hamming weights of all possible codeword in a linear code  $\mathbf{C}$ .

**Definition 9.** The weight enumerator of a linear code  $\mathbf{C}$  over  $\mathbb{F}_q^n$  is given by

$$W(x)_\mathbf{C} = A_0 + A_1x + A_2x^2 + \dots + A_{n-2}x^{n-2} + A_{n-1}x^{n-1} + A_nx^n,$$

where  $A_i$  is the number of codewords of  $\mathcal{C}$  whose Hamming weight is  $i$ .

From [Table 2.1](#), the  $[7, 4, 3]$  Hamming code will its weight enumerator function given as [Equation 2.13](#).

$$W(x)_{[7,4,3]} = 1 + 7x^3 + 7x^4 + x^7. \quad (2.13)$$

## Chapter 3

# Introduction to Quantum Computing

In this chapter, we will begin with covering all of the required mathematical knowledge required for quantum computing. We will then introduce the mathematical description of a qubit, quantum phenomena, quantum gates and conclude with quantum circuits.

### 3.1 Hilbert Space

In this section, we will introduce the definition of dot product, norm and distance of an Euclidean space [03, chap. 3.2]. We will then provide the definitions of inner product, norm and distance of a Hilbert space. Given two vectors in an Euclidean space of  $\mathbb{R}^n$ ,

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix},$$

the *dot* product is defined as

$$\mathbf{a} \cdot \mathbf{b} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \cdot \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = a_1 \cdot b_1 + a_2 \cdot b_2 + \cdots + a_n \cdot b_n,$$

the norm (magnitude or length) of  $\mathbf{a}$ , denoted as  $\|\cdot\|$ , is

$$\|\mathbf{a}\| = \sqrt{a_1^2 + a_2^2 + \cdots + a_n^2},$$

and the Euclidean distance between  $\mathbf{a}$  and  $\mathbf{b}$ , denoted as  $\text{DIST}(\cdot, \cdot)$ , is

$$\text{DIST}(\mathbf{a}, \mathbf{b}) = \|\mathbf{a} - \mathbf{b}\| = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \dots + (a_n - b_n)^2}.$$

The *inner* (equivalent to dot) product in a Hilbert space,  $\mathcal{H}$ , is defined as

$$\langle \mathbf{a} | \mathbf{b} \rangle = \sum_{k=1}^n a_k^* b_k,$$

where

$$\langle \mathbf{a} | = \begin{bmatrix} a_1^* & a_2^* & \dots & a_n^* \end{bmatrix}_{1 \times n}, \quad |\mathbf{b}\rangle = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}_{n \times 1},$$

$a_k^*$  is the complex conjugate of  $a_k$

Here are some useful properties [37, chap. 4.3]

1.  $\langle \mathbf{a} | \mathbf{b} \rangle^* = \langle \mathbf{b} | \mathbf{a} \rangle$ .
2. The inner product is not necessarily commutative, *i.e.*, there exist  $\mathbf{a}$  and  $\mathbf{b}$  for which  $\langle \mathbf{a} | \mathbf{b} \rangle \neq \langle \mathbf{b} | \mathbf{a} \rangle$ .
3. For a given  $c \in \mathbb{C}$ :
  - (a) The inner product is linear in the second position

$$c \langle \mathbf{a} | \mathbf{b} \rangle = \langle \mathbf{a} | c\mathbf{b} \rangle.$$

- (b) The inner product is anti-linear in the first position

$$c \langle \mathbf{a} | \mathbf{b} \rangle = \langle c^* \mathbf{a} | \mathbf{b} \rangle.$$

In Hilbert space, the norm of a vector  $\mathbf{a}$  is defined as

$$\|\mathbf{a}\| = \sqrt{\sum_{k=1}^n |a_k|^2} = \sqrt{\sum_{k=1}^n a_k^* a_k},$$

and the distance between vectors  $\mathbf{a}$  and  $\mathbf{b}$  is defined as

$$\text{DIST}(\mathbf{a}, \mathbf{b}) = \|\mathbf{b} - \mathbf{a}\| = \sqrt{\sum_{k=1}^n |b_k - a_k|^2}.$$

We now can provide a rigorous definition of a Hilbert space [37, chap. 4.4].

**Definition 10.** *A Hilbert space is a complete real or complex (finite or infinite) vector space where the inner product is defined. The completeness property is defined as any Cauchy Sequence of vectors in the space converges to some vector which is also in the space.*

A Cauchy Sequence is rigorously defined as [50, chap. 1]:

**Definition 11.** *Let  $(X, d)$  be a metric space where  $X$  is a set and  $d$  is the metric (distance) function acting on  $X$  defined as*

$$d : X \times X \rightarrow \mathbb{R}.$$

*A sequence  $x_1, x_2, \dots \in X$  is a Cauchy Sequence if for every  $\varepsilon \in \mathbb{R}$  and  $\varepsilon > 0$ , there exists a  $k \in \mathbb{Z}^+$  such that for  $i, j \in \mathbb{Z}^+$  and  $i, j > k$ ,  $\text{DIST}(x_i, x_j) < \varepsilon$ .*

Throughout the thesis, we will not use bold kets or bras and adapt to the convention that they are vectors by default.

## 3.2 Mathematical Preliminaries

In this section, we will be discussing the required mathematical concepts for this thesis. Specifically, we will introduce the tensor (Kronecker) product, kets, bras, and brackets as well as Hilbert space. The tensor product, denoted as  $\otimes$ , is applied to two matrices of arbitrary size.<sup>1</sup> Given two matrices  $A$  and  $B$ ,

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}_{m \times n}, \quad B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1q} \\ b_{21} & b_{22} & \cdots & b_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ b_{p1} & b_{p2} & \cdots & b_{pq} \end{bmatrix}_{p \times q},$$

---

<sup>1</sup>By matrices, row vectors column and vectors are also valid.

then, the tensor product of  $A \otimes B$  is:

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix}_{(mp) \times (nq)},$$

where the resulting matrix is an  $mp$  by  $nq$  matrix [45, chap. 1.2]. For example, the tensor of  $A$  and  $B$  matrices is:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},$$

$$A \otimes B = \begin{bmatrix} 1 \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} & 0 \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} & 0 \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} & 0 \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \\ 0 \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} & 1 \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} & 0 \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} & 0 \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \\ 0 \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} & 0 \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} & 0 \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} & 1 \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \\ 0 \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} & 0 \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} & 1 \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} & 0 \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \end{bmatrix}.$$

It is worth mentioning that the tensor product is not necessarily commutative, *i.e.*,  $A \otimes B \neq B \otimes A$ . A useful notation when it comes to applying multiple tensor products in Hilbert space  $\mathcal{H}$  is to use  $\mathcal{H}^{\otimes n} = \underbrace{\mathcal{H} \otimes \mathcal{H} \otimes \cdots \otimes \mathcal{H}}_{n \text{ times}} \in \mathcal{H}^n$ . Kets, bras, and brackets are Dirac's notation, named after the famous physicist, Paul Dirac. Kets, which are denoted by  $|\cdot\rangle$ , are a compact form of a column vector in  $\mathcal{H}$  whereas bras are denoted as  $\langle\cdot|$  and represents a row vector in the dual Hilbert space,  $\mathcal{H}^*$ . The bra is the Hermitian conjugate of a ket. The bracket, denoted by  $\langle a|b\rangle$ , is the inner product of  $\langle a|$  with  $|b\rangle$  resulting in a scalar in  $\mathbb{C}$ . For example, let  $\langle a| \in \mathcal{H}^*$  and  $|b\rangle \in \mathcal{H}$ ,

$$\langle a|b\rangle = b_1 a_1^* + b_2 a_2^* + \cdots + b_n a_n^*.$$



In this thesis, we will be dealing with tensoring column vectors frequently. The next notations will all be equivalent to each other:

$$\underbrace{|a_1\rangle \otimes |a_2\rangle \otimes \cdots \otimes |a_n\rangle}_{\text{Tensoring } n \text{ times}} = \underbrace{|a_1\rangle |a_2\rangle \cdots |a_n\rangle}_{\text{Tensoring } n \text{ times without } \otimes \text{ symbol}} = \underbrace{|a_1 a_2 \dots a_n\rangle}_{\text{A compact notation of tensoring } n \text{ times}} . \quad (3.1)$$

### 3.3 An Overview of a Qubit

A quantum binary digit (*qubit*) is the atomic unit of a quantum computer for storing quantum information. It is the equivalent of a *bit* in a classical computer. A qubit can represent the value “0” as the  $|0\rangle$  quantum state or the value of “1” as the  $|1\rangle$  quantum state. Moreover, a qubit can in a *superposition* of  $|0\rangle$  and  $|1\rangle$ . This means that a qubit can simultaneously occur in the  $|0\rangle$  state and the  $|1\rangle$  state. Rigorously, a qubit  $|\psi\rangle$  is defined as

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle , \quad (3.2)$$

where  $\alpha, \beta \in \mathbb{C}$  and  $|\alpha|^2 + |\beta|^2 = 1$ . The  $|\alpha|^2$  denotes the probability of the qubit  $|\psi\rangle$  evaluating to  $|0\rangle$  whereas  $|\beta|^2$  is the probability of the qubit  $|\psi\rangle$  evaluating to  $|1\rangle$ . The coefficients  $\alpha$  and  $\beta$  are referred to as probability amplitudes (or complex amplitudes) whereas  $|0\rangle$  and  $|1\rangle$  are known as computational basis vectors (or basis states) and Equation 3.2 is a quantum state. Similar to a classical bit, once a qubit is measured, it will collapse to a “0” ( $|0\rangle$ ) with probability of  $|\alpha|^2$  or “1” ( $|1\rangle$ ) with probability of  $|\beta|^2$  [45, chap. 1.2, 60, chap. 5.1]. In general, measuring a qubit will cause it to collapse only one of its computational basis vectors. This means the qubit will no longer be in superposition and will act like a classical bit after measurement. This is a crucial point to keep in mind, as the measurement of a qubit will disturb its state. The kets  $|0\rangle$  and  $|1\rangle$  are a compacted form of two different column vectors. They are equivalent to

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} , \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} .$$

With that being said, we can rewrite  $|\psi\rangle$  as

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

A simple way to understand how the values of the column vector are obtained is through the following: the label inside  $|\cdot\rangle$  denotes the zero-based column in binary system. So,  $|0\rangle$  will have a “1” in the zeroth cell whereas  $|1\rangle$  will have a “1” in the subsequent cell. The same kets are now associated with labels (in base 2):

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{matrix} \mathbf{0_2} \\ \mathbf{1_2} \end{matrix}, \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{matrix} \mathbf{0_2} \\ \mathbf{1_2} \end{matrix}.$$

These labels will help us understand how multiqubits are represented further in this section. Visualizing a single bit is something very trivial and simple. Before we begin, it is worth noting that a bit can be expressed using the same notation as a qubit, since a bit is a special case of a qubit. For example, we can express the “0” bit by the qubit state:

$$|\psi\rangle = 1 \cdot |0\rangle + 0 \cdot \cancel{|1\rangle}^0.$$

Similarly, we can express the “1” bit by the qubit state:

$$|\psi\rangle = 0 \cdot \cancel{|0\rangle}^0 + 1 \cdot |1\rangle.$$

With that, we have established a way to represent a bit in a 2D plane in  $\mathbb{R}^2$ . A simple visualization is shown in [Figure 3.1](#). It is important to notice that the unit vectors (with a scalar of 1) are the only two valid vectors here. The  $|0\rangle$  ket denotes a bit with a value of 0 and the  $|1\rangle$  vector denotes a bit with a value of 1. The visualization of a single qubit is far more complex than a bit. The state of a qubit can be expressed in a Bloch sphere. A Bloch sphere (also known as *Poincaré sphere*) is a visualization of a single qubit in 3D space [45, chap. 1.2].<sup>2</sup> It is assumed to have a radius of 1 and the qubit is denoted as a unit vector of length 1. The Bloch sphere is shown

<sup>2</sup>For two or more qubits, it is exponentially difficult to visualize as the dimension increases.

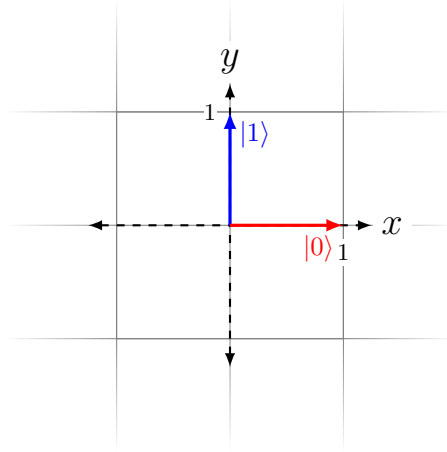


Figure 3.1: The visualization of a bit in a 2D plane

in Figure 3.2. In a Bloch sphere, a qubit in the form of Equation 3.2 can be described

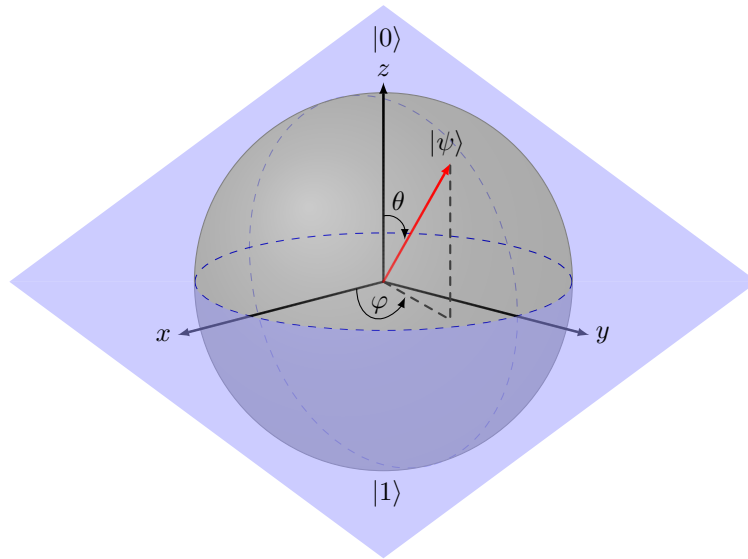


Figure 3.2: The Bloch sphere representation with the equatorial plane passing through the origin. The red vector is the qubit state  $|\psi\rangle$

in the equation:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right) |0\rangle + e^{i\varphi} \sin\left(\frac{\theta}{2}\right) |1\rangle, \quad (3.3)$$

where  $0 \leq \theta < \pi$  and  $0 \leq \varphi < 2\pi$  [48, chap. 2.1]. Note that we simply substituted  $\alpha$  with  $\cos\left(\frac{\theta}{2}\right)$  and  $\beta$  with  $e^{i\varphi} \sin\left(\frac{\theta}{2}\right)$ . The two angles are sufficient to define a point on the Bloch sphere. A qubit can lie along the  $z$ -axis denoting  $|0\rangle$  or straight down denoting  $|1\rangle$ . Figure 3.3 depicts the two states visually. Moreover, there are infinitely

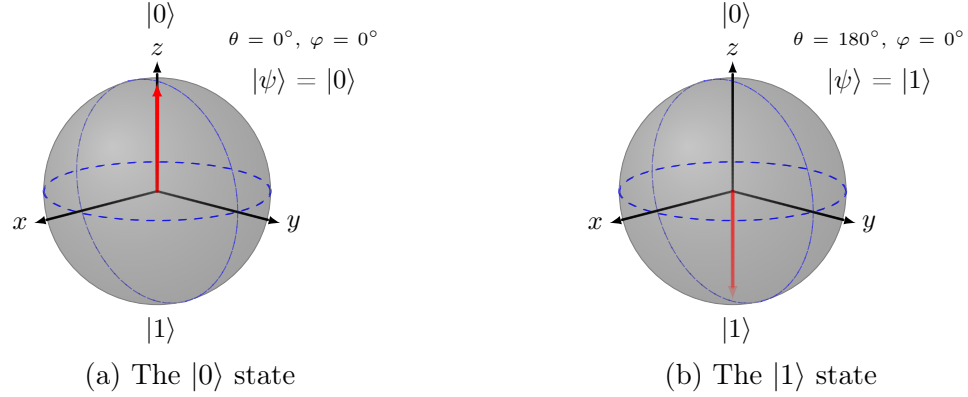


Figure 3.3: The two main states,  $|0\rangle$  on the left side and  $|1\rangle$  on the right side

many states that can be represented as  $\alpha$  and  $\beta$  have an infinite range of values. The most common states are:

$$\begin{aligned}
 |\psi\rangle = |+\rangle &= \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle, & |\psi\rangle = |-\rangle &= \frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle \\
 |\psi\rangle = |i+\rangle &= \frac{1}{\sqrt{2}} |0\rangle + \frac{i}{\sqrt{2}} |1\rangle, & |\psi\rangle = |i-\rangle &= \frac{1}{\sqrt{2}} |0\rangle - \frac{i}{\sqrt{2}} |1\rangle.
 \end{aligned} \tag{3.4}$$

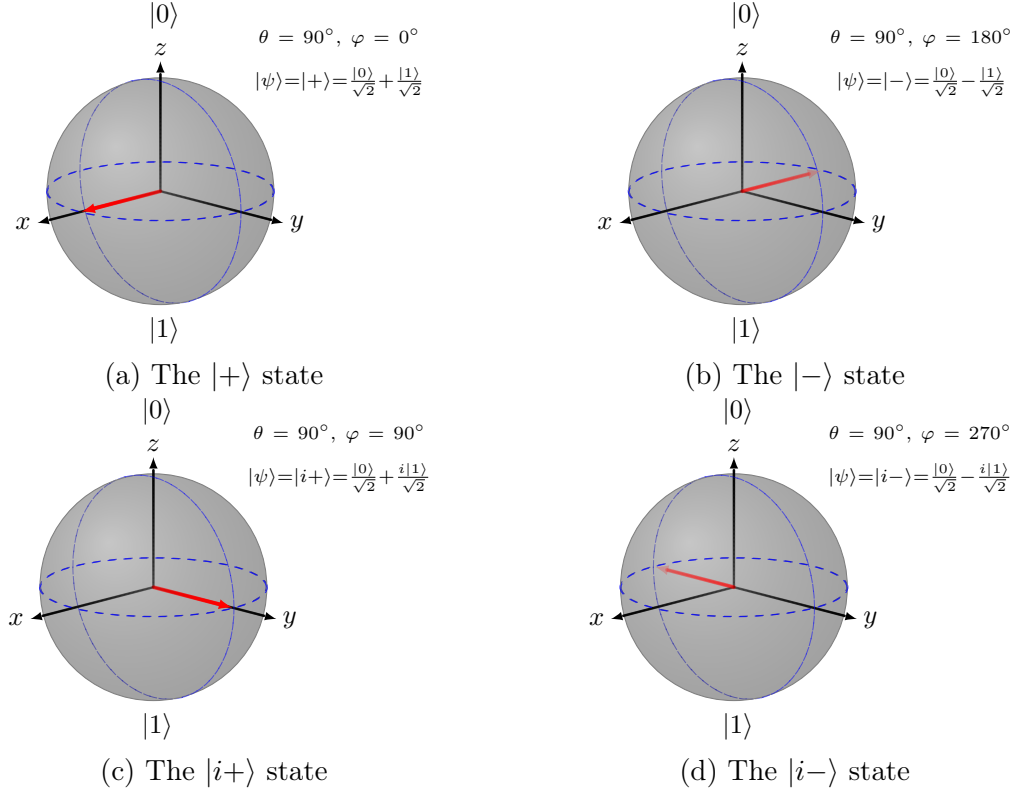
They all lie on the axes of the Bloch sphere as shown in [Figure 3.4](#). The more generalized qubits states of [Equation 3.4](#) are:

$$\begin{aligned}
 |\psi\rangle &= \alpha |0\rangle + \beta |1\rangle, & |\psi\rangle &= \alpha |0\rangle - \beta |1\rangle \\
 |\psi\rangle &= \alpha |0\rangle + i\beta |1\rangle, & |\psi\rangle &= \alpha |0\rangle - i\beta |1\rangle.
 \end{aligned} \tag{3.5}$$

So far, we have only discussed the characteristics of a single qubit. Let's extend our definition to include a two-qubit system. A two-qubit system, which lies in  $\mathcal{H} \otimes \mathcal{H} \in \mathcal{H}^2$ , will be in the form of

$$|\psi\rangle = \alpha |00\rangle + \beta |01\rangle + \gamma |10\rangle + \delta |11\rangle. \tag{3.6}$$

where  $\alpha, \beta, \gamma, \delta \in \mathbb{C}$  and  $|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$ . In this system,  $|\alpha|^2, |\beta|^2, |\gamma|^2, |\delta|^2$  denote the probability of the qubit  $|\psi\rangle$  evaluating to  $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ , respectively. The kets  $|ij\rangle$  are the basis in this system. They represent a compact form of a column vector obtained via tensor product of smaller column vectors. The four bases can be

Figure 3.4: The four common states other than  $|0\rangle$  and  $|1\rangle$ 

written as

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

In terms of the column vectors obtained, we can simply label them as follows to see a clear pattern:

$$|00\rangle = \begin{bmatrix} \alpha \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{matrix} \mathbf{00}_2 \\ \mathbf{01}_2 \\ \mathbf{10}_2 \\ \mathbf{11}_2 \end{matrix}, \quad |01\rangle = \begin{bmatrix} 0 \\ \beta \\ 0 \\ 0 \end{bmatrix} \begin{matrix} \mathbf{00}_2 \\ \mathbf{01}_2 \\ \mathbf{10}_2 \\ \mathbf{11}_2 \end{matrix}, \quad |10\rangle = \begin{bmatrix} 0 \\ 0 \\ \gamma \\ 0 \end{bmatrix} \begin{matrix} \mathbf{00}_2 \\ \mathbf{01}_2 \\ \mathbf{10}_2 \\ \mathbf{11}_2 \end{matrix}, \quad |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \delta \end{bmatrix} \begin{matrix} \mathbf{00}_2 \\ \mathbf{01}_2 \\ \mathbf{10}_2 \\ \mathbf{11}_2 \end{matrix}.$$

We will now expand our definition to generalize an  $n$ -qubit system. A system of  $n$  qubits lies in  $\underbrace{\mathcal{H} \otimes \mathcal{H} \otimes \cdots \otimes \mathcal{H}}_{n \text{ times}} \in \mathcal{H}^n$ , will have a basis of with a dimension of  $2^n$ . Its

qubit state will be given as

$$|\psi\rangle = \underbrace{\alpha_{00\dots 01}}_{\text{length } n} \overbrace{|00\dots 01\rangle}^{\text{length } n} + \underbrace{\alpha_{00\dots 10}}_{\text{length } n} \overbrace{|00\dots 10\rangle}^{\text{length } n} + \dots + \underbrace{\alpha_{11\dots 11}}_{\text{length } n} \overbrace{|11\dots 11\rangle}^{\text{length } n}, \quad (3.7)$$

where  $|\alpha_{00\dots 01}|^2 + |\alpha_{00\dots 10}|^2 + \dots + |\alpha_{11\dots 11}|^2 = 1$ .

### 3.4 Global Phase versus Relative Phase

A phase is in the form of  $re^{i\theta}$  or  $e^{i\theta}$  where  $r = 1$  when dealing with normalization constraint setting. A qubit can encounter two types of phases: global phase and relative phase. A global phase factor is a complex number of unit modulus multiplying the state (*i.e.*,  $|z|^2 = 1$ ). In quantum computing, two qubits that differ by a global phase from each other are considered equivalent. This is because a global phase is physically indistinguishable [19]. A qubit is represented as vector of norm 1, *i.e.*,  $\langle\psi|\psi\rangle = 1$  [19]. Consider the following two arbitrary qubits:

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle \quad (3.8)$$

$$|\phi\rangle = e^{i\theta} |\psi\rangle = e^{i\theta} \alpha |0\rangle + e^{i\theta} \beta |1\rangle. \quad (3.9)$$

For Equation 3.8, let

$$\alpha = a_1 + ib_1 \quad \alpha^* = a_1 - ib_1 \quad \beta = a_2 + ib_2 \quad \beta^* = a_2 - ib_2. \quad (3.10)$$

Then,

$$\begin{aligned} |\alpha|^2 &= \alpha\alpha^* & |\beta|^2 &= \beta\beta^* \\ &= (a_1 + ib_1)(a_1 - ib_1) & &= (a_2 + ib_2)(a_2 - ib_2) \\ &= a_1^2 - ia_1b_1 + ia_1b_1 - i^2b_1^2 & &= a_2^2 - ia_2b_2 + ia_2b_2 - i^2b_2^2 \\ &= a_1^2 + b_1^2. & &= a_2^2 + b_2^2. \end{aligned}$$

Hence, in Equation 3.8, the probability of measuring  $|0\rangle$  is  $a_1^2 + b_1^2$  and the probability of measuring  $|1\rangle$  is  $a_2^2 + b_2^2$ , provided that  $a_1^2 + b_1^2 + a_2^2 + b_2^2 = 1$ . For Equation 3.9,

we can use [Equation 3.10](#) to calculate the probability amplitudes:

$$\begin{aligned}
 |e^{i\theta}\alpha|^2 &= (e^{i\theta}\alpha)(e^{i\theta}\alpha)^* \\
 &= (e^{i\theta}\alpha)(e^{-i\theta}\alpha^*) \\
 &= (e^{i\theta}(a_1 + ib_1))(e^{-i\theta}(a_1 - ib_1)) \\
 &= e^{i\theta}a_1 \cdot e^{-i\theta}a_1 - ie^{i\theta}a_1 \cdot e^{-i\theta}b_1 + ie^{i\theta}a_1 \cdot e^{-i\theta}b_1 - i^2e^{i\theta} \cdot e^{-i\theta}b_1^2 \\
 &= a_1^2 + b_1^2.
 \end{aligned}$$

$$\begin{aligned}
 |e^{i\theta}\beta|^2 &= (e^{i\theta}\beta)(e^{i\theta}\beta)^* \\
 &= (e^{i\theta}\beta)(e^{-i\theta}\beta^*) \\
 &= (e^{i\theta}(a_2 + ib_2))(e^{-i\theta}(a_2 - ib_2)) \\
 &= e^{i\theta}a_2 \cdot e^{-i\theta}a_2 - ie^{i\theta}a_2 \cdot e^{-i\theta}b_2 + ie^{i\theta}a_2 \cdot e^{-i\theta}b_2 - i^2e^{i\theta} \cdot e^{-i\theta}b_2^2 \\
 &= a_2^2 + b_2^2.
 \end{aligned}$$

We can see that both calculations yield the same results with or without a global phase, *i.e.*,  $a_1^2 + b_1^2$  and  $a_2^2 + b_2^2$ . On the other hand, relative phase occurs when *some* of the basis states are affected by some phase  $e^{i\theta}$ . Consider the case of  $|+\rangle$  and  $|-\rangle$ :

$$|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \qquad |-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle = \underbrace{\frac{1}{\sqrt{2}}}_{\alpha}|0\rangle + \underbrace{e^{i\pi}\frac{1}{\sqrt{2}}}_{\beta}|1\rangle,$$

in  $|-\rangle$ ,  $\beta$  is affected by  $e^{i\pi}$  whereas  $\alpha$  is not. Hence, they differ by a relative phase. Furthermore, two probability amplitudes  $\alpha$  and  $\beta$  are said to differ by a relative phase if there exists a real non-zero number  $\theta$  such that  $\alpha = e^{i\theta}$  [45, sec. 2.2.7]. It is important to note that  $|+\rangle$  and  $|-\rangle$  *do not* differ by a global phase even though calculating the probability amplitudes yield the same magnitude. This is because relative phase targets basis-state level [45, sec. 2.2.7]. Unlike global phase, relative phase can be observed physically. In [Section 4.2.2](#), we will see how relative phase in the form of a sign flip plays a significant role in quantum error correction.

### 3.5 Quantum Phenomena

*Superposition* is a non-existence classical phenomenon where a qubit can be a classical “0” or a classical “1” or in a combined “0” and “1” state (in a *superposition*). The caveat is that once the qubit is measured, it will be seen/read as a classical 0 or classical 1. As long as the qubit is not measured, the superposition state will not be disrupted.<sup>3</sup> *Entanglement* is a physical phenomenon where qubits cannot be described independently of each other.

**Definition 12.** A state  $|\psi\rangle$  is said to be entangled if it cannot be expressed as the tensor product of smaller states.

In other words, entangled qubits will always have some sort of a relationship with one another, regardless of the physical distance between them. Furthermore, manipulating one qubit will spontaneously alter the state of the other qubit. This is the essence of entanglement-assisted quantum error correction codes (EAQECC). To mathematically prove this, consider the following the state

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

We can use proof by contradiction to show it is in the entangled state and cannot be expressed as the tensor product of two single-qubits in the form of

$$|\phi_1\rangle = \alpha_1 |0\rangle + \beta_1 |1\rangle \text{ and } |\phi_2\rangle = \alpha_2 |0\rangle + \beta_2 |1\rangle.$$

Assume  $|\psi\rangle$  can be written as the tensor product of  $|\psi_1\rangle$  with  $|\psi_2\rangle$ , then

$$\begin{aligned} |\psi\rangle &= |\phi_1\rangle \otimes |\phi_2\rangle \\ &= (\alpha_1 |0\rangle + \beta_1 |1\rangle) \otimes (\alpha_2 |0\rangle + \beta_2 |1\rangle) \\ &= (\alpha_1 |0\rangle \otimes \alpha_2 |0\rangle) + (\alpha_1 |0\rangle \otimes \beta_2 |1\rangle) + (\beta_1 |1\rangle \otimes \alpha_2 |0\rangle) + (\beta_1 |1\rangle \otimes \beta_2 |1\rangle) \\ &= \alpha_1 \alpha_2 |00\rangle + \alpha_1 \beta_2 |01\rangle + \beta_1 \alpha_2 |10\rangle + \beta_1 \beta_2 |11\rangle \end{aligned}$$

Since  $|\alpha_1 \alpha_2|^2 + |\alpha_1 \beta_2|^2 + |\beta_1 \alpha_2|^2 + |\beta_1 \beta_2|^2 = 1$ ,  $\alpha_1$  or  $\beta_2$  and  $\beta_1$  or  $\alpha_2$  must be 0, which is a contradiction. On the other hand, a non-entangled state (also called separable state) is a quantum state which can be expressed as the tensor of two or more states. For example,  $|10\rangle$  is a separable state, as it is the tensor of  $|1\rangle$  with  $|0\rangle$ .

---

<sup>3</sup>By saying “not being disrupted”, we are assuming the hardware of the quantum computer and qubits are perfect and without any noise.



**Definition 13.** A quantum state  $|\psi\rangle \in \mathcal{H}_1 \otimes \mathcal{H}_2$  is separable if and only if there exists a state  $|\phi_1\rangle \in \mathcal{H}_1$  and  $|\phi_2\rangle \in \mathcal{H}_2$  such that

$$|\psi\rangle = |\phi_1\rangle \otimes |\phi_2\rangle \in \mathcal{H}_1 \otimes \mathcal{H}_2.$$

Otherwise,  $|\psi\rangle$  is entangled [51, thm. 4.2].

The *No-Cloning Theorem* is an important principle in the quantum error correction field. It simply states that we cannot clone an arbitrary unknown quantum state. A simple proof by contradiction suffices [23, pg. 7]. Suppose we have a way to clone our qubit  $|\psi\rangle$  to

$$|\psi\rangle \rightarrow |\psi\rangle \otimes |\psi\rangle,$$

then, we can also clone some other qubit  $|\phi\rangle$  to

$$|\phi\rangle \rightarrow |\phi\rangle \otimes |\phi\rangle.$$

Since the mathematical structures of quantum mechanics are based on linear spaces, the transformation of states must be linear [30, pg. 1],

$$|\psi\rangle + |\phi\rangle \rightarrow (|\psi\rangle \otimes |\psi\rangle) + (|\phi\rangle \otimes |\phi\rangle).$$

Since our definition requires

$$|\psi\rangle + |\phi\rangle \rightarrow (|\psi\rangle + |\phi\rangle) \otimes (|\psi\rangle + |\phi\rangle),$$

hence,

$$(|\psi\rangle \otimes |\psi\rangle) + (|\phi\rangle \otimes |\phi\rangle) \neq (|\psi\rangle + |\phi\rangle) \otimes (|\psi\rangle + |\phi\rangle),$$

we arrive at a contradiction.

## 3.6 Quantum Gates

This section begins by giving us an intuition of how classical gates can be represented as matrices. We will then dive deeper into single-qubit gates and finish the section with two-qubit gates. All classical gates can be represented as matrices. We can

represent the **NOT** gate as the matrix:

$$\text{NOT} = \left[ \begin{array}{cc|c} \text{Input} & & \\ \hline 0 & 1 & 0 \\ 1 & 0 & 1 \end{array} \right] \text{Output}.$$

Applying **NOT** on  $|0\rangle$  gives us

$$\text{NOT} |0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle.$$

Applying it on  $|1\rangle$  will result in  $|0\rangle$ . Moreover, we can represent the **AND** gate as

$$\text{AND} = \left[ \begin{array}{cccc|c} \text{Input} & & & & \\ \hline 00 & 01 & 10 & 11 & \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{array} \right] \text{Output}.$$

Applying **AND** on  $|00\rangle$ ,  $|10\rangle$ ,  $|10\rangle$  gives us  $|0\rangle$  while applying it on  $|11\rangle$  outputs  $|1\rangle$ . The following matrices represent the **OR**, **XOR**, **NAND** and **XOR** gates, respectively:

$$\text{OR} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}, \quad \text{XOR} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}, \quad \text{NAND} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}.$$

Similar to classical computing, quantum computing also requires logical gates to manipulate qubits. The restriction here is that a quantum logical gate must be unitary (*i.e.*,  $UU^\dagger = U^\dagger U = I$ ). It means that all quantum gates must be reversible. Classically, the **NOT** gate is the only reversible gate since we can map the obtained output to its input (*i.e.*, output = 0  $\rightarrow$  input = 1 and vice versa). On the other hand, the **OR** gate, for example, isn't reversible since receiving the output 1 implies the input could have been 01, 10, or 11. The identity gate is the simplest single-qubit gate, which doesn't change the state of the basis states. Its matrix is given by:

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.11}$$

The Pauli Gates are three quantum gates that operate on a single qubit. The first gate, denoted as  $X$ , is the equivalent of a **NOT** gate. The second and third gates are  $Y$  and  $Z$ . The three gates are associated with the following matrices:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \quad (3.12)$$

Given our qubit state  $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ , applying the Pauli Gates gives us:

$$X|\psi\rangle = X(\alpha|0\rangle + \beta|1\rangle) = \alpha|1\rangle + \beta|0\rangle \quad (3.13)$$

$$Y|\psi\rangle = Y(\alpha|0\rangle + \beta|1\rangle) = i\alpha|1\rangle - i\beta|0\rangle = i(\alpha|1\rangle - \beta|0\rangle) \quad (3.14)$$

$$Z|\psi\rangle = Z(\alpha|0\rangle + \beta|1\rangle) = \alpha|0\rangle - \beta|1\rangle. \quad (3.15)$$

We can think about  $X$  as a bit flip,  $Z$  as a phase shift and  $Y$  as a combination of both.<sup>4</sup> Moreover, here are some useful algebraic relations between the gates:

$$\begin{aligned} X^2 &= I & , & & Y^2 &= I & , & & Z^2 &= I \\ XY &= iZ & , & & YX &= -iZ & , & & YZ &= iX \\ ZY &= -iX & , & & ZX &= -iY & , & & XZ &= -iY \end{aligned} \quad (3.16)$$

The Hadamard Gate,  $H$ , is the only gate that converts a quantum basis state into a superposition of  $|0\rangle$  and  $|1\rangle$ . It is given by the matrix:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \quad (3.17)$$

Given our qubit states  $|\psi\rangle = |0\rangle$  and  $|\phi\rangle = |1\rangle$ , applying the Hadamard gate gives us:

$$H|\psi\rangle = H|0\rangle = |+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \quad (3.18)$$

$$H|\phi\rangle = H|1\rangle = |-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle. \quad (3.19)$$

The last gate we will present is the  $P$  gate, which is also known as the *phase* gate. It affects the  $|1\rangle$  state by multiplying it by a factor of  $i$ . The  $P$  gate will map  $|0\rangle$ ,  $|1\rangle$

---

<sup>4</sup>We exclude the global phase factor  $i$  in the subsequent chapters of the thesis as it doesn't have any observable effect, as discussed in [Section 3.4](#).

and states in Equation 3.4 to:

$$\begin{aligned} P|0\rangle &\rightarrow |0\rangle, & P|1\rangle &\rightarrow i|1\rangle, & P|+\rangle &\rightarrow |i+\rangle, \\ P|-\rangle &\rightarrow |i-\rangle, & P|i+\rangle &\rightarrow |-\rangle, & P|i-\rangle &\rightarrow |+\rangle. \end{aligned} \quad (3.20)$$

Each gate we have discussed has its own circuit representation. Table 3.1 lists each gate with its corresponding circuit. We will now introduce a couple of two-qubit gates.

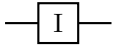



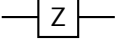
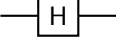

Gate Name	Circuit Representation
I	
X	 or 
Y	
Z	
H	
P	

Table 3.1: All single-qubit gates associated with their circuit representation

A Two-qubit gate requires two input qubits and yields two outputs. The most common gate is the **CNOT** gate. It flips the second qubit if the first (control bit) is true. In other words, it leaves the first qubit unchanged and xors the second qubit with the first. It has a circuit symbol and matrix representation, which are given below:

$$\begin{array}{lcl} Q_1 & |\psi\rangle & \text{---} \bullet \text{---} |\psi'\rangle = |\psi\rangle \\ & & | \\ Q_2 & |\phi\rangle & \text{---} \oplus \text{---} |\phi'\rangle = |\psi \oplus \phi\rangle \end{array}, \quad \text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \quad (3.21)$$

Figure 3.5: CNOT gate circuit representation

The CNOT gate can also be represented as a truth table: The second two-qubit gate is the **CZ** (controlled-Z) gate. It will act similar to a CNOT gate but will apply a Z gate to the target qubit if the control qubit is  $|1\rangle$ . The circuit representation of a CZ

Control Qubit		Target Qubit	
$ \psi\rangle$	$ \phi\rangle$	$ \psi'\rangle$	$ \phi'\rangle$
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Table 3.2: The truth table for CNOT gate

is

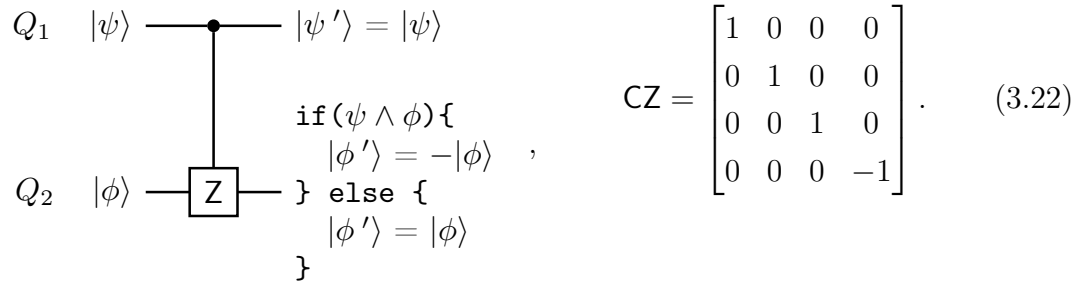


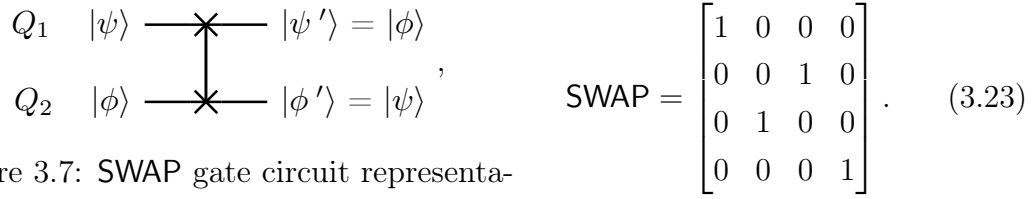
Figure 3.6: CZ gate circuit representation

The CZ gate can also be represented as a truth table, as shown in Table 3.3. The

Control Qubit		Target Qubit	
$ \psi\rangle$	$ \phi\rangle$	$ \psi'\rangle$	$ \phi'\rangle$
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	-1

Table 3.3: The truth table for CZ gate

last two-qubit gate is the **SWAP** gate. It simply swaps the two inputs given, (*i.e.*,  $|\psi\rangle \mapsto |\phi\rangle$  and  $|\phi\rangle \mapsto |\psi\rangle$ ). The circuit and matrix representation are given as:



$$\text{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (3.23)$$

Figure 3.7: SWAP gate circuit representation

The truth table for the SWAP is described in [Table 3.4](#).

$ \psi\rangle$	$ \phi\rangle$	$ \psi'\rangle$	$ \phi'\rangle$
0	0	0	0
0	1	1	0
1	0	0	1
1	1	1	1

Table 3.4: The truth table for SWAP gate

## 3.7 Introduction to Quantum Circuits

In this section, we will be introducing the basics of designing quantum circuits. We will then discuss their mathematical representation as matrices. The previous section explained how basic quantum gates manipulate qubits in the state  $|0\rangle$  or  $|1\rangle$ , making it simple to form truth tables. However, it neglected to show how multigate matrices were established and how qubits in the state of superposition are affected. Hence, we will be giving some insight on how gate matrices are formed and exploring how the gates affect qubits in superposition.

Throughout the thesis, the convention of combining qubits is from top to bottom where the topmost qubit in the circuit will be represented as the far-left digit in the ket and the bottom most qubit will be represented as the far-right digit in the ket. This is illustrated in [Figure 3.8](#).

We will be eliminating the I gates from quantum circuits as they will be assumed to be there. For example, [Figure 3.9](#) shows two equivalent circuits where the left circuit contains I and the right circuit assumes I exists without explicitly including it.

Another point to consider is the representation of a multiqubit gate. We can construct our customized multiqubit gate, which acts on some or all of the qubits in the circuit. A multigate itself is a compact form of a quantum circuit, where

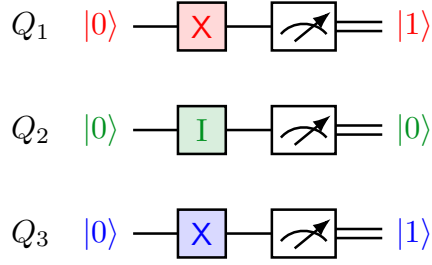


Figure 3.8: A quantum circuit that flips the first and last qubits. It starts with  $|\psi\rangle = |0\rangle \otimes |0\rangle \otimes |0\rangle = |000\rangle$  and outputs  $|101\rangle$  after applying the gates

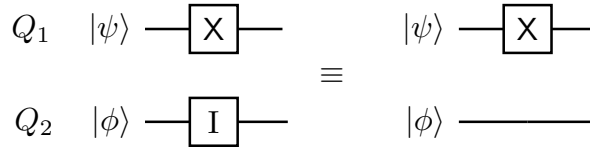


Figure 3.9: The left circuit contains the I gate for completeness whereas the right circuit doesn't because it is assumed to be implied

the number of inputs and outputs must be equal. This is due to the condition that quantum gates are reversible. An example of a gate that flips three qubits is illustrated in [Figure 3.10](#). We can now use our bit-flipper gate in other quantum circuits as

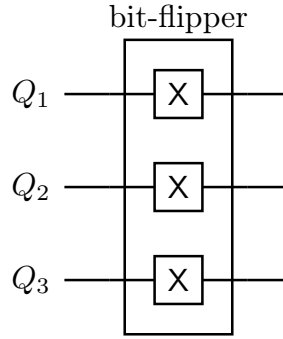


Figure 3.10: Creating a customized gate which has three inputs and three. The matrix of  $U$  is  $U = X \otimes X \otimes X$

shown in [Figure 3.11](#). Any quantum gate  $U$  applied twice consecutively will cancel itself because  $UU = I$ , which is equivalent to applying the I gate.

Furthermore, the order of applying gates is applied inside-out. For example, the circuit in [Figure 3.12](#) will (in order) apply  $X$ ,  $Y$  and lastly  $Z$  on  $Q_1$ . Mathematically, we will need to first apply  $X$  on  $Q_1$  to get  $Q'_1$ , then apply  $Y$  on  $Q'_1$  to get  $Q''_1$  and

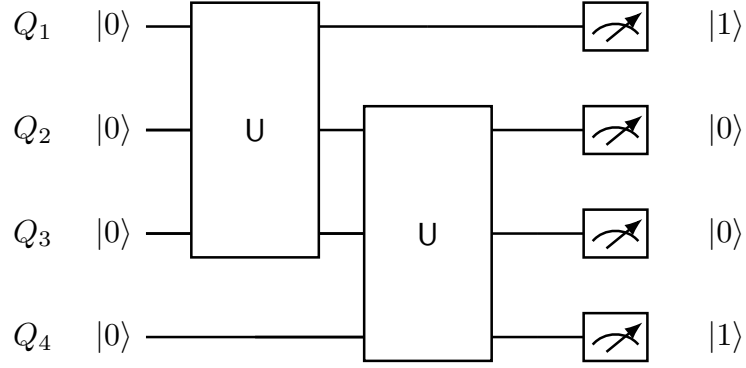
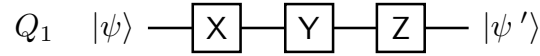
Figure 3.11: Using the customized gate  $U$  created in [Figure 3.10](#)

Figure 3.12: The order of applying gates is done following a left-to-right fashion

lastly, apply  $Z$  on  $Q_1''$ , as such:

$$\underbrace{\underbrace{(Z(Y(\underbrace{X|Q_1\rangle}_{\text{Evaluated 1st}}))}_{\text{Evaluated 2nd}})}_{\text{Evaluated 3rd}}).$$

Another point to mention is the way we can read/translate the tensor products of some operations to circuits (and vice-versa). For example, the circuit in [Figure 3.13](#) has three qubits and the operations applied on each qubit are found in [Equation 3.24](#).

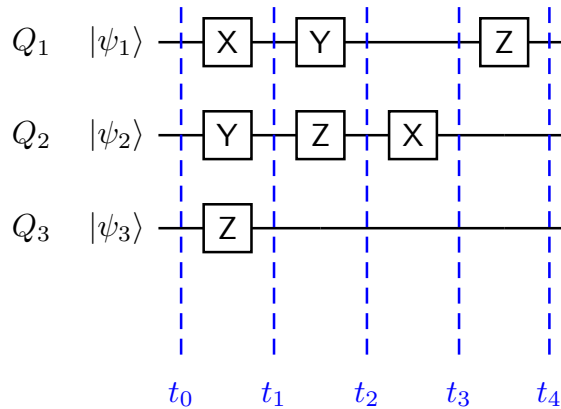


Figure 3.13: The order of applying gates on each qubit is done in a left-to-right fashion



$$\begin{array}{rcccl}
Q_1 & = & X & Y & I & Z \\
Q_2 & = & Y & Z & X & I \\
Q_3 & = & Z & I & I & I
\end{array} \quad (3.24)$$

$t_0 - t_1 \quad t_1 - t_2 \quad t_2 - t_3 \quad t_3 - t_4$

Each row in Equation 3.24 shows the gates applied on each individual qubit, where the columns show the unitary gate that is applied on all qubits at the specific snapshot. For example,  $Q_1$  in Figure 3.13 will have X, Y, I and Z (in order) applied on it whereas from  $t_0$  to  $t_1$ , we can represent the matrix applied on all the qubits as  $X \otimes Y \otimes Z$ .

Throughout this chapter, we used the control element (  $\text{---}\bullet\text{---}$  ) without explicitly explaining what it is. A control element will simply execute its associated gate if the wire (input) is  $|1\rangle$  or a superposition of  $|1\rangle$ . Another element to discuss is the anticontrol (  $\text{---}\circ\text{---}$  ). An anticontrol element will execute its associated gate if the input wire (input) is  $|0\rangle$  or a superposition of  $|0\rangle$ .<sup>5</sup> The anti-CNOT gate is the equivalent of leaving the anticontrol qubit unchanged and set the target qubit to the XNOR of both qubits. The circuit is shown in Figure 3.14. The truth table of the

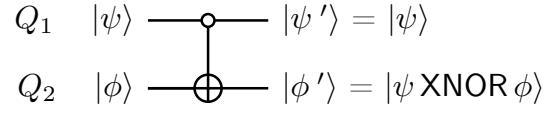


Figure 3.14:  $Q_1$  (the anticontrol qubit) will perform an XNOR operation on  $Q_2$  (the target qubit) with  $Q_1$  if  $Q_1$  is  $|0\rangle$  or a superposition of  $|0\rangle$

anti-CNOT is given in Table 3.5.

$ \psi\rangle$	$ \phi\rangle$	$ \psi'\rangle$	$ \phi'\rangle$
0	0	0	1
0	1	0	0
1	0	1	0
1	1	1	1

Table 3.5: The truth table for anti-CNOT

We know that, for example, applying CNOT on  $|10\rangle$  gives  $|11\rangle$ . The logic might not be clear when we are dealing with qubits in superposition as we have only discussed qubits being only  $|0\rangle$  or only  $|1\rangle$ . Consider the circuit in Figure 3.15. The first qubit

<sup>5</sup>For both control and anti control elements, the classical wire only works for qubits that are  $|0\rangle$  only or  $|1\rangle$  only and not in superposition. A quantum wire will work with  $|0\rangle$ ,  $|1\rangle$  or superposition of both.

will be in a superposition state after the **H** gate gets applied. Let us explain the steps of how the **CNOT** gate will be applied.

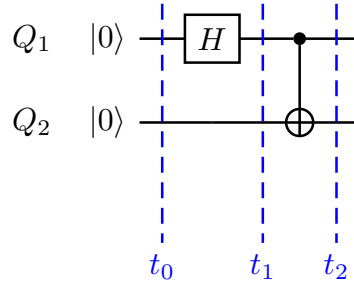


Figure 3.15: A qubit in superposition acting as the control qubit in a **CNOT** gate

1. We start with  $|00\rangle$ .
2.  $Q_1$  gets mapped to  $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$  and  $Q_2$  is left untouched by the following unitary matrix:

$$(\mathbf{H} \otimes \mathbf{I})|00\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|10\rangle.$$

3. We need to perform **CNOT** operation where  $Q_1$  is the control qubit and is in superposition. It is important to note that  $Q_1$  is in the  $|1\rangle$  state only 50% of the time. In the case where a flip happens, only 50% of a flip will occur. We can apply **CNOT** operation on  $|\psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|10\rangle$  to obtain:

$$\mathbf{CNOT}|\psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle.$$

Generally, when we have the control qubit in superposition, we will obtain the following states when using the **CNOT** gate:

$$\begin{aligned} |00\rangle &\rightarrow \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle \\ |01\rangle &\rightarrow \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|10\rangle \\ |10\rangle &\rightarrow \frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|11\rangle \\ |11\rangle &\rightarrow \frac{1}{\sqrt{2}}|01\rangle - \frac{1}{\sqrt{2}}|10\rangle. \end{aligned} \tag{3.25}$$

On the other hand, the **CZ** gate will affect the phase (sign) of the target qubit if the

control qubit is  $|1\rangle$  or superposition of  $|1\rangle$  and will leave  $|0\rangle$  unchanged. More precisely, the control qubit produces a phase flip on the target qubit if the control qubit is  $|1\rangle$  or a superposition of  $|1\rangle$ , as shown [Figure 3.6](#). There will be no row swaps in the matrix but a sign change. In other words, the affected rows in the column representation of the vector will be multiplied by  $-1$ . We can rewrite [Table 3.3](#) as [Table 3.6](#). The CZ

$ \psi\phi\rangle$	$\rightarrow$	$ \psi'\phi'\rangle$
$ 00\rangle$	$\rightarrow$	$ 00\rangle$
$ 01\rangle$	$\rightarrow$	$ 01\rangle$
$ 10\rangle$	$\rightarrow$	$ 10\rangle$
$ 11\rangle$	$\rightarrow$	$- 11\rangle$

Table 3.6: An equivalent representation of the truth table for CZ gate

gate will keep  $|00\rangle$ ,  $|01\rangle$  and  $|10\rangle$  unchanged but will map  $|11\rangle$  to  $-|11\rangle$  and vice-versa. When  $Q_1$  is in superposition, a similar approach will occur as the one encountered earlier in [Figure 3.15](#). We will discuss the procedure found in [Figure 3.16](#).

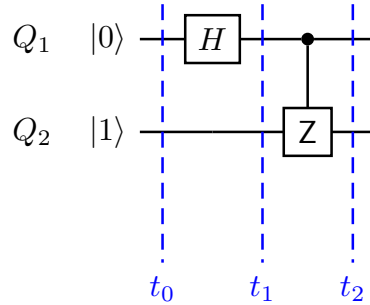


Figure 3.16: A qubit in superposition acting as the control qubit in a CZ gate

1. We start with  $|10\rangle$ .
2.  $Q_1$  gets mapped to  $\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$  and  $Q_2$  is left untouched by the following unitary matrix:

$$(H \otimes I) |01\rangle = \frac{1}{\sqrt{2}} |01\rangle + \frac{1}{\sqrt{2}} |11\rangle.$$

3. We need to perform CZ operation where  $Q_1$  is the control qubit and is in superposition. It is important to note that  $Q_1$  is in the  $|1\rangle$  state only 50% of the time. In the case where a phase-flip happens, only 50% of a phase-flip will

occur. We can apply CZ operation on  $\frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|11\rangle$  to obtain:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{1}{\sqrt{2}}|01\rangle - \frac{1}{\sqrt{2}}|11\rangle.$$

Generally, when we have the control qubit in superposition, we will obtain these states when using the CZ gate:

$$\begin{aligned} |00\rangle &\rightarrow \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|01\rangle \\ |01\rangle &\rightarrow \frac{1}{\sqrt{2}}|01\rangle - \frac{1}{\sqrt{2}}|11\rangle \\ |10\rangle &\rightarrow \frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|10\rangle \\ |11\rangle &\rightarrow \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|11\rangle. \end{aligned} \tag{3.26}$$

We can further expand our logic of using CZ gate, which contains one control qubit and two target qubits as shown in [Figure 3.17](#).

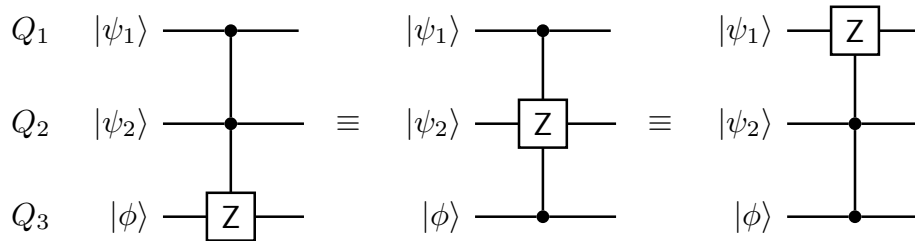


Figure 3.17: Equivalent CZ circuits with two control qubits and one target qubit

In [Figure 3.17](#), the control qubits are  $Q_1$  and  $Q_2$  in the first circuit,  $Q_1$  and  $Q_3$  in the second circuit gate and  $Q_2$  and  $Q_3$  in the third circuit. They are all equivalent and yield the same matrix representation. The first circuit requires that  $Q_1$  and  $Q_2$  to be  $|1\rangle$  or superposition of  $|1\rangle$ .  $Q_3$  must also be  $|1\rangle$  or a superposition to be affected as the  $Z$  gate doesn't affect  $|0\rangle$ . Hence, all of  $Q_1$ ,  $Q_2$  and  $Q_3$  must be  $|1\rangle$  or superposition of  $|1\rangle$ . The same explanation is valid to justify the two remaining circuits. The conclusion is that only  $|111\rangle$  will be mapped to  $-|111\rangle$ , whereas the other states are left unchanged.

Lastly, the **SWAP** gate represents a compact circuit consisting of three **CNOT**s as

shown in [Figure 3.18](#).

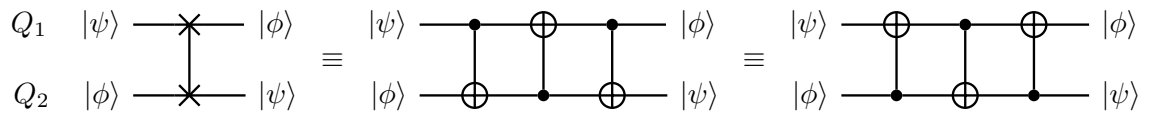


Figure 3.18: The SWAP gate represented as three CNOT gates

## Chapter 4

# Introduction to Quantum Error Correction

This chapter will discuss how errors affect quantum states. We will first see how we can represent any error as a combination of the Pauli gates. Then, we introduce the three-bit quantum repetition code we discuss and how to detect as well as correct a single bit-flip. Moreover, we will show a similar technique to detect and correct a single phase-flip as well. Lastly, we will introduce Shor's code, which can correct a bit-flip and/or a phase-flip.

### 4.1 Digitalizing Environment Noise on a Qubit

Qubits are prone to the noise of the surrounding environment. Furthermore, errors affecting qubits are continuous (*i.e.*, qubits do not *directly* encounter a bit-flip or phase-flips). Qubits would instead experience some angular shift by some arbitrary angle [17, sec. 2.2]. It is possible to prove that any continuous evolution of a single qubit coupled throughout time with the environment can be discretely represented as a linear combination of X, Y and Z operators. This is a requirement for quantum error correction theory [01, sec. 4.1]. The angular shift may occur due to inaccurate control over the qubits or by the interaction of the qubits with the environment. An *environment* is defined as everything other than the physical premise of the qubit. A *quantum channel* describes how qubits in a given setting are affected by their environment [31, sec. 10.4]. Over some time, a qubit  $|\psi\rangle = |0\rangle$  can evolve due to its interaction with an environment in the state  $|E\rangle$ , which leads to a superposition state

in the form of [59, sec 2.2.2]:

$$|0\rangle \otimes |E\rangle \rightarrow \gamma_1 (|0\rangle \otimes |E_1\rangle) + \gamma_2 (|1\rangle \otimes |E_2\rangle),$$

where  $\gamma_1$  is the probability that the qubit will remain in the basis state  $|0\rangle$  and some state  $|E_1\rangle$ , whereas  $\gamma_2$  is the probability that the qubit will remain in the basis state  $|1\rangle$  and some state  $|E_2\rangle$ . This generic definition also applies for the qubit  $|\psi\rangle = |1\rangle$  and its interaction with an environment in the state  $|E\rangle$ :

$$|1\rangle \otimes |E\rangle \rightarrow \gamma_3 (|0\rangle \otimes |E_3\rangle) + \gamma_4 (|1\rangle \otimes |E_4\rangle).$$

In general, a qubit  $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$  will interact with the environment  $|E\rangle$  yielding [59, sec 2.2.2]:

$$\begin{aligned} |\psi\rangle \otimes |E\rangle &\rightarrow (\alpha |0\rangle + \beta |1\rangle) \otimes |E\rangle \\ &\rightarrow \alpha \left( \gamma_1 (|0\rangle \otimes |E_1\rangle) + \gamma_2 (|1\rangle \otimes |E_2\rangle) \right) + \beta \left( \gamma_3 (|0\rangle \otimes |E_3\rangle) + \gamma_4 (|1\rangle \otimes |E_4\rangle) \right) \\ &\rightarrow \alpha \gamma_1 (|0\rangle \otimes |E_1\rangle) + \alpha \gamma_2 (|1\rangle \otimes |E_2\rangle) + \beta \gamma_3 (|0\rangle \otimes |E_3\rangle) + \beta \gamma_4 (|1\rangle \otimes |E_4\rangle). \end{aligned} \tag{4.1}$$

Let's see how the Pauli gates will affect an arbitrary qubit  $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$  (which doesn't contain any errors) [59, sec 2.2.2]:

$$\begin{aligned} \mathbf{I} |\psi\rangle &= \alpha |0\rangle + \beta |1\rangle \\ \mathbf{X} |\psi\rangle &= \alpha |1\rangle + \beta |0\rangle \\ \mathbf{Y} |\psi\rangle &= \alpha |1\rangle - \beta |0\rangle \\ \mathbf{Z} |\psi\rangle &= \alpha |0\rangle - \beta |1\rangle. \end{aligned}$$

We can generate a set of orthonormal states,  $B$ , that can represent the result of the Pauli gates when acted on  $|\psi\rangle$ , *i.e.*,

$$B = \{ \alpha |0\rangle + \beta |1\rangle, \alpha |1\rangle + \beta |0\rangle, \alpha |1\rangle - \beta |0\rangle, \alpha |0\rangle - \beta |1\rangle \}.$$

With the usage of the elements in set  $B$ , Equation 4.1 can be rewritten as [31, sec. 10.4]:

$$\begin{aligned} & \frac{1}{2}[(\alpha|0\rangle + \beta|1\rangle) \times (\gamma_1|E_1\rangle + \gamma_3|E_3\rangle) + \\ & (\alpha|0\rangle - \beta|1\rangle) \times (\gamma_1|E_1\rangle - \gamma_3|E_3\rangle) + \\ & (\alpha|1\rangle + \beta|0\rangle) \times (\gamma_2|E_2\rangle + \gamma_4|E_4\rangle) + \\ & (\alpha|1\rangle - \beta|0\rangle) \times (\gamma_2|E_2\rangle - \gamma_4|E_4\rangle)]. \end{aligned} \quad (4.2)$$

Furthermore, Equation 4.2 can be written using the Pauli gates:

$$\begin{aligned} & \frac{1}{2}[(I|\psi\rangle) \times (\gamma_1|E_1\rangle + \gamma_3|E_3\rangle) + \\ & (Z|\psi\rangle) \times (\gamma_1|E_1\rangle - \gamma_3|E_3\rangle) + \\ & (X|\psi\rangle) \times (\gamma_2|E_2\rangle + \gamma_4|E_4\rangle) + \\ & (Y|\psi\rangle) \times (\gamma_2|E_2\rangle - \gamma_4|E_4\rangle)]. \end{aligned} \quad (4.3)$$

Hence, the interaction between a qubit and an environment can be written as a linear combination of Pauli gates, as shown in Equation 4.3.

## 4.2 Three-Qubit Quantum Repetition Code

We will now discuss how the quantum repetition code is derived. First, we will discuss how to correct a bit-flip. Secondly, we will examine the process of correcting a phase-flip. Lastly, we will introduce Shor's Code, which corrects a single arbitrary error.

### 4.2.1 Bit-flip Correction Code

A quantum circuit can be constructed to represent the correction of a single bit-flip at most. First, we will explain the process behind the creation of such circuit. Assume that  $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$  needs to be sent over an error prone channel, which may cause the qubit to experience one bit-flip at most. In the encoding stage, we transform  $|\psi\rangle$  as such [59, sec 2.2.3]:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \xrightarrow{\text{encoded}} \overline{|\psi\rangle} = \alpha|000\rangle + \beta|111\rangle. \quad (4.4)$$

In the case where no error occurs,  $\overline{|\psi\rangle}$  will stay unchanged. If an error occurs on the first qubit,  $\overline{|\psi\rangle}$  will be transformed to  $\alpha|100\rangle + \beta|011\rangle$ . If an error occurs on the second



qubit,  $\overline{|\psi\rangle}$  becomes  $\alpha |0\mathbf{1}0\rangle + \beta |1\mathbf{0}1\rangle$ . Lastly, if an error occurs on the third qubit,  $\overline{|\psi\rangle}$  becomes  $\alpha |00\mathbf{1}\rangle + \beta |11\mathbf{0}\rangle$ . The circuit will have five stages: encoding, transmitting data through erroneous channel, syndrome extraction, syndrome correction and decoding. The quantum circuit in Figure 4.1 can be analyzed further using the timestamps

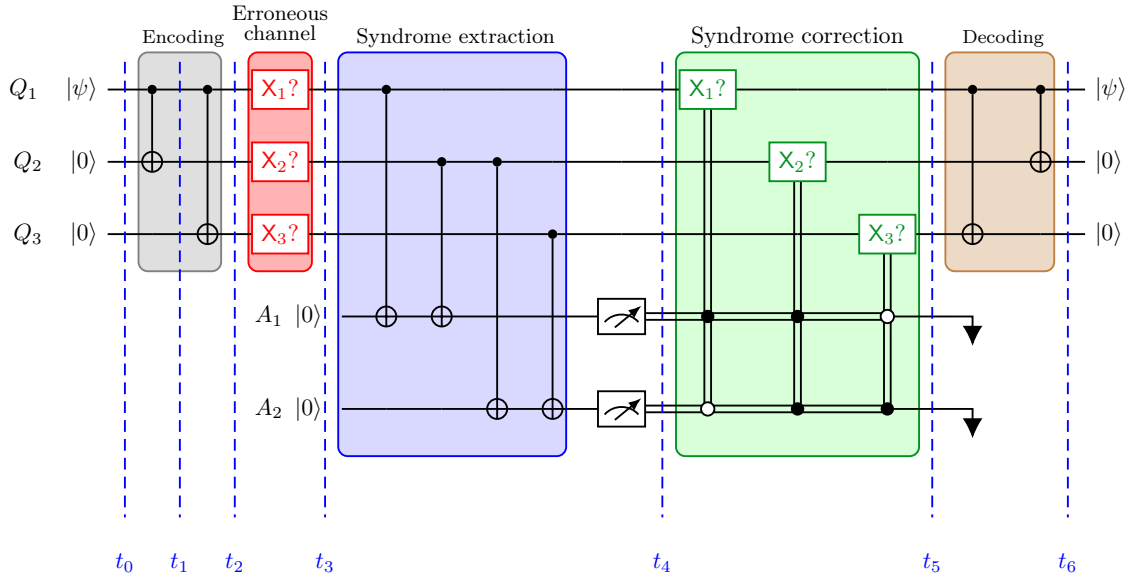


Figure 4.1: Repetition code single bit-flip correction circuit using CNOT gates to find the syndromes

$t_0, \dots, t_6$ .

1. We start the encoding process by preparing three qubits.  $Q_1$  is in superposition, and both  $Q_2$  and  $Q_3$  are zeros. We transform  $|\psi\rangle$  to  $\overline{|\psi\rangle}$ :

$$\begin{aligned}
 |\psi\rangle &\rightarrow \overline{|\psi\rangle} = |\psi\rangle \otimes |0\rangle \otimes |0\rangle \\
 &= (\alpha |0\rangle + \beta |1\rangle) \otimes |0\rangle \otimes |0\rangle \\
 &= (\alpha |00\rangle + \beta |10\rangle) \otimes |0\rangle \\
 &= \alpha |000\rangle + \beta |100\rangle.
 \end{aligned}$$

It is important to keep in mind that the wire of  $Q_1$  will manipulate the left-most digit in the kets (*i.e.*,  $\alpha |0\mathbf{0}0\rangle + \beta |1\mathbf{0}0\rangle$ ). Similarly,  $Q_2$  and  $Q_3$  will manipulate the middle digit ( $\alpha |0\mathbf{0}0\rangle + \beta |1\mathbf{0}0\rangle$ ) and right-most digit ( $\alpha |00\mathbf{0}\rangle + \beta |10\mathbf{0}\rangle$ ) in the kets, respectively.

2. We applied  $U_1 = \text{CNOT} \otimes I$  on  $\overline{|\psi\rangle}$ .

$$U_1 \overline{|\psi\rangle} = \alpha |000\rangle + \beta |110\rangle.$$

3. We applied  $U_2 = I \otimes \text{CNOT}$  on  $\overline{|\psi\rangle}$ .

$$U_2 \overline{|\psi\rangle} = \alpha |000\rangle + \beta |111\rangle.$$

The encoding procedure is complete and the qubit  $\overline{|\psi\rangle}$  can be send over.

4. Our qubit was sent through an erroneous channel where a single bit-flip (at most) can occur. In the case where a bit-flip error occurred randomly and independently from each qubit. The qubit can be received as one of [59, sec 2.2.3]:

$$\alpha |000\rangle + \beta |111\rangle \tag{4.5}$$

$$\alpha |100\rangle + \beta |011\rangle \tag{4.6}$$

$$\alpha |010\rangle + \beta |101\rangle \tag{4.7}$$

$$\alpha |001\rangle + \beta |110\rangle \tag{4.8}$$

$$\alpha |110\rangle + \beta |001\rangle \tag{4.9}$$

$$\alpha |101\rangle + \beta |010\rangle \tag{4.10}$$

$$\alpha |011\rangle + \beta |100\rangle \tag{4.11}$$

$$\alpha |111\rangle + \beta |000\rangle \tag{4.12}$$

Only four states can be corrected, namely, Equations (4.5) to (4.8) as we can correct one error at most.

5. We start with the syndrome extraction procedure. Two ancilla (temporary) qubits  $A_1$  and  $A_2$  were prepared as  $A_1 \otimes A_2 = |0\rangle \otimes |0\rangle$  for detecting and correcting the error. The four **CNOT** operation will act on the ancilla qubits to obtain the syndrome. We can think of these **CNOT** operations as an operator  $S$ . Given

$$\overline{|\psi\rangle} = \alpha |x_1 x_2 x_3\rangle + \beta |\overline{x_1} \overline{x_2} \overline{x_3}\rangle, \text{ where } \overline{x_i} = x_i + 1 \pmod{2},$$

we can define the operator  $S$  as:

$$S : \alpha |x_1 x_2 x_3\rangle + \beta |\overline{x_1} \overline{x_2} \overline{x_3}\rangle \rightarrow \underbrace{|x_1 \oplus x_2\rangle}_{A_1} \otimes \underbrace{|x_2 \oplus x_3\rangle}_{A_2},$$

or equivalently as:

$$S : \alpha |x_1 x_2 x_3\rangle + \beta |\overline{x_1} \overline{x_2} \overline{x_3}\rangle \rightarrow \underbrace{|\overline{x_1} \oplus \overline{x_2}\rangle}_{A_1} \otimes \underbrace{|\overline{x_2} \oplus \overline{x_3}\rangle}_{A_2}.$$

The syndromes are listed in Table 4.1 with the operation needed to be correct the error.

Syndrome ( $ A_1\rangle \otimes  A_2\rangle$ )	Error	Operation
$ 0\rangle \otimes  0\rangle$	No error encountered	Nothing
$ 0\rangle \otimes  1\rangle$	Bit-flip on third qubit	Apply $X_3$
$ 1\rangle \otimes  0\rangle$	Bit-flip on first qubit	Apply $X_1$
$ 1\rangle \otimes  1\rangle$	Bit-flip on second qubit	Apply $X_2$

Table 4.1: Showing possible syndromes and with the error caused on the  $|\psi\rangle$  and the operation needed to correct it

6. We have now entered the recovery stage. We can use the error syndrome to correct an error occurred using *only* one of  $X_1$ ?,  $X_2$ ? or  $X_3$ ?. The operators are given as:

$$U_{S_{00}} = I \otimes I \otimes I$$

$$U_{S_{01}} = I \otimes I \otimes X$$

$$U_{S_{10}} = X \otimes I \otimes I$$

$$U_{S_{11}} = I \otimes X \otimes I,$$

where  $U_{S_{ij}}$  corresponds to the syndrome  $ij$ . Applying the appropriate  $U_{S_{ij}}$  on  $|\psi\rangle$  will correct the error. For example, if  $i = 1$  and  $j = 0$ , then  $S$  yields  $|1\rangle \otimes |0\rangle$ , hence,  $X_1$  is applied.

7. After the correction, we should have  $\overline{|\psi\rangle} = \alpha |000\rangle + \beta |111\rangle$ . Our goal is to have  $\overline{|\psi\rangle} = \alpha |000\rangle + \beta |100\rangle$  which is achieved by applying  $U_1(U_2\overline{|\psi\rangle})$ , *i.e.*, first

applying  $U_2$  on  $\overline{|\psi\rangle}$  resulting

$$\overline{|\psi\rangle} = \alpha |000\rangle + \beta |110\rangle,$$

then, applying  $U_1$  on  $\overline{|\psi\rangle}$  resulting

$$\overline{|\psi\rangle} = \alpha |000\rangle + \beta |100\rangle,$$

or equivalently as

$$\overline{|\psi\rangle} = |\psi\rangle \otimes |0\rangle \otimes |0\rangle.$$

### 4.2.2 Phase-flip Correction Code

The phase-flip, on the other hand, is dealt with differently. This is because a phase flip changes the sign rather than the bit itself. More specifically, a phase-flip will only affect  $|1\rangle$ . Applying  $Z$  on  $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$  gives:

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

We are more interested in the sign of the qubit rather than the bit value (0 or 1). We can fundamentally change the basis states of  $\{|0\rangle, |1\rangle\}$  to something more suitable. Consider the Hadamard gate, which transform the basis states of  $|\psi\rangle$  (*i.e.*,  $|0\rangle$  and  $|1\rangle$ ) to

$$H |0\rangle = |+\rangle$$

$$H |1\rangle = |-\rangle.$$

The relationship between  $|+\rangle$  and  $|-\rangle$  can be seen using the  $Z$  operator:

$$Z |+\rangle = |-\rangle$$

$$Z |-\rangle = |+\rangle.$$

Hence, it is more suitable to use  $\{|+\rangle, |-\rangle\}$  as the basis when correcting phase flips.

Furthermore, it is important to see how a phase-flip can be turned into a bit flip using the Hadamard gate. Let's say we have our qubit as  $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$ . The first step is to write the state in terms of the basis  $\{|+\rangle, |-\rangle\}$ . We can apply the

Hadamard gate on our qubit  $|\psi\rangle$ :

$$H|\psi\rangle = |\overline{\psi}\rangle = \frac{\alpha}{\sqrt{2}}|0\rangle + \frac{\beta}{\sqrt{2}}|1\rangle. \quad (4.13)$$

Assume the qubit encountered a phase-flip, *i.e.*, the Z gate was applied on  $|\overline{\psi}\rangle$ :

$$Z|\overline{\psi}\rangle = \frac{\alpha}{\sqrt{2}}|0\rangle - \frac{\beta}{\sqrt{2}}|1\rangle. \quad (4.14)$$

Note that Equation 4.13 and Equation 4.14 are *almost* the states  $|+\rangle$  and  $|-\rangle$ , respectively, but differ in the amplitudes. The next step is to apply the Hadamard gate on  $|\overline{\psi}\rangle$ :

$$H|\overline{\psi}\rangle = \beta|0\rangle + \alpha|1\rangle$$

We can see that our qubit is almost the same as what we started with except that it now has a bit-flip applied on it. The last step is to correct the bit-flip by applying X on  $|\overline{\psi}\rangle$  to obtain our original qubit  $|\psi\rangle$ . So far, we have explained the process of a single qubit encountering a phase-flip. We can extend our approach to using three qubits. Moreover, we are able to use the decoding logic given in Figure 4.1 to correct what started as a phase-flip then turned into a bit flip. Assume that the qubit  $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$  needs to be sent over an erroneous channel, where one phase flip (at most) can occur. We transform  $|\psi\rangle$  to  $\alpha|000\rangle + \beta|111\rangle$  and then perform three Hadamard operations:

$$\begin{aligned} |\psi\rangle &\rightarrow |\overline{\psi}\rangle = (H \otimes H \otimes H)|\psi\rangle \\ &= \alpha'|++\rangle + \beta'|---\rangle, \end{aligned}$$

where  $\alpha' = \frac{\alpha}{2\sqrt{2}}$  and  $\beta' = \frac{\beta}{2\sqrt{2}}$ . The circuit in Figure 4.2 can be further explained with the timestamps  $t_0, \dots, t_4$ .

1. Similar to the bit-flip encoding circuit, we start the encoding process by preparing three qubits.  $Q_1$  is in superposition, and both  $Q_2$  and  $Q_3$  are zeros. We start with  $|\psi\rangle$  and encode the state by performing CNOT operations to obtain

$$|\overline{\psi}\rangle = \alpha|000\rangle + \beta|111\rangle.$$

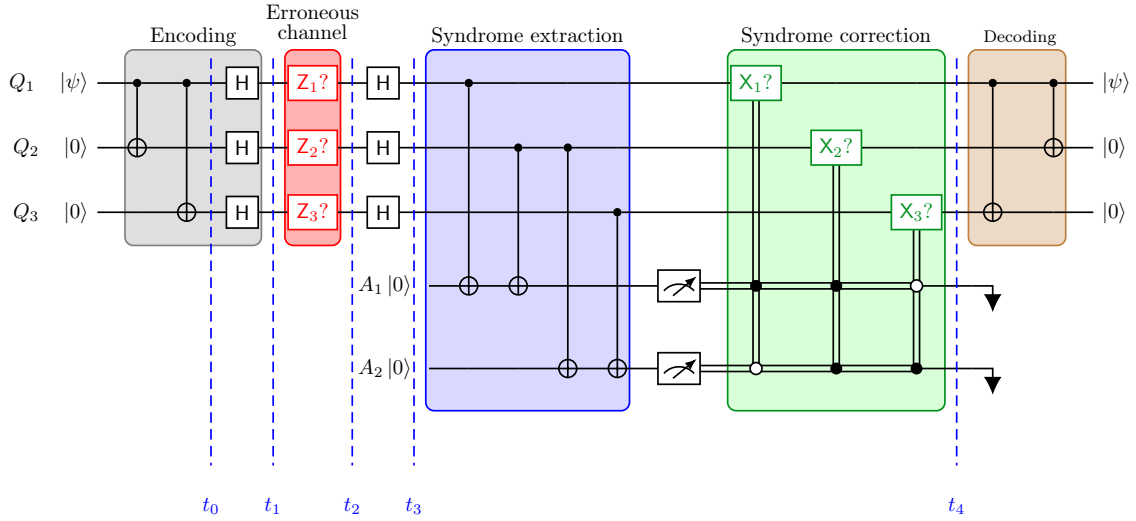


Figure 4.2: The phase-flip circuit correction for the three-qubit repetition code using CNOT gates for syndrome correction

2. We applied  $U_1 = H \otimes H \otimes H$  on  $|\overline{\psi}\rangle$  to obtain [59, sec 2.2.4]

$$\frac{\alpha}{2\sqrt{2}} |+++ \rangle + \frac{\beta}{2\sqrt{2}} |-- - \rangle.$$

3. Our qubit was sent through an erroneous channel where a single phase flip (at most) can occur. In the case where a phase-flip error occurred, it is assumed to change the basis (+/- signs in the kets) independently from each qubit. The qubit can be received as one of:

$$\alpha |+++ \rangle + \beta |-- - \rangle \quad (4.15)$$

$$\alpha |-++ \rangle + \beta |+- - \rangle \quad (4.16)$$

$$\alpha |+- + \rangle + \beta |-- + - \rangle \quad (4.17)$$

$$\alpha |++ - \rangle + \beta |-- - + \rangle \quad (4.18)$$

$$\alpha |-- + \rangle + \beta |++ - \rangle \quad (4.19)$$

$$\alpha |-+ - \rangle + \beta |+- + \rangle \quad (4.20)$$

$$\alpha |+- - \rangle + \beta |-- + + \rangle \quad (4.21)$$

$$\alpha |-- - - \rangle + \beta |++ + \rangle \quad (4.22)$$

Note that Equations (4.15) to (4.18) are the only states that can be corrected.

4. Applying the Hadamard gate on each of the three qubits will cancel the first Hadamard gate applied in  $t_2$  if a phase-flip did not occur. Rather, it will transform the error that occurred to a bit-flip, as explained earlier.
5. After extracting the syndrome, we are able to correct the bit-flip. This step is equivalent to the syndrome extraction process that was seen when correcting the bit-flip error in Figure 4.1.

In Figure 4.1 and Figure 4.2, the syndrome extraction component treated  $\text{---}\bigcirc\text{---}$  as a “0” and  $\text{---}\bullet\text{---}$  as a “1” to which the syndrome was derived from. Another useful approach is to use CZ gates in the syndrome extraction segment instead of CNOTs. The syndrome produced and operations required to correct the phase flip error are the same in Table 4.1. In addition to using CZ gates, the ancilla qubits will have H as the first and last gate in order to detect if a bit-flip occurred (as discussed in the beginning of Section 4.2.2). The complete circuit is shown in Figure 4.3.

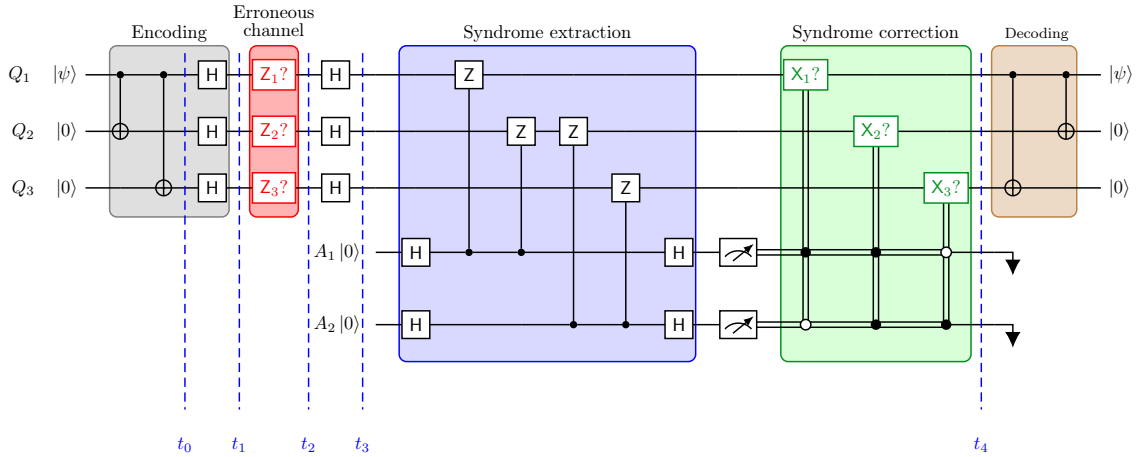


Figure 4.3: The phase-flip circuit correction for the three-qubit repetition code using CZ gates for syndrome correction

To further explain the syndrome segment in Figure 4.3, we will focus on  $Q_1$ ,  $Q_2$  and  $A_1$ . The two CZ gates will not affect  $Q_1$  and  $Q_2$  if both qubits are in: the  $|0\rangle$  state (due to the nature of the gate), the state of  $|1\rangle$  or a superposition of  $|1\rangle$  (as applying two CZ gates on the state of  $|1\rangle$  or a superposition of  $|1\rangle$  will cancel one another). However, if  $Q_1$  differs from  $Q_2$ , then the CZ gates will not cancel each other and will result in  $|A_1\rangle$  to flip to  $|1\rangle$ . This syndrome circuit can be replaced with the syndrome circuit in Figure 4.1 and still yield the same result. The complete circuit is found in Figure A.1.1 under Section A.1.

### 4.2.3 Shor's Code

Shor's Code is a quantum error correction code that encodes one logical qubit in nine physical qubits to correct a single error  $\mathcal{E} \in \{X, Y, Z\}$  at most. This code concatenates the three-qubit repetition code three times. The states  $|0\rangle$  and  $|1\rangle$  are encoded as:

$$\begin{aligned} |0\rangle &\rightarrow \overline{|0\rangle} = \left(\frac{1}{\sqrt{2}}\right)^3 ((|000\rangle + |111\rangle)^{\otimes 3}) \\ |1\rangle &\rightarrow \overline{|1\rangle} = \left(\frac{1}{\sqrt{2}}\right)^3 ((|000\rangle - |111\rangle)^{\otimes 3}), \end{aligned} \tag{4.23}$$

which is equivalent to

$$\begin{aligned} |0\rangle &\rightarrow \overline{|0\rangle} = \frac{1}{2\sqrt{2}} ((|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle)) \\ &= \frac{1}{2\sqrt{2}} (|000000000\rangle + |000000111\rangle + |000111000\rangle + |000111111\rangle \\ &\quad + |111000000\rangle + |111000111\rangle + |111111000\rangle + |111111111\rangle) \\ |1\rangle &\rightarrow \overline{|1\rangle} = \frac{1}{2\sqrt{2}} ((|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle)), \\ &= \frac{1}{2\sqrt{2}} (|000000000\rangle - |000000111\rangle - |000111000\rangle + |000111111\rangle \\ &\quad - |111000000\rangle + |111000111\rangle + |111111000\rangle - |111111111\rangle). \end{aligned} \tag{4.24}$$

To send a qubit  $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ , we encode it as  $|\psi\rangle = \alpha\overline{|0\rangle} + \beta\overline{|1\rangle}$ . In order to correct a bit-flip error, the nine qubits are divided into three equal groups:

$$\{Q_1, Q_2, Q_3\}, \{Q_4, Q_5, Q_6\}, \{Q_7, Q_8, Q_9\},$$

where each group detects and corrects the bit-flip that may occur within the group. For example, if a bit-flip occurred on any of  $Q_1, Q_2$  or  $Q_3$ , then the first group will correct it. This is done using the same syndrome extraction circuit and syndrome correction used in [Figure 4.3](#). Similarly, correcting a bit-flip error in  $Q_4, Q_5$  or  $Q_6$  will use the same approach as well as correcting a bit-flip error in  $Q_7, Q_8$  or  $Q_9$ . The total syndromes that must be checked are:

$$\underbrace{\{Z_1Z_2, Z_2Z_3\}}_{\text{group 1}}, \underbrace{\{Z_4Z_5, Z_5Z_6\}}_{\text{group 2}}, \underbrace{\{Z_7Z_8, Z_8Z_9\}}_{\text{group 3}}.$$



On the other hand, a phase-flip error is detected by all nine qubits but corrected using  $Q_1$ ,  $Q_4$  and  $Q_7$ . Similar to a bit-flip, the nine qubits are divided into the same three groups to identify the error. Let us focus on the first group (*i.e.*,  $Q_1$ ,  $Q_2$  and  $Q_3$ ). Any of  $Z_1$ ,  $Z_2$  or  $Z_3$  errors will transform

$$\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)_{Q_1, Q_2, Q_3} \text{ to } \frac{1}{\sqrt{2}}(|000\rangle - |111\rangle)_{Q_1, Q_2, Q_3}^1,$$

Similarly, a  $Z_4$ ,  $Z_5$  or  $Z_6$  error maps

$$\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)_{Q_4, Q_5, Q_6} \text{ to } \frac{1}{\sqrt{2}}(|000\rangle - |111\rangle)_{Q_4, Q_5, Q_6}.$$

Lastly, a  $Z_7$ ,  $Z_8$  or  $Z_9$  error transforms

$$\frac{1}{\sqrt{2}}(|000\rangle - |111\rangle)_{Q_7, Q_8, Q_9} \text{ to } \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)_{Q_7, Q_8, Q_9}.$$

We can use the two following syndromes:

$$\underbrace{Q_1, Q_2, Q_3}_{\text{group 1}} \underbrace{Q_4, Q_5, Q_6}_{\text{group 2}} \text{ and } \underbrace{Q_4, Q_5, Q_6}_{\text{group 2}} \underbrace{Q_7, Q_8, Q_9}_{\text{group 3}},$$

to find which group contains the phase error. More precisely,

$$\{X_1X_2X_3X_4X_5X_6, X_4X_5X_6X_7X_8X_9\}$$

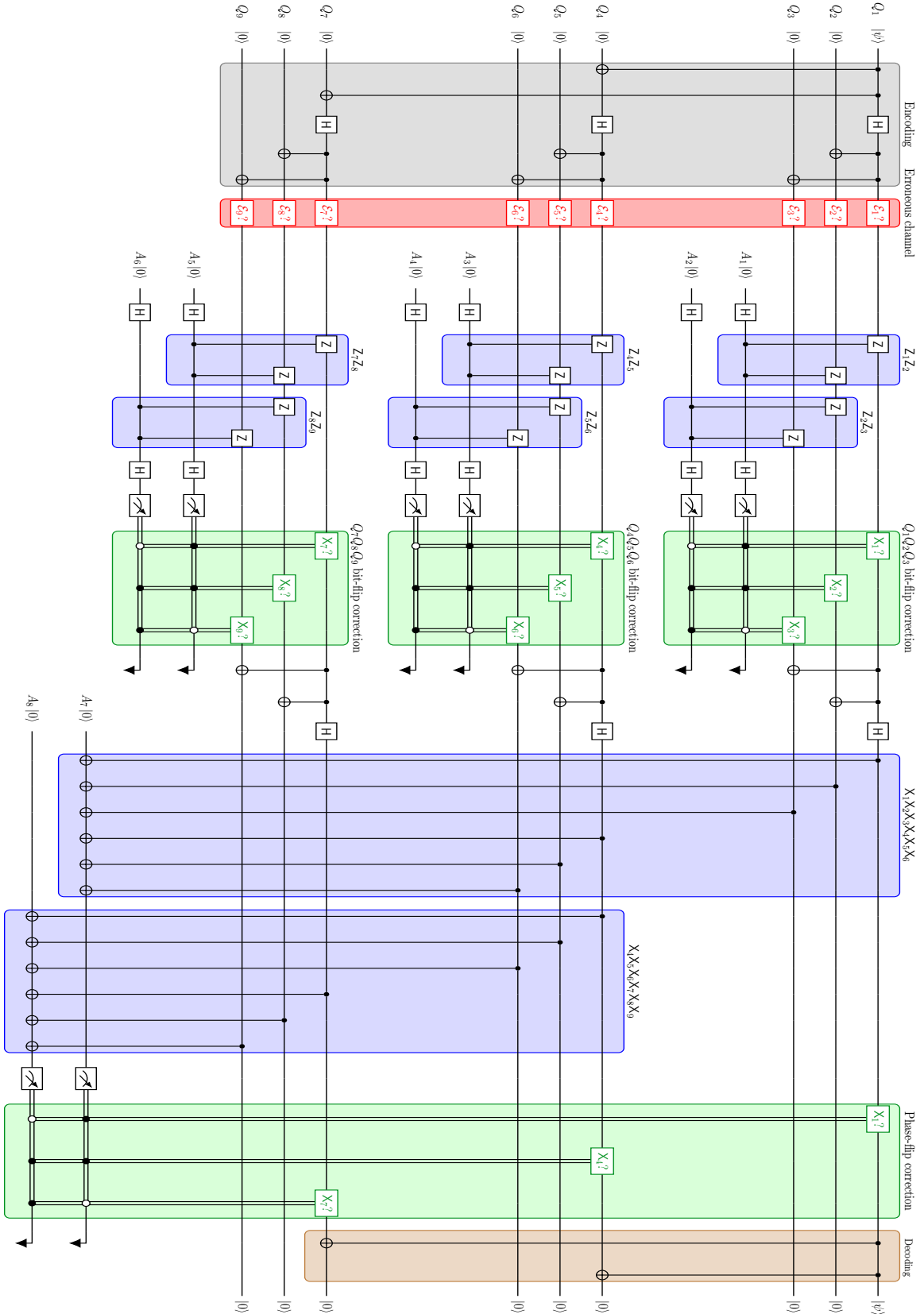
are the operators needed to extract these syndromes. For example, the syndrome  $|0\rangle \otimes |1\rangle$  implies that  $X_1X_2X_3X_4X_5X_6$  did not detect an error, while  $X_4X_5X_6X_7X_8X_9$  did, hence, the error occurred in group 3. The complete circuit for Shor code is given by Figure 4.4. Furthermore, Table 4.2 shows the output of all ancilla qubits in Figure 4.4.

---

<sup>1</sup>The subscript  $Q_1, Q_2, Q_3$  represents the qubits we are focusing on.

Ancilla Qubits	Syndrome	Error	Operation
$ A_1\rangle \otimes  A_2\rangle$	$ 0\rangle \otimes  0\rangle$	No bit-flip error in Group 1	Nothing
	$ 0\rangle \otimes  1\rangle$	Bit-flip on $Q_3$	Apply $X_3$
	$ 1\rangle \otimes  0\rangle$	Bit-flip on $Q_1$	Apply $X_1$
	$ 1\rangle \otimes  1\rangle$	Bit-flip on $Q_2$	Apply $X_2$
$ A_3\rangle \otimes  A_4\rangle$	$ 0\rangle \otimes  0\rangle$	No bit-flip error in Group 2	Nothing
	$ 0\rangle \otimes  1\rangle$	Bit-flip on $Q_6$	Apply $X_6$
	$ 1\rangle \otimes  0\rangle$	Bit-flip on $Q_4$	Apply $X_4$
	$ 1\rangle \otimes  1\rangle$	Bit-flip on $Q_5$	Apply $X_5$
$ A_5\rangle \otimes  A_6\rangle$	$ 0\rangle \otimes  0\rangle$	No bit-flip error in Group 3	Nothing
	$ 0\rangle \otimes  1\rangle$	Bit-flip on $Q_9$	Apply $X_9$
	$ 1\rangle \otimes  0\rangle$	Bit-flip on $Q_7$	Apply $X_7$
	$ 1\rangle \otimes  1\rangle$	Bit-flip on $Q_8$	Apply $X_8$
$ A_7\rangle \otimes  A_8\rangle$	$ 0\rangle \otimes  0\rangle$	No phase-flip error occurred	Nothing
	$ 0\rangle \otimes  1\rangle$	Phase-flip in $Q_6Q_7Q_8$	Apply $X_7$
	$ 1\rangle \otimes  0\rangle$	Phase-flip in $Q_1Q_2Q_3$	Apply $X_1$
	$ 1\rangle \otimes  1\rangle$	Phase-flip in $Q_4Q_5Q_6$	Apply $X_4$

Table 4.2: All possible syndromes for Shor's Code with their effect on the encoded qubit  $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$  and the operation needed to correct it

Figure 4.4: Nine-qubit Shor Code circuit which corrects at most one of  $\mathcal{E} \in \{X, Y, Z\}$

## Chapter 5

# Introduction to Stabilizer Codes

### 5.1 Repetition Bit-flip Code Revisited

In the previous chapter, we thoroughly explained how the repetition code works. The error-correcting concept depended on the qubit states themselves. Stabilizer codes, however, depend on the operator rather than the states of the qubits. We will start with informally introducing the concept of stabilizer codes using the repetition bit-flip code and then rigorously establish the formalism of stabilizer codes. Recall the encoding given in [Equation 4.4](#) is

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \rightarrow \overline{|\psi\rangle} = \alpha|000\rangle + \beta|111\rangle. \quad (5.1)$$

The syndrome extraction circuit in [Figure A.1.1](#) used the two following operators to detect a bit-flip error:

$$S = \begin{bmatrix} Z_1 Z_2 & Z_2 Z_3 \end{bmatrix}. \quad (5.2)$$

We can use these syndrome operators to detect a bit-flip error. These operators are equivalent to

$$\begin{aligned} Z_1 \otimes Z_2 \otimes I_3 &\equiv Z_1 \otimes Z_2 \equiv Z_1 Z_2, \\ I_1 \otimes Z_2 \otimes Z_3 &\equiv Z_2 \otimes Z_3 \equiv Z_2 Z_3, \end{aligned} \quad (5.3)$$

where  $U_i$  is applying the  $U$  gate on the  $i^{th}$  qubit. Expanding the unitary matrices given in Equation 5.3 and  $|\overline{\psi}\rangle$  yields

$$Z_1 Z_2 = U_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, Z_2 Z_3 = U_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, |\overline{\psi}\rangle = \begin{bmatrix} \alpha \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \beta \end{bmatrix}. \quad (5.4)$$

In the case where there are no errors, then applying both syndromes on  $|\overline{\psi}\rangle$  gives:

$$U_1 |\overline{\psi}\rangle = |\overline{\psi}\rangle, \quad U_2 |\overline{\psi}\rangle = |\overline{\psi}\rangle.$$

The possible errors that can occur are

$$\begin{aligned} X_1 \otimes I_2 \otimes I_3 &\equiv X_1, \\ I_1 \otimes X_2 \otimes I_3 &\equiv X_2, \\ I_1 \otimes I_2 \otimes X_3 &\equiv X_3. \end{aligned} \quad (5.5)$$

For the sake of argument, let us assume  $X_1$  occurred during transmission, hence

$$X_1 |\overline{\psi}\rangle = |\overline{\psi}\rangle_{X_1} = \begin{bmatrix} 0 & 0 & 0 & \beta & \alpha & 0 & 0 & 0 \end{bmatrix}^T \quad (5.6)$$

Applying  $U_1$  and  $U_2$  from Equation 5.4 on  $|\overline{\psi}\rangle_{X_1}$  yields:

$$U_1 |\overline{\psi}\rangle_{X_1} = \begin{bmatrix} 0 & 0 & 0 & -\beta & -\alpha & 0 & 0 & 0 \end{bmatrix}^T = -|\overline{\psi}\rangle_{X_1} \quad (5.7)$$

$$U_2 |\overline{\psi}\rangle_{X_1} = \begin{bmatrix} 0 & 0 & 0 & \beta & \alpha & 0 & 0 & 0 \end{bmatrix}^T = |\overline{\psi}\rangle_{X_1} \quad (5.8)$$

We can see the eigenvalues of  $U_1 |\overline{\psi}\rangle_{X_1}$  and  $U_2 |\overline{\psi}\rangle_{X_1}$  are  $-1$  and  $+1$ , respectively. We can use the eigenvalue mechanism to detect an error. We can append a new column to Table 4.1 regarding eigenvalue, as given by Table 5.1, which maps

$$|0\rangle \rightarrow +1 \quad \text{and} \quad |1\rangle \rightarrow -1. \quad (5.9)$$

It is important to note that when no errors are encountered there is an all  $+1$  syndrome vector.

Syndrome	Eigenvalue	Error	Operation
$ 0\rangle \otimes  0\rangle$	$[+1 \ +1]$	No error encountered	Nothing
$ 0\rangle \otimes  1\rangle$	$[+1 \ -1]$	Bit-flip on third qubit	Apply $X_3$ on $Q_3$
$ 1\rangle \otimes  0\rangle$	$[-1 \ +1]$	Bit-flip on first qubit	Apply $X_1$ on $Q_1$
$ 1\rangle \otimes  1\rangle$	$[-1 \ -1]$	Bit-flip on second qubit	Apply $X_2$ on $Q_2$

Table 5.1: Showing possible syndromes for quantum repetition code as eigenvalues, with the error caused on the  $|\psi\rangle$  and the operation needed to correct it

## 5.2 Stabilizer Codes

Stabilizer Codes were discovered by Daniel Gottesman in 1997. Gottesman's Ph.D. thesis [23] was dedicated to this topic which is where this chapter is inspired from. An operator that acts on a code and yields  $+1$  as the eigenvalue has no effect on the qubit state and is said to *stabilize* or *fix*<sup>1</sup> the code. For example, the two operators in Equation 5.3 stabilize the repetition bit-flip code, provided there was no error encountered. Recall that the Pauli operators in Equation 3.16 form a group under multiplication with  $\pm 1$  and  $\pm i$  as overall phases and including  $I$ . Let

$$\begin{aligned} \mathcal{P}^{\otimes 1} &= \{\pm 1, \pm i\} \times \{I, X, Y, Z\} \\ &= \{I, -I, iI, -iI, X, -X, iX, -iX, Y, -Y, iY, -iY, Z, -Z, iZ, -iZ\}, \end{aligned} \quad (5.10)$$

be a Pauli group acting on a single qubit. Let

$$\mathcal{P}^{\otimes n} = \{\pm 1, \pm i\} \times \{I, X, Y, Z\}^{\otimes n} \quad (5.11)$$

be an  $n$ -fold Pauli group of  $\mathcal{P}^{\otimes 1}$  under matrix multiplication acting on  $n$  qubits. An element  $A \in \mathcal{P}^{\otimes n}$  is of the form

$$A = cA_1 \otimes A_2 \otimes \cdots \otimes A_{n-1} \otimes A_n, \text{ where } A_i \in \{I, X, Y, Z\} \text{ and } c \in \{\pm 1, \pm i\}. \quad (5.12)$$

The cardinality of  $\mathcal{P}^{\otimes n}$  without global phases is  $4^n$  elements and with overall phases is  $4 \times 4^n = 4^{n+1}$  elements. Let us define the commutator and anti-commutator relations between elements in  $\mathcal{P}^{\otimes n}$  as follows:

$$\begin{aligned} \text{Commutator} \quad [A, B] &= +1 \quad \text{if } AB = +BA, \quad \text{for } A, B \in \mathcal{P}^{\otimes n}, \\ \text{Anti-commutator} \quad \{A, B\} &= -1 \quad \text{if } AB = -BA, \quad \text{for } A, B \in \mathcal{P}^{\otimes n}. \end{aligned} \quad (5.13)$$

<sup>1</sup>The word fix doesn't mean to correct, rather, "holds in place".

Some of the properties  $\mathcal{P}^{\otimes n}$  include:

- 1) For any  $A \in \mathcal{P}^{\otimes n}$ :
  - I) the square of  $A$  yields  $\pm I^{\otimes n}$ .
  - II)  $A$  is Hermitian or anti-Hermitian. A Hermitian matrix is  $(A^*)^T = A^\dagger = A$ , *i.e.*, taking the complex conjugate of  $A$  and transposing it yields  $A$ . An example of an anti-Hermitian element in  $\mathcal{P}^{\otimes n}$  is  $(iI^{\otimes n})^\dagger = -iI^{\otimes n}$ .
  - III)  $A$  is always a unitary matrix, *i.e.*,  $A^\dagger A = AA^\dagger = I^{\otimes n}$ .
- 2) For any  $A, B \in \mathcal{P}^{\otimes n}$ , either  $[A, B] = +1$  or  $\{A, B\} = -1$ , *i.e.*, either commute or anti-commute.

Squaring any element from  $\mathcal{P}^{\otimes n}$  yields an eigenvalue of  $+1$ . These elements are composed of  $\pm I, \pm X, \pm Y$  and  $\pm Z$ . We include the global phase  $\pm i$  in the definition of  $\mathcal{P}^{\otimes n}$  but don't use it in our formalism as it makes  $\mathcal{P}^{\otimes n}$  a group under multiplication [22, sec. 1]. The commutator and anti-commutator relation of  $\mathcal{P}^{\otimes 1}$  elements are given by Table 5.2. Let  $\mathcal{S}$  be an Abelian subgroup under multiplication of  $\mathcal{P}^{\otimes n}$  with the

$\begin{array}{c c} & B \\ \hline A & \end{array}$	I	X	Y	Z
I	+1	+1	+1	+1
X	+1	+1	-1	-1
Y	+1	-1	+1	-1
Z	+1	-1	-1	+1

Table 5.2: The commutation relation of Pauli elements with themselves

following definition

$$\mathcal{S} = \{M_1, M_2, \dots, M_{2^{n-k}-1}, M_{2^{n-k}} \mid \forall M_i, M_j : [M_i, M_j] = +1 \wedge \forall M_i : M_i \neq -I^{\otimes n}\}. \quad (5.14)$$

We can see that  $\mathcal{S}$  must be a commutative subgroup of  $\mathcal{P}^{\otimes n}$  and the element  $-I^{\otimes n}$  cannot be in the set. This is because  $-I^{\otimes n}$  yields a syndrome with an eigenvalue of  $-1$ , which doesn't stabilize any state. We can see that  $I, X, Y$  and  $Z$  with  $+1$  or  $-1$

as global phase stabilizes  $|0\rangle, |1\rangle$  and the main states mentioned in Equation 3.4:

$$\begin{aligned}
Z|0\rangle &= |0\rangle, & -Z|1\rangle &= |1\rangle, \\
X|+\rangle &= |+\rangle, & -X|-\rangle &= |-\rangle, \\
Y|i+\rangle &= |i+\rangle, & -Y|i-\rangle &= |i-\rangle, \\
I &\text{ stabilizes } |0\rangle, |1\rangle, |+\rangle, |-\rangle, |i+\rangle \text{ and } |i-\rangle, \\
-I &\text{ stabilizes none.}
\end{aligned} \tag{5.15}$$

**Definition 14.** Let  $\mathcal{C}_S \subseteq \mathcal{H}^{\otimes n}$  be the coding space. The stabilizer  $\mathcal{S}$  that is associated with the codespace  $\mathcal{C}_S$  is given as

$$\mathcal{S}(\mathcal{C}_S) = \{M \in \mathcal{P}^{\otimes n} \mid M|\psi\rangle = |\psi\rangle, \forall |\psi\rangle \in \mathcal{C}_S\}, \tag{5.16}$$

where  $|\psi\rangle$  is an arbitrary codeword in the codespace  $\mathcal{C}_S$ .

It should be noted that Definition 14 describes that given a code  $\mathcal{C}_S$ , there might exist some set of  $M_i$  that stabilizes the codespace. We can define the other direction such that given a stabilizer set, that set will stabilize some codewords which will define/create the codespace [56, sec. 2.2.2]. We can define  $\mathcal{C}$  as a coding subspace of  $\mathcal{H}^{\otimes n}$  where all states in  $\mathcal{C}$  are stabilized by all elements in the set  $\mathcal{S}$  [56, sec. 2.2.2].

**Definition 15.** Let  $\mathcal{S} \subseteq \mathcal{P}^{\otimes n}$  be an Abelian subgroup under multiplication of  $\mathcal{P}^{\otimes n}$ . Let  $\mathcal{C}(\mathcal{S}) \subseteq \mathcal{H}^{\otimes n}$  be the coding space associated with  $\mathcal{S}$  such that

$$\mathcal{C}(\mathcal{S}) = \{|\psi\rangle \mid M|\psi\rangle = |\psi\rangle, \forall M \in \mathcal{S}\}, \tag{5.17}$$

where  $|\psi\rangle$  is a codeword in the coding subspace  $\mathcal{C}(\mathcal{S})$ .

A stabilizer code ensures the stabilizer set  $\mathcal{S}$  that fixes some subspace  $\mathcal{C}$  is the codespace  $\mathcal{C}_S$ , which is the same codespace  $\mathcal{C}(\mathcal{S}(\mathcal{C}_S))$  generates [56, sec. 2.2.2]. Throughout this chapter, we will use the notation  $\mathcal{C}$  to describe a stabilizer code that is stabilized by some  $\mathcal{S} \subseteq \mathcal{P}^{\otimes n}$ , which is Abelian subgroup under multiplication of  $\mathcal{P}^{\otimes n}$ . We can discuss some properties of the stabilizer set  $\mathcal{S}$ :

- 1) As mentioned above, the operator  $-I$  is not a part of  $\mathcal{S}$ . This means that all other Pauli operators cannot have the negative phase in  $\mathcal{S}$ , i.e.,  $-X, -Y, -Z \notin \mathcal{S}$ . This ensures that  $\mathcal{S}$  is a group.
- 2) Since  $\mathcal{S}$  is an Abelian group under multiplication, for  $M_1, M_2 \in \mathcal{S}$ , then  $M_1 M_2, M_2 M_1 \in \mathcal{S}$ .



3) Since  $\mathcal{S}$  stabilizes all codewords  $|\psi\rangle \in \mathcal{C}$ , then

$$M_1 M_2 |\psi\rangle = M_2 M_1 |\psi\rangle = |\psi\rangle. \quad (5.18)$$

From the stabilizer set  $\mathcal{S}$  given by Equation 5.14, there exist a subset of  $\mathcal{S}$  that *generates* all the elements in the set [56, sec. 2.2.2].

**Definition 16.** Let  $\mathcal{S}$  be a stabilizer set, as defined in Equation 5.14. There exist a generator set  $\mathbf{G} = \langle g_1, g_2, \dots, g_{n-k-1}, g_{n-k} \rangle$  that generates the set  $\mathcal{S}$  such that any element  $M \in \mathcal{S}$  can be written as the product of some elements in  $\mathbf{G}$ .

The elements in the generator set  $\mathbf{G}$  are independent in the sense that an element  $g_i \in \mathbf{G}$  cannot be expressed as the product of elements in  $\mathbf{G}$ . It should be noted that if the cardinality of  $\mathbf{G}$  is  $n - k$ , then this means the cardinality of  $\mathcal{S}$  will be  $2^{n-k}$ . We can represent the elements in the generator set as a binary string where “0” denotes it is not included and “1” denote it is included. This is because an element  $M \in \mathcal{S}$  can be written in the form of

$$M = g_1^{c_1} \times g_2^{c_2} \times \dots \times g_{n-k-1}^{c_{n-k-1}} \times g_{n-k}^{c_{n-k}}, \text{ for } g_i \in \mathbf{G} \text{ and } c_i \in \{0, 1\}. \quad (5.19)$$

The codewords in the codespace can be generated by a basis of  $k$  independent codewords, giving us a total  $2^k$  codewords in the codespace [56, sec. 2.1.3]. Next, we can define the centralizer of a stabilizer set  $\mathcal{S}$ .

**Definition 17.** The set of elements in  $\mathcal{P}^{\otimes n}$  that commute with all of  $\mathcal{S}$  is the centralizer  $\mathcal{Z}(\mathcal{S})$  of  $\mathcal{S}$  [12].

We are ready to give the characteristics of a stabilizer code.

**Definition 18.** Let  $\mathcal{S}$  be a stabilizer set, as defined in Equation 5.14 and  $\mathcal{C}_{\mathcal{S}} \subseteq \mathcal{H}^{\otimes n}$  be the codespace stabilized by  $\mathcal{S}$ . A stabilizer code encodes  $k$  logical qubits into  $n$  physical qubits of information will have a dimension of  $2^k$  with the cardinality of  $\mathcal{S}$  as  $2^{n-k}$  elements and the cardinality of  $\mathbf{G}$  is  $n - k$  [22, thm. 5].

**Definition 19.** The weight of some element  $A \in \mathcal{P}^{\otimes n}$ , denoted as  $\text{WT}(A)$ , is the number of tensor elements that is not the identity  $I$  [23, sec. 2.3].

**Definition 20.** The distance  $d$  of a stabilizer code is the minimum weight of any operator in [23, sec. 3.2, 32]

$$\mathcal{N}(\mathcal{S}) \setminus \mathcal{S} = \{A | A \in \mathcal{N}(\mathcal{S}) \wedge A \notin \mathcal{S}\}, \quad (5.20)$$

where  $\mathcal{N}(\mathcal{S})$  is the normalizer of  $\mathcal{S}$ .

We will use the notation of  $[[n, k, d]]$  to denote a quantum code that encodes  $k$  logical qubits into  $n$  physical qubits with  $d$  being the minimum distance of the code. Such code can correct up to  $t = \lfloor \frac{d-1}{2} \rfloor$  errors [23, sec. 3.2].

### 5.2.1 Syndrome Extraction Procedure

Let  $\mathcal{E} \subseteq \mathcal{P}^{\otimes n}$  be a set  $\mathcal{E} = \{E_1, E_2, \dots, E_{f-1}, E_f\}$  denoting all correctable errors. An error  $E \in \mathcal{E}$  will anticommute with any element  $M \in \mathcal{S}$ , *i.e.*,

$$ME|\psi\rangle = -EM|\psi\rangle = -E|\psi\rangle. \quad (5.21)$$

In this case, the error syndrome of a stabilizer code will be  $n - k$  binary row vector in the following form<sup>2</sup>

$$\begin{bmatrix} M_1 \circ E & M_2 \circ E & \cdots & M_{n-k-1} \circ E & M_{n-k} \circ E \end{bmatrix}, \quad (5.22)$$

where

$$M_i \circ E = \begin{cases} 0, & \text{if } [M_i, E] \\ 1, & \text{if } \{M_i, E\}. \end{cases}$$

## 5.3 Criteria for Quantum Error Correction

The codespace  $\mathcal{C}$  can be described by a basis  $\mathcal{C}_{\text{basis}} = \{|\psi_1\rangle, |\psi_2\rangle, \dots, |\psi_{k-1}\rangle, |\psi_k\rangle\}$ . A quantum code can correct two errors  $E_i$  and  $E_j$  if we can distinguish the two errors that are acting on two basis  $|\psi_x\rangle$  and  $|\psi_y\rangle$  [23, sec. 3.2]. This condition is satisfied if and only if  $E_i|\psi_x\rangle$  is orthogonal to  $E_j|\psi_y\rangle$  [23, sec. 3.2], *i.e.*,

$$\langle\psi_x| E_i^\dagger E_j |\psi_y\rangle = 0, \quad \forall i \neq j. \quad (5.23)$$

Such distinguishable errors are correctable.<sup>3</sup> When performing a measurement on the error, we cannot know anything about the state of the code as this will destroy the superposition of the basis states. we can use the following measurement instead [23, sec. 3.2]:

$$\langle\psi_x| E_i^\dagger E_j |\psi_x\rangle = 0, \quad \forall E_i, E_j \in \mathcal{E}. \quad (5.24)$$

<sup>2</sup>We can map the eigenvalues  $-1$  to  $0$  and  $+1$  to  $1$  to resort to binary setting.

<sup>3</sup>In the case where only a single error occurs, we can simply let  $E_i^\dagger$  or  $E_j$  to be  $I$ .

Furthermore, Equation 5.24 should also hold true for all codewords in the basis,

$$\langle \psi_x | E_i^\dagger E_j | \psi_x \rangle = \langle \psi_y | E_i^\dagger E_j | \psi_y \rangle, \quad \forall E_i, E_j \in \mathcal{E} \wedge |\psi_x\rangle, |\psi_y\rangle \in \mathbb{C}_{\text{basis}}. \quad (5.25)$$

We are able to combine Equations (5.24) and (5.25) to obtain [23, sec. 3.2]

$$\langle \psi_x | E_i^\dagger E_j | \psi_y \rangle = C_{ij} \delta_{xy}, \quad (5.26)$$

where  $C_{ij} \in \mathbb{C}$  and  $\delta_{xy} \in \{0, 1\}$  is the Kronecker delta function defined as:

$$\delta_{xy} = \begin{cases} 1, & \text{if } x = y, \\ 0, & \text{if } x \neq y. \end{cases}$$

It is important to mention that  $C_{ij}$  is independent of  $|\psi_x\rangle$  and  $|\psi_y\rangle$  [23, sec. 3.2]. All in all, a stabilizer code will correct a set of errors  $\mathcal{E} = \{E_1, E_2, \dots, E_{f-1}, E_f\}$  if [22, thm. 5, 12]

$$E_i^\dagger E_j \in \mathcal{S} \cup (\mathbf{G} - \mathcal{Z}(\mathcal{S})), \quad \forall E_i E_j. \quad (5.27)$$

### 5.3.1 Examples of Stabilizer Codes

The first example of a stabilizer code is the  $[[9, 1, 3]]$  Shor's Code. The subgroup  $\mathcal{S}$  will have  $n - k = 9 - 1 = 8$  elements,

$$\mathcal{S} = \{M_1, M_2, \dots, M_8\}. \quad (5.28)$$

They are given in Equation 5.29.

	$Q_1$	$Q_2$	$Q_3$	$Q_4$	$Q_5$	$Q_6$	$Q_7$	$Q_8$	$Q_9$						
$M_1 =$	Z	⊗	Z	⊗	I	⊗	I	⊗	I	⊗	I	⊗	I	⊗	I
$M_2 =$	I	⊗	Z	⊗	Z	⊗	I	⊗	I	⊗	I	⊗	I	⊗	I
$M_3 =$	I	⊗	I	⊗	I	⊗	Z	⊗	Z	⊗	I	⊗	I	⊗	I
$M_4 =$	I	⊗	I	⊗	I	⊗	I	⊗	Z	⊗	Z	⊗	I	⊗	I
$M_5 =$	I	⊗	I	⊗	I	⊗	I	⊗	I	⊗	I	⊗	Z	⊗	Z
$M_6 =$	I	⊗	I	⊗	I	⊗	I	⊗	I	⊗	I	⊗	I	⊗	Z
$M_7 =$	X	⊗	X	⊗	X	⊗	X	⊗	X	⊗	X	⊗	I	⊗	I
$M_8 =$	I	⊗	I	⊗	I	⊗	X	⊗	X	⊗	X	⊗	X	⊗	X

(5.29)

The syndrome  $S$  will be

$$S = \begin{bmatrix} M_1 & M_2 & M_3 & M_4 & M_5 & M_6 & M_7 & M_8 \end{bmatrix}. \quad (5.30)$$

In the case where no error occurs, the syndrome vector will be

$$S = \begin{bmatrix} +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 \end{bmatrix}. \quad (5.31)$$

Assume there was a phase flip occurred on  $Q_2$ , then the syndrome vector will be:

$$S = \begin{bmatrix} -1 & -1 & +1 & +1 & +1 & +1 & +1 & +1 \end{bmatrix}. \quad (5.32)$$

## Chapter 6

# Entanglement-Assisted QECC

Stabilizer codes allows us to import certain classical linear codes and turn them into quantum codes. The limitation here is that the classical code must be self-dual. Entanglement-assisted quantum error correction codes are the generalization of stabilizer codes, established by Todd Brun, Igor Devetak and Min-hsiu Hsieh [12]. The self-duality condition is not required and binary as well as quaternary linear codes can be constructed. Furthermore, the set of generators can contain anti-commuting elements. This is achieved by the usage of bipartite entanglement. Bipartite entanglement is the idea of having an entangled qubit (ebit) shared between the sender *Alice* and the receiver *Bob*. The ebit is given as

$$|\Phi^+\rangle = \frac{|00\rangle^{AB} + |11\rangle^{AB}}{\sqrt{2}}. \quad (6.1)$$

The red portion of the ebit is controlled by Alice whereas the blue portion is controlled by Bob. The two operators that stabilizes  $|\Phi^+\rangle$  are:

$$\begin{aligned} X_{A_1} \otimes X_{B_1} |\Phi^+\rangle &= |\Phi^+\rangle \\ Z_{A_1} \otimes Z_{B_1} |\Phi^+\rangle &= |\Phi^+\rangle \end{aligned} \quad (6.2)$$

where  $X_{A_1}$  and  $Z_{A_1}$  are  $X_1$  and  $Z_1$  applied on Alice's portion and  $X_{B_1}$  and  $Z_{B_1}$  are  $X_1$  and  $Z_1$  applied on Bob's portion. Furthermore, the two operators commute with each other:

$$[X_{A_1} \otimes X_{B_1}, Z_{A_1} \otimes Z_{B_1}] = 0. \quad (6.3)$$

Moreover, having Alice and Bob individually apply operators on their portion of the ebit gives:

$$\begin{aligned}
 \mathbf{X}_{A_1} \otimes \mathbf{I} |\Phi^+\rangle &= |\Psi^+\rangle \\
 \mathbf{Z}_{A_1} \otimes \mathbf{I} |\Phi^+\rangle &= |\Phi^-\rangle \\
 \mathbf{I} \otimes \mathbf{X}_{B_1} |\Phi^+\rangle &= |\Psi^+\rangle \\
 \mathbf{I} \otimes \mathbf{Z}_{B_1} |\Phi^+\rangle &= |\Phi^-\rangle
 \end{aligned} \tag{6.4}$$

where,

$$|\Psi^+\rangle = \frac{|\mathbf{01}\rangle^{AB} + |\mathbf{10}\rangle^{AB}}{\sqrt{2}} \quad \text{and} \quad |\Phi^-\rangle = \frac{|\mathbf{00}\rangle^{AB} - |\mathbf{11}\rangle^{AB}}{\sqrt{2}}.$$

Our interest is that the four operators anticommute as follows:

$$\begin{aligned}
 \{\mathbf{X}_{A_1} \otimes \mathbf{I}, \mathbf{Z}_{A_1} \otimes \mathbf{I}\} &= 0 \\
 \{\mathbf{I} \otimes \mathbf{X}_{B_1}, \mathbf{I} \otimes \mathbf{Z}_{B_1}\} &= 0
 \end{aligned} \tag{6.5}$$

We can use the notation  $\boxed{\bullet \quad \bullet}$  to denote the pair operators that stabilizes the ebit shared between both parties. The left cell denotes Alice's portion and the right denotes Bob's. We can represent the operations acted on Equation 6.2 as  $\boxed{\mathbf{Z}_1 \mathbf{Z}_1}$  and  $\boxed{\mathbf{X}_1 \mathbf{X}_1}$ . The relation above can help us to bypass the commutativity constraint by allowing our set of generators to contain anticommutative elements and use ebits to resolve the anticommutativity. An entanglement-assisted code  $[[n, k, d; c]]$  of distance  $d$  encodes  $k$  logical qubits into  $n$  physical qubits with the help of  $c$  ebits and  $a$  ancilla qubits. Let  $\mathcal{S}$  be arbitrary subgroup of  $\mathcal{P}^{\otimes n}$  with  $2^{2a+c}$  distinct elements up to overall phase, then there exists a generator set

$$\mathbf{G} = \underbrace{\langle \overline{\mathbf{Z}}_{a+1}, \dots, \overline{\mathbf{Z}}_{a+c}, \overline{\mathbf{X}}_{a+1}, \dots, \overline{\mathbf{X}}_{a+c}, \overline{\mathbf{Z}}_1, \dots, \overline{\mathbf{Z}}_a \rangle}_{2a+c \text{ elements}} \tag{6.6}$$

to generate  $\mathcal{S}$  with the following commutation characteristics:

$$\begin{aligned}
 [\overline{\mathbf{Z}}_i, \overline{\mathbf{Z}}_j] &= 0, & \forall i, j \\
 [\overline{\mathbf{X}}_i, \overline{\mathbf{X}}_j] &= 0, & \forall i, j \\
 [\overline{\mathbf{X}}_i, \overline{\mathbf{Z}}_j] &= 0, & \forall i \neq j \\
 \{\overline{\mathbf{X}}_i, \overline{\mathbf{Z}}_j\} &= 0, & \forall i
 \end{aligned} \tag{6.7}$$

The elements  $\overline{Z}_i$  and  $\overline{X}_i$  are the equivalent form of  $2a + c$  elements in  $\mathcal{S}$ . They are the Z gate and X gate being applied on the  $i$  qubit. The generator set  $\mathbf{G}$  determines the number of  $a$  ancilla qubits and  $c$  ebits qubits required for the code which is obtained from the symplectic Gram-Schmidt orthogonalization algorithm [57, pg. 23]. This algorithm performs row operations on the generators in the code to output an equivalent code. The algorithm is optimized in the sense that it will find the minimal number of ebits required for the code. The nonabelian group  $\mathcal{S}$  can be split into two subgroups which are the isotropic subgroup  $\mathbf{G}_I$  and entanglement subgroup  $\mathbf{G}_E$ :

$$\begin{aligned}\mathbf{G}_I &= \langle \overline{Z}_1, \dots, \overline{Z}_a \rangle \\ \mathbf{G}_E &= \langle \overline{Z}_{a+1}, \dots, \overline{Z}_{a+c}, \overline{X}_{a+1}, \dots, \overline{X}_{a+c} \rangle\end{aligned}\tag{6.8}$$

The isotropic group is the commuting subgroup and the entanglement group is the one with anti-commuting pairs. Let  $\mathcal{E} \subseteq \mathcal{P}^{\otimes n}$  be a set  $\mathcal{E} = \{E_1, E_2, \dots, E_{f-1}, E_f\}$  denoting all correctable errors, then an entanglement-assisted quantum error correction code can correct

$$E_i^\dagger E_j \in \mathbf{G}_I \quad \text{or} \quad E_i^\dagger E_j \in \mathcal{P}^{\otimes n} - \mathcal{Z}(\langle \mathbf{G}_I, \mathbf{G}_S \rangle) \quad \forall E_i, E_j \in \mathcal{E}.\tag{6.9}$$

The practical approach to using this quantum communication system is by starting with:

1. Having Alice and Bob share  $c$  ebits prior to the beginning of the communication.
2. Alice would also have  $a$  ancilla qubits.
3. The unencoded state is a simultaneous +1-eigenstate of the following operators:

$$\left\{ \boxed{\textcolor{red}{Z}_{a+1}} \boxed{\textcolor{blue}{Z}_1}, \dots, \boxed{\textcolor{red}{Z}_{a+c}} \boxed{\textcolor{blue}{Z}_c}, \boxed{\textcolor{red}{X}_{a+1}} \boxed{\textcolor{blue}{X}_1}, \dots, \boxed{\textcolor{red}{X}_{a+c}} \boxed{\textcolor{blue}{X}_c}, \textcolor{red}{Z}_1, \dots, \textcolor{red}{Z}_a \right\}.\tag{6.10}$$

The operators  $\textcolor{red}{Z}_1, \dots, \textcolor{red}{Z}_a$  are used for the ancilla qubits on Alice's side. We have a +1-eigenstate because applying the  $c$ -pair operators on  $c$  ebits in the state  $|\Phi^+\rangle^{\otimes c}$  leaves all  $c$  ebits unchanged, as discussed in Equation 6.2. We stated that the unencoded state is a simultaneous +1-eigenvalue, by that, we mean the operators can be applied on the quantum state in any order and will still give +1-eigenvalue.

4. Alice encodes her  $k$  information qubits with  $a$  ancilla qubits and her half of the

$c$  ebits. This will transform the operators from Equation 6.10 to Equation 6.11:

$$\left\{ \boxed{\overline{Z}_{a+1}} \boxed{Z_1}, \dots, \boxed{\overline{Z}_{a+c}} \boxed{Z_c}, \boxed{\overline{X}_{a+1}} \boxed{X_1}, \dots, \boxed{\overline{X}_{a+c}} \boxed{X_c}, \overline{Z}_1, \dots, \overline{Z}_a \right\}. \quad (6.11)$$

The operators in Equation 6.11 are mathematically equivalent operators to the ones in Equation 6.10. They are obtained from the symplectic Gram-Schmidt orthogonalization algorithm. The algorithm minimizes the number of ebits required which in turn yield the optimal number of ebits.

5. The encoding step has been completed now. Alice will send her  $n$  qubits which consists of  $k = n - a - c$  logical qubits,  $a$  ancilla qubits and her portion of the  $c$  ebits. It is important to note that the erroneous channel is assumed to affect only the  $n$  qubits Alice has sent. Bob's portion of the  $c$  ebits is assumed to be perfect and immune to errors.
6. Bob receives the  $n$  qubits Alice has sent. Along with his  $c$  ebits, Bob has  $n + c$  qubits and will measure all of them to detect the errors that may occur.
7. Bob then performs the necessary correction to obtain the original message sent.

Entanglement-assisted quantum error correction tends to focus on the parity check matrix of the code rather than the generator matrix. This is because there is a clever isomorphism between Pauli Gates and binary strings. The rest of this section is influenced by [57, sec 2.4]. A string in the form of  $\mathbb{Z}_2^n$  is a binary string of length  $n$ . Consider  $[A]$  as the set of equivalence classes of some operator  $A$  of the same phase:

$$[A] = \{\alpha A \mid \alpha \in \mathbb{C} \wedge |\alpha| = 1\}. \quad (6.12)$$

Let  $[\mathcal{P}^{\otimes 1}]$  be the equivalence set of all Pauli operators in  $\mathcal{P}^{\otimes 1}$  defined as

$$[\mathcal{P}^{\otimes 1}] = \{[A] \mid A \in \mathcal{P}^{\otimes 1}\}. \quad (6.13)$$

We can now define  $N$  as the map

$$N : \mathbb{Z}_2^2 \mapsto \mathcal{P}^{\otimes 1} \quad (6.14)$$



with the following mapping:

$$\begin{array}{rcl}
 \mathbb{Z}_2^2 & & \mathcal{P}^{\otimes 1} \\
 00 & \mapsto & \mathbf{I} \\
 01 & \mapsto & \mathbf{X} \\
 10 & \mapsto & \mathbf{Z} \\
 11 & \mapsto & \mathbf{Y}
 \end{array} \quad (6.15)$$

To see how this is useful, let us use the following notation to denote a binary string in  $\mathbb{Z}_2^2$ : for  $u, v \in \mathbb{Z}_2^2 = \{[0|0], [0|1], [1|0], [1|1]\}$ , let  $u = [z_u|x_u]$  and  $v = [z_v|x_v]$  where  $z_u, x_u, z_v, x_v \in \{0, 1\}$ . For example, if  $u = [0|1]$  and  $v = [1|1]$ , then  $N(u) = \mathbf{X}$  and  $N(v) = \mathbf{Y}$ , respectively. Moreover, the xor of vectors in  $\mathbb{Z}_2^2$  is equivalent to the multiplication of Pauli operators up to a global phase:

$$N[(u \oplus v)] = [N(u)][N(v)]. \quad (6.16)$$

Let  $\odot$  denote the symplectic product between two vectors  $u, v \in \mathbb{Z}_2^2$  as

$$u \odot v = u_z v_x - v_z u_x, \quad (6.17)$$

which yields the following output: The symplectic product yields the commutation

$\odot$	00	01	10	11
00	0	0	0	0
01	0	0	1	1
10	0	1	0	1
11	0	1	1	0

Table 6.1: The symplectic product of two binary vectors

relations of gates in  $\mathcal{P}^{\otimes 1}$ :

$$N(u)N(v) = (-1)^{u \odot v} N(v)N(u). \quad (6.18)$$

We discussed the map between Pauli gates in  $\mathcal{P}^{\otimes 1}$  and binary strings of length 2. We can generalize this map isomorphism to describe Pauli gates in  $\mathcal{P}^{\otimes n}$  and binary strings of length  $2n$ . Let

$$\mathbf{A} = A_1 \otimes A_2 \otimes \cdots \otimes A_n \quad (6.19)$$

be an arbitrary gate in  $\mathcal{P}^{\otimes n}$ . Consider  $[\mathbf{A}]$  as the set of equivalence classes of some operator  $\mathbf{A}$  of the same phase:

$$[\mathbf{A}] = \{\alpha \mathbf{A} \mid \alpha \in \mathbb{C} \wedge |\alpha| = 1\}. \quad (6.20)$$

Let  $[\mathcal{P}^{\otimes n}]$  be the equivalence set of all Pauli operators in  $\mathcal{P}^{\otimes n}$  defined as

$$[\mathcal{P}^{\otimes n}] = \{[\mathbf{A}] \mid \mathbf{A} \in \mathcal{P}^{\otimes n}\}. \quad (6.21)$$

In this case,  $[\mathcal{P}^{\otimes n}]$  is an equivalence class which forms a commutative group under the  $\star$  operation because for any  $\mathbf{A}, \mathbf{B} \in [\mathcal{P}^{\otimes n}]$ ,

$$[\mathbf{A}] \star [\mathbf{B}] = [\mathbf{AB}] = [A_1] \star [B_1] \otimes [A_2] \star [B_2] \otimes \cdots \otimes [A_n] \star [B_n]. \quad (6.22)$$

The binary string of length  $2n$  is a  $2n$ -dimensional vector space of the form

$$\mathbb{Z}_2^{2n} = \{(\mathbf{z}, \mathbf{x}) \mid \mathbf{z}, \mathbf{x} \in \mathbb{Z}_2^n\}, \quad (6.23)$$

forms a commutative group under the xor operator  $\oplus$ . Let  $\mathbf{u} = [\mathbf{z}_u | \mathbf{x}_u]$  and  $\mathbf{v} = [\mathbf{z}_v | \mathbf{x}_v]$  be two vectors in  $\mathbb{Z}_2^{2n}$  and each of  $\mathbf{z}_u, \mathbf{x}_u, \mathbf{z}_v, \mathbf{x}_v$  as a binary vector of length  $n$  of the form  $(b_1, b_2, \dots, b_n)$ :

$$\begin{aligned} \mathbf{z}_u &= (z_{u1}, z_{u2}, \dots, z_{un}) \\ \mathbf{x}_u &= (x_{u1}, x_{u2}, \dots, x_{un}) \\ \mathbf{z}_v &= (z_{v1}, z_{v2}, \dots, z_{vn}) \\ \mathbf{x}_v &= (x_{v1}, x_{v2}, \dots, x_{vn}) \end{aligned} \quad (6.24)$$

The symplectic product of  $\mathbf{u} \odot \mathbf{v}$  is given as

$$\mathbf{u} \odot \mathbf{v} = \sum_{i=1}^n \mathbf{u}_{z_i} \mathbf{v}_{x_i} - \mathbf{v}_{z_i} \mathbf{u}_{x_i} = \sum_{i=1}^n u_i \odot v_i, \quad (6.25)$$

where  $u_i = [\mathbf{u}_{z_i} | \mathbf{u}_{x_i}]$  and  $v_i = [\mathbf{v}_{z_i} | \mathbf{v}_{x_i}]$ . Let the map  $\mathbf{N}$  as

$$\mathbf{N} : \mathbb{Z}_2^{2n} \mapsto \mathcal{P}^{\otimes n} \quad (6.26)$$

be the isomorphism from binary vectors to Pauli gates.

$$\mathbf{N}(\mathbf{u}) = N(u_1) \otimes N(u_2) \otimes \cdots \otimes N(u_n), \quad (6.27)$$

where  $\mathbf{u} = [z_{u_1}, z_{u_2}, \dots, z_{u_n} \mid x_{u_1}, x_{u_2}, \dots, x_{u_n}] \in Z_n^{2n}$ . Furthermore, define

$$\begin{aligned} \mathbf{Z}(\mathbf{z}) &= Z^{z_1} \otimes Z^{z_2} \otimes \cdots \otimes Z^{z_n} \\ \mathbf{X}(\mathbf{x}) &= X^{x_1} \otimes X^{x_2} \otimes \cdots \otimes X^{x_n} \end{aligned} \quad (6.28)$$

where  $z_i$  and  $x_i$  are the values of a vector

$$\mathbf{v} = [z_1, z_2, \dots, z_n \mid x_1, x_2, \dots, x_n] \in Z_n^{2n}. \quad (6.29)$$

With that,  $\mathbf{N}(\mathbf{u})$  and  $\mathbf{Z}(\mathbf{z})\mathbf{X}(\mathbf{x})$  are a part of the same equivalence class

$$[\mathbf{N}(\mathbf{u})] = [\mathbf{Z}(\mathbf{z})\mathbf{X}(\mathbf{x})]. \quad (6.30)$$

Hence,

$$[\mathbf{N}(\mathbf{u} \oplus \mathbf{v})] = [\mathbf{N}(\mathbf{u})][\mathbf{N}(\mathbf{v})], \quad \forall \mathbf{u}, \mathbf{v} \in \mathbb{Z}_2^{2n}. \quad (6.31)$$

Furthermore, for any operator  $\mathbf{N}(\mathbf{u})$  and  $\mathbf{N}(\mathbf{v})$ , the symplectic product preserves the following commutation:

$$\mathbf{N}(\mathbf{u})\mathbf{N}(\mathbf{v}) = (-1)^{u \odot v} \mathbf{N}(\mathbf{v})\mathbf{N}(\mathbf{u}). \quad (6.32)$$

We will later see how we can import a classical linear quaternary codes as an entanglement-assisted quantum code.

## 6.1 The Encoding/Decoding Algorithm

The algorithm in [58] determines the encoding and decoding circuits for the set of Pauli generators. The perk of this algorithm is that we do not need to know the number of ebits required and it also finds the optimal (least) number of ebits needed. Suppose there are  $r$  generators to generate our nonabelian subgroup  $\mathcal{S}$ . We need to convert these generators to the binary form using our Pauli to binary mapping. Our

parity check matrix  $H$  will be of the form

$$H = \left[ \begin{array}{cccc|cccc} Z_{1,1} & Z_{1,2} & \dots & Z_{1,n} & X_{1,1} & X_{1,2} & \dots & X_{1,n} \\ Z_{2,1} & Z_{2,2} & \dots & Z_{2,n} & X_{2,1} & X_{2,2} & \dots & X_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ Z_{b,1} & Z_{b,2} & \dots & Z_{b,n} & X_{b,1} & X_{b,2} & \dots & X_{b,n} \end{array} \right], \quad (6.33)$$

where  $n$  is the length of the code,  $b$  is the dimension of the code which is  $2k - n + c$ , the entries  $Z_{i,j} \in \{0, 1\}$  and  $X_{i,j} \in \{0, 1\}$  represent if  $Z$  or  $X$  operator should be placed, respectively. Let us use the same example used in [12, 57]. Let  $\mathcal{S}$  be a nonabelian subgroup of  $\mathcal{P}^{\otimes 4}$  with the following generators:

$$\begin{array}{rcll} & Q_1 & Q_2 & Q_3 & Q_4 \\ M_1 & = & Z & \otimes & X & \otimes & Z & \otimes & I \\ M_2 & = & Z & \otimes & Z & \otimes & I & \otimes & Z \\ M_3 & = & X & \otimes & Y & \otimes & X & \otimes & I \\ M_4 & = & X & \otimes & X & \otimes & I & \otimes & X \end{array} \quad (6.34)$$

We need to find an equivalent set of generators that satisfy the commutation relation in Equation 6.7. The modified set is

$$\begin{array}{rcll} & Q_1 & Q_2 & Q_3 & Q_4 \\ M_1 & = & Z & \otimes & X & \otimes & Z & \otimes & I \\ M_2 & = & Z & \otimes & Z & \otimes & I & \otimes & Z \\ M'_3 & = & Y & \otimes & X & \otimes & X & \otimes & Z \\ M'_4 & = & Z & \otimes & Y & \otimes & Y & \otimes & X \end{array} \quad (6.35)$$

The generators  $M_i$  are obtained by:

- Leaving  $M_1$  unchanged
- Leaving  $M_2$  unchanged
- $M_3$  changed to  $M'_3 = M_2 \times M_3$
- $M_4$  changed to  $M'_4 = M_1 \times M_2 \times M'_3 \times M_4$

We now have the following commutation relations:

$$\begin{aligned}
 \{M_1, M_2\} &= 0 \\
 [M_1, M_3'] &= 0 \\
 [M_1, M_4'] &= 0 \\
 [M_2, M_3'] &= 0 \\
 [M_2, M_4'] &= 0 \\
 [M_3', M_4'] &= 0
 \end{aligned} \tag{6.36}$$

After applying the encoding algorithm, we obtain a unitarily equivalent set of generators in [Equation 6.35](#).

$$\begin{array}{rcll}
 & Q_1 & Q_2 & Q_3 & Q_4 \\
 M_1'' &= & \mathbf{X} & \otimes & \mathbf{I} & \otimes & \mathbf{I} & \otimes & \mathbf{I} \\
 M_2'' &= & \mathbf{Z} & \otimes & \mathbf{I} & \otimes & \mathbf{I} & \otimes & \mathbf{I} \\
 M_3'' &= & \mathbf{I} & \otimes & \mathbf{Z} & \otimes & \mathbf{I} & \otimes & \mathbf{I} \\
 M_4'' &= & \mathbf{I} & \otimes & \mathbf{I} & \otimes & \mathbf{Z} & \otimes & \mathbf{I}
 \end{array} . \tag{6.37}$$

We can expand the number of qubits and add one ebit ( $c_1$ ) to resolve the anticommutativity between  $M_1''$  and  $M_2''$ :

$$\begin{array}{rcll}
 & Q_1 & Q_2 & Q_3 & Q_4 & c_1 \\
 M_1'' &= & \mathbf{X} & \otimes & \mathbf{I} & \otimes & \mathbf{I} & \otimes & \mathbf{I} & \otimes & \mathbf{X} \\
 M_2'' &= & \mathbf{Z} & \otimes & \mathbf{I} & \otimes & \mathbf{I} & \otimes & \mathbf{I} & \otimes & \mathbf{Z} \\
 M_3'' &= & \mathbf{I} & \otimes & \mathbf{Z} & \otimes & \mathbf{I} & \otimes & \mathbf{I} & \otimes & \mathbf{I} \\
 M_4'' &= & \mathbf{I} & \otimes & \mathbf{I} & \otimes & \mathbf{Z} & \otimes & \mathbf{I} & \otimes & \mathbf{I}
 \end{array} . \tag{6.38}$$

The generators in [Equation 6.38](#) stabilize the encoded system  $\overline{|\psi\rangle}$ :

$$\overline{|\psi\rangle} = |\Phi^+\rangle \otimes |0\rangle \otimes |0\rangle \otimes |\psi\rangle, \tag{6.39}$$

where  $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$  is the qubit to be transmitted. The encoding algorithm will apply row and column operations on the parity check matrix  $H$  in the form of [Equation 6.33](#) to obtain the reduction. This reduction yields an equivalent code. The column operation uses P, H, CNOT and SWAP gates with the following effect on the binary matrix  $H$  (using  $H_z$  as the left part of the matrix and  $H_x$  as the right part):

- P on qubit  $i$ : adds column  $i$  in  $H_x$  to column  $i$  in  $H_z$ .

- H on qubit  $i$ : swap column  $i$  in  $H_z$  with column  $i$  in  $H_x$ .
- CNOT from qubit  $i$  to  $j$ : add column  $j$  to column  $i$  in  $H_z$  and add column  $i$  to column  $j$  in  $H_x$ .
- SWAP on qubit  $i$  and  $j$ : swap column  $i$  with column  $j$  in  $H_z$  as well as swap column  $i$  with column  $j$  in  $H_x$ .

As for the decoding algorithm, the exact steps of the encoding algorithm are applied in the reverse order since applying some unitary gate  $U$  twice is mathematically equivalent of applying the  $I$  gate.

## Chapter 7

# Introduction to Quaternary Fields and Codes

Let  $\mathbb{F}_q$  denote the finite (Galois) field of order  $q$  where  $q = p^k$  for some prime number  $p$  and  $k$  is a positive integer. A quaternary field is in the form  $\mathbb{F}_{2^2} = \mathbb{F}_4$ <sup>1</sup> of order 4 with the following elements:  $\{0, 1, \omega, \bar{\omega}\}$ , where  $\bar{\omega} = \omega + 1 = \omega^2$  and  $\omega^3 = 1$ . For any finite field of the form  $\mathbb{F}_{p^2}$ , the complex conjugation of an element  $x \in \mathbb{F}_{p^2}$  is defined as  $x^{\sqrt{q}} \in \mathbb{F}_{p^2}$  [47, chap. 1.2]. We will use the notation  $x^\dagger$  to denote an element that is complex conjugated. In the quaternary field, the conjugation is defined as  $x^2 \in \mathbb{F}_{2^2}$ . The addition, multiplication and conjugation of elements in  $\mathbb{F}_4$  are given in Table 7.1.

+	0	1	$\omega$	$\bar{\omega}$	$\times$	0	1	$\omega$	$\bar{\omega}$	$x$	$x^\dagger$
0	0	1	$\omega$	$\bar{\omega}$	0	0	0	0	0	0	0
1	1	0	$\bar{\omega}$	$\omega$	1	0	1	$\omega$	$\bar{\omega}$	1	1
$\omega$	$\omega$	$\bar{\omega}$	0	1	$\omega$	0	$\omega$	$\bar{\omega}$	1	$\omega$	$\bar{\omega}$
$\bar{\omega}$	$\bar{\omega}$	$\omega$	1	0	$\bar{\omega}$	0	$\bar{\omega}$	1	$\omega$	$\bar{\omega}$	$\omega$

Table 7.1: The addition table (left), multiplication (middle) and conjugation (right) of elements in  $\mathbb{F}_4$

Denote a quaternary code  $[n, k, d]_4$  with  $n$ ,  $k$  and  $d$  as length, dimension and minimum distance, respectively. For example, consider the following generator matrix of the  $[6, 2, 4]_4$  code:

$$G = \left[ \begin{array}{cc|cccc} 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & \omega & \bar{\omega} \end{array} \right] \begin{matrix} \textcircled{1} \\ \textcircled{2} \end{matrix}, \quad (7.1)$$

this will generate  $4^k = 4^2 = 16$  codewords. That is because we also take the multiple

<sup>1</sup>the notation  $GF(4)$  is frequently used.

of each codeword which is something we didn't do in binary codes. The first row in the generator matrix  $[1 \ 0 \ 0 \ 1 \ 1 \ 1]$  is a codeword and so are the following two vectors:

$$\begin{aligned}\omega [1 \ 0 \ 0 \ 1 \ 1 \ 1] &= [\omega \ 0 \ 0 \ \omega \ \omega \ \omega] \\ \bar{\omega} [1 \ 0 \ 0 \ 1 \ 1 \ 1] &= [\bar{\omega} \ 0 \ 0 \ \bar{\omega} \ \bar{\omega} \ \bar{\omega}]\end{aligned}\tag{7.2}$$

The 16 codewords are shown in Table 7.2.

#	Linear combinations
00	None
01	① in $G$
02	$\omega$ ①
03	$\bar{\omega}$ ①
04	② in $G$
05	$\omega$ ②
06	$\bar{\omega}$ ②
07	① + ②
08	$\omega$ ① + ②
09	$\bar{\omega}$ ① + ②
10	① + $\omega$ ②
11	$\omega$ ① + $\omega$ ②
12	$\bar{\omega}$ ① + $\omega$ ②
13	① + $\bar{\omega}$ ②
14	$\omega$ ① + $\bar{\omega}$ ②
15	$\bar{\omega}$ ① + $\bar{\omega}$ ②

Table 7.2: All of the  $4^k = 4^2 = 16$  codewords of the  $[6, 2, 4]_4$  code

**Definition 21.** *The Euclidean dual code of a quaternary code  $C$  is defined as [47, chap. 1.3]*

$$C^\perp = \{\mathbf{x} \in \mathbb{F}_4^n \mid \langle \mathbf{x}, \mathbf{y} \rangle = 0, \forall \mathbf{y} \in C\},\tag{7.3}$$

where  $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^n \mathbf{x}_i \mathbf{y}_i$  for vectors  $\mathbf{x}, \mathbf{y} \in \mathbb{F}_4^n$ .

We are also interested in the *Hermitian dual code* of a code. A Hermitian dual code is defined over some  $[n, k, d]_{q^2}$  code. In our case,  $q = 2$  so we can define it as the following:

**Definition 22.** *The Hermitian dual code of a quaternary code  $C = [n, k, d]_4$  is defined as [47, chap. 1.2]*

$$C^{\perp_H} = \{\mathbf{x} \in \mathbb{F}_4^n \mid \langle \mathbf{x}, \mathbf{y} \rangle_H = 0, \forall \mathbf{y} \in C\},\tag{7.4}$$

where  $\langle \mathbf{x}, \mathbf{y} \rangle_H = \sum_{i=1}^n \mathbf{x}_i \mathbf{y}_i^\dagger$  for vectors  $\mathbf{x}, \mathbf{y} \in \mathbb{F}_4^n$ .



We are able to give the definitions for Euclidean linear complementary dual (LCD) and Hermitian LCD.

**Definition 23.** A code  $\mathcal{C}$  over  $\mathbb{F}_q$  is called LCD if  $\mathcal{C} \cap \mathcal{C}^\perp = \{\mathbf{0}\}$  [40].

**Definition 24.** A code  $\mathcal{C}$  over  $\mathbb{F}_q$  with a generator matrix  $G$  is Euclidean LCD if  $GG^T$  is nonsingular [15].

For example, consider the following generator matrix of the  $[6, 3, 3]_4$  code:

$$G = \left[ \begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & \omega \\ 0 & 0 & 1 & 1 & 0 & 1 \end{array} \right]. \quad (7.5)$$

Since  $\text{DET}(GG^T) = 1 \neq 0$ ,  $GG^T$  is invertible, hence nonsingular.

**Definition 25.** A code  $\mathcal{C}$  over  $\mathbb{F}_{q^2}$  is Hermitian LCD if and only if  $GG^\dagger$  is nonsingular for a generator matrix  $G$  of the code  $\mathcal{C}$  [15].

For example, consider the following generator matrix of the  $[7, 4, 3]_4$  code:

$$G = \left[ \begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & \omega \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & \omega \end{array} \right]. \quad (7.6)$$

Since  $\text{DET}(GG^\dagger) = 1 \neq 0$ ,  $GG^\dagger$  is invertible, hence nonsingular.

In this thesis, the objective is to find Hermitian LCD codes. The Entanglement-assisted quantum error correction code formalism opens the door to the construction of entanglement assisted quantum codes from  $\mathbb{F}_4$  classical codes. Let us mention Corollary 3.6.2 from [57, pg. 37]:

**Lemma 3.** Given a classical  $[n, k, d]_4$  code with a parity check matrix  $H$ , we can import it as a  $[[n, 2k - n + c, d; c]]$  entanglement-assisted quantum code where  $c = \text{RANK}(HH^\dagger)$ .

The operation  $H^\dagger$  is the conjugate transpose of the matrix  $H$ . The proof of the lemma can be found in [57, pg. 38].

**Definition 26.** A Hermitian LCD code  $[n, k, d]_4$  is said to be optimal if there doesn't exist some other Hermitian linear complementary code dual  $[n, k, d']_4$  where  $d' > d$  [28].

**Definition 27.** An entanglement-assisted quantum code  $[[n, 2k - n + c, d; c]]$  imported from a classical  $[n, k, d]_4$  is said to be optimal if there doesn't exist some other entanglement-assisted quantum code  $[[n, 2k - n + c, d'; c]]$  where  $d' > d$  [28].

The field  $\mathbb{F}_4$  is used as an entanglement-assisted quantum code because there is an isomorphism between the quaternary elements and the binary strings used to represent Pauli gates:

$$\begin{array}{ccccc}
 x \in \mathbb{F}_4 & & \mathbb{Z}_2^2 & & \mathcal{P}^1 \\
 0 & \mapsto & 00 & \mapsto & I \\
 \omega & \mapsto & 01 & \mapsto & X \\
 \bar{\omega} & \mapsto & 10 & \mapsto & Z \\
 1 & \mapsto & 11 & \mapsto & Y
 \end{array} \tag{7.7}$$

The sole purpose of this thesis is to find the optimal Hermitian LCD codes.

## Chapter 8

# Programming Implementation of Thesis

In this chapter, we will share the construction method for finding optimal and non-optimal Hermitian linear complementary dual codes, the implementation of the Java code and the binary manipulation used to manipulate the quaternary vector. There are multiple parts to this program: vector generator, storage of all  $4^k$  linear combination of the codewords generated by the  $[n, k, d]_4$  code, determining whether or not  $GG^\dagger$  is nonsingular and the actual engine that pieces everything together. Furthermore, we will add the Bitwise operations used to achieve fast calculations for vector multiplication and the dot product of two vectors in  $\mathbb{F}_4$ .

### 8.1 Construction Method for Finding Optimal Hermitian Linear Complementary Codes

This thesis uses a similar construction to the one found in [28]. All  $[n, k, d]_4$  codes are equivalent in the sense that they all can be generated using a generator matrix in standard form [28]. Recall from Chapter 2 that a generator matrix  $G$  is said to be in standard form if

$$G = [I_{k \times k} \mid P_{k \times (n-k)}]_{k \times n}, \quad (8.1)$$

where  $I$  is the identity matrix and  $P$  is a  $k \times (n - k)$  matrix. We are able to assume the following characteristics for the submatrix  $P$  with that notation  $r_i$  as the  $i^{th}$  row in  $P$ :

- The top row of  $G$  is a fixed row vector of the form

$$\underbrace{[1 \ 0 \ \dots \ 0]}_{n-k-d+1} \underbrace{[1 \ 1 \ \dots \ 1]}_{d-1}. \quad (8.2)$$

- The first nonzero digit from the left is 1 for all vectors in the generator matrix (*i.e.*, neither  $\omega$  nor  $\bar{\omega}$ ).
- $\forall r_i : \text{WT}(r_i) \geq d - 1$ . Note that these weights don't have to be equal to  $d$  because the identity portion of the matrix includes a 1 making the overall minimum weight of the generator matrix vectors  $d$ .
- We use a lexicographic ordering so that the vectors are sorted in an ascending order when  $d \geq 3$ :

$$r_1 < r_2 < \dots < r_k, \quad (8.3)$$

or the following lexicographic ordering if  $d \not\geq 3$ :

$$r_1 \leq r_2 \leq \dots \leq r_k. \quad (8.4)$$

The types of codes obtained in this approach contains a set of all inequivalent codes [28]. This restriction will ensure that we only visit the unique possible combinations of vectors in the generator matrix because the generator matrix will generate the same codewords regardless of the ordering of the rows. Having this will *significantly* cut down the search space.

### 8.1.1 Comparing our Modified Method to the Original Procedure

The original construction was provided by Masaaki Harada in [28]. The approach was to construct the submatrix  $P$  and the generator matrix  $G$  dynamically in terms of the rows and columns. It will start with having the top row fixed. Then, for each iteration  $m = 2, \dots, k - 1$ , the number of rows increases as well as the columns, appropriately. The generator matrix will be constructed as follows:

$$\left[ \begin{array}{c|c} I_{m \times m} & \begin{matrix} r_1 \\ \vdots \\ r_m \end{matrix} \end{array} \right]. \quad (8.5)$$

Furthermore, at each step, the constructed code will be in the form of  $[n + m - k, m]_4$  and the minimum distance of the set of codewords is at least  $d$ . Our construction is slightly different in the sense that the length (*i.e.*,  $n$ ) of the code is constant and the

generator matrix will dynamically add rows that satisfy the orthogonality constraint. Also, each row that is added with its linear combinations must have a minimum weight of least  $d$ .

Chapter 9 of this thesis discusses the parameters [28] found along with other parameters obtained from other papers. Our contributions are found in Section 9.1.

## 8.2 The Programming Implementation of the Thesis

This thesis uses Java as the only programming language to implement an efficient approach to find optimal quaternary Hermitian linear complementary dual codes. MATLAB R2021a is used for the sake of confirming the matrices found are indeed invertible and have full rank. Furthermore, the programming code implemented doesn't rely on any external packages, just Java's standard libraries. Version 8 of Java is used but Java 7 can also be used with minor changes to the code. The program is designed to use a `long` to represent the codewords in the generator matrix  $G$  which is of length 64-bits. Since each quaternary value requires two bits to be represented, the maximum value for  $n$ , the length of the code, is 32. The quaternary values are mapped using the same mapping we have seen to the 2-bit binary values:  $0_4 \mapsto 00_2$ ,  $1_4 \mapsto 01_2$ ,  $\omega_4 \mapsto 10_2$  and  $\bar{\omega}_4 \mapsto 11_2$ .

### 8.2.1 Vector Generator

The vector generator focuses on generating the rows of the submatrix  $P$  of  $G$  and appending the appropriate row of the identity matrix to obtain a vector of length  $n$ . All vectors generated here ensure that the minimum distance requirement is met. All the vectors of the submatrix  $P$  will have weight of at least  $d - 1$ , which ensures that all rows of the  $G$  have weight of at least  $d$ . It has the option to generate all possible vectors without restriction or to ensure that all generated vectors have a 1 as the first nonzero digit.

### 8.2.2 Storing Linear Combinations

In addition to storing the generator matrix itself, all  $4^k$  linear combinations of the rows of the generator matrix must also be stored. The storage of these linear combination of the vectors in the generator matrix requires its own logic. This is because, in Java, a 1-Dimensional array uses an `int` as its index which is 32-bits. This is sufficient when  $k \leq 15$ . When  $k$  is larger than 15, we cannot store all linear combination due to

the maximum index constraint. To overcome this, a 2-Dimensional array is used to mimic multiple 1-Dimensional arrays stitched together. It is worth mentioning that the amount of RAM memory needed for mid-range  $k$  values is immense. For example, if  $k = 15$ , then  $4^{15} \times 64\text{bits} \approx 8.5\text{GB}$ . Furthermore, 8.8TB of memory is required when  $k = 20$  and 9PB<sup>1</sup> when  $k = 25$ .

### 8.2.3 Determining Whether if a Matrix is Invertible

To test if a matrix is nonsingular, it suffices to find whether it is invertible or not. A matrix is invertible if its determinant is not equal to 0. To do that, this thesis used Bareiss algorithm [04] which is named after Erwin H. Bareiss. This algorithm is a fraction-free method for computing the determinant of a matrix. Since our domain of values is the quaternary digits, *i.e.*, 0, 1,  $\omega$  and  $\bar{\omega}$ , we are able to use it. The issue with this algorithm is that it has the potential to encounter division by zero. In [36], the proposed a simple solution to overcome this which is to swap the current row which causes the issue with the row below it. In the case where swapping with all the rows still yield division by zero, then the matrix is not invertible. Furthermore, it is done with  $\mathcal{O}(n^3)$  operations [36]. The pseudocode of the algorithm is given in Algorithm 1. All the elementary operators on the elements in the matrix such as addition, multiplication and division are done in  $\mathbb{F}_4$ .

### 8.2.4 The Weight of a Quaternary Vector

One of the most frequent operations used in this thesis is finding the weight of a quaternary vector. Java 8's implementation for finding the number of 1's in a 64-bit long value is found in Listing 8.1 [46].<sup>2</sup>

```
i = i - ((i >>> 1) & 0x5555555555555555L);
i = (i & 0x3333333333333333L) + ((i >>> 2) & 0x3333333333333333L);
i = (i + (i >>> 4)) & 0x0F0F0F0F0F0F0F0FL;
i = i + (i >>> 8);
i = i + (i >>> 16);
i = i + (i >>> 32);
return (int)i & 0x7F;
```

Listing 8.1: Java 8's implementation for finding the number of 1's in a 64-bit long value

---

<sup>1</sup>9PB is 9000TB.

<sup>2</sup>The operator >>> is used instead of >> because we need to ensure that the new bit value of the far-left digit is set to 0 after the shifting.

It accepts `long i` and returns the weight of `i`. To find the weight of a quaternary vector, the standard approach doesn't work. For example, the quaternary vector  $[\bar{\omega} \ \omega \ 1 \ 0]$  has a weight of 3. The binary representation is  $[11 \ 10 \ 01 \ 00]$  which gives a weight of 4 with the standard approach. The focus is on  $\bar{\omega}$  which contains two 1's in its binary representation. The solution is to use the mask  $[01 \ 01 \ 01 \ 01]$ . First, shift  $v$  once to the right to obtain  $v'$  then logical-and it with the mask to get  $x_1$ . This will accurately capture how many  $\omega$  and  $\bar{\omega}$  present in the original vector  $v$ . Next, logical-and  $v$  with the mask to get  $x_2$ . This will capture how many 1 and  $\bar{\omega}$  in the original vector  $v$ . Lastly, logical-or  $x_1$  with  $x_2$  to get  $x$  which contains the correct number of 1's. Apply Java's implementation to obtain the weight of the vector  $v$ . This is shown in [Equation 8.6](#).

$$\begin{array}{rcl} \begin{array}{r} v' \ 01 \ 11 \ 00 \ 10 \\ \text{mask} \ 01 \ 01 \ 01 \ 01 \ \wedge \\ \hline 01 \ 01 \ 00 \ 00 \ x_1 \end{array} & \begin{array}{r} v \ 11 \ 10 \ 01 \ 00 \\ \text{mask} \ 01 \ 01 \ 01 \ 01 \ \wedge \\ \hline 01 \ 00 \ 01 \ 00 \ x_2 \end{array} & \text{WT}(v) = \text{WT}(x_1 \vee x_2) = 3. \quad (8.6) \end{array}$$

For an arbitrary quaternary vector  $v$ , we can use the approach in [Listing 8.2](#) which requires 21 operations to find the weight.

```
i = ((i >>> 1) & 0x5555555555555555L) | (i & 0x5555555555555555L);
i = i - ((i >>> 1) & 0x5555555555555555L);
i = (i & 0x3333333333333333L) + ((i >>> 2) & 0x3333333333333333L);
i = (i + (i >>> 4)) & 0x0F0F0F0F0F0F0F0FL;
i = i + (i >>> 8);
i = i + (i >>> 16);
i = i + (i >>> 32);
return (int)i & 0x7F;
```

Listing 8.2: Our implementation for finding the number of 1's in a quaternary value of length 32

### 8.2.5 The Core Engine of the Program

The core engine that starts the search is done recursively. It will accept the current row index to be populated in the generator matrix as well as a vector generator to be associated with the current run. The pseudocode is found in [Algorithm 2](#). The basic idea is to use backtracking when trying to choose the vectors. We first start with placing smallest valid vector which contains  $d - 1$  1's on the right side in the first row of the generator matrix. We also store the linear combinations of the vector. Next, generate the next potential vector which is the next lexicographical vector. We

---

**Algorithm 1:** Bareiss Algorithm which takes in a quaternary  $n \times n$  matrix represented by binary values and return its determinant which can either be 0, 1,  $\omega$  or  $\bar{\omega}$  or  $00_2$ ,  $01_2$ ,  $10_2$  and  $11_2$  in binary, respectively

---

**Input** :  $M_{n \times n}$  – A square matrix to find the determinant of.

**Output:**  $\det(M_{n \times n}) \in \{0, 1, 2, 3\}$

```

1 Function bareissAlgorithm( $M_{n \times n}$ ):
2    $pivot \leftarrow 1$ 
3   for  $k \leftarrow 0$  to  $n - 2$  do
4     if  $M[k][k] = 0$  then           // Division-by-zero encountered
5       Avoid division-by-zero by swapping current row with another one
6        $isFixed \leftarrow \text{false}$ 
7       for  $r \leftarrow k + 1$  to  $n - 1$  do
8         if  $M[r][k] \neq 0$  then
9           swap row  $k$  with  $r$ 
10           $isFixed \leftarrow \text{true}$ 
11          break
12        end
13      end
14    end
15    if  $isFixed = \text{false}$  then      // We still have division-by-zero
16      matrix is not invertible
17      return 0
18    end
19    for  $i \leftarrow k + 1$  to  $n - 1$  do
20      for  $j \leftarrow k + 1$  to  $n - 1$  do
21         $result \leftarrow \frac{M[i][j] \times M[k][k] - M[i][k] \times M[k][j]}{pivot}$ 
22         $M[i][j] \leftarrow result$ 
23      end
24    end
25     $pivot \leftarrow M[k][k]$ 
26  end
27  return  $M[n - 1][n - 1] \in \{1, 2, 3\}$ 
28 End Function

```

---



check if it is orthogonal to all the linear combinations found. In the case where it is, then we add it to the generator matrix in the second row along with the new linear combinations. We need to populate the third row in the matrix. It will be the next vector lexicography. In the case where it doesn't satisfy the minimum weight requirement or it is not orthogonal with at least one of the combinations, it will be skipped and we check the next vector. Assume we checked all possible vectors until we reached the vector  $[1 \ \bar{\omega} \ \bar{\omega} \ \dots \ \bar{\omega} \ \bar{\omega}]^3$ . Since there are no more other vectors to check, we realize that we cannot have three vectors that satisfy the code requirement. We then backtrack to the second row and change it by choosing the next lexicographical vector.

### 8.2.6 Verifying Results with MATLAB

MATLAB supports Galois fields of the form  $2^q$  where  $1 \leq q \leq 16$ . In our case,  $q = 2$ . It also has a method `ctranspose(A)` (or equivalently `A'`) which computes the conjugate transpose of the matrix `A`. Unfortunately, these methods work for matrices that are not Galois field matrices. In the case where `A` is a Galois field matrix, then applying `ctranspose(A)` will result in `A` unmodified [42]. To overcome this, we need to manually take the complex conjugation of each element in the matrix then transpose the matrix. An example of finding the complex conjugate transpose of the generator matrix is found in Listing 8.3.

```
% An example of how to check the conjugate transpose of [12, 6, 5]
% generator matrix code.
q = 2;
A = gf([
    1 0 0 0 0 0 0 0 1 1 1 1;
    0 1 0 0 0 0 0 1 0 1 1 2;
    0 0 1 0 0 0 0 1 1 0 2 1;
    0 0 0 1 0 0 0 1 1 2 1 0;
    0 0 0 0 1 0 0 1 2 1 0 1;
    0 0 0 0 0 1 1 0 0 1 1 3
], q);
APrime = transpose(A.^2); % The conjugate transpose of A.
det(A * APrime) % Outputs 1
rank(A * APrime) % Outputs 6
```

Listing 8.3: A simple MATLAB script for testing the conjugate transpose of the generator matrix `A`

---

<sup>3</sup>This is the row of the submatrix of  $G$ , *i.e.*, without the identity portion

---

**Algorithm 2:** The main engine of the program that starts the search to conclude whether or not a  $[n, k, d]_4$  exists

---

**Data** :  $n, k, d, G_{k \times n}$

**Input** :  $r$  – The current row of  $G$  being filled and it is zero-based,  $0 \leq r \leq k$ .

*vectorGenerator* – The vector generator that generates potential codewords to insert in  $G$ .

**Output** : **true** if  $[n, k, d]_4$  code exists, **false** otherwise

```

1 Function isCodeValid( $r, vectorGenerator$ ):
2   if  $r \geq k$  then                                     // Base case
3     if  $\det(G \times G^t) \neq 0$  then                       // Valid generator matrix
4       return true
5     else
6        $G[r - 1] \leftarrow 0$ 
7       return false
8     end
9   end
10  while true do
11     $vector \leftarrow vectorGenerator.nextVector()$ 
12    // Check if vector generator cannot generate new vectors
13    if  $\neg vectorGenerator.isValid()$  then
14      return false
15    end
16    // Check if vector is orthogonal to  $4^r$  codewords found
17    if  $isOrthogonal(vector)$  then
18       $G[r] \leftarrow vector$ 
19      // Create a new instance of the vector generator and
        start at the value of  $vector + 1$ 
20       $x \leftarrow new\ vectorGenerator(vector + 1)$ 
21      if  $isCodeValid(r + 1, x)$  then                       // Recursive call
22        return true
23      end
24    end
25  end
26 End Function

```

---

## Chapter 9

### Previous work and new Results

Recently, there has been an increase in focus on Hermitian linear complementary dual codes because they can be converted to an entanglement-assisted quantum code. Hence, we will review relevant literature regarding optimal Hermitian LCD codes:

Year	Paper	Description
February 2004	[10]	Determined the upper and lower bounds of optimal quaternary codes but not Hermitian LCD codes for $n \leq 12$
October 2006	[13]	The establishment of the entanglement-assisted quantum error correction formalism
March 2009	[09]	Determined all optimal quaternary codes but not Hermitian LCD codes for $n \leq 13$ except one parameter
January 2011	[34]	Determined suboptimal and optimal quaternary Hermitian LCD codes for $n \leq 15$ . About 50% of the parameters are optimal
February 2013	[35]	Contributed a few more suboptimal and optimal quaternary Hermitian LCD codes parameters compared to [34] for $n \leq 15$ . The parameters are: $[9, 4, 5]_4$ , $[9, 5, 4]_4$ , $[10, 4, 6]_4$ , $[11, 5, 6]_4$ , $[11, 6, 5]_4$ , $[12, 2, 9]_4$ , $[12, 8, 4]_4$ , $[13, 2, 10]_4$ and $[15, 13, 2]_4$ .
August 2013	[05]	Similar to [09] but now the length is extended to $n \leq 15$
March 2014	[20]	Focused on optimal quaternary Hermitian LCD codes with $k = 4$ and large $n$ values
June 2015	[38]	A significant improvement for optimal quaternary Hermitian LCD codes and almost all optimal parameters for $n \leq 15$ are found. Also, the range was extended up to $n \leq 20$ .

April 2018	[08]	Proved some quaternary codes (not Hermitian LCD) do not exist as well as showed that a $[12, 6, 6]_4$ quaternary code exists
June 2018	[33]	Contributed a few more suboptimal and optimal quaternary Hermitian LCD codes parameters compared to [38] for $n \leq 15$ .
December 2019	[28]	Showed six quaternary Hermitian LCD codes parameters exist: $[14, 6, 7]_4$ , $[15, 7, 7]_4$ , $[17, 6, 9]_4$ , $[17, 7, 8]_4$ , $[19, 7, 9]_4$ and $[20, 7, 10]_4$ . Also, it was mentioned that a quaternary Hermitian LCD code $[12, 6, 6]_4$ doesn't exist.
October 2020	[39]	A significant improvement for extending optimal quaternary Hermitian LCD codes to $n \leq 25$ with the following optimal codes: $[18, 7, 9]_4$ , $[19, 7, 9]_4$ and $[20, 7, 10]_4$
January 2021	[27]	Found these suboptimal quaternary Hermitian LCD codes parameters: $[21, 8, 9]_4$ , $[21, 10, 8]_4$ and $[21, 11, 7]_4$ as well as the following optimal parameter: $[22, 8, 10]_4$ . The paper also found other suboptimal parameters for $n \geq 26$ .
June 2021	[02]	Found the following optimal quaternary Hermitian LCD codes parameters: $[19, 9, 8]_4$ , $[20, 7, 10]_4$ and $[23, 7, 12]_4$

All the above papers either [www.codetables.de](http://www.codetables.de) and/or Magma to obtain their results. Code tables is a collection of largest known distance for various types of codes [24]. Magma is a well-trusted package designed for coding theory and other mathematical branches that use finite fields [11]. It contains a colossal database of codes and parameters as well as offers clever support for building codes from already existing ones found. In this thesis, we will focus on the optimal parameters found from [02, 27, 28, 33, 34, 38, 39]. We ran the optimal parameters found for  $2 \leq k \leq 14$  and  $2 \leq n \leq 14$  using our implementation and matched almost all of the parameters found. There seems to be some typos and inaccuracies we will also mention as well as the accurate optimal parameters found. The parameters:

$[12, 7, 5]_4$	from [38]	should be $[12, 7, 4]_4$	because of [33]	,
$[13, 11, 3]_4$	from [39]	should be $[13, 11, 2]_4$	because of [33, 38]	,
$[14, 10, 4]_4$	from [33, 38, 39]	should be $[14, 10, 3]_4$	we will explain next	,
$[14, 11, 4]_4$	from [39]	should be $[14, 11, 3]_4$	because of [34, 38]	.

The parameter  $[12, 7, 5]_4$  should have 4 as its largest distance because [33] stated

that the maximum distance is 4. The parameter  $[13, 11, 3]_4$  has a large distance of 2 and both [33, 38] concluded this. Moreover,  $[14, 11, 4]_4$  should have 3 for largest distance because [34, 38] established this fact. Our program also matched the modified largest distance for these three parameters. The optimal distance of  $[14, 10, 4]_4$  was first mentioned in [38]. It didn't include a generator matrix nor the other papers it cited. Paper [33] also has the same parameter without a generator matrix. There exists a  $[14, 10, 4]_4$  quaternary code which is given in Equation 9.1 with its weight enumerator in Equation 9.2.

$$G_{[14, 10, 4]_4} = \left[ \begin{array}{cccccccccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \omega & \bar{\omega} \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \bar{\omega} & \omega \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \omega & \bar{\omega} \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \bar{\omega} & \omega \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & \omega & \omega \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & \bar{\omega} & \bar{\omega} \end{array} \right] \quad (9.1)$$

$$\begin{aligned} \text{WT}(x)_{[14, 10, 4]_4} = & 1 + 483x^4 + 1,680x^5 + 8,526x^6 + 29,568x^7 + 77,511x^8 + 151,200x^9 \\ & + 235,620x^{10} + 247,296x^{11} + 191,541x^{12} + 86,352x^{13} + 18,798x^{14} \end{aligned} \quad (9.2)$$

However,  $G_{[14, 10, 4]_4}$  is not a Hermitian LCD because  $\text{DET} \left( G_{[14, 10, 4]_4} G_{[14, 10, 4]_4}^\dagger \right) = 0$ . Our program brute-forced through all the possible combinations based on the construction we mentioned in the previous chapter and found 7995 generator matrices, all of which had determinant of 0.

## 9.1 New Suboptimal and Optimal Parameters

Our program was able to conclude there were nine suboptimal parameters found. Furthermore, we brute-forced through the possibility of obtaining  $[17, 11, 6]_4$  and  $[17, 13, 4]_4$  but concluded the known largest distances (which are 5 and 3, respectively) are the optimal distances. They are found in Table 9.2. We also included the generator matrices and weight enumerators of the codes mentioned.

$\begin{smallmatrix} k \\ n \end{smallmatrix}$	8	9	10	11	12	13	14
17				5		3	
22		$\frac{9}{10}$		$\frac{8}{9}$	$\frac{7}{8}$		
23	$\frac{11}{12}$		$\frac{9}{10}$			$\frac{7}{8}$	
24			$\frac{10}{11}$	$\frac{9}{10}$			$\frac{7}{8}$
25					$\frac{9}{10}$	$\frac{8}{9}$	

Table 9.2: Suboptimal and optimal parameters we have found. The notation  $\frac{x}{y}$  is the lower bound ( $x$ ) and upper bound  $y$  (both inclusive) of the largest distance of the code

$$G_{[22,9,9]_4} = \left[ \begin{array}{c|cccccccccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right] \begin{array}{cccccccccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & \omega & \omega & \omega & \omega & \omega & \omega \\ 0 & 0 & 0 & 1 & \omega & 1 & \omega & 0 & 1 & \omega & 0 & 1 & \omega & \omega & \omega & \omega \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & \omega & 1 & \omega & \omega & \omega & \omega & \omega \\ 0 & 0 & 1 & 0 & \omega & 1 & 0 & \omega & \omega & \omega & \omega & \omega & \omega & \omega & \omega & \omega \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & \omega & \omega & \omega & \omega & \omega & \omega & \omega & \omega & \omega \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & \omega & \omega & \omega & \omega & \omega & \omega & \omega \\ 0 & 1 & 0 & 1 & 0 & \omega & 1 & \omega & \omega & \omega & \omega & \omega & \omega & \omega & \omega & \omega \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & \omega & 0 & \omega & \omega & \omega & \omega & \omega & \omega \end{array} \quad (9.3)$$

$$\begin{aligned} \text{WT}(x)_{[22,9,9]_4} = & 1 + 414x^9 + 954x^{10} + 2,334x^{11} + 5,781x^{12} + 12,483x^{13} \\ & + 22,920x^{14} + 35,016x^{15} + 45,801x^{16} + 48,888x^{17} \\ & + 41,622x^{18} + 28,194x^{19} + 13,161x^{20} + 3,999x^{21} + 576x^{22} \end{aligned} \quad (9.4)$$



$$G_{[23,10,9]_4} = \left[ \begin{array}{c|cccccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \right| \begin{array}{cccccccccccc} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & \omega & \omega & \omega \\ 0 & 0 & 0 & 1 & \omega & 1 & \omega & 0 & 1 & \omega & 0 & 1 & \omega \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & \omega & 1 & \omega & \bar{\omega} \\ 0 & 0 & 1 & 0 & \omega & 1 & 0 & \omega & \bar{\omega} & \omega & \omega & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & \bar{\omega} & \omega & \omega & 1 & 0 & \omega \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & \bar{\omega} & \omega & 1 & \bar{\omega} \\ 0 & 1 & 0 & 1 & 0 & \omega & 1 & \omega & \omega & \bar{\omega} & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & \omega & 0 & \bar{\omega} & \omega & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & \omega & 1 & 1 & \omega & 1 & 1 & 0 \end{array} \right] \quad (9.9)$$

$$\begin{aligned} \text{WT}(x)_{[23,10,9]_4} &= 1 + 540x^9 + 1,398x^{10} + 4,032x^{11} + 11,040x^{12} + 27,237x^{13} \\ &\quad + 57,126x^{14} + 102,417x^{15} + 155,403x^{16} + 193,542x^{17} + 197,562x^{18} \\ &\quad + 156,714x^{19} + 92,772x^{20} + 38,169x^{21} + 9,498x^{22} + 1,125x^{23} \end{aligned} \quad (9.10)$$

$$G_{[23,13,7]_4} = \left[ \begin{array}{c|cccccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \right| \begin{array}{cccccccccccc} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & \omega & \omega & \bar{\omega} & & & & \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & \bar{\omega} & \bar{\omega} & \omega & & & & \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & \omega & & & & \\ 0 & 0 & 1 & \omega & 1 & 0 & \bar{\omega} & \omega & 0 & 1 & & & & \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & \omega & \bar{\omega} & \omega & & & & \\ 0 & 1 & 0 & 1 & 0 & 0 & \bar{\omega} & 1 & \bar{\omega} & \bar{\omega} & & & & \\ 0 & 1 & 0 & \omega & \omega & 1 & \bar{\omega} & \bar{\omega} & \bar{\omega} & \omega & & & & \\ 0 & 1 & 1 & 0 & 1 & \bar{\omega} & 0 & 0 & \omega & \omega & & & & \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & \bar{\omega} & \omega & \bar{\omega} & & & & \\ 1 & 0 & 0 & 1 & 0 & 1 & \omega & 0 & 1 & \omega & & & & \\ 1 & 0 & 0 & \bar{\omega} & 1 & \omega & 0 & \bar{\omega} & 0 & 1 & & & & \\ 1 & 0 & 1 & \bar{\omega} & \bar{\omega} & 1 & 0 & 0 & 0 & \omega & & & & \end{array} \right] \quad (9.11)$$

$$\begin{aligned} \text{WT}(x)_{[23,13,7]_4} &= 1 + 891x^7 + 3,606x^8 + 15,609x^9 + 64,101x^{10} + 225,780x^{11} \\ &\quad + 683,346x^{12} + 1,737,903x^{13} + 3,727,791x^{14} + 6,722,268x^{15} \\ &\quad + 10,072,965x^{16} + 12,424,119x^{17} + 12,411,699x^{18} + 9,816,240x^{19} \\ &\quad + 5,900,466x^{20} + 2,527,809x^{21} + 686,361x^{22} + 87,909x^{23} \end{aligned} \quad (9.12)$$



[illegible]

$$\begin{aligned} \text{WT}(x)_{[24, 10, 10]_4} = & 1 + 798x^{10} + 1,704x^{11} + 5,280x^{12} + 14,445x^{13} + 34,560x^{14} \\ & + 69,357x^{15} + 118,953x^{16} + 166,746x^{17} + 194,886x^{18} + 184,170x^{19} \\ & + 136,896x^{20} + 78,681x^{21} + 32,364x^{22} + 8,673x^{23} + 1,062x^{24} \end{aligned} \quad (9.14)$$

[illegible]

$$\begin{aligned} \text{WT}(x)_{[24, 11, 9]_4} = & 1 + 708x^9 + 2,085x^{10} + 6,795x^{11} + 20,880x^{12} \\ & + 58,257x^{13} + 139,062x^{14} + 278,934x^{15} + 471,624x^{16} \\ & + 669,798x^{17} + 778,473x^{18} + 735,975x^{19} + 549,438x^{20} \\ & + 313,653x^{21} + 130,092x^{22} + 34,056x^{23} + 4,473x^{24} \end{aligned} \quad (9.16)$$



$$\begin{aligned}
\text{WT}(x)_{[25, 12, 9]_4} = & 1 + 942x^9 + 3,174x^{10} + 11,715x^{11} + 40,401x^{12} + 122,373x^{13} \\
& + 317,172x^{14} + 701,367x^{15} + 1,310,532x^{16} + 2,083,821x^{17} \\
& + 2,775,750x^{18} + 3,060,105x^{19} + 2,761,647x^{20} + 1,973,343x^{21} \\
& + 1,076,928x^{22} + 420,477x^{23} + 105,051x^{24} + 12,417x^{25}
\end{aligned} \tag{9.20}$$

$$G_{[25, 13, 8]_4} = \left[ \begin{array}{cccccccccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \middle| \begin{array}{cccccccccccc} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & \omega & \omega & \bar{\omega} & \bar{\omega} & \bar{\omega} & \bar{\omega} & \bar{\omega} \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & \bar{\omega} & \bar{\omega} & \omega & \omega & \omega & \omega & \omega \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & \omega & \bar{\omega} & \bar{\omega} & \bar{\omega} & \bar{\omega} \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & \omega & 1 & \bar{\omega} & \omega & \bar{\omega} & \bar{\omega} & \bar{\omega} & \bar{\omega} \\ 0 & 0 & 1 & 0 & 1 & 0 & \omega & 1 & 1 & 0 & 1 & \omega & \bar{\omega} & \bar{\omega} & \bar{\omega} \\ 0 & 0 & 1 & 1 & 0 & \omega & \omega & 1 & \bar{\omega} & \omega & \bar{\omega} & \bar{\omega} & \bar{\omega} & \bar{\omega} & \bar{\omega} \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & \omega & 0 & \bar{\omega} & \bar{\omega} & \bar{\omega} & \omega & \bar{\omega} & \omega \\ 0 & 1 & 0 & 0 & 1 & 0 & \bar{\omega} & \omega & \bar{\omega} & \omega & \bar{\omega} & \omega & 0 & \omega & \bar{\omega} \\ 0 & 1 & 0 & 1 & \omega & \omega & 1 & 0 & \bar{\omega} & \bar{\omega} & \bar{\omega} & 0 & 1 & \bar{\omega} & \omega \\ 0 & 1 & 1 & 1 & 1 & \omega & \bar{\omega} & 0 & \bar{\omega} & 1 & \bar{\omega} & \bar{\omega} & \omega & \bar{\omega} & \omega \\ 1 & 0 & 0 & 0 & 0 & 1 & \omega & 0 & \bar{\omega} & \omega & 1 & \bar{\omega} & \bar{\omega} & \bar{\omega} & \bar{\omega} \\ 1 & 0 & 0 & 1 & \omega & 0 & \omega & \bar{\omega} & \bar{\omega} & 1 & \omega & \bar{\omega} & \bar{\omega} & \bar{\omega} & \bar{\omega} \end{array} \right] \tag{9.21}$$

$$\begin{aligned}
\text{WT}(x)_{[25, 13, 8]_4} = & 1 + 1,050x^8 + 3,408x^9 + 14,523x^{10} + 53,154x^{11} \\
& + 177,600x^{12} + 510,885x^{13} + 1,276,797x^{14} + 2,776,260x^{15} \\
& + 5,186,355x^{16} + 8,260,923x^{17} + 11,074,101x^{18} \\
& + 12,301,794x^{19} + 11,124,636x^{20} + 7,959,195x^{21} \\
& + 4,300,659x^{22} + 1,649,304x^{23} + 394,614x^{24} + 43,605x^{25}
\end{aligned} \tag{9.22}$$

To the best of our knowledge, the tightest suboptimal and optimal largest distances for  $n \leq 25$  are found in [Table 9.3](#).

$n \backslash k$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
3	3	2																							
4	3	2	1																						
5	5	3	2	2																					
6	5	4	3	2	1																				
7	7	5	4	3	2	2																			
8	7	6	5	4	3	2	1																		
9	9	6	6	5	4	3	2	2																	
10	9	7	6	6	5	4	3	2	1	1															
11	11	8	7	6	6	5	4	3	2	2	1														
12	11	9	8	7	6	5	4*	4	3	2	2	1													
13	13	10	9	8	7	6	5	4	4	3	2*	2													
14	13	10	9	8	$\frac{7}{8}$	7	6	5	4	3*	3*	2	1												
15	15	11	10	9	8	7	7	6	5	4	4	3	2	2											
16	15	12	11	10	9	8	$\frac{7}{8}$	$\frac{6}{7}$	6	5	5	4	3	2	1										
17	17	13	12	11	$\frac{9}{10}$	9	8	$\frac{7}{8}$	$\frac{6}{7}$	6	<b>5</b>	4	<b>3</b>	3	2	2									
18	17	14	13	$\frac{11}{12}$	$\frac{10}{11}$	$\frac{9}{10}$	9	$\frac{8}{9}$	$\frac{7}{8}$	$\frac{6}{7}$	$\frac{5}{6}$	5	4	3	3	2	1								
19	19	14	13	$\frac{12}{13}$	11	$\frac{10}{11}$	9	$\frac{8}{9}$	8	7	$\frac{6}{7}$	$\frac{5}{6}$	5	4	3	3	2	2							
20	19	15	14	13	12	$\frac{11}{12}$	10	$\frac{8}{9}$	$\frac{7}{8}$	$\frac{6}{7}$	$\frac{5}{6}$	$\frac{4}{5}$	$\frac{3}{4}$	3	2	2	1	1							
21	21	16	15	14	12	12	$\frac{10}{11}$	$\frac{9}{10}$	$\frac{8}{9}$	$\frac{7}{8}$	6	$\frac{5}{6}$	5	4	3	2	2	2	1						
22	22	17	15	14	13	$\frac{12}{13}$	$\frac{11}{12}$	10	<b>9</b>	$\frac{8}{9}$	$\frac{7}{8}$	<b>7</b>	$\frac{6}{7}$	6	$\frac{5}{6}$	$\frac{4}{5}$	4	3	2	2	2	1			
23	23	18	16	15	14	13	12	<b>11</b>	$\frac{9}{10}$	<b>9</b>	$\frac{8}{9}$	$\frac{7}{8}$	<b>7</b>	6	$\frac{5}{6}$	$\frac{4}{5}$	4	3	2	2	2	1			
24	24	18	17	16	15	14	$\frac{12}{13}$	$\frac{11}{13}$	$\frac{10}{12}$	<b>10</b>	$\frac{9}{11}$	$\frac{8}{9}$	<b>7</b>	<b>7</b>	$\frac{6}{7}$	$\frac{5}{6}$	$\frac{4}{5}$	4	3	2	2	2	1		
25	25	19	18	17	15	$\frac{14}{15}$	$\frac{13}{14}$	$\frac{12}{13}$	$\frac{11}{13}$	$\frac{10}{12}$	$\frac{9}{11}$	<b>9</b>	<b>8</b>	7	$\frac{6}{8}$	$\frac{5}{7}$	$\frac{4}{6}$	4	3	2	2	2	2		

Table 9.3: The updated table for optimal values for  $n \leq 25$  all along with the results found in this thesis in bold. The values with an asterisk denotes the accurate largest distances

## Chapter 10

# Thesis Conclusion and Future Work

This thesis started with discussing the basics of classical linear binary codes error correction. Then, we discussed the essence of quantum computing by going over the fundamental mathematics required as well as quantum circuits. Quantum phenomena such as superposition and entanglement were discussed next. The subsequent chapter examined quantum error correction and explained Shor's code in-depth. The next one was dedicated to entanglement-assisted quantum error correction which introduced the formalism of using ebits to resolve the anticommutativity constraint. We then discussed the implementation of the program that was created to search through possible quaternary code parameters. The results chapter shed light on previous works as well as the new suboptimal and optimal entanglement-assisted quantum codes from quaternary Hermitian linear complementary dual codes obtained through the program implemented in this thesis.

### 10.1 Summary of Contribution

One of the main contributions of this thesis is creating a Java program specifically designed for finding optimal quaternary Hermitian linear complementary dual codes based on the  $n$ ,  $k$  and  $d$  values of the code to be examined. The program will give the matrix if the parameters exist or displays that the code doesn't exist. Furthermore, the implementation is hosted on GitHub and can be accessed through

<https://github.com/Aljumaily/MScThesis>.

We used the above program to contribute the following suboptimal parameters:  $[22, 9, 9 \text{ or } 10]_4$ ,  $[22, 12, 7 \text{ or } 8]_4$ ,  $[23, 8, 11 \text{ or } 12]_4$ ,  $[23, 10, 9 \text{ or } 10]_4$ ,  $[23, 13, 7 \text{ or } 8]_4$ ,  $[24, 10, 10 \text{ or } 11]_4$ ,  $[24, 11, 9 \text{ or } 10]_4$ ,  $[24, 14, 7 \text{ or } 8]_4$ ,  $[25, 12, 9 \text{ or } 10]_4$ ,  $[25, 13, 8 \text{ or } 9]_4$ , as well as the optimal parameters for  $[17, 11, 5]_4$  and  $[17, 13, 3]_4$ .

## 10.2 Future Work

As for future work, we believe that using C as the main programming language would allow for larger lengths of  $n$  to be tested. This is because the GNU Compiler Collections (GCC) compiler supports 128-bit values since version 4.7.2.

There should be the functionality of having the program to start with pre-populated rows in the generator matrix and continuing the search from there. This will be useful for two reasons: the first is that we can concurrently run multiple instances of the same program and give each process different starting point(s) for some rows to speed up the overall time required. Also, it will aid in having the ability to save the current state of the program and run it again at a later time.

Another point to keep in mind is to improve the bit manipulation used when multiplying matrices and finding the determinant. It could be optimized further. Also, there isn't a method for finding the parity check matrix for some generator matrix. Such addition will smoothen the transition from the classical quaternary form to the set of generators used in an entanglement-assisted quantum code. The implementation can be modified to also include the option of finding quaternary codes, not just hermitian quaternary LCD codes.

Lastly, there could be the ability to not store the linear combinations in RAM but recalculate them every time they are needed. Although it will increase the time, it will not require petabytes of RAM when  $n$  is 25 or larger.

# Bibliography

- [00] A.D. Aczel. *Entanglement: The Greatest Mystery in Physics*. Raincoast Books, 2002. ISBN: 9781551926476. URL: <https://books.google.ca/books?id=F-7I4ioxbzC>.
- [01] Yashar Alizadeh. “Quantum Error Correction Methods”. MA thesis. Eastern Mediterranean University, 2016. URL: <https://pdfs.semanticscholar.org/4502/ebb59f945ccacea060a6fa55091914950e0c.pdf>.
- [02] A. Allahmadi et al. “New constructions of entanglement-assisted quantum codes”. In: *Cryptography and Communications* (June 2021). ISSN: 1936-2455. DOI: [10.1007/s12095-021-00499-7](https://doi.org/10.1007/s12095-021-00499-7). URL: <https://doi.org/10.1007/s12095-021-00499-7>.
- [03] H. Anton and C. Rorres. *Elementary Linear Algebra: Applications Version 11<sup>th</sup> Edition*. Wiley, 2013. ISBN: 9781118474228.
- [04] Erwin H Bareiss. “Sylvester’s identity and multistep integer-preserving Gaussian elimination”. In: *Mathematics of computation* 22.103 (1968), pp. 565–578.
- [05] Daniele Bartoli et al. *The nonexistence of an additive quaternary [15,5,9]-code*. 2013. arXiv: [1308.2108](https://arxiv.org/abs/1308.2108) [math.CO].
- [06] D.J. Baylis. *Error Correcting Codes: A Mathematical Introduction*. Chapman and Hall/CRC, 1997. ISBN: 9780412786907.
- [07] János A. Bergou and Mark Hillery. “Decoherence and Quantum Error Correction”. In: *Graduate Texts in Physics*. Springer New York, 2013, pp. 133–148. DOI: [10.1007/978-1-4614-7092-2\\_9](https://doi.org/10.1007/978-1-4614-7092-2_9). URL: [https://doi.org/10.1007/978-1-4614-7092-2\\_9](https://doi.org/10.1007/978-1-4614-7092-2_9).
- [08] Juergen Bierbrauer, Stefano Marcugini, and Fernanda Pambianco. *Additive quaternary codes related to exceptional linear quaternary codes*. 2018. arXiv: [1804.02468](https://arxiv.org/abs/1804.02468) [math.CO].
- [09] Jürgen Bierbrauer et al. “Short Additive Quaternary Codes”. In: *IEEE Transactions on Information Theory* 55.3 (2009), pp. 952–954. DOI: [10.1109/TIT.2008.2011447](https://doi.org/10.1109/TIT.2008.2011447).
- [10] A Blokhuis and A.E Brouwer. “Small additive quaternary codes”. In: *European Journal of Combinatorics* 25.2 (2004). In memory of Jaap Seidel, pp. 161–167. ISSN: 0195-6698. DOI: [https://doi.org/10.1016/S0195-6698\(03\)00096-9](https://doi.org/10.1016/S0195-6698(03)00096-9). URL: <https://www.sciencedirect.com/science/article/pii/S0195669803000969>.
- [11] WIEB BOSMA, JOHN CANNON, and CATHERINE PLAYOUST. “The Magma Algebra System I: The User Language”. In: *Journal of Symbolic Computation* 24.3 (1997), pp. 235–265. ISSN: 0747-7171. DOI: <https://doi.org/10.1006/jsco.1996.0125>. URL: <https://www.sciencedirect.com/science/article/pii/S074771719690125X>.

- [12] Todd Brun, Igor Devetak, and Min-Hsiu Hsieh. “Correcting Quantum Errors with Entanglement”. In: *Science* 314.5798 (2006), pp. 436–439. DOI: [10.1126/science.1131563](https://doi.org/10.1126/science.1131563). eprint: <https://www.science.org/doi/pdf/10.1126/science.1131563>. URL: <https://www.science.org/doi/abs/10.1126/science.1131563>.
- [13] Todd Brun, Igor Devetak, and Min-Hsiu Hsieh. “Correcting Quantum Errors with Entanglement”. In: *Science* 314.5798 (2006), pp. 436–439. ISSN: 0036-8075. DOI: [10.1126/science.1131563](https://doi.org/10.1126/science.1131563). eprint: <https://science.sciencemag.org/content/314/5798/436.full.pdf>. URL: <https://science.sciencemag.org/content/314/5798/436>.
- [14] A Robert Calderbank et al. “Quantum error correction via codes over GF (4)”. In: *IEEE Transactions on Information Theory* 44.4 (1998), pp. 1369–1387.
- [15] Claude Carlet et al. “Linear Codes Over  $\mathbb{F}_q$  Are Equivalent to LCD Codes for  $q \neq 3$ ”. In: *IEEE Transactions on Information Theory* 64.4 (2018), pp. 3010–3017. DOI: [10.1109/TIT.2018.2789347](https://doi.org/10.1109/TIT.2018.2789347).
- [16] John G. Cramer. *The Quantum Handshake*. Springer International Publishing, 2016. DOI: [10.1007/978-3-319-24642-0](https://doi.org/10.1007/978-3-319-24642-0). URL: <https://doi.org/10.1007/978-3-319-24642-0>.
- [17] Simon J Devitt, William J Munro, and Kae Nemoto. “Quantum error correction for beginners”. In: *Reports on Progress in Physics* (2013). DOI: [10.1088/0034-4885/76/7/076001](https://doi.org/10.1088/0034-4885/76/7/076001). URL: <https://doi.org/10.1088/0034-4885/76/7/076001>.
- [18] Artur Ekert, PM Hayden, and Hitoshi Inamori. “Basic concepts in quantum computation”. In: *Coherent atomic matter waves*. Springer, 2001, pp. 661–701.
- [19] Artur Ekert et al. “Geometric quantum computation”. In: *Journal of modern optics* 47.14-15 (2000), pp. 2501–2513.
- [20] Qiang Fu et al. “Entanglement-assisted Quantum Codes of Distance Four Constructed from Caps in PG(5,4) and PG(6,4)”. In: *Proceedings of the 2014 International Conference on Future Computer and Communication Engineering*. Atlantis Press, 2014/03, pp. 153–156. ISBN: 978-94-6252-005-9. DOI: <https://doi.org/10.2991/icfcce-14.2014.38>. URL: <https://doi.org/10.2991/icfcce-14.2014.38>.
- [21] Rubén Jesús García-Hernández and Dieter Kranzlmüller. “NOMAD VR: Multiplatform virtual reality viewer for chemistry simulations”. In: *Computer Physics Communications* 237 (2019), pp. 230–237. ISSN: 0010-4655. DOI: <https://doi.org/10.1016/j.cpc.2018.11.013>. URL: <http://www.sciencedirect.com/science/article/pii/S0010465518304053>.
- [22] Daniel Gottesman. *An Introduction to Quantum Error Correction and Fault-Tolerant Quantum Computation*. 2009. arXiv: [0904.2557](https://arxiv.org/abs/0904.2557) [quant-ph].
- [23] Daniel Gottesman. *Stabilizer Codes and Quantum Error Correction*. 1997. arXiv: [quant-ph/9705052](https://arxiv.org/abs/quant-ph/9705052) [quant-ph].
- [24] Markus Grassl. *Codetables.de*. URL: <http://codetables.de/>.
- [25] R. W. Hamming. “Error Detecting and Error Correcting Codes”. In: *Bell System Technical Journal* 29.2 (1950), pp. 147–160. DOI: [10.1002/j.1538-7305.1950.tb00463.x](https://doi.org/10.1002/j.1538-7305.1950.tb00463.x).



- [26] Richard W Hamming. “Error detecting and error correcting codes”. In: *The Bell system technical journal* 29.2 (1950), pp. 147–160.
- [27] Masaaki Harada. “Construction of binary LCD codes, ternary LCD codes and quaternary Hermitian LCD codes”. In: *CoRR* abs/2101.11821 (2021). arXiv: [2101.11821](https://arxiv.org/abs/2101.11821). URL: <https://arxiv.org/abs/2101.11821>.
- [28] Masaaki Harada. “Some optimal entanglement-assisted quantum codes constructed from quaternary Hermitian linear complementary dual codes”. In: *International Journal of Quantum Information* 17.07 (2019), p. 1950053. DOI: [10.1142/S0219749919500539](https://doi.org/10.1142/S0219749919500539). eprint: <https://doi.org/10.1142/S0219749919500539>. URL: <https://doi.org/10.1142/S0219749919500539>.
- [29] W. Cary Huffman and Vera Pless. “Basic concepts of linear codes”. In: *Fundamentals of Error-Correcting Codes*. Cambridge University Press, 2003. ISBN: 9780511807077. DOI: [10.1017/CB09780511807077.002](https://doi.org/10.1017/CB09780511807077.002).
- [30] T.F. Jordan. *Linear Operators for Quantum Mechanics*. Dover Books on Physics Series. Dover Publications, 2006. ISBN: 9780486453293.
- [31] Phillip Kaye, Raymond Laflamme, and Michele Mosca. *An Introduction to Quantum Computing*. Oxford University Press, Inc., 2007. ISBN: 0198570007.
- [32] Emanuel Knill, Raymond Laflamme, and Lorenza Viola. “Theory of Quantum Error Correction for General Noise”. In: *Physical Review Letters* 84.11 (Mar. 2000), pp. 2525–2528. DOI: [10.1103/physrevlett.84.2525](https://doi.org/10.1103/physrevlett.84.2525). URL: <https://doi.org/10.1103/physrevlett.84.2525>.
- [33] Ching-Yi Lai and Alexei Ashikhmin. “Linear Programming Bounds for Entanglement-Assisted Quantum Error-Correcting Codes by Split Weight Enumerators”. In: *IEEE Transactions on Information Theory* 64.1 (2018), pp. 622–639. DOI: [10.1109/TIT.2017.2711601](https://doi.org/10.1109/TIT.2017.2711601).
- [34] Ching-Yi Lai, Todd A. Brun, and Mark M. Wilde. “Dualities and identities for entanglement-assisted quantum codes”. In: (2011), pp. 1–14. URL: <https://arxiv.org/pdf/1010.5506v2.pdf>.
- [35] Ching-Yi Lai, Todd A. Brun, and Mark M. Wilde. “Duality in Entanglement-Assisted Quantum Error Correction”. In: *IEEE Transactions on Information Theory* 59.6 (2013), pp. 4020–4024. DOI: [10.1109/TIT.2013.2246274](https://doi.org/10.1109/TIT.2013.2246274).
- [36] Deanna Leggett and John Perry. “Fraction-Free Computation of Determinants”. In: (2011).
- [37] M. Loceff. *A Course in Quantum Computing (For the Community College)*. Foothill College, 2015.
- [38] Liangdong Lu et al. “Maximal entanglement entanglement-assisted quantum codes constructed from linear codes”. In: *Quantum Information Processing* 14.1 (Jan. 2015), pp. 165–182. ISSN: 1573-1332. DOI: [10.1007/s11128-014-0830-y](https://doi.org/10.1007/s11128-014-0830-y). URL: <https://doi.org/10.1007/s11128-014-0830-y>.
- [39] Liangdong Lu et al. *Optimal Quaternary Hermitian LCD codes*. 2020. arXiv: [2010.10166](https://arxiv.org/abs/2010.10166) [cs.IT].
- [40] James L Massey. “Linear codes with complementary duals”. In: *Discrete Mathematics* 106 (1992), pp. 337–342.

- [41] Patrick Gerland Mathieu Pageau and Nancy McGirr. “Data Processing Strategies: Hardware and Software for Data Entry Editing and Tabulating”. In: *NIDI/IUSSP* (1992).
- [42] Mathworks, ed. *Complex conjugate*. URL: <https://www.mathworks.com/help/matlab/ref/conj.html> (visited on 09/24/2021).
- [43] Razavy Mohsen. *Quantum Theory of tunneling*. World Scientific, 2013.
- [44] Gordon E Moore et al. *Cramming more components onto integrated circuits*. 1965.
- [45] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information 10<sup>th</sup> Edition*. Cambridge University Press, 2009. DOI: [10.1017/cbo9780511976667](https://doi.org/10.1017/cbo9780511976667). URL: <https://doi.org/10.1017/cbo9780511976667>.
- [46] Oracle, ed. *Java™ PlatformStandard Ed. 8*. URL: <https://docs.oracle.com/javase/8/docs/api/java/lang/Long.html> (visited on 09/24/2021).
- [47] Eric M Rains and Neil JA Sloane. “Self-dual codes”. In: *arXiv preprint math/0208001* (2002).
- [48] M. Reed. *Entanglement and Quantum Error Correction with Superconducting Qubits*. Yale University, 2013. ISBN: 9781304084866.
- [49] Robert B. Leighton Richard Feynman and Matthew Sands. *The Feynman Lectures on Physics New Millennium edition*. Vol. III. Basic Books, 2013.
- [50] W. Rudin. *Functional Analysis 2<sup>nd</sup> Edition*. McGraw-Hill Education, 1991. ISBN: 9780070542365.
- [51] Wolfgang Scherer. *Mathematics of Quantum Computing*. Springer International Publishing, 2019. DOI: [10.1007/978-3-030-12358-1](https://doi.org/10.1007/978-3-030-12358-1). URL: <https://doi.org/10.1007/978-3-030-12358-1>.
- [52] Claude E Shannon. “A mathematical theory of communication”. In: *The Bell system technical journal* 27.3 (1948), pp. 379–423.
- [53] P. W. Shor. “Algorithms for Quantum Computation: Discrete Logarithms and Factoring”. In: *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*. SFCS ’94. USA: IEEE Computer Society, 1994, pp. 124–134. ISBN: 0818665807. DOI: [10.1109/SFCS.1994.365700](https://doi.org/10.1109/SFCS.1994.365700). URL: <https://doi.org/10.1109/SFCS.1994.365700>.
- [54] Vladimir Silva. *Practical Quantum Computing for Developers: Programming Quantum Rigs in the Cloud Using Python, Quantum Assembly Language and IBM QExperience*. Apress, 2018.
- [55] Christoph Simon and Julia Kempe. “Robustness of multiparty entanglement”. In: *Physical Review A* 65.5 (May 2002). DOI: [10.1103/physreva.65.052327](https://doi.org/10.1103/physreva.65.052327). URL: <https://doi.org/10.1103/physreva.65.052327>.
- [56] Theerapat Tansuwannont. “Flag fault-tolerant error correction for cyclic CSS codes”. MA thesis. University of Waterloo, 2018.
- [57] Mark M. Wilde. *Quantum Coding with Entanglement*. 2008. arXiv: [0806.4214](https://arxiv.org/abs/0806.4214) [quant-ph].
- [58] Mark M. Wilde, Hari Krovi, and Todd A. Brun. “Convolutional entanglement distillation”. In: *2010 IEEE International Symposium on Information Theory* (June 2010). DOI: [10.1109/ISIT.2010.5513666](https://doi.org/10.1109/ISIT.2010.5513666). URL: <http://dx.doi.org/10.1109/ISIT.2010.5513666>.

- 
- [59] Yixuan Xie. “Quantum Error Correction and Stabilizer Codes.” PhD thesis. University of New South Wales, Sydney, Australia, 2016. URL: <https://pdfs.semanticscholar.org/4502/ebb59f945ccacea060a6fa55091914950e0c.pdf>.
- [60] Noson S. Yanofsky and Mirco A. Mannucci. *Quantum Computing for Computer Scientists*. Cambridge University Press, 2008. DOI: [10.1017/cbo9780511813887](https://doi.org/10.1017/cbo9780511813887). URL: <https://doi.org/10.1017/cbo9780511813887>.
- [61] Peide Ye, Thomas Ernst, and Mukesh V. Khare. “The last silicon transistor: Nanosheet devices could be the final evolutionary step for Moore’s Law”. In: *IEEE Spectrum* 56.8 (Aug. 2019), pp. 30–35. DOI: [10.1109/mspec.2019.8784120](https://doi.org/10.1109/mspec.2019.8784120). URL: <https://doi.org/10.1109/mspec.2019.8784120>.

# Appendices

## A.1 Repetition Code-bit Flip Circuit Using **CZ** Gates

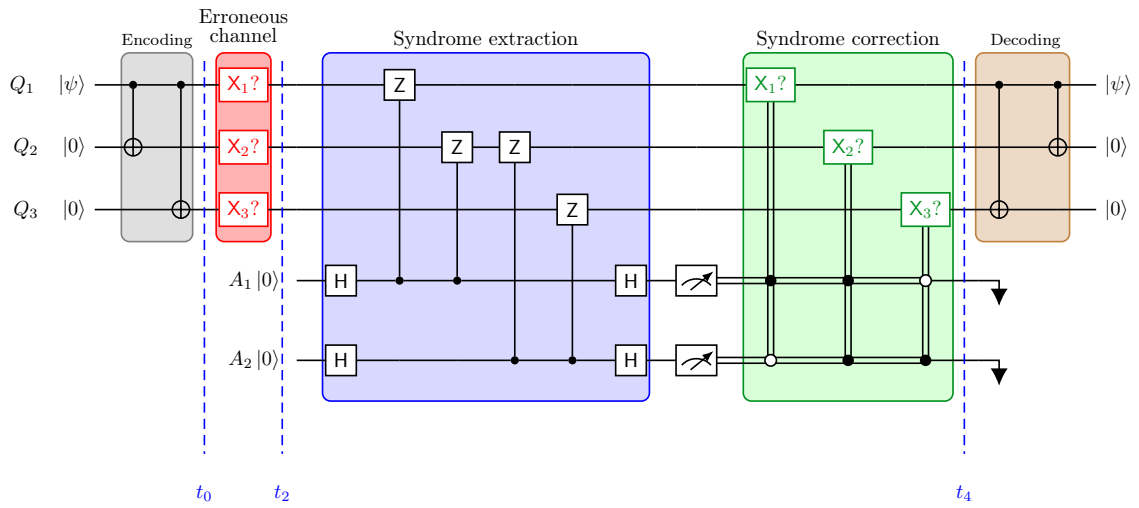


Figure A.1.1: Repetition code single bit-flip correction circuit using CZ gates to find the syndromes