Simulating Fault Tolerance in Quantum Stabilizer Codes

A Thesis

submitted to
Indian Institute of Science Education and Research Pune
in partial fulfillment of the requirements for the
BS-MS Dual Degree Programme

by

Pranav Maheshwari

Indian Institute of Science Education and Research Pune
Dr. Homi Bhabha Road,
Pashan, Pune 411008, INDIA.

May, 2024

Supervisor: Dr. Ankur Raina

Certificate

This is to certify that this dissertation entitled **Simulating Fault Tolerance in Quantum Stabilizer Codes** towards the partial fulfilment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research, Pune represents study carried out by Pranav Maheshwari at Indian Institute of Science Education and Research, Bhopal under the supervision of Dr. Ankur Raina, Assistant Professor, Department of EECS, during the academic year 2023 – 2024.

Dr. Ankur Raina

Committee:

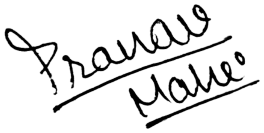
Dr. Ankur Raina

Prof. M. S. Santhanam

This thesis is dedicated to my parents.

Declaration

I hereby declare that the matter embodied in the report entitled **Simulating Fault Tolerance in Quantum Stabilizer Codes** are the results of the work carried out by me at the Department of EECS, Indian Institute of Science Education and Research, Bhopal, under the supervision of Dr. Ankur Raina and the same has not been submitted elsewhere for any other degree. Wherever others contribute, every effort is made to indicate this clearly, with due reference to the literature and acknowledgment of collaborative research and discussions.



Pranav Maheshwari
20191181

Acknowledgments

I express profound gratitude to Dr. Ankur Raina, my invaluable mentor, whose unwavering support has been instrumental throughout this thesis. Engaging in insightful discussions and brainstorming sessions with him has simplified complex concepts and served as the primary source of innovative ideas. His provision of resources and freedom to explore at will has been immensely appreciated. Dr. Raina's encouragement during moments of uncertainty and his insistence on exploring diverse approaches have been pivotal in my progress. Under his guidance, I have cultivated patience, honed essential skills, and profoundly understood the research process.

I extend my gratitude to Prof. M. S. Santhanam for his invaluable research exposure and guidance, which significantly contributed to laying the foundation for this academic journey. I am immensely grateful to Dr. Krishnakumar Sabapathy for his invaluable contributions to our research journey. His motivation to study the Bare ancilla fault tolerance method, valuable inputs, and thought-provoking discussions have been instrumental in guiding us and clearing our doubts.

I am thankful to Prof. Sunil Mukhi and Prof. T. S. Mahesh for their exceptional teaching of Introductory Quantum Mechanics course, laying a solid foundation and covering latest advancements in the field, which piqued my interest in quantum computing and information. I sincerely thank Prof. R. G. Bhat, Prof. Srinivas Hotha, Prof. M. S. Madhusudhan, Prof. Bhas Bapat, Dr. Seema Sharma, Dr. Bijay Agarwalla, and Dr. Sreejith G. J. for making physics an intriguing and captivating subject for me. Their teaching and guidance have been invaluable in shaping my academic journey and fostering a deep interest in quantum physics. I am deeply grateful to all the faculties who have imparted their knowledge and wisdom, shaping my academic journey and inspiring my intellectual growth.

I sincerely appreciate my esteemed research group members, Sanidhya, Harsh, and Mainak, for their support and availability throughout my thesis journey. The group discussions, interactions at coffee tables, and weekly meetings have allowed me to explore various topics and be updated on advancements in the research field. Sanidhya has been more than a friend; he has been my steadfast companion at every step of my life at IISER Bhopal. Harsh has been an inspiring figure, constantly motivating me to persevere in the face of challenges. I am particularly thankful to him for generously sharing the [[8,1,3]] code, a crucial asset in our exploration of its fault-tolerant properties. Mainak's academic guidance has been indispensable, as he consistently advised me to make thoughtful and fruitful decisions. Furthermore, I extend my gratitude to Yogesh and Pallavi for fostering a supportive and conducive lab environment where creativity and collaboration flourish.

I am profoundly grateful to my parents for their firm belief in me, their understanding, and their continuous encouragement throughout this journey. Their love, guidance, and unwavering faith in my abilities have been a consistent source of strength and motivation. They have always shielded me from external pressures and wholeheartedly encouraged me to pursue my passions. I extend my sincere thanks to Maittri for her resolute support, constant availability, and steadfast belief in my capabilities. Her presence and encouragement have provided immense comfort and motivation during challenging times. She has ensured that I never feel alone and has always made me feel cherished and valued.

I extend my heartfelt gratitude to Lokender, Nivesh, Anish, Jatin, Hritik, and once again to Sanidhya for their friendship, which has made every moment of this period truly enjoyable and unforgettable. Their companionship has been the cornerstone of my student life experience at IISER Bhopal, filling my final year with joyous trips, lively dining table conversations, and countless memorable moments.

Special thanks go to Kaustubh and Ravish for their trusted companionship despite the distance and for being integral to my special moments. Kaustubh has played the role of an informal mentor, consistently motivating me to work hard while also reminding me to savor moments of personal life. His wisdom and support during times of confusion or difficulty have been invaluable in academic and non-academic pursuits. Ravish has been a constant source of inspiration and confidence. His fearless approach to challenges and adaptability to any situation has kept me motivated and true to myself. I am deeply grateful to both of them for always being just a phone call away, understanding me during my lows, and

celebrating my highs. Their presence has been a constant source of strength and reassurance throughout my journey.

I am much grateful to Jezer and Shivam for being my constant companions to discuss quantum computing with. Pranjal has always been a great friend and supporter of me. To Tridash, Vishal, Yadram, Anitha, Suyash, Lokendra, Abhinav, and Shivansh, your presence has filled these years with joy making them truly unforgettable. To all my friends who have shared in this journey, your warmth and laughter have made every moment precious and enriching.

Acknowledging the profound influence of Shree Krishna's blessings on this thesis is essential. I take pride in India's rich spiritual heritage and its profound impact on my dedication to my work and interactions with those around me. This thesis is a testament to the spiritual wisdom that permeates the land of India and its transformative effect on my commitment to authenticity and service to others.

Lastly, I extend heartfelt appreciation to all those who have supported and encouraged me throughout this journey. Your unwavering belief in me has served as the driving force behind my accomplishments.

Contents

List of Figures

List of Tables

Notations

n	Number of physical qubits.
k	Number of logical qubits.
d	Minimum Hamming distance of the code.
H	Check matrix of the code.
$\mathcal{S}$	Stabilizer subgroup.
g or M	Generators of the stabilizer group.
$\mathcal{C}$	Code or Codespace.
$\mathbf{X}_i, \mathbf{Y}_j, \mathbf{Z}_k$	Pauli $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ on qubits i, j, k respectively.
$\mathbf{P}_i$	A Pauli matrix on i^{th} qubit.
$\mathbf{I}$	Identity operator.
$\mathbf{H}$	Hadamard gate.
$\mathbf{S}$	Phase gate.
$\|\phi\rangle$	k qubit message state.
$\|\psi_i\rangle$	n qubit initial state.
$\|\psi_e\rangle$	n qubit encoded state.
$\|\psi_d\rangle$	n qubit decoded state.
$\mathbb{F}_2$	Binary field $\{0, 1\}$.

Abstract

Quantum computing becomes achievable only assuming that fault-tolerant quantum error correction codes (QECCs) can be effectively implemented on physical hardware, offering significantly lower error rates than those inherent in the system. Quantum stabilizer codes encompass various families of quantum codes and furnish a group-theoretic framework for quantum error correction (QEC). Each code is distinguished by a pseudo-threshold value, indicative of its performance under a selected noise model and decoding scheme. We establish a framework capable of generating error correction circuits and providing error rates and pseudo-thresholds, if applicable, for any stabilizer code. Additionally, we introduce a unitary encoding scheme capable of encoding any stabilizer code, overcoming the limitation of disregarding the phase of a stabilizer generator. Furthermore, we devise a syndrome extraction circuit employing a single ancilla qubit, which accelerates simulation while minimizing the consumption of qubit resources.

Fault tolerance stands as a crucial requirement that renders QECCs practically useful. Among the myriad techniques to achieve fault tolerance, we concentrate on the bare ancillary fault tolerance scheme, which boasts the least overhead observed to date. Our focus involves a review of the reordering stabilizer generator trick for syndrome extraction, with an extension aimed at overcoming limitations inherent in the original method proposed by Brown *et. al.* Furthermore, we delve into a study of the fault-tolerant properties exhibited by an eight-qubit code, obtaining pseudo-threshold values under the standard depolarizing and anisotropic noise models. Additionally, we undertake a comparative analysis of various error rates under the bare ancillary fault tolerance method, elucidating why a noise-free error correction step is often employed in the literature despite its practical infeasibility.

Our work offers a comprehensive introduction to fault-tolerant QEC and extends it to the simulation of quantum error-correcting codes. The techniques outlined in this thesis complement existing literature and surmount limitations to achieve lower error rates in physical systems.

Contributions

The study encapsulated in the thesis has resulted in a paper, titled **Fault Tolerance of the [[8,1,3]] non-CSS code**, accepted in the IEEE Symposium in Information Technology (ISIT) Quantum Information Knowledge (QuIK) Workshop 2024. The pre-print version is accessible at `https://arxiv.org/abs/2402.19389`.

This work introduces an enhanced encoding scheme applicable to any stabilizer code, with a primary focus on non-CSS codes. It presents a modified approach to fault tolerance utilizing a single ancillary qubit and demonstrates the performance of the [[8,1,3]] code under this modified approach.

For those interested in implementing the procedure and reproducing the results, Python code is available at `https://github.com/maheshwaripranav/Bare_Ancillary_Codes`. Additionally, a framework for implementing any stabilizer code using Qiskit has been developed and can be accessed at `https://github.com/maheshwaripranav/Stabilizer_Codes`.

Chapter 1

Introduction

Once upon a time, in the not-so-distant past, humanity witnessed the birth of computing. It was a marvel—a tool of immense power, capable of processing vast amounts of information with unprecedented speed. From solving complex equations to simulating the behavior of atoms, computers have become indispensable in nearly every facet of modern life. Nevertheless, as our thirst for knowledge grew, so did the need for even greater computational horsepower. We found ourselves at a crossroads, pushing the boundaries of what current technology could achieve regarding speed, efficiency, and sheer processing power.

As scientists delved deeper into the mysteries of the universe, they stumbled upon the enigmatic realm of quantum physics. Here, reality seemed to take on a surreal quality, with particles existing in multiple states simultaneously and information teleporting across vast distances instantaneously. It was a realm governed not by the familiar laws of classical physics but by the bizarre and wondrous principles of quantum mechanics. Intrigued by this newfound frontier, researchers began to wonder: could the strange phenomena of quantum physics be harnessed to build a new kind of computer, one that operated not with bits but with quantum bits, or qubits, capable of existing in multiple states simultaneously? Thus began the quest for quantum computing—a journey into the advanced future fueled by curiosity and ambition.

The promise of quantum computing is indeed tantalizing. Quantum computers, challenging the classical limits set by the Church-Turing Thesis, can solve problems exponentially faster than classical machines. With algorithms like Shor's and Grover's, they open

doors to deciphering unbreakable codes and conducting lightning-fast searches through vast databases [1, 2]. Moreover, quantum supremacy, exemplified by Google's Sycamore, demonstrates their capacity to outperform classical counterparts in specific tasks [3]. Quantum cryptography ensures secure communication through protocols like quantum key distribution [4], while quantum machine learning accelerates optimization and pattern recognition processes. Additionally, quantum simulation allows for precise modeling of molecular behavior and complex quantum phenomena, promising transformative breakthroughs in fields ranging from materials science to drug discovery [5]. The quantum computing applications appear boundless, heralding a new era of computational prowess and innovation.

However, lurking in the shadows are formidable challenges. At the forefront of these challenges lies the pervasive issue of noise and errors, which incessantly disrupt the precision of qubits within physical systems. The delicate nature of qubits make them prone to errors, requiring sophisticated error-correction techniques to maintain their reliability. Nevertheless, the dream persists—a dream of a future where quantum computers would unlock new frontiers in science, technology, and beyond.

This thesis delves into QEC's realm, examining its theoretical underpinnings and practical implications. Chapter 1 lays the groundwork by elucidating the fundamental concepts of quantum computing, error correction, and fault tolerance. Building upon this foundation, Chapter 2 introduces the stabilizer formalism and delineates the circuits necessary for simulating QECCs. We present an optimized encoding scheme and elaborate on criteria for assessing the performance of quantum codes.

Chapter 3 delves into extensively studied fault tolerance techniques, focusing on minimizing overhead by utilizing only one to two ancillary qubits. Furthermore, we propose enhancements to the single ancilla fault tolerance technique to improve code performance beyond conventional methods. In Chapter 4, we present our findings from implementing bare ancillary fault tolerance on [[7,1,3]] and [[8,1,3]] non-CSS codes. A comprehensive discussion ensues, covering suitable simulation methodologies and the performance of codes under various noise models.

Finally, Chapter 5 serves as a culmination of our research, offering a summary of our findings, the outcomes of our innovations, and a contemplation of future avenues in the realm of fault-tolerant QEC.

1.1 Quantum Computation

Unlike classical computers that operate with bits in a binary state (0 or 1), quantum computing harnesses the principles of quantum mechanics. It processes information through quantum bits, or qubits, capable of existing in multiple states ($|0\rangle$ and $|1\rangle$) simultaneously. Akin to logic gates, certain unitary operations on qubits provide us with computational capability.

Qubits can be realized in various physical architectures [6], for example,

- *Superconducting Qubits* - where the presence or absence of current in a superconducting loop represents the qubit states, and the operations are applied through microwave pulses.
- *Trapped Ion Qubits* - where the internal energy levels of ions trapped using electromagnetic fields are the qubit states, manipulated using laser pulses.
- *Photonic Qubits* - where horizontal or vertical polarization states of photons represent the qubit states, controlled through optical elements such as phase shifters and beam splitters.

Various computation models have been proposed for quantum computing, such as *Circuit Model* where operations are performed through Quantum Gates, *Measurement Based Model* which harnesses entanglement and measurement for computations, and *Adiabatic Model*, which evolves the Hamiltonian of the system such that desired computed result is its ground state [6]. This thesis will focus on the circuit model of quantum computing.

1.1.1 Circuit Model

Considering an initial quantum state (generally $|0\rangle$), a set of instructions is applied for computation; these instructions are represented by gates. These can be unitary, such as *Pauli gates*, or non-unitary, such as *measurement gates*, in nature. A set of instructions combined together is called a *circuit* [6].

Gate	Matrix	Description
I	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	Identity Gate
X	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	Pauli **X** Gate
Y	$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$	Pauli **Y** Gate
Z	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	Pauli **Z** Gate
H	$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$	Hadamard Gate
S	$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$	Phase Gate
CX	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$	Controlled **X** Gate
CZ	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$	Controlled **Z** Gate
$\boxed{\|\mathbf{0}\rangle}$	–	Reset Gate
	–	Measurement Gate

Table 1.1: A list of quantum gates.

A universal set of unitary gates in quantum computing is a minimal collection of quantum gates that can perform any quantum computation with arbitrary precision. In quantum computing, single qubit rotation gates, $\{R_x(\alpha), R_y(\beta), R_z(\gamma)\}$, alongwith **CX** gate forms a set of universal quantum gates. A more practical and widely acclaimed set of universal quantum gates is $\{\mathbf{CX}, \mathbf{H}, \mathbf{T}\}$, where **T** is the $\pi/8$ phase gate. Furthermore, $\{\mathbf{CX}, \mathbf{H}, \mathbf{S}\}$ gates form a set of operations called *Clifford Group*, which maps the Pauli group $\{\mathbf{I}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}\}$ to itself [6]. We would use only the Clifford group and a few non-unitary gates to create circuits. Table 1.1 presents a list of gates we shall use in our circuits.

1.1.2 Noise

All quantum computing and information can be modeled using quantum operations formalism in which density matrices represent states, and operators are maps from one density matrix to another satisfying specific properties. Applying a unitary operation U on state $|\psi\rangle$ is equivalent to applying a map $\mathcal{E}$ on density matrix ρ such that

$$\mathcal{E}(\rho) = U|\psi\rangle\langle\psi|U^\dagger = U|\psi\rangle \text{ [6].} \tag{1.1}$$

Physical systems are prone to environmental noise, which can cause errors in quantum computation. We can model noise due to the environment as a unitary quantum operation on a combined system (state + environment). Let the initial state be $\rho \otimes \rho_{\text{env}}$ and the noise operation be U. Then the final state would be $U(\rho \otimes \rho_{\text{env}})U^\dagger$. To eliminate the environment in this final combined state, we need to take a partial trace over the environment, giving us

$$\mathcal{E}(\rho) = \text{tr}_{\text{env}}\big[U(\rho \otimes \rho_{\text{env}})U^\dagger\big], \tag{1.2}$$

where $\mathcal{E}$ is the noise operation [6].

Quantum operations can also be represented in another form known as *Operator Sum Representation*. Let $|e_k\rangle$ be an orthonormal basis for the environment and $\rho_{\text{env}} = |e_0\rangle\langle e_0|$ be initial state of the environment. Then from Equation 1.2,

$$\mathcal{E}(\rho) = \sum_k \langle e_k|U[\rho \otimes \rho_{\text{env}}]U^\dagger|e_k\rangle, \tag{1.3}$$

$$= \sum_k E_k \rho E_k^\dagger, \tag{1.4}$$

where $E_k = \langle e_k|U|e_0\rangle$ is an operator on the state space of the principal system. These E_k operation elements are known as *Kraus Operators* [6]. Any noise-prone operation can be considered as a completely positive trace-preserving operator, $\mathcal{E}$ acting on a state ρ as-

$$\mathcal{E}(\rho) = \sum_i E_i \rho E_i^\dagger \;\; s.t. \;\; \sum_i E_i E_i^\dagger = \mathbf{I}. \tag{1.5}$$

To simulate the behavior of physical systems, we model the noise through certain error operators. Consider that we have a noise model that causes an error on any one qubit. This

error can be represented by some 2×2 matrix, which can always be written as

$$\mathcal{E} = a\mathbf{X} + b\mathbf{Y} + c\mathbf{Z} + d\mathbf{I}, \tag{1.6}$$

where $a, b, c, d \in \mathbb{C}$. Hence, we only need to worry about a basis of errors, which is *Pauli matrices* in our case [7].

Various techniques are used to reduce the effects of noise in quantum computation [8],

- *Error Suppresion* - hardware level handling of errors using knowledge of potential undesirable effects.
- *Error Mitigation* - post-processing of results to reduce the effects of noise, such as extrapolating the noise-free result from noisy results.
- *Error Correction* - encoding the information so that errors due to noise can be detected and corrected.

While error suppression and error mitigation are most prevalent for present *Noisy Intermediate Scale Quantum (NISQ)* devices, accurate quantum computation aims to achieve fault tolerance through error correction. In this thesis, we shall focus on QEC and fault tolerance procedures.

1.2 Error Correcting Codes

Richard Hamming thought of communicating information in an encoded form so that the original information could be retrieved despite any noise in the communication channel. The core idea was to store original information in more bits than required, and the challenge was to keep the extra bits to a minimum, giving rise to the field of *Error Correction* [9].

Redundancy in data is created by generating copies of information in extra bits. There is a particular structure for storing information in these bits, which is harnessed to identify which bits are errorful during information processing or communication over a noisy channel. The identified error could then be corrected to retain the original data of interest.

The simplest example is creating three copies of a single bit and comparing the received bits. Using majority voting, we infer the actual value of the bit sent. Let us elucidate using an example. The original bit value was 0; the three bits are encoded as 000. If the error occurs on any bit with a probability p, then with probability p, two of the bits have a value of 0, and thus we can obtain the original value. Only with probability p^2 or p^3, two or three bits get errorful, and we obtain wrong results. Hence, the encoding scheme reduces the error rate from p to p^2. This code is referred to as *Repetition Code* [6]. We will now introduce the structure that any classical code follows.

1.2.1 Classical Error Correcting Codes

In classical error correction, a *linear code*, encodes k bits of information into n bits and is denoted as an $[n, k]$ code [6, 7, 10]. The code is specified by a matrix of size $n \times k$ called the *Generator Matrix*, G, whose entries are elements of $\mathbb{F}_2$. To encode all messages uniquely, we require columns of G to be linearly independent.

A codespace, $\mathcal{C}$, of an $[n, k]$ code is defined to consist of all vectors, $x \in \mathbb{F}_2^n$, of length n, such that $Hx = 0$ where H is a matrix of size $n - k \times n$, known as *Parity Check Matrix*. We can also say that codespace is the kernel space of H. By *Rank-Nullity Theorem*, H should have linearly independent rows. The matrices G and H are related as $HG = 0$.

Hamming Distance, $d(x, y)$, between two codewords x and y is defined to be the number of places at which x and y differ. The *weight* of a codeword (any vector in codespace) is defined to be the distance from the string of all zeroes, i.e.,

$$\mathrm{wt}(x) = d(x, 0_n) \quad \forall x \in \mathcal{C}. \tag{1.7}$$

Distance of a code, d, is the minimum distance between any two codewords.

$$d(\mathcal{C}) = \min\{d(x, y) \ \mid \ x, y \in \mathcal{C} \ \text{ and } \ x \neq y\}, \tag{1.8}$$

which implies $d(\mathcal{C}) = \min\{\mathrm{wt}(x) | x \in \mathcal{C}\}$. This defines an $[n, k, d]$ code.

Consider a codeword $y = Gx$, but an error is induced such that $y' = y + e$. Since $Hy = 0$, we get $Hy' = He = s$, called the *syndrome*. If for all possible errors e_j, we

know corresponding s_j; we can match the error syndromes obtained to identify the error and perform correction accordingly. For a code to correct errors up to weight t, the minimum distance of the code should be $2t+1$ so that y' can be corrected to y satisfying $d(y, y') \leq t$ [6, 7, 10].

Example

Consider a [6,3,3] code with the generator matrix and parity check matrix as follows:

$$G = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \quad H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}. \tag{1.9}$$

If the initial k bit vector was x given by, $x^T = [111]$, then the encoded state $y = Gx$ is given by $y^T = [111000]$. Let us consider an error occurs on the first bit, and it flips to 0, giving us $y'^T = [011000]$. On obtaining y', we can calculate $s = Hy' = [110]$, which can be matched to a standard table of this code to indicate an error on the first qubit. We can thus obtain the correct codeword $y^T = [111000]$, leading to the corrected initial vector $x^T = [111]$.

1.2.2 Quantum Error Correcting Codes

In quantum computation, n length vectors are replaced by n qubit state vectors, and operators replace the matrices. Akin to the parity check matrix, H, we create an $n-k \times 2n$ check matrix with independent rows. These rows correspond to $n-k$ independent operators acting on n qubits, called the *Stabilizer generators*. The span of these operators generates a group called the *Stabilizer group*, $\mathcal{S}$. QECCs harnessing stabilizers are called *Stabilizer Codes*, and we refer to the corresponding error correction mechanism as *Stabilizer Formalism* [7].

Mapping Operators to Matrices

Throughout the thesis, we will be concerned with Pauli operators and their tensor products. Let us consider an operator $A = A_1 \otimes A_2 \cdots \otimes A_n$ acting on n qubits. We shall create a row of $2n$ entries, all initialized to 0. We shall flip the i^{th} entry to 1 $\forall i$ such that $A_i = \mathbf{X}$ or $\mathbf{Y}$. And flip the $(n+j)^{th}$ entry to 1 $\forall j$ such that $A_j = \mathbf{Z}$ or $\mathbf{Y}$. Let us represent $A = [A^x|A^z]$. Consider another operator $B = [B^x|B^z]$. We shall define *symplectic product* of A and B as

$$A \odot B = A^x B^z + A^z B^x \pmod 2 \text{ [11]}. \tag{1.10}$$

Henceforth, all calculations would be modulo two unless specified. This symplectic product also gives the commutation relation between any two operators,

$$AB = -1^{(A \odot B)} BA. \tag{1.11}$$

Given the stabilizer generators, we can thus write the *check matrix*, H. The i^{th} row of H is the binary representation of the i^{th} stabilizer generator. H could be divided into two blocks, H^x and H^z, corresponding to $\mathbf{X}$ and $\mathbf{Z}$ operators respectively in the stabilizer generators. To obtain syndrome, we shall take the symplectic product of the operator corresponding to an error and each row of the check matrix, equivalent to finding the stabilizer generators that anti-commute with the error operator [11].

Properties of a Quantum Code

For each code, there exists a set of correctable errors, represented by $\mathcal{E}$. To correct two errors E_a and E_b, their action on each basis codeword should lead to orthogonal states. Thus, $\forall \left|\psi_i\right\rangle, \left|\psi_j\right\rangle \in \mathcal{C}$ and $\forall E_a, E_b \in \mathcal{E}$,

$$\left\langle\psi_i\right| E_a^\dagger E_b \left|\psi_j\right\rangle = C_{ab}\delta_{ij}. \tag{1.12}$$

A code satisfies Equation 1.12 $\forall E_a \in \mathcal{E}$ iff the code can correct all errors in $\mathcal{E}$. A Code for which matrix C is non-singular, that is, each correctable error has a distinct syndrome, is called a non-degenerate code [7].

The weight of an operator is defined as the number of non-identity operators in its tensor

product representation. The weight of the smallest operator for which Equation 1.12 does not hold is the distance of the code. Each code has a distance of at least one, and a t error *correcting* code must have distance, $d \geq 2t+1$. An s error *detecting* code must have $d \geq s+1$ as we only need to differentiate E_a with $E_b = \mathbf{I}$. Correcting r located errors in a quantum erasure channel requires $d \geq r + 1$. A code correcting t errors, r additional located errors, and detects further s errors must have distance $d \geq 2t + r + s + 1$ [7].

1.2.3 Family of Quantum Codes

QECCs are organized into distinct families based on their structures and properties. Each family provides a certain *threshold* value, which is used to characterize the performance of the codes within that family. Individual codes may belong to multiple families and can have different *pseudo-threshold* values for different noise models. We present a few families of quantum codes below:

- *Perfect Quantum Codes* - codes which harness all the possible syndrome values to correct all errors up to a certain weight, e.g., [[5,1,3]] code [12].
- *CSS Codes* - codes whose generators contain only either **X** operators or **Z** operators treating **X** and **Z** errors separately. For example, [[7, 1.3]] Steane Code and the Shor's [[9,1,3]] repetition code [13, 14].
- *Color Codes* - Geometric codes with trivalent vertices and each face denoting different color or stabilizer generators. The smallest color code is a [[7,1,3]] codes [15].
- *Entanglement Assisted Codes* - Codes which utilize pre-shared entanglement resource to create a quantum code from any given set of operators [11].
- *Subsystem Codes* - Quantum codes which utilize certain gauge freedoms to reduce the number of stabilizers in certain codes such as the [[9,1,3,3]] Bacon-Shor code with four stabilizer generators [16].

CSS codes were the first to be discovered and have been widely studied. Examples of CSS codes have been found in various families of codes, such as color codes, surface codes, etc. We are inclined to study quantum codes in a more general approach that covers CSS

as well as Non-CSS codes. The space of Non-CSS codes is allegedly larger, and there are many discovered codes whose stabilizer generators contain $\mathbf{X}, \mathbf{Y}$, and $\mathbf{Z}$ together. We have modified traditional simulation schemes to include Non-CSS codes, and we aim to compare the performance of CSS and Non-CSS codes having the same *code rate*, $(\frac{k}{n})$.

1.3 Fault Tolerance in Error Correction

Integrating error correction introduces errors due to noisy gates, thereby underscoring the significance of fault tolerance. Fault tolerance ensures that the system remains resilient against these newly induced errors, providing an additional layer of protection to maintain the integrity of the quantum computation. Many fault-tolerant schemes have been proposed that require two or more ancillary qubits to correct errors arising while implementing QEC procedure [17].

Definition 1.1 (Fault Tolerance)**.** A procedure is said to be fault-tolerant if the failure of only one component with probability p causes at most one error with probability $\mathcal{O}(p)$ in each encoded block or codeblock of qubits output [10].

The component here is not only an encoded gate but also measurements, quantum wires, state preparation, etc. We also require that if only one component fails, then the measurement result reported after fault tolerant error correction must have a probability of error of $\mathcal{O}(p^2)$, where p is the maximum probability of a failure in any one of the components used to implement the QEC procedure [6].

Thus, fault tolerance requires certain properties followed by encoded operations, known as *transversality* [10].

Definition 1.2 (Transversal Operations)**.** Operations which satisfies one of the following two conditions:

1. Applies single qubit gates to qubits in a codeblock.

2. Interacts i^{th} qubit in one codeblock to only i^{th} qubit in another codeblock.

It is evident that a single qubit gate can generate only one error. Furthermore, the second condition in Definition 1.2 ensures that a controlled gate either propagates one error to the same codeblock or another codeblock or to both. Since each codeblock will have a maximum of one error, transversal operations are fault-tolerant [10].

Curiously enough, if we can confine errors to a single error in a codeblock, we can also apply another level of encoding to each physical qubit of a codeblock, and again encode each qubit of encoded codeblocks thus generated. We can keep encoding recursively and perform error correction to get further reduced error rates. The *Threshold Theorem* in quantum computing asserts that if the error rate in quantum computation is below a certain threshold, then it is possible to perform arbitrarily long quantum computations reliably by using QECCs [18].

Theorem 1.1 (Threshold Theorem)**.** *A quantum circuit containing $p(n)$ gates may be simulated with probability of error at most ϵ using,*

$$\mathcal{O}\Big(poly\Big(\frac{\log p(n)}{\epsilon}\Big)p(n)\Big),$$

gates when maximum probability of failure of any component, p, is below some constant threshold, p_{th} [6, 18].

The threshold theorem guarantees efficient computation incorporating QEC. The major challenge of devising a fault-tolerant protocol is eliminating propagated errors. Certain two or more qubit gates can propagate errors from one qubit to another, spreading the error within the circuit. The error correction scheme itself can lead to further errors due to the propagation of errors [19]. Considering only **CX** and **CZ** gates, the Pauli **X** and **Z** errors propagate as shown in Figure 1.1[1].

A single error on a two-qubit gate means an error on both the control and the target, and if any error propagates through other gates in the procedure, a single error can lead to an exponentially large number of errors on the data qubits. We shall consider a protocol to be t-fault tolerant if $k \leq t$ errors (faults) lead to $l \leq t$ errors on the data qubits.

Making a QECC fault tolerant requires using extra ancillary qubits to identify faults in the procedure. The ratio of physical qubits to logical qubits is referred to as *Overhead*

[1]All circuit diagrams are created using quantikz package [20].

(a) Error on control on **CX** Gate

(b) Error on target on **CX** Gate

(c) Error on **CZ** Gate

Figure 1.1: Propagation of an **X** and **Z** Pauli error due to controlled gates, **CX** and **CZ**. Figures on the left indicates that **X** errors propagate only through a control, while on the right indicates that **Z** error propagates only through a target of **CX**. The gates are considered to be noise-free and do not add any error.

in fault-tolerant quantum computation [21]. The first of the fault-tolerant techniques had much higher overhead, such as fault tolerance in a code was shown by Shor in 1996 [22], requiring up to $w+1$ ancillary qubits for a stabilizer of weight, w. In 1997, Steane proposed a fault tolerance method that uses n ancillary qubits for an $[[n, k, d]]$ code [23]. Reducing the overhead has become the most prominent challenge in modern times. The quest for the least overhead continued with continuous advancements in the field. In 2017, Brown *et. al* presented a method that does not require more than one ancillary qubit to achieve fault tolerance [24]. This method has its limitations and cannot be applied to all codes. Soon after, flag fault tolerance method was proposed, which uses only two ancillae for fault-tolerant QEC. A general approach using flag qubits was later presented, which can tolerate up to t faults using $2t + 1$ ancillary qubits [25].

We will now delve into the theoretical intricacies of a quantum stabilizer code and understand its error correction mechanism. The next chapter details how to build a quantum circuit to implement an error-correcting code and evaluate its effectiveness on noise models inspired by physical systems.

Chapter 2

Stabilizer Formalism

A QECC encodes k qubits with information in 2^k dimensional Hilbert space to n qubits with information in 2^n dimensional Hilbert space. Doing computation in 2^k dimensions requires a very small number of operators than can be used in 2^n dimensional space. We model the extra operators as error operators which can be detected and eliminated. This chapter details the stabilizer formalism of QEC which provide a picture to understand how various operators can be used to detect errors and perform computations in the codespace.

2.1 Theoretical Framework

A single qubit can exist in a 2^1 dimensional complex vector space, denoted by $\mathcal{V}_1$. Correspondingly, the operators acting in such a state can be expressed as a linear combination of *Pauli matrices* forming a basis group, $\mathcal{G}_1$. For a single qubit, we can define *Pauli basis group* as:

$$\mathcal{G}_1 \equiv \{\pm\mathbf{I}, \pm i\mathbf{I}, \pm\mathbf{X}, \pm i\mathbf{X}, \pm\mathbf{Y}, \pm i\mathbf{Y}, \pm\mathbf{Z}, \pm i\mathbf{Z}\}. \tag{2.1}$$

This group can be generated by operators $i\mathbf{I}, \mathbf{X}, \mathbf{Z}$ represented as $\mathcal{G}_1 = \langle i\mathbf{I}, \mathbf{X}, \mathbf{Z}\rangle$. For n qubits, Pauli group ($\mathcal{G}_n$) contains n-fold tensor products of Pauli matrices with multiplicative factors $\{\pm 1, \pm i\}$ [6]. Tensor products of Pauli matrices follow the following two properties:

1. For any two elements $g_1, g_2 \in \mathcal{G}_n$, either they commute or anti-commute, i.e., either

$[g_1, g_2] = 0$ or $\{g_1, g_2\} = 0$.

2. Each element is Hermitian or Anti-Hermitian i.e., $\forall g \in \mathcal{G}_n,\ g^\dagger = \pm g$. Also, $g^2 = \pm \mathbf{I}$.

Let us find group structure in the space of all possible operators, $\mathcal{G}_n$, acting on n qubits vector space. For each group, there exists a set of independent elements such that group operations among these elements can generate the whole group. Such elements are called *generators* of the group.

Proposition 2.1. *The cardinality of a group generated by l independent elements belonging to group $\mathcal{G}_n$ is 2^l.*

Proof. Consider the generators be, $\{g_1, g_2, \ldots, g_l\}$. Any element, g, in the group thus generated will be the product of these generators. Thus $g = g_1^{i_1} g_2^{i_2} \ldots g_l^{i_l}$. Using properties given above, we only need to choose if a particular g_j constitutes g or not, i.e., $i_k \in \{0, 1\}\ \ \forall k \in \{1, \ldots, l\}$. Number of g of form g_j is $\binom{l}{1}$, and of form $g_j g_k$ is $\binom{l}{2}$ and so on. Therefore,

$$|\langle g_1, g_2, \ldots, g_l \rangle| = \binom{l}{1} + \binom{l}{2} + \cdots + \binom{l}{l} = 2^l. \tag{2.2}$$

Hence, the cardinality of a group generated by l independent elements belonging to group $\mathcal{G}_n$ is 2^l. □

This property will be much used in further discussion to find out the cardinality of groups and cosets.

Definition 2.1 (Stabilizer subgroup)**.** If $\mathcal{G}_n$ is a group acting on a set (or vector space) $\mathcal{V}_n$ and $\mathcal{V}_\mathcal{S}$ is a subspace of $\mathcal{V}_n$, the stabilizer subgroup of $\mathcal{V}_\mathcal{S}$ is the group

$$\mathcal{S} = \{g \in \mathcal{G}_n \quad | \quad g\,|\psi\rangle = |\psi\rangle \ \ \forall\, |\psi\rangle \in \mathcal{V}_\mathcal{S}\}. \tag{2.3}$$

The codespace, $\mathcal{C}$ must be 2^k dimensional. We wish that the stabilizer subgroup $\mathcal{S}$ be such that the corresponding stabilized subspace $\mathcal{V}_\mathcal{S} \equiv \mathcal{C}$.

Proposition 2.2. *Let $\mathcal{V}_\mathcal{S}$ be the vector space stabilized by $\mathcal{S} = \langle g_1, \ldots, g_{n-k} \rangle$, generated by $n - k$ independent and commuting elements from $\mathcal{G}_n$, such that $-\mathbf{I} \notin \mathcal{S}$, then $\mathcal{V}_\mathcal{S}$ is a 2^k dimensional vector space [6].*

Proof. Consider the group $\mathcal{S}$ be generated by r elements, i.e., $\mathcal{S} = \langle g_1, \ldots, g_r \rangle$. None of the generators can be $\mathbf{I}$ unless $\mathcal{S} \equiv \mathbf{I}$. Note that $\text{tr}(A \otimes B) = \text{tr}(A)\text{tr}(B)$ and $\text{tr}(\mathbf{P}) = 0$, for a Pauli matrix $\mathbf{P} \in \{\mathbf{X}, \mathbf{Y}, \mathbf{Z}\}$. It gives that $\text{tr}(g_1) = 0$ and since eigenvalues of g_1 are only ± 1, it must have 2^{n-1} degeneracy for each. So g_1 divides $\mathcal{V}_n$ into two halves corresponding to +1 and -1 eigenspaces of g_1 each having dimensions 2^{n-1}.

Take $P_+(g_1) = \frac{1}{2}(\mathbf{I} + g_1)$ to be the projector onto +1 eigenspace of g_1. Using $\text{tr}(A+B) = \text{tr}(A) + \text{tr}(B)$, we find $\text{tr}(P_+(g_1) \cdot g_2) = 0$. Thus g_r further divides +1 eigenspace of g_1 into two halves corresponding to +1 and -1 eigenspaces of g_2 each having dimensions 2^{n-2}. Inductively, g_2 divides +1 eigenspace of $g_1, \ldots, g_{r-1}$ in two halves corresponding to +1 and -1 eigenspaces of g_r each having dimensions 2^{n-r}.

For $\mathcal{S} = \langle g_1, \ldots, g_{n-k} \rangle$, $\mathcal{V}_\mathcal{S}$ has dimensions $2^{n-(n-k)} = 2^k$. □

Trivially, the *Stabilizer subgroup*, generated by $n-k$ elements, is *Abelian*. From Proposition 2.1, $|\mathcal{S}| = 2^{n-k}$. We can say that given $|\psi\rangle \in \mathcal{C}$, any $g \in \mathcal{S}$ acts as an identity on state $|\psi\rangle$.

Proposition 2.3. *For a stabilizer subgroup $\mathcal{S} = \langle g_1, \ldots, g_{n-k} \rangle$, fix $i \in \{1, \ldots, n-k\}$. There exists $e \in \mathcal{G}_n$ such that $\forall j \neq i$,*

$$\{e, g_i\} = 0, \quad \textit{and} \qquad [e, g_j] = 0. \tag{2.4}$$

Proof. Since $\mathcal{S}$ consists of $n-k$ independent operators, the check matrix corresponding to $\mathcal{S}$ also has $n-k$ linearly independent rows. Thus there exists a matrix $\Lambda_{n \times n-k}$ such that

$$H_{n-k \times n} \Lambda_{n \times n-k} = \mathbf{I}_{n-k}. \tag{2.5}$$

We can restructure Λ such that the symplectic product of H and Λ gives identity. This Λ can be mapped to $n-k$ operators, $\{e_1, \ldots, e_{n-k}\}$, each of which gives a column of $\mathbf{I}$ on symplectic product with H. Hence, for each $i, j \in \{1, \ldots, n-k\}, \exists e_i$ such that $\{e_i, g_i\} = 0$, and $[e_i, g_j] = 0 \ \ \forall j \neq i$. □

There must exist $n-k$ such operators and these act as generators of the group of correctable errors, $\mathcal{E}$. The fact that each, except identity operator, anti-commutes with at least one stabilizer generator helps us determine a bit-string called *syndrome* as defined below.

Distinct syndromes correspond to distinct errors and thus help us in detecting the specific error that has occurred on the codeword. Further for some $g \in \mathcal{S}, e \in \mathcal{E}$, and $\forall\, |\psi\rangle \in \mathcal{C}$,

$$g(e\,|\psi\rangle) = -eg\,|\psi\rangle = -1(e\,|\psi\rangle). \tag{2.6}$$

Since $(e\,|\psi\rangle)$ corresponds to the -1 eigenstate of a stabilizer generator, we conclude that an error, e takes a codeword out of the codespace.

Definition 2.2 (Syndrome)**.** Let $\mathcal{S} = \langle g_1, \ldots, g_{n-k}\rangle$ be the stabilizer subgroup and $e \in \mathcal{G}_n$ be an error. The syndrome $syn(e)$ for e is the bit string $l_1, \ldots, l_{n-k}$, determined by

$$l_i = \begin{cases} 0 & \text{if} \quad [e, g_i] = 0 \\ 1 & \text{if} \quad \{e, g_i\} = 1 \end{cases} \qquad (i = 0, \ldots, n-k). \tag{2.7}$$

Definition 2.3 (Normalizer subgroup)**.** Normalizer of a subgroup $\mathcal{S}$ in group $\mathcal{G}_n$ is the the set of elements $\mathcal{N}$ given by

$$\mathcal{N}_{\mathcal{G}_n}(\mathcal{S}) = \{g \in G_n \quad | \quad g\mathcal{S}g^{-1} \in \mathcal{S}\}. \tag{2.8}$$

For Pauli operators, the normalizer subgroup is equivalent to the centralizer subgroup and thus, $g\mathcal{S}g^{-1} = \mathcal{S}$. The normalizer subgroup contains all elements that commute with the stabilizer subgroup but do not necessarily commute with each other.

Theorem 2.1. *Two errors $e_1, e_2 \in \mathcal{G}_n$ have the same error syndrome $l = l_1 \ldots l_{n-k}$ if and only if they belong to the same coset of $\mathcal{N}_{\mathcal{G}_n}(\mathcal{S})$ [10].*

Proof. ($\Longrightarrow$) Let $S(e_1) = S(e_2) = l$. Then, for each generator g_i,

$$e_1 e_2 g_i = (-1)^{l_i} e_1 g_i e_2 = (-1)^{2l_i} g_1 e_1 e_2 = g_i e_1 e_2, \tag{2.9}$$

for all $i \in \{1, \ldots, n-k\}$. Thus $[e_1e_2, g_i] = 0 \implies e_1e_2 \in \mathcal{N}_{\mathcal{G}_n}(\mathcal{S})$. Since, $e_1^\dagger = \pm e_1$, $e_1^\dagger e_2 \in \mathcal{N}_{\mathcal{G}_n}(\mathcal{S})$. Let $e_1^\dagger e_2 = c$, then $e_2 = e_1 c \in e_1\mathcal{N}_{\mathcal{G}_n}(\mathcal{S})$. But, $e_2 \in e_2\mathcal{N}_{\mathcal{G}_n}(\mathcal{S})$. Since cosets do not overlap, $e_1\mathcal{N}_{\mathcal{G}_n}(\mathcal{S}) = e_2\mathcal{N}_{\mathcal{G}_n}(\mathcal{S})$. Thus both errors belong to the same coset.

($\Longleftarrow$) Let $e_1, e_2 \in e_1\mathcal{N}_{\mathcal{G}_n}(\mathcal{S})$. Thus, $\exists c \in e_1\mathcal{N}_{\mathcal{G}_n}(\mathcal{S})$ s.t. $e_2 = e_1 c$ and $e_1^\dagger e_2 = c \in \mathcal{N}_{\mathcal{G}_n}(\mathcal{S})$.

Thus,

$$[e_1^\dagger e_2, g_i] = 0 \implies [e_1 e_2, g_i] = 0. \tag{2.10}$$

let $S(e_1) = l, S(e_2) = l'$, then for any generator g_i,

$$e_1 e_2 g_i = (-1)^{l'_i} e_1 g_i e_2 = (-1)^{l'_i + l_i} g_i e_1 e_2. \tag{2.11}$$

From Equation 2.10 and Equation 2.11, we get $l_i + l'_i = 0 \implies l' = l$. Hence, $S(e_1) = S(e_2)$ [10]. □

Proposition 2.4. *The normalizer subgroup, $\mathcal{N}_{\mathcal{G}_n}(\mathcal{S})$ corresponding to above defined $\mathcal{S}$ is generated by $n + k$ elements and $|\mathcal{N}_{\mathcal{G}_n}(\mathcal{S})| = 4 \cdot 2^{n+k}$ [10].*

Proof. From Theorem 2.1, $\mathcal{N}_{\mathcal{G}_n}(\mathcal{S})$ partition $\mathcal{G}_n$. Since each coset has a unique syndrome, the number of cosets can not exceed the total number of syndrome values $= 2^{n-k}$.
Thus, the index $[\mathcal{G}_n : \mathcal{N}_{\mathcal{G}_n}(\mathcal{S})] = 2^{n-k}$ as here exists 2^{n-k} errors each having a different syndromes. Using Lagrange's theorem,

$$\begin{aligned} |\mathcal{N}_{\mathcal{G}_n}(\mathcal{S})| =& [\mathcal{G}_n : \mathcal{N}_{\mathcal{G}_n}(\mathcal{S})]/|\mathcal{G}_n|, \\ =& 2^{n-k}/2^{2n+2}, \\ =& 2^{n+k+2}. \end{aligned}$$

Therefore, $|\mathcal{N}_{\mathcal{G}_n}(\mathcal{S})| = 4 \cdot 2^{n+k}$. □

If we quotient out the center of $\mathcal{G}_n$, $\mathcal{Z}(\mathcal{G}_n) = \{\pm I, \pm iI\}$, we get

$$|\mathcal{N}_{\mathcal{G}_n}(\mathcal{S})/\mathcal{Z}(\mathcal{G}_n)| = 2^{n+k}. \tag{2.12}$$

The normalizer subgroup can be partitioned by a coset of the stabilizer subgroup. Thus, upto centralizer $\mathcal{Z}(\mathcal{G}_n)$,

$$|\mathcal{N}_{\mathcal{G}_n}(\mathcal{S})/\mathcal{S}| = 2^{n+k}/2^{n-k} = 2^{2k}. \tag{2.13}$$

Proposition 2.1 thus tells that we require $2k$ operators to generate the normalizer subgroup. These operators are often denoted by $\mathcal{N}_{\mathcal{G}_n}(\mathcal{S}) = \langle \bar{\mathbf{X}}_1, \ldots, \bar{\mathbf{X}}_k, \bar{\mathbf{Z}}_1, \ldots, \bar{\mathbf{Z}}_k \rangle$.

Further, the normalizer subgroup partitions the stabilizer subgroup. Consider $A \in \mathcal{N}_{\mathcal{G}_n}(\mathcal{S})$, $g \in \mathcal{S}$, and $|\psi\rangle \in \mathcal{V}_{\mathcal{S}}$, thus $[A, M] = 0$, and $A^\dagger M A = \pm M A^\dagger A = \pm M$. There-

fore,

$$M(A|\psi\rangle) = AM|\psi\rangle = (A|\psi\rangle) \implies A|\psi\rangle \in \mathcal{V}_{\mathcal{S}}. \tag{2.14}$$

Thus, the operators in the normalizer subgroup keep a state in the codespace within itself. Opposed to stabilizers, they do not fix each state in $\mathcal{C}$, instead, they take one state in the codespace to another state in the codespace. We call them the *logical operators*.

Theorem 2.2. *Let $\mathcal{S}$ be the stabilizer for a code $\mathcal{C}$. Suppose $\{E_j\}$ is a set of operators in $\mathcal{G}_n$ such that ${E_j}^\dagger E_k \notin \mathcal{N}_{\mathcal{G}_n}(\mathcal{S}) \setminus \mathcal{S}\ \forall j,k$. Then $\{E_j\}$ is a correctable set of errors. [6].*

Proof. Consider $E_j = E_j^\dagger$ and w.l.g., we only need to consider the case when $E_jE_k \notin \mathcal{N}_{\mathcal{G}_n}(\mathcal{S})\setminus \mathcal{S}$. Let P_+ be the projector onto $+1$ eigenspace of $\mathcal{S}$. Consider $E_jE_k \in \mathcal{S}$, then $P_+E_jE_kP_+ = P_+$ as projector will be invariant under $\mathcal{S}$.

Consider $E_jE_k \in \mathcal{G}_n \setminus \mathcal{N}_{\mathcal{G}_n}(\mathcal{S})\setminus\mathcal{S}$, from Proposition 2.3, $\exists g_1 \in \mathcal{S}$ such that $\{g_1, E_jE_k\} = 0$. Thus,

$$P_+E_jE_kP_+ = 0. \tag{2.15}$$

Thus E_jE_k satisfies Equation 1.12 and is a correctable error. □

For correctable errors, measure the generators g_1 through g_{n-k} and obtain the error syndrome with corresponding results β_1 through β_{n-k}. Suppose error E_j occur on qubit j, and the syndrome result is β_l corresponding to g_l,

- If E_j is the unique error operator with the obtained syndrome, we can recover the state by applying $E_j^\dagger$.
- If E_j and $E_{j'}$ both have the same obtained error syndrome, $E_jPE_j^\dagger = E_{j'}PE_{j'}^\dagger$ implies that applying $E_j^\dagger$ will recover the state.

A rigorous understanding of the stabilizer formalism can be best obtained through Chapters 2 and 3 in [10] and Chapter 10 in [6]. A shorter, nonetheless, robust review could be obtained from [26]. The original motivation for the stabilizer structure is given in [7], it also provides a few alternate mathematical interpretations of the same.

Alternate Picture

Consider a single qubit, all operations[1] on this qubit can be written in terms of Pauli group generated by $\langle \mathbf{X}, \mathbf{Z} \rangle$ operators. Now, we have n such qubits, $\{q_1, q_2, \ldots, q_n\}$ and the corresponding generators of all operators would be

$$\langle \mathbf{X}_1, \ldots, \mathbf{X}_n, \mathbf{Z}_1, \ldots, \mathbf{Z}_n \rangle.$$

Map each operator, $\mathbf{P}_i$, from the above generators to a n qubit operator denoted by $\bar{\mathbf{P}}_i$[2],

$$\langle \mathbf{X}_1, \ldots, \mathbf{X}_n, \mathbf{Z}_1, \ldots, \mathbf{Z}_n \rangle \longrightarrow \langle \bar{\mathbf{X}}_1, \ldots, \bar{\mathbf{X}}_n, \bar{\mathbf{Z}}_1, \ldots, \bar{\mathbf{Z}}_n \rangle.$$

It follows certain properties as following:

1. $[\bar{\mathbf{Z}}_i, \bar{\mathbf{Z}}_j] = 0 \ \ \forall i, j \in \{1, \ldots, n-k\}$.

2. $\{\bar{\mathbf{Z}}_i, \bar{\mathbf{X}}_i\} = 0$ while, $[\bar{\mathbf{Z}}_i, \bar{\mathbf{X}}_j] = 0$ and $[\bar{\mathbf{X}}_i, \bar{\mathbf{Z}}_j] = 0 \ \ \forall i \neq j \in \{n-k+1, \ldots, n\}$.

3. $[\bar{\mathbf{Z}}_i, \bar{\mathbf{Z}}_j] = 0$ and $[\bar{\mathbf{Z}}_i, \bar{\mathbf{X}}_j] = 0 \ \ \forall i \in \{1, \ldots, n-k\}, j \in \{n-k+1, \ldots, n\}$.

4. $\{\bar{\mathbf{Z}}_i, \bar{\mathbf{X}}_i\} = 0$ while, $[\bar{\mathbf{Z}}_i, \bar{\mathbf{X}}_j] = 0$ and $[\bar{\mathbf{X}}_i, \bar{\mathbf{Z}}_j] = 0 \ \ \forall i \neq j \in \{1, \ldots, n-k\}$.

These correspond to the properties of the stabilizer subgroup, normalizer subgroup, coset structure of normalizer-stabilizer, and relation of errors and stabilizers respectively. A visualization is presented in Figure 2.1, where all the relations are exactly as they should be if $\bar{P}_i = P_i$. This forms the picture of k logical qubits encoded in n physical qubits where the blue $n-k$ operators act as an identity, the green $2k$ operators act as logical operators akin to operators in $\mathcal{G}_k$ and the red $n-k$ operators represent the physical errors.

The stabilizer formalism gives a nice visual structure to QECCs. In the next section, we will discuss how to implement a stabilizer code on qubits and design a circuit to implement any code given its stabilizers.

[1] We are excluding factors of $\pm 1, \pm i$

[2] $\bar{\mathbf{P}}_i$ is not some Pauli matrix, but a notation to an n qubit operator.

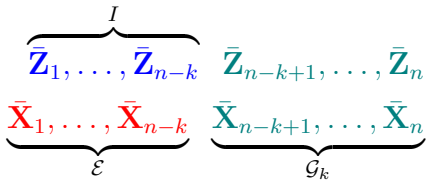


Figure 2.1: Visualization of operators in $\mathcal{G}_n$ corresponding to operators in $\mathcal{G}_k$. Stabilizers act as identity, Normalizers up to stabilizers act as $\mathbf{X}$ and $\mathbf{Z}$ operators on k qubits, and the rest up to stabilizers are the correctable errors.

2.2 Encoding

The most crucial step of error correction is to map information stored in 2^k dimensional Hilbert space of k qubits to a 2^k dimensional subspace ($\mathcal{C}$) of 2^n dimensional Hilbert Space of n qubits. This process is known as *Encoding*. The encoded state lies in the codespace, $\mathcal{C} \equiv \mathcal{V}_{\mathcal{S}}$. This encoded state is prone to errors due to the physical environment which alters the state. A set of procedures that detect this error and correct the altered state back to the encoded state is known as *Decoding*. We can further map the decoded state back to the original k qubit quantum state.

The implementation outline is shown in figure 2.2. The decoding step consists of two steps, as we call them, *error detection* and *error correction*. Error detection requires ancilla qubits (extra qubits) which capture the syndrome as defined above. Based on syndrome outcomes, the Detector applies certain operators on the altered state to bring it back to the encoded state.

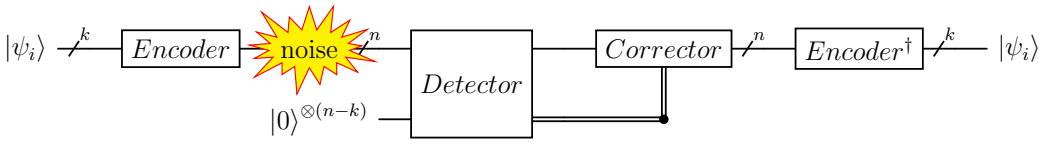


Figure 2.2: Error correction procedure for an [[n,k]] stabilizer code.

In the next sections, we will discuss two ways of encoding: a non-unitary method based on the measurement of stabilizer generators, and a unitary method based on the standard form of *check matrix* corresponding to stabilizer generators. We will further advance the latter method to overcome its limitations and make it applicable to any stabilizer code.

2.2.1 Measurement Based Encoding

Recall that in Proposition 2.2, we have shown that $\mathcal{C}$ corresponds to $+1$ eigenspace of all the stabilizer generators. Thus, encoding here involves projecting the initial state to $+1$ eigenspace of each stabilizer generator sequentially. Note that $P_+(M) = \frac{1}{2}(\mathbf{I} + M)$ is the projector operator which projects any state onto $+1$ eigenspace of M, while $P_-(M) = \frac{1}{2}(\mathbf{I} - M)$ is the projector operator which projects any state into -1 eigenspace of M. Our goal is to create a quantum circuit that evolves an initial state, $|\psi_i\rangle$, to an encoded state, $|\psi_e\rangle$,

$$|\psi_e\rangle = \frac{1}{2^{n-k}}(\mathbf{I} + M_{n-k}) \cdots (\mathbf{I} + M_1) |\psi_i\rangle , \tag{2.16}$$

where, $\{M_1, \ldots, M_{n-k}\}$ are the stabilizer generators. Throughout this section, we will denote stabilizer generators by notation M.

In the encoding circuit, we have a quantum register of n qubits, which we call the *Data* register and an ancillary register of $n-k$ qubits which we call the *Ancilla* register. We shall apply a controlled-M_i operator, having control on ancilla initialized in $|+\rangle$ state and target on the data qubits. It creates a superposition of $P_+(M_i)$ and $P_-(M_i)$ operator on the data qubits. Moreover, the data register is now entangled with the ancilla register such that if on measurement, ancilla collapses to state $|+\rangle$ then the data state is a $+1$ eigenstate of M_1 else a -1 eigenstate of M_1.

Consider any initial n qubit state $|\psi_i\rangle$. It need not contain any information to be encoded, and thus a preferable choice is to take $|\psi_i\rangle = |0\rangle^{\otimes n}$. The circuit as outlined above is shown in Figure 2.3 for two stabilizer generators. Ancilla is initialised in $|0\rangle$ state and acted upon by $\mathbf{H}$ to get $|+\rangle$ state. The measurement is in $\mathbf{Z}$ basis and thus an $\mathbf{H}$ is required to change $\mathbf{X}$ basis to $\mathbf{Z}$ basis. Notice how the state evolves

$$\begin{aligned}
|0\rangle |\psi_i\rangle &\xrightarrow{\mathbf{H}_0} \frac{1}{\sqrt{2}}\Big[|0\rangle |\psi_i\rangle + |1\rangle |\psi_i\rangle \Big] \\
&\xrightarrow{cM_1} \frac{1}{\sqrt{2}}\Big[|0\rangle |\psi_i\rangle + |1\rangle (M_1 |\psi_i\rangle)\Big] \\
&\xrightarrow{\mathbf{H}_0} \frac{1}{\sqrt{4}}\Big[|0\rangle |\psi_i\rangle + |1\rangle |\psi_i\rangle + |0\rangle (M_1 |\psi_i\rangle) - |1\rangle (M_1 |\psi_i\rangle)\Big] \\
&= \frac{1}{2} |0\rangle \Big[|\psi_i\rangle + (M_1 |\psi_i\rangle)\Big] + \frac{1}{2} |1\rangle \Big[|\psi_i\rangle - (M_1 |\psi_i\rangle)\Big], \\
\therefore |\psi_1\rangle &= |0\rangle \Big[\frac{1}{2}(\mathbf{I} + M_1) |\psi_i\rangle \Big] + |1\rangle \Big[\frac{1}{2}(\mathbf{I} - M_1) |\psi_i\rangle \Big].
\end{aligned} \tag{2.17}$$

Repeating the same with another ancilla qubit, we get,

$$\begin{aligned}|0\rangle|\psi_1\rangle \xrightarrow{\mathbf{H}_1 cM_2 \mathbf{H}_1} |0\rangle|0\rangle\left[\frac{1}{4}(\mathbf{I}+M_2)(\mathbf{I}+M_1)|\psi_i\rangle\right] \\ +|0\rangle|1\rangle\left[\frac{1}{4}(\mathbf{I}+M_2)(\mathbf{I}-M_1)|\psi_i\rangle\right] \\ +|1\rangle|0\rangle\left[\frac{1}{4}(\mathbf{I}-M_2)(\mathbf{I}+M_1)|\psi_i\rangle\right] \\ +|1\rangle|1\rangle\left[\frac{1}{4}(\mathbf{I}-M_2)(\mathbf{I}-M_1)|\psi_i\rangle\right].\end{aligned} \tag{2.18}$$

Repeat the process with all $n-k$ generators. The evolved state $|\psi_{n-k}\rangle$ would thus be

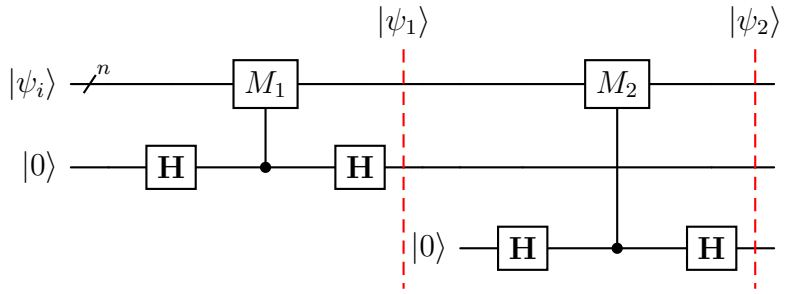


Figure 2.3: Projecting an n qubit state onto a superposition of +1 and -1 eigenspace of M_1 and M_2 operators, using ancillary qubits.

$$|\psi_{n-k}\rangle = \Big[\prod_{j=1}^{n-k}\sum_{t\in\{+,-\}} P_t(M_j)\Big]|\psi_i\rangle\,. \tag{2.19}$$

Measuring the ancilla qubits will change the state to

$$|\psi_{n-k}\rangle = \prod_{j=1}^{n-k} P_{+/-}(M_j)|\psi_i\rangle\,, \tag{2.20}$$

where the index sign of P depends on the measurement outcome. If ancilla measures 0, the sign is $+$ else $-$ sign appears. The desired data state is

$$|\psi_e\rangle = \prod_{j=1}^{n-k} P_+(M_j)|\psi_i\rangle\,. \tag{2.21}$$

To get this state, we need to change every $P_-(M_j)$ to $P_+(M_j)$ in the obtained state,

$|\psi_{n-k}\rangle$.

Proposition 2.3 guarantees that for each $M_j, \exists E_j$ such that $\{E_j, M_j\} = 0$, and $[E_j, M_i] = 0$ for all $i \neq j$. Create an operator $\prod_{j=1}^{n-k} E_j^{t_j}$, where $t_j \in \{0, 1\}$ is the measurement outcome for j^{th} ancilla. Note that $E_j P_-(M_j) = P_+(M_j) E_j$ and thus

$$\prod_{j=1}^{n-k} E_j^{t_j} \prod_{j=1}^{n-k} P_{+/-}(M_j) |\psi_i\rangle = \prod_{j=1}^{n-k} P_+(M_j) \prod_{j=1}^{n-k} E_j^{t_j} |\psi_i\rangle . \tag{2.22}$$

Since the initial state was supposed to be a random state, we can call that the state $\prod_{j=1}^{n-k} E_j^{t_j} |\psi_i\rangle$ is the initial n qubit state and hence, the final state is an encoded state $|\psi_e\rangle \in \mathcal{C}$.

Continuing with the example above, say, we get measurement results, $|1\rangle |0\rangle$, and data state $\frac{1}{4}(\mathbf{I} - M_2)(\mathbf{I} + M_1) |\psi\rangle$. We apply an operator E_2 such that $\{E_2, M_2\} = 0, [E_2, M_1] = 0$. Therefore,

$$|\psi_2\rangle = \frac{1}{4} E_2 (\mathbf{I} - M_2)(\mathbf{I} + M_1) |\psi\rangle = \frac{1}{4} (\mathbf{I} + M_2)(\mathbf{I} + M_1)(E_2 |\psi_i\rangle). \tag{2.23}$$

The measurement-based encoding procedure takes in any random input state and returns some state in the codespace. We can be more resourceful and keep using the same qubit as ancilla for measurement as shown in figure 2.4. This circuit is much more efficient if a fast qubit reset is allowed. Since we keep measuring ancilla after each controlled-M_i, simulation is required to store fewer superposition states, further, we can apply E_i as soon as we detect 1 in measurement outcome and keep the state in $+1$ eigenspace of each generator at each step.

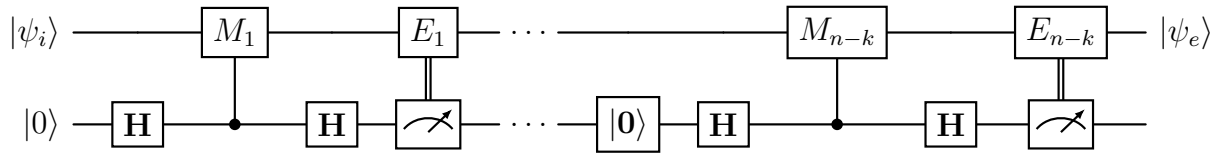


Figure 2.4: Encoding any n qubit initial state, $|\psi_i\rangle$ to a logical state $|\psi_e\rangle$ by projecting onto $+1$ eigenspace of stabilizer generators, $\{M_1, \ldots, M_{n-k}\}$.

Be aware that we haven't yet encoded the k qubits' information. $|\psi_e\rangle$ obtained above is some random state in the codespace. We need to perform further operations to map

this state to information in k qubits. This is achieved by applying the $2k$ logical operators (Normalizers).

We shall apply projectors of logical $\bar{\mathbf{Z}}$ operators. A logical $|\bar{0}_j\rangle$ state corresponds to +1 eigenspace of logical $\bar{\mathbf{Z}}_j$ while logical $|\bar{1}_j\rangle$ state corresponds to -1 eigenspace of logical $\bar{\mathbf{Z}}_j$. To obtain an encoded $|\overline{c_1 c_2 \dots c_k}\rangle$ state, where $j \in \{0, 1\}$, apply the same procedure with stabilizer generators replaced by logical $\bar{\mathbf{Z}}$ operators and measure ancilla outcomes $m_1, \dots, m_k$ for $\bar{\mathbf{Z}}_1, \dots, \bar{\mathbf{Z}}_k$ operators. If $c_j \neq m_j$, for any j, apply logical $\bar{\mathbf{X}}_j$ gate to the data register. This ensures that any k qubit state can be mapped to the codespace, $\mathcal{C}$ [6]. The complete procedure is shown in Figure 2.5.

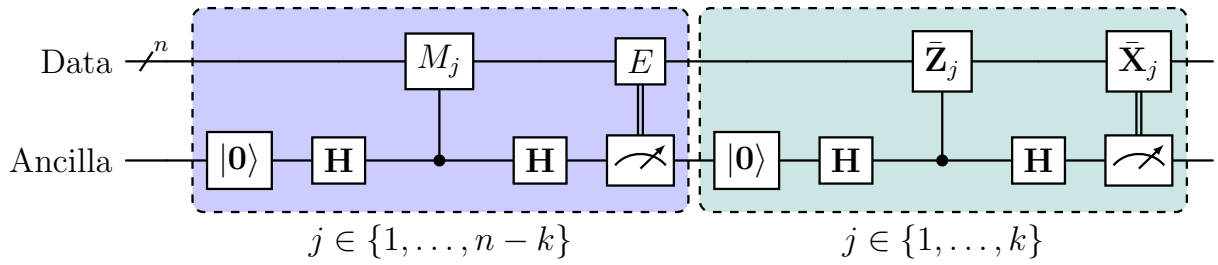


Figure 2.5: Projecting any n qubit data state to +1 eigenspace of stabilizer and logical $\bar{\mathbf{Z}}$ generators to generate logical $|\bar{0}\rangle$ state.

Measurement-based encoding is *irreversible* as we apply some gates based on measurement outcomes. Since the measurement is a non-unitary gate, the given encoding scheme is also non-unitary. In the next section, we will discuss an efficient approach to encoding which is unitary.

2.2.2 Efficient Encoding

A unitary Encoder will be free from ancilla qubits and measurement outcomes. Projecting onto +1 eigenspace of stabilizer generators requires sequential application of $\frac{1}{2}(\mathbf{I} + M_i)$ operators. We shuffle the generators in a particular order to obtain another set of generators by converting the check matrix, H to its standard form given in Equation 2.24.

$$\left[\begin{array}{c|c} H^x_{n-k\times n} & H^z_{n-k\times n} \end{array}\right] \longrightarrow \left[\begin{array}{ccc|ccc} I_{r\times r} & A_{r\times l} & A'_{r\times k} & B_{r\times r} & 0_{r\times l} & C_{r\times k} \\ 0_{l\times r} & 0_{l\times l} & 0_{l\times k} & D_{l\times r} & I_{l\times l} & E_{l\times k} \end{array}\right]. \tag{2.24}$$

Here, $l = n-k-r$. The transformation is obtained by *Gaussian Elimination* of H involving the addition of rows and swapping of columns. The standard form, H', signifies another set of stabilizer generators, $M'_1, \ldots, M'_{n-k}$ as

1. Addition of row i to row j is equivalent to multiplying M_i to M_j giving yet another stabilizer generator.

2. Swapping column i to j is equivalent to relabeling qubit i as j and j as i.

We can also obtain the new logical operators, $\bar{\mathbf{X}}'$ and $\bar{\mathbf{Z}}'$, as shown in Equation 2.25.

$$\left[\begin{array}{c} \bar{\mathbf{X}}' \\ \bar{\mathbf{Z}}' \end{array}\right] = \left[\begin{array}{ccc|ccc} 0_{k\times r} & E^T_{k\times l} & I_{k\times k} & C^T_{k\times r} & 0_{k\times l} & 0_{k\times k} \\ 0_{k\times r} & 0_{k\times l} & 0_{k\times k} & A'^T_{k\times r} & 0_{k\times l} & I_{k\times k} \end{array}\right]. \tag{2.25}$$

The structure of new generators is such that for $i \in \{1, \ldots, r\}$, M'_i contains an $\mathbf{X}$ or $\mathbf{Y}$ at i^{th} qubit and no where else until qubit r and the generators $M'_{r+1}, \ldots, M'_{n-k}$ contain only $\mathbf{Z}$ operators on any position. This gives us an advantage in building the Encoder circuit explained below.

We shall start with $|0\rangle^{\otimes n}$ state. This state is already in the $+1$ eigenspace of the last l stabilizer generators and thus operators $M'_{r+1}, \ldots, M'_{n-k}$ do not alter the state. Apply a $\mathbf{H}$ gate on the first qubit. And controlled on it apply M'_{1j} operator on qubit j where $j \in \{2, \ldots, n\}$. Notice the state has changed to

$$\frac{1}{\sqrt{2}}(|0\rangle^{\otimes n} + M'_1 |0\rangle^{\otimes n}) = \frac{1}{\sqrt{2}}(\mathbf{I} + M'_1) |0\rangle^{\otimes n}. \tag{2.26}$$

For M'_2, apply $\mathbf{H}$ on the second qubit. Controlled on it, apply M'_2 on other qubits. The state has now changed to

$$\frac{1}{2}(|0\rangle^{\otimes n} + M'_2 |0\rangle^{\otimes n} + M'_1 |0\rangle^{\otimes n} + M'_2 M'_1 |0\rangle^{\otimes n}) = \frac{1}{2}(\mathbf{I} + M'_2)(\mathbf{I} + M'_1) |0\rangle^{\otimes n}. \tag{2.27}$$

Repeating the same until M'_r, we get an encoded state. This is crucially due to $I_{r\times r}$ in the standard form which does not flip state $|0\rangle$ until qubit r. The obtained state is a logical $|0\rangle$ state. Since stabilizer generators and logical operators commute, we are free to apply them in any order. Consider that we first apply the logical $\bar{\mathbf{Z}}'$ which has only $\mathbf{Z}$ operators on the first r and last k qubits and no $\mathbf{X}$ operators. The qubits are in $|0\rangle$ state and thus, we need

not apply any operator at all.

To encode a basis state, we would be required to apply respective logical $\bar{\mathbf{Z}}'$ and $\bar{\mathbf{X}}'$ operators. That is, encoded $|\overline{c_1 \dots c_k}\rangle$ is related to encoded $|\bar{0}\rangle$ as

$$|\overline{c_1 \dots c_k}\rangle = \left[\alpha \bar{\mathbf{X}}_1^{c_1} \dots \bar{\mathbf{X}}_k^{c_k} \times \prod_{j=1}^{n-k} (\mathbf{I} + M_j)\right] |0\rangle^{\otimes n} . \tag{2.28}$$

Through the standard form of logical operators, we can obtain $|\overline{c_1 \dots c_k}\rangle$ directly without the above action. To do so, we should first initialize the last k qubits to state of interest $|\phi\rangle = |c_1 \dots c_k\rangle$, then apply controlled $\bar{\mathbf{X}}_j$ gates controlled on qubit $l + r + j$ where $j \in \{1, \dots, k\}$. This is because the standard form of $\bar{\mathbf{X}}_j$ has 1 at the respective data qubit (c_j). We will be only be required to apply $\mathbf{CX}$ gates from the bottom k qubits to the middle l qubits corresponding to logical $\bar{\mathbf{X}}'$ given in Equation 2.25 as $\mathbf{Z}$ operators due to C^T will not affect $|0\rangle$ states. Hence, up to a normalization factor α,

$$|\overline{c_1 \dots c_k}\rangle = \alpha \times \prod_{j=1}^{n-k} (\mathbf{I} + M_j) \bar{\mathbf{X}}_1'^{c_1} \dots \bar{\mathbf{X}}_k'^{c_k} |0\rangle^{\otimes n-k} |c_1 \dots c_k\rangle . \tag{2.29}$$

Due to the linear nature of quantum mechanics, any superposition of basis states with any phase would propagate to the same linear superposition of encoded basis states. This removes any requirements to apply logical $\bar{\mathbf{Z}}$ gates. Consequently, the efficient Encoder is capable of encoding any k qubit state to an n qubit state in the codespace, $\mathcal{C}$ [27, 28].

Let us take an example to elucidate the encoding scheme. Figure 2.6 presents an illustration of the efficient Encoder circuit with $r = 2, l = 1$. Starting with $|\psi_i\rangle = |00\rangle |0\rangle |\phi\rangle$,

the state evolution is as follows,

$$
\begin{aligned}
|00\rangle\,|0\rangle\,|\phi\rangle \xrightarrow{c\bar{\mathbf{X}}'_1} & |00\rangle\,(\bar{\mathbf{X}}'_1\,|0\rangle)\,|\phi\rangle\,, \\
& \text{(Similarly, all } X_i \text{ only apply } cX \text{ with target on } l \text{ qubits)} \\
\xrightarrow{\mathbf{H}_1} & \frac{1}{\sqrt{2}}\Big[\,|00\rangle\,(\bar{\mathbf{X}}'_1\,|0\rangle)\,|\phi\rangle + |10\rangle\,(\bar{\mathbf{X}}'_1\,|0\rangle)\,|\phi\rangle\,\Big], \\
\xrightarrow{cM_1} & \frac{1}{\sqrt{2}}(\mathbf{I}+M'_1)\Big[\,|00\rangle\,(\bar{\mathbf{X}}'_1\,|0\rangle)\,|\phi\rangle\,\Big], \\
\xrightarrow{\mathbf{H}_2} & \frac{1}{2}\Big[(\mathbf{I}+M'_1)\big(\,|00\rangle\,(\bar{\mathbf{X}}'_1\,|0\rangle)\,|\phi\rangle\,\big) + (\mathbf{I}+M'_1)\big(\,|01\rangle\,(\bar{\mathbf{X}}'_1\,|0\rangle)\,|\phi\rangle\,\big)\Big], \\
\xrightarrow{cM_2} & \frac{1}{2}(\mathbf{I}+M'_2)(\mathbf{I}+M'_1)\Big[\,|00\rangle\,(\bar{\mathbf{X}}'_1\,|0\rangle)\,|\phi\rangle\,\Big].
\end{aligned}
$$

Therefore, the encoded state is,

$$
|\psi_e\rangle = (\bar{\mathbf{X}}_1^{\phi_1}\cdots\bar{\mathbf{X}}_k^{\phi_k})(\mathbf{I}+M'_r)\cdots(\mathbf{I}+M'_1)\,|\psi_i\rangle\,. \tag{2.30}
$$

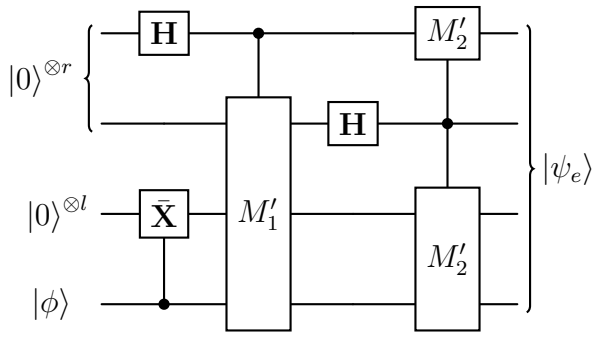


Figure 2.6: An schematic of the efficient Encoder example for 2 stabilizers (M'_1, M'_2) and logical $\bar{\mathbf{X}}$ in standard form.

We note that the standard form generates a set of stabilizer generators but it is different from the original generators as certain column swap operations have relabeled the qubits. If we wish to use the original labels, we need to apply **SWAP** gates in the reverse order of column swaps while creating standard form. It ensures that the qubits are labeled back to their original form.

We can further decrease the number of gates by removing any **CZ** operations in the Encoder circuit, whose control is qubit i and the target is qubit j such that $i < j < r$, as qubit j would be in state $|0\rangle$ which remains unchanged on applying **CZ** gate [29]. This

encoding scheme uses at most r $\mathbf{H}$ gates, $n-1$ controlled gates for each r qubits, and l controlled gates for each of k logical $\bar{\mathbf{X}}$. Thus, the time complexity of the circuit is $\mathcal{O}(n^2)$ as

$$r + r(n-1) + k(n-k-r) = (k+r)(n-k) \leq n(n-k). \tag{2.31}$$

Since the input itself has $n-k$ stabilizer generators each with n operators, the proposed encoding scheme is efficient [10].

2.2.3 Efficient Encoding for all Codes

The efficient encoding mechanism in [27] defines $\mathbf{Y} \coloneqq \mathbf{XZ}$ and omits any associated phase[10]. We notice that $\mathbf{Y}^2 = \mathbf{I}$ but $(\mathbf{XZ})^2 = -\mathbf{I}$, which might change the syndrome values and the encoded state. Thus, the above method suits perfectly to CSS codes but fails with certain non-CSS codes.

To incorporate the required sign, we add another column, *phase column*, in the check matrix, H. Corresponding to $\{+, i, -, -i\}$, phase of the stabilizer, we add values $\{0, 1, 2, 3\}$ respectively in the last column. Similar row operations (modulo 4 for phase columns) are applicable leading to the standard phase column of the standard matrix.

Each Pauli $\mathbf{Y}$ in the standard form stabilizers devours a multiple of i from the phase column. We can then obtain the standard stabilizer generators and corresponding phases. While creating circuit, we add an $\mathbf{S}$ gate after $\mathbf{H}$ gate to encode $(\mathbf{I}+\mathbf{Y})\,|0\rangle$. And to account for the stabilizer phase, add an $\mathbf{S}, \mathbf{Z}$ or $\mathbf{SZ}$ gate respectively for $i, -1$, and $-i$ in the phase column. We check if the state has been mapped to $+1$ eigenspace of all stabilizer generators. Through additional $\mathbf{S}$ and $\mathbf{Z}$ operators, we can correct any phase difference in the encoded state leading to unexpected syndrome values [17].

2.3 Decoding

An encoded state is used to perform any quantum computation that must be error-free. Environmental noise might take an encoded state out of the codespace. Decoding concerns with bringing the state back to the codespace without altering the encoded information. We

first need to extract the syndrome corresponding to a state and then correct the error that occurred on the state. Error correction can be done using different techniques, however, we shall only discuss the *lookup-table* decoding method.

Syndrome extraction is similar to measurement-based encoding as the premise is to project onto +1 eigenspace of stabilizer generators. Following the example above, consider the encoded state to be $|\psi_2\rangle$ and initialize an ancilla qubit in state $|0\rangle$. Now $|\psi_2\rangle$ is a +1 eigenstate of all stabilizer elements. We can then expect to get syndrome $|0\rangle$ when any stabilizer is measured.

$$\begin{aligned}
|0\rangle |\psi_2\rangle &\xrightarrow{\mathbf{H}_0 c M_1 \mathbf{H}_0} |0\rangle \left[\frac{1}{2}(\mathbf{I}+M_1)|\psi_2\rangle\right] + |1\rangle \left[\frac{1}{2}(\mathbf{I}-M_1)|\psi_2\rangle\,,\right] \\
&= |0\rangle \left[\frac{1}{2}(\mathbf{I}+\mathbf{I})|\psi_2\rangle\right] + |1\rangle \left[\frac{1}{2}(\mathbf{I}-\mathbf{I})|\psi_2\rangle\right], \\
&= |0\rangle |\psi_2\rangle\,.
\end{aligned}$$

And we obtain the syndrome value as expected. We can continue for all $n-k$ stabilizer generators to obtain the complete syndrome, l. This step is the syndrome extraction step. The application of g based on syndrome qubits is the same as *error correction*.

Each code can correct certain errors only which are limited to the maximum number of syndrome values possible, i.e., 2^{n-k}. In general, the distance of a code is an indicator of how many errors can the quantum code correct. For example, $d=3$ represents the code that can correct single-qubit errors while $d=5$ represents that the code can correct up to two-qubit errors. For a lookup table decoder, we need to find syndromes for each of these possible errors.

Take a correctable error e, and find its commutation relation with each stabilizer generator M_i. This is equivalent to finding the symplectic product of e with each M_i giving us l_i for each $i \in \{1,\ldots,n-k\}$. Thus, we obtain syndrome $l(e)$. Similarly, we can obtain syndromes for each correctable error and create a table of syndromes and corresponding error operators. This is known as the *lookup table*. We need to ensure $\mathrm{wt}(e_i) \leq \lfloor \frac{d-1}{2} \rfloor$. If two correctable errors $e_i \neq e_j$ are such that $l(e_i) = l(e_j)$, then the code is said to be degenerate and the same correction operator is used. Traditionally, a lookup table consists of an error operator corresponding to each possible syndrome value. It could be done by noting down $l(e)$ for all $e \in \mathcal{G}_n$ in ascending order of weight. For each possible l, we assign an error operator and hence lookup table can correct up to 2^{n-k} errors.

Once we obtain the lookup table for the code of interest. We search, classically, for the syndrome obtained in the previous step in the lookup table and apply the corresponding operator on the data register to correct the error detected.

Examples

We present stabilizers of a few codes of interest to us. Our experiments involved many more codes primarily taken from [30].

Stabilizers					
M_1	X	Z	Z	X	I
M_2	I	X	Z	Z	X
M_3	X	I	X	Z	Z
M_4	Z	X	I	X	Z
Logicals					
$\bar{\mathbf{X}}_1$	X	X	X	X	X
$\bar{\mathbf{Z}}_1$	Z	Z	Z	Z	Z

Table 2.1: [[5,1,3]] Perfect code.

Stabilizers							
M_1	I	I	I	X	X	X	X
M_2	I	X	X	I	I	X	X
M_3	X	I	X	I	X	I	X
M_4	I	I	I	Z	Z	Z	Z
M_5	I	Z	Z	I	I	Z	Z
M_6	Z	I	Z	I	Z	I	Z
Logicals							
$\bar{\mathbf{X}}_1$	X	X	X	X	X	X	X
$\bar{\mathbf{Z}}_1$	Z	Z	Z	Z	Z	Z	Z

Table 2.2: [[7,1,3]] Steane code.

2.4 Performance Analysis

Once a complete error-correcting circuit is created with all its components - encoding, extraction of the syndrome, correcting errors based on the syndrome, and decoding, the natural

Stabilizers									
M_1	Z	Z	I	I	I	I	I	I	I
M_2	I	Z	Z	I	I	I	I	I	I
M_3	I	I	I	Z	Z	I	I	I	I
M_4	I	I	I	I	Z	Z	I	I	I
M_5	I	I	I	I	I	I	Z	Z	I
M_6	I	I	I	I	I	I	I	Z	Z
M_7	X	X	X	X	X	X	I	I	I
M_8	I	I	I	X	X	X	X	X	X
Logicals									
$\bar{\mathbf{X}}_1$	X	X	X	X	X	X	X	X	X
$\bar{\mathbf{Z}}_1$	Z	Z	Z	Z	Z	Z	Z	Z	Z

Table 2.3: [[9,1,3]] Shor code.

course of action is to implement the circuit in the presence of errors. The entire process of QEC can be summarized as follows:

1. Initialize an n qubit state, $|0\rangle^{\otimes n-k}|\phi\rangle$ where $|\phi\rangle$ is the k qubit state to be encoded.
2. Apply a unitary quantum circuit called *Encoder* which maps the above n qubit state to the simultaneous $+1$ eigenspace of all stabilizer generators, giving the $|\psi_e\rangle$ state.
3. Take an ancilla qubit in $|+\rangle$ state and apply controlled M_1 operation on the encoded state, controlled on the ancilla.
4. Measure the ancilla qubit in $\mathbf{X}$[3] basis and store the outcome. Reset the ancilla to $|+\rangle$ state and repeat step 3 for all M_i where $i \in \{2, \ldots, n-k\}$.
5. Combine the stored outcomes in order and call it the *syndrome*. Feed the syndrome to a Corrector function, which applies the appropriate recovery operation.
6. Since the action of the Encoder is a unitary operation, we can apply the inverse of each operator in the Encoder, in reverse order, to obtain the decoded state, $|\psi_d\rangle$.

A schematic of the steps involved in QEC is presented in Fig. 2.7. We wish to evaluate the performance of a given code using noise models based on real-world noise in physical systems.

[3]Change of basis is achieved by $\mathbf{H}$ operation.

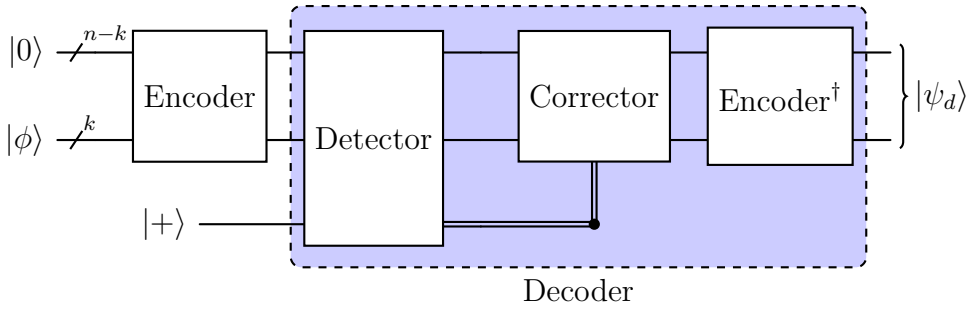


Figure 2.7: Block diagram description of an $[[n, k]]$ quantum error correcting code, using efficient unitary Encoder, and measurement decoder requiring single ancilla qubit with fast reset. Message state $|\phi\rangle$ is entangled with $|0\rangle^{\otimes n-k}$ state and the encoded state thus obtained passes through error correcting decoder giving us the decoded state, $|\psi_d\rangle$.

2.4.1 Noise Models

We model noise as a quantum operation acting on the state space of the principal system. These operations, represented by Kraus operators, can also be modeled through Pauli matrices in the case of Clifford operations. We shall discuss below some of the most common noise models in quantum computing.

Bit Flip & Phase Flip Noise

Bit flip noise flips the qubit with probability p. The corresponding Kraus operators will be

$$E_0 = \sqrt{p}\mathbf{I}; \quad E_1 = \sqrt{1-p}\mathbf{X}. \tag{2.32}$$

The phase flip operator flips the phase from $+$ to $-$ and vice versa. The Kraus operators for phase flip noise are

$$E_0 = \sqrt{p}\mathbf{I}; \quad E_1 = \sqrt{1-p}\mathbf{Z}. \tag{2.33}$$

In simulations, it can be modeled by applying an $\mathbf{X}$ or $\mathbf{Z}$ gate with probability p, for bit-flip and phase-flip noise respectively.

Amplitude Damping Noise

In physical systems, one of the states of a qubit is preferred over the other. For example, a preferred ground state or loss of a photon to the environment. The net result is a decrease in the amplitude of a state, or an increase in amplitude of a preferred state, leading to amplitude damping noise. The corresponding Kraus operators are

$$E_0 = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{bmatrix}, \text{ and } E_1 = \begin{bmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{bmatrix}. \tag{2.34}$$

These operators are non-unitary. E_0 keeps state $|0\rangle$ unchanged and reduce the amplitude of state $|1\rangle$, while E_1 changes $ket0$ to $|1\rangle$ with probability γ. It can be modeled using controlled RY gates and CX gates.

Phase Damping Noise

It is also possible that the total energy of the system is conserved but there is a loss of quantum information. For example, scattering of a photon in a waveguide or interaction of an electron with some distant electron causes perturbation. Such errors could be summarized by the following Kraus operators,

$$E_0 = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{bmatrix}, \text{ and } E_1 = \begin{bmatrix} 0 & 0 \\ 0 & \sqrt{\gamma} \end{bmatrix}. \tag{2.35}$$

It can be modeled using a controlled RY gate.

Standard depolarizing Noise

Depolarization is a widely encountered error in which a qubit is replaced by a completely mixed state $\mathbf{I}/2$ with probability p giving us

$$\mathcal{E}(\rho) = \frac{p\mathbf{I}}{2} + (1-p)\rho. \tag{2.36}$$

Since $\frac{I}{2} = \frac{\rho+\mathbf{X}\rho\mathbf{X}+\mathbf{Y}\rho\mathbf{Y}+\mathbf{Z}\rho\mathbf{Z}}{4}$, we can write the depolarizing channel as

$$\mathcal{E}(\rho) = \Big(1 - \frac{3p}{4}\Big)\rho + \frac{p}{4}(\mathbf{X}\rho\mathbf{X} + \mathbf{Y}\rho\mathbf{Y} + \mathbf{Z}\rho\mathbf{Z}). \tag{2.37}$$

It is often convenient to represent the depolarizing channel as

$$\mathcal{E}(\rho) = (1-p)\rho + \frac{p}{3}(\mathbf{X}\rho\mathbf{X} + \mathbf{Y}\rho\mathbf{Y} + \mathbf{Z}\rho\mathbf{Z}). \tag{2.38}$$

Equation 2.38 is equivalent to applying $\mathbf{X}, \mathbf{Y}, \text{and}, \mathbf{Z}$ gates with probability $p/3$ each [6].

Noise models are used to simulate the effect of noise on the quantum circuit. It is at the discretion of the user to choose the noise model based on the physical system being used. Circuit level noise considers each element in the circuit to be noisy including gates, measurements, reset operations, as well as idle locations where no gate is applied, to mimic the physical systems as closely as possible. We shall apply noise only at particular elements of the circuit and evaluate the performance of the code.

2.4.2 Error Rates and Thresholds

The criterion used to characterize different codes is *error rate* which tells us the rate of loss of fidelity or simply how many times we didn't get the expected output.

- **Physical Error Rate:** Error rate that occurs naturally in a physical system due to decoherence i.e., the probability with which noise occurs.
- **Logical Error Rate:** Error rates obtained after implementing error correction, due to logical errors created.
- **Total Error Rate:** Error rates obtained due to any uncorrectable error including logical errors.
- **Fidelity:** Probability of error corrected state to be same as the initial state.
- **Threshold:** The value of the error rate below which a suitably configured quantum computer can be used to perform arbitrarily long computations. Threshold exists only for a family of codes. For a single code, the threshold value is called *pseudo-threshold*

We first present an analytical approach to calculate the error rates of a code. Consider a physical system in which a qubit decoheres with probability p. We'll use an error correcting scheme, using a $[[n, k, d]]$ code, such that the probability of an *error-full* final state, $p_{\text{error}} < p$.

Let us use a code of distance, $d = 3$, i.e., it can correct a single qubit error. Thus,

$$p_{\text{error}} = 1 - \{(1-p)^n + n \cdot p(1-p)^{n-1}\} \sim \mathcal{O}(n^2 p^2). \tag{2.39}$$

Similarly, for a code having $d = 5$,

$$p_{\text{error}} = 1 - \left\{(1-p)^n + \binom{n}{1} \cdot p(1-p)^{n-1} + \binom{n}{2} \cdot p^2(1-p)^{n-2}\right\} \sim \mathcal{O}(np^2). \tag{2.40}$$

For a code having $d = 2l + 1$,

$$p_{\text{error}} = 1 - \left\{(1-p)^n + \binom{n}{1} \cdot p(1-p)^{n-1} + \cdots + \binom{n}{l} \cdot p^l(1-p)^{n-2}\right\} \sim \mathcal{O}(np^2). \tag{2.41}$$

Note that, for a fixed d, p_{error} is a function of (n, p), i.e., $p_{\text{error}}(n, p)$. The value of p such that $p_{\text{error}} = p$ is called the *pseudo-threshold* of an $[[n, k, d]]$ code. Thus,

$$p_{\text{th}} = p_{\text{error}}(n, p_{\text{th}}). \tag{2.42}$$

For $[[n, 1, 3]]$ codes, we evaluated $p_{\text{th}} = 0.132, 0.058, 0.043, 0.033$ for $n = 5, 7, 8, 9$ respectively. These values correspond to total error rates in simulation.

We shall now present a simulation-based approach to calculate error rates and thresholds. Once the circuit for QEC has been created, we chose to use a standard depolarizing noise model to add noise to the circuit. To match the above analytical approach, we shall only introduce a layer of noise after the Encoder as shown in Figure 2.2. The simulation procedure is as follows:

1. For a given value of p, run the simulation circuit 10^5 times (preferably $100/p$ times) and note down the number of logical errors, total errors, and no errors.

2. Define logical error rate = no. of logical errors/run of times the circuit was run, and similarly obtain total error rate and fidelity.

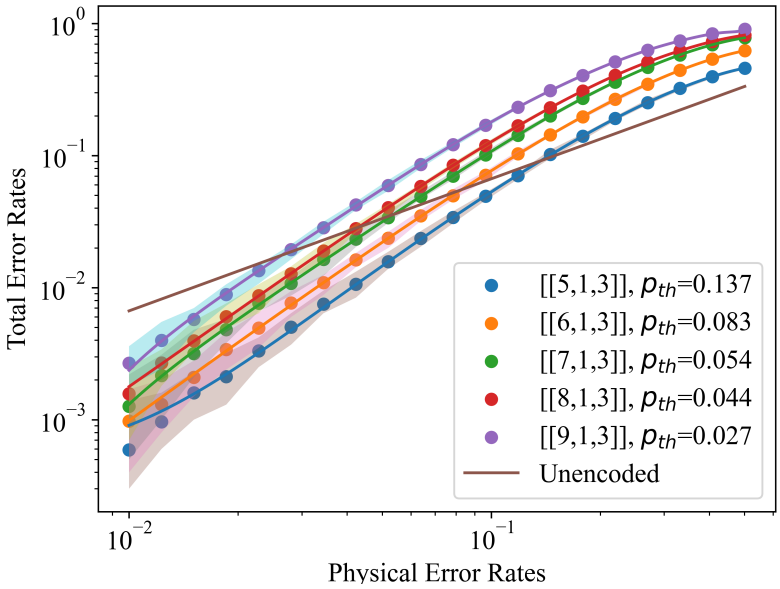


Figure 2.8: Total error rates vs. physical error rates for $[[n, 1, 3]]$ codes. The value at which the error rate of code crosses the unencoded error rate is the pseudo-threshold value, p_{th}.

3. Obtain logical error rate, total error rate, and fidelity for several values for physical error, p.

4. For each p value, run step 1 and 2 for m rounds, we used rounds=20.

5. Fit a polynomial curve to the obtained rates and compare it to unencoded rates to obtain pseudo-thresholds.

Figure 2.8 presents the error rates for $[[n, 1, 3]]$ codes. We can see that the pseudo-threshold values are decreasing with increasing n as expected. The values are also in agreement with the analytical approach with an error of ± 0.005. A clearer picture is obtained by plotting the leading order of the error rates as shown in Figure 2.9. The higher the leading order, the lower the pseudo-threshold value. Figure 2.10 presents the fidelity of the code for different values of p.

In physical settings, errors may occur at any stage of the circuit. This is where fault-tolerance chips in, reducing errors generated during the error correction procedure. We shall now present fault-tolerant techniques and simulate them to obtain error rates while allowing noise in the Detector circuit.

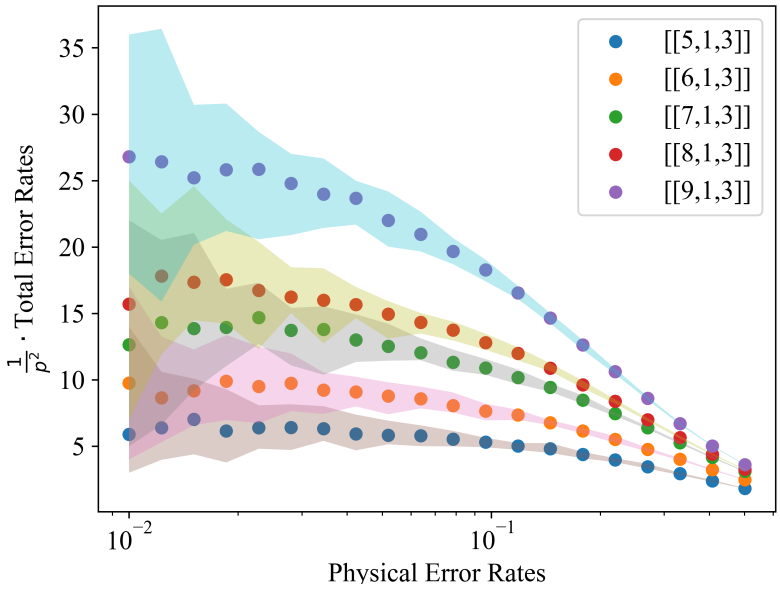


Figure 2.9: Presenting $\frac{1}{p^2}\times$ total error rates to show leading order of quadratic polynomial fit of total error rates for [[n,1,3]] codes.

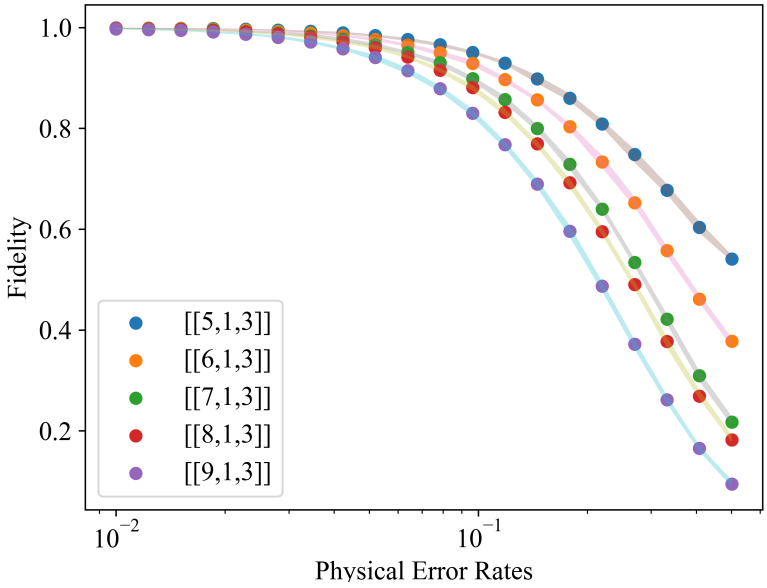


Figure 2.10: Fidelity of error corrected state against initial state for [[n,1,3]] codes.

Chapter 3

Fault Tolerance Techniques

Error correcting procedures are shown to be efficient in getting rid of errors on the encoded state, lest there occurs a logical error. Computation can be done directly on the encoded states without the need for decoding. We replace each original qubit to an encoded block of qubits, called a *logical qubit*, and each original gate to an encoded gate across encoded blocks of qubits. This way we can keep applying error correction at regular intervals to get rid of errors that are created during computation [6].

Nonetheless, errors can also prop up while applying an encoded gate or during the error-correcting procedure. These errors can also propagate to create errors of larger weight which might not be correctable by the error-correcting code used. Further, the errors generated during syndrome measurement can defile the error-less encoded state. Fault Tolerance in QECCs ensures such unexpected errors do not lead to uncorrectable errors.

Fault Tolerance is achieved by performing additional operations, often requiring more qubits to the QEC procedure, called *ancillary qubits*, which are used to detect and correct faults (errors during the procedure). We will discuss a few prominent fault tolerance techniques and will focus primarily on fault tolerance with the least overhead. We shall discuss the *Bare* method proposed by Brown *et. al* [24], which requires only a single ancilla qubit, and the *Flag* method by Chao and Reichardt [31], which requires two ancilla qubits, and present an upgraded version of *Bare* method having enhanced error correction capabilities [17].

3.1 Fault Tolerant QEC Gadgets

QEC circuits are designed using certain sub-circuits, which are often termed as *gadgets*. The first such gadget was the *DiVincezo-Shor* gadget given in Figure 3.1, which we have already used in the Detector circuit, to project a state onto eigenspace of stabilizer generators [32]. The controlled gates can propagate errors as described for **X** and **Z** errors in Figure 1.1, and thus this QEC gate is not fault-tolerant.

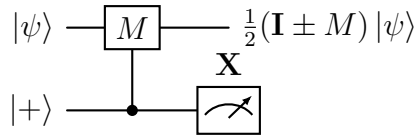


Figure 3.1: DiVincezo-Shor gadget which projects the input state $|\psi\rangle$ to either +1 or -1 eigenspace of operator M corresponding to measurement outcome of 0(+) or 1(-) respectively.

Shor proposed to add $w + 1$ ancillary qubits for fault-tolerant implementation of a DiVincenzo-Shor gadget having $\mathrm{wt}(M) = w$. A *cat* state $\frac{1}{\sqrt{2}}(|0\rangle^{\otimes w} + |1\rangle^{\otimes w})$ is prepared on the ancillary qubits and is verified for its correctness using another ancillary qubit. The controlled gates are then added from each ancillary qubit to the data qubits instead of only one ancillary qubit, utilizing the transversality of **CX** gates. Since each controlled gate begins from a different qubit, the error could not propagate to data qubits [22]. The number of physical gate operations in Shor's fault tolerance procedure is very high and thus performance is not much improved.

Steane proposed a fault-tolerant QEC gadget that uses pre-prepared logical $|\bar{0}\rangle$ states thus requiring at least n ancillary qubits for an $[[n, k, d]]$ code. It uses transversal gates to extract **X** and **Z** error syndromes independently [23]. The fault-tolerant Steane's gadget is shown in Figure 3.2. The Steane gadget requires the $|\bar{0}\rangle$ state to be prepared with high fidelity. The syndrome extraction procedure is also required to be repeated multiple times to obtain reliable syndrome values. Further, the method is suitable only for CSS codes.

A quantum teleportation-based fault-tolerant QEC gadget has been proposed by *Emmanuel Knill*, which utilizes high fidelity encoded *Bell States*, thus requiring a minimum of two codeblocks. This method does not need to obtain syndromes and teleports the error-corrected state to an ancillary codeblock [33].

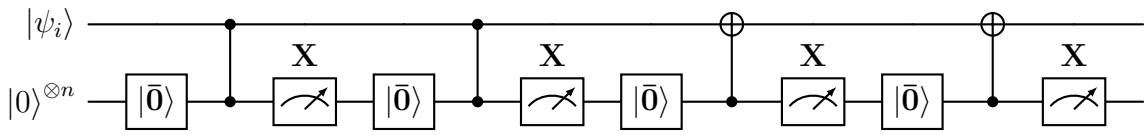


Figure 3.2: Fault tolerance scheme proposed by Steane which requires transversal operations among the n qubit data state and logical $|\bar{0}\rangle$. Each block is repeated multiple times to obtain true syndrome values through measurement.

Advances to the proposed schemes have been made continuously in the past three decades. DiVincenzo and Aliferis improved Shor's gadget by replacing the verification step with a *post-processing* step, using a carefully designed decoding circuit [34]. Stephens, Yoder and Kim further reduced the number of ancillary qubits required for the Shor's gadget. Plenio *et. al* proposed a mechanism to reduce the number of syndrome extraction rounds in Steane's gadget. Aschauer *et. al* and Dawson *et. al* provided methods for verification and entanglement purification respectively for high fidelity state preparation required for Steane's and Knill's gadgets [35, 36]. And here we are discussing the fault tolerance techniques that have the least overhead - the *Bare* method requiring single ancilla and *Flag* method requiring two ancillae.

3.2 Bare Ancillary Fault Tolerance

To reduce qubits overhead, an efficient technique would not require any more ancillary qubits. It requires that the code should be able to correct all propagated errors in addition to correctable errors. This could be achieved in two ways:

1. Propagated Errors are Correctable Errors
2. Propagated Errors have Distinct Syndromes

Bare Ancillary Fault Tolerance method [24] requires a single ancilla qubit used for syndrome extraction. The fault-tolerant nature is hidden in the carefully chosen stabilizer generators for the code. Finding such a set of stabilizers is not possible for all codes. We shall look next into both the requirements given above and how does bare method works for a non-CSS [[7,1,3]] code.

Propagated Errors are Correctable Errors

Let us consider a code that can correct any single qubit error with a stabilizer generator $g = s_1 s_2 s_3$ where $s_i \in \{\mathbf{X}_i, \mathbf{Y}_i, \mathbf{Z}_i\}$, i.e., some operator s on i^{th} qubit. The syndrome extraction circuit for the same is shown in Figure 3.3. Notice that the propagated errors are single-qubit errors up to the stabilizer g and hence are correctable errors for a code having distance, $d \geq 3$.

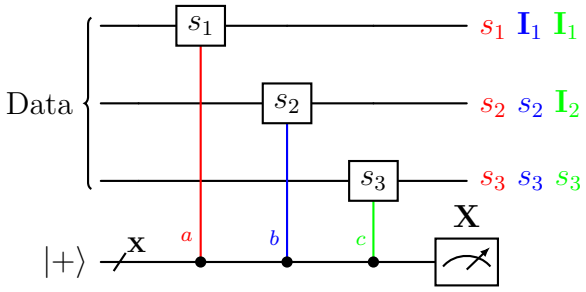


Figure 3.3: If an $\mathbf{X}$ error occurs on the syndrome qubit and gets propagated through gates a (red), b (blue), and c (green), the propagated errors are as shown in red, blue, and green columns respectively. A $\mathbf{Z}$ error would not propagate to data from the control.

Similarly, for a two-qubit error correcting code, errors propagated through stabilizer generators having weight ≤ 5 would be correctable errors. We can generalize it to say that if $\text{wt}(g) \leq \min(d)$ for a code of distance d, then all propagated errors are correctable. A few such low-distance surface codes are given in [37].

Propagated Errors have Distinct Syndromes

When the weight of a stabilizer generator is higher than the distance of the code, we need to ensure that the propagated errors have distinct syndromes than those of correctable errors. Stabilizer generators for bare [[7,1,3]] code achieve fault tolerance without requiring any ancillary qubit other than the one for syndrome extraction. Its stabilizers and logical operators are given in Table 3.1 [24].

Notice that four of the stabilizers are of weight 2 and thus propagate errors on syndrome only to correctable errors on data. For stabilizer $\mathbf{Z}_0\mathbf{Z}_1\mathbf{Z}_2\mathbf{X}_3\mathbf{Z}_4\mathbf{Z}_5$, and $\mathbf{Z}_2\mathbf{Z}_3\mathbf{Y}_5\mathbf{Y}_6$, the propagated errors are shown in row 3 of Table 3.1. Comparing the syndromes of correctable

Stabilizers	$\mathbf{Z}_0\mathbf{Z}_1\mathbf{Z}_2\mathbf{X}_3\mathbf{Z}_4\mathbf{Z}_5$	$\mathbf{Z}_2\mathbf{Z}_3\mathbf{Y}_5\mathbf{Y}_6$	$\mathbf{X}_0\mathbf{X}_4$	$\mathbf{X}_1\mathbf{X}_4$	$\mathbf{X}_2\mathbf{X}_5$	$\mathbf{X}_3\mathbf{X}_6$
Normalizers	$\mathbf{X}_1\mathbf{X}_2\mathbf{X}_3$	$\mathbf{Z}_0\mathbf{Z}_1\mathbf{Z}_4$				
Errors	$\mathbf{Z}_0\mathbf{Z}_1\mathbf{Z}_2$	$\mathbf{Z}_0\mathbf{Z}_1$	$\mathbf{Z}_4\mathbf{Z}_5$	$\mathbf{Z}_2\mathbf{Z}_3$		

Table 3.1: Stabilizer generators, normalizer generators, and data errors due to propagation of $\mathbf{X}$ error on syndrome ancilla for the bare ancillary fault tolerant non-CSS [[7,1,3]] code.

errors with the uncorrectable propagated errors, we shall find that the syndrome of $\mathbf{Z}_0\mathbf{Z}_1$ error is the same as that of $\mathbf{Z}_4$ error. Thus, our code might identify the propagated error $\mathbf{Z}_0\mathbf{Z}_1$ as $\mathbf{Z}_4$ and after the correction step, the resultant error will become $\mathbf{Z}_0\mathbf{Z}_1\mathbf{Z}_4$, a logical error. We can get rid of this logical error by applying a technique of reordering controlled gates in the circuit.

3.2.1 Reordering

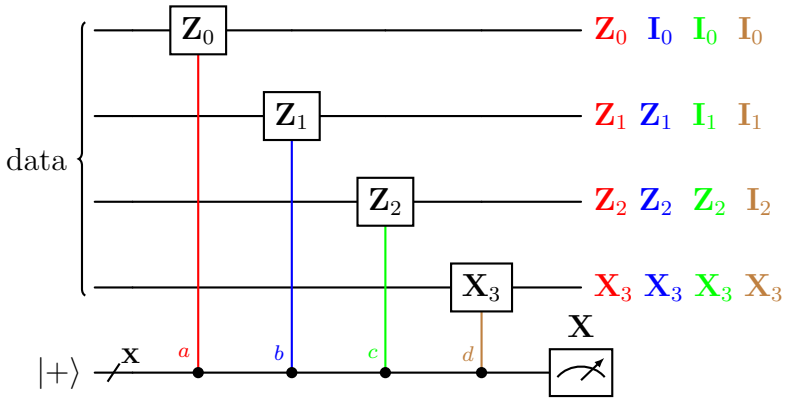
In the above code, the stabilizer $\mathbf{Z}_0\mathbf{Z}_1\mathbf{Z}_2\mathbf{X}_3\mathbf{Z}_4\mathbf{Z}_5$ could be reordered as $\mathbf{Z}_0\mathbf{Z}_2\mathbf{X}_3\mathbf{Z}_1\mathbf{Z}_4\mathbf{Z}_5$ as shown in Figure 3.4b. The propagated errors in the reordered circuit are shown in Table 3.2.

Errors before Reordering	$\mathbf{Z}_0\mathbf{Z}_1\mathbf{Z}_2$	$\mathbf{Z}_0\mathbf{Z}_1$	$\mathbf{Z}_4\mathbf{Z}_5$	$\mathbf{Z}_2\mathbf{Z}_3$
Errors after Reordering	$\mathbf{Z}_0\mathbf{Z}_2\mathbf{X}_3$	$\mathbf{Z}_0\mathbf{Z}_2$	$\mathbf{Z}_4\mathbf{Z}_5$	$\mathbf{Z}_2\mathbf{Z}_3$

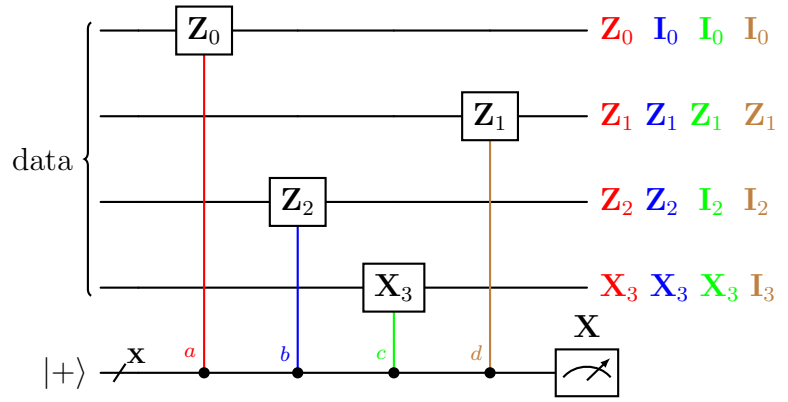
Table 3.2: Propagated Errors after sequential and reordered measurement of gates in Detector circuit for the stabilizer $\mathbf{Z}_0\mathbf{Z}_1\mathbf{Z}_2\mathbf{X}_3\mathbf{Z}_4\mathbf{Z}_5$.

The propagated errors $\mathbf{Z}_0\mathbf{Z}_1$ have been replaced by $\mathbf{Z}_0\mathbf{Z}_2$ and the new errors now have distinct syndromes from all the correctable errors. Whenever an error occurs on the syndrome qubit, it would now be correctable by the code itself. Any possible error on the data qubit can only propagate to become a $\mathbf{Z}$ error on the syndrome qubit and does not propagate to any other data qubit. Thus one fault can lead to a maximum of one error on the data qubit. We note that such is not possible with any perfect code such as the [[5,1,3]] code as all possible syndromes correspond already to correctable errors and no possible syndrome value is left to account for propagated errors.

To ensure that we do not end up applying the wrong Corrector operator, syndrome measurement should take place twice. If the measurement outcomes are the same, then apply the corresponding error correction operator, else repeat the syndrome measurement until we get the same syndrome consecutively. It guarantees that an error has not occurred



(a) Sequential application of gates for syndrome extraction



(b) Reordered application of gates for syndrome extraction

Figure 3.4: Measuring syndrome for $\mathbf{Z}_0\mathbf{Z}_1\mathbf{Z}_2\mathbf{X}_3\mathbf{Z}_4\mathbf{Z}_5$ stabilizer generator. Only the first four gates are shown as others will be sequentially applied as in (a). Each column following the circuit shows propagated errors if an $\mathbf{X}$ fault occurs before the gate of the corresponding color.

on the syndrome and that the data error has been correctly captured by the syndrome qubits. This procedure ensures any single qubit error and any error propagated from the ancillary qubit to data qubits would be corrected, giving us the fault-tolerant [[7,1,3]] code using a single ancillary qubit.

3.2.2 Modified Bare Ancillary Fault Tolerance

A two-qubit error might also occur following a two-qubit gate, which can not be handled by the bare mechanism and would lead to a logical error. We proposed a modified approach to

bare ancillary fault tolerance, we not only consider errors on the syndrome qubit but also the data qubit simultaneously [17].

We shall consider errors of type $\mathbf{P}\otimes\mathbf{X}$ and $\mathbf{P}\otimes\mathbf{Z}$ after any two-qubit gate instead of only $\mathbf{X}$ or $\mathbf{Z}$ error on the ancilla. While obtaining syndrome value corresponding to a stabilizer, say, $M = \mathbf{X}_1\mathbf{Y}_2\mathbf{Z}_3$, a $\mathbf{P}\otimes\mathbf{X}$ error after the three gates propagates to the following errors: $\mathbf{P}_1\mathbf{Y}_2\mathbf{Z}_3$, $\mathbf{P}_2\mathbf{Z}_3$, and $\mathbf{P}_3$ respectively as shown in Figure 3.5. We shall now choose a reordering of gates such that all possible propagated errors of the above form with $\mathbf{P}_i \in \{\mathbf{I}_i, \mathbf{X}_i, \mathbf{Y}_i, \mathbf{Z}_i\}$ have distinct syndromes.

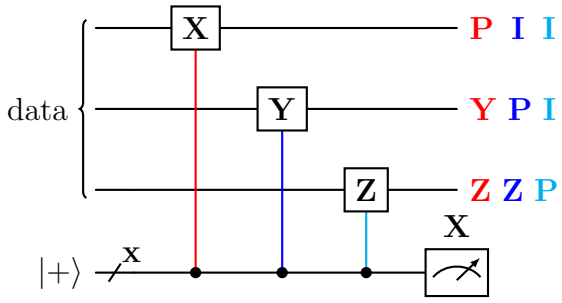


Figure 3.5: Propagation of error on data qubits when two-qubit gates are followed by a $\mathbf{P}\otimes\mathbf{X}$ error, where $\mathbf{P} \in \{\mathbf{X}, \mathbf{Y}, \mathbf{Z}\}$. Errors of the kind $\mathbf{P}\otimes\mathbf{Z}$ lead only to $\mathbf{P}$ error on data qubits. Each color denotes the gate that gets faulty and the corresponding error on data qubits.

We shall use the modified bare ancillary method to achieve fault tolerance in the $[[8, 1, 3]]$ code which we discovered using cluster state measurements. The method is limited in its application as all codes might not have stabilizers that can be reordered to have distinct syndromes, for example, the bare $[[7, 1, 3]]$ code described does not suit the modified method. Further, there exists a class of non-CSS $[[7, 1, 3]]$ color codes [15] which can not be made fault tolerant using the bare method. We shall look into a different approach to achieve fault tolerance in such codes using flag qubits.

3.3 Flag Qubits for Fault Tolerance

To catch faults in the syndrome ancilla that lead to errors of weight two or more, we can add extra ancillary qubits, hereby called, *flags*. These flags are capable of localizing where the error has occurred. A technique to achieve fault tolerance using only two ancillary qubits- the

syndrome ancilla and the flag has been presented in [31]. Consider the Detector sub-circuit for the stabilizer **XZZXI** of the [[5,1,3]] code shown in Figure 3.6.

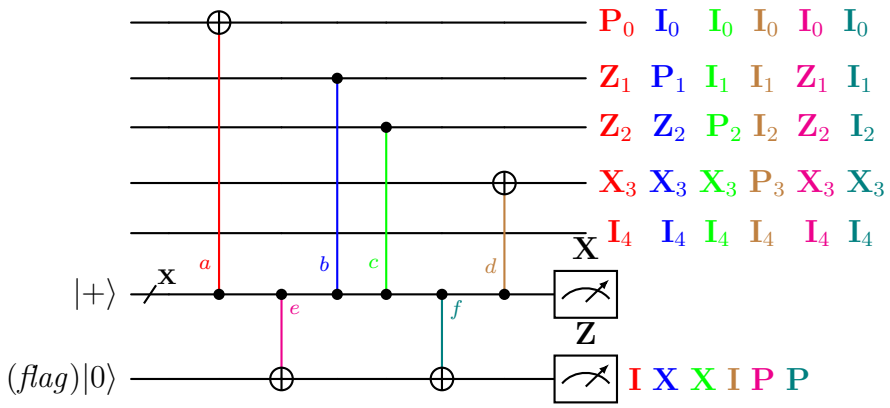


Figure 3.6: Flagged syndrome extraction for [[5,1,3]] code. If a $\mathbf{P} \otimes \mathbf{X}$ error occurs after any one of the colored gates, the propagated errors on data qubits are given by the respective colored columns.

The flag qubit is activated (measures out 1) when the error occurs on the gate b or c leading to $\mathbf{I}_0 \otimes \mathbf{P}_1$ or $\mathbf{P}_2 \otimes \mathbf{X}_3$ error respectively, where $\mathbf{P} \in \{\mathbf{I}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}\}$. Other errors are one qubit correctable errors and the two qubit errors are captured by the flag qubit. There are still possibilities that the syndrome qubit gets a $\mathbf{Z}$ error or flag gets a $\mathbf{P}$ error measuring to 1 instead of 0. These can be taken care of if we run an error correction procedure as soon as either flag or syndrome measures 1, it guarantees that data error will be corrected, if any, or does not alter.

To correct the two or more qubits errors captured by the flags, we need to ensure that none of these errors have the same syndrome. If so happens, we shall use the reordering trick as outlined in Section 3.2.1 such that all higher-weight propagated errors have distinct syndromes and can be corrected. This requires a new lookup table for flag-captured errors. Notice that syndromes of these errors can be the same as correctable errors as once the flag is activated, we are sure that the propagated error is not one of the correctable errors and hence the correction corresponds to captured errors (higher weight errors) only. The error correction protocol, replacing the Detector and Corrector procedures in the standard QEC scheme, is outlined in Algorithm 1. The flag qubit scheme ensures that a single error during the error correction procedure leads to at max a single error in the data qubits.

In the bare [[7,1,3]] code, the flag method can be applied to measure syndrome values.

Algorithm 1 Fault Tolerant Error Correction using Flag Qubits

```
function ERROR CORRECTION(lookup)
    Measure Syndrome without flag qubits
    Match syndrome values with lookup
    Apply corresponding error correcting operator
end function
function FLAG FAULT TOLERANCE
    for s ∈ stabilizers do
        Measure syndrome and flag qubit for s
        if flag ← 1 then
            ERROR CORRECTION(new lookup)          ▷ Two Qubit Errors Lookup
        else if syndrome ← 1 then
            ERROR CORRECTION(old lookup)          ▷ Single Qubit Errors Lookup
        end if
    end for
end function
```

For the stabilizer generators of weight 2, we need not require any flag and for the weight 4 generator, the flag procedure will be similar to that of the [[5,1,3]] code. The stabilizer having weight six will require a reordering of gates to ensure that the propagated errors have distinct syndromes. The sub-circuit for the stabilizer $\mathbf{Z}_0\mathbf{Z}_1\mathbf{Z}_2\mathbf{X}_3\mathbf{Z}_4\mathbf{Z}_5$ is shown in Figure 3.7.

The flag method for fault tolerance, presented here, is non-adaptive and may not work for every code. A more general approach is given in [25] which requires $2t + 1$ flag qubits to correct up to t faults in the error correction procedure. This method is adaptive and can be used for any stabilizer code.

The flag method has been studied extensively and has shown its uses in quantum computation [38]. An updated flag method has been proposed in [39] which presents a flag method scheme for any code of arbitrary distance. The method has also been utilized to achieve lower error rates in CSS color codes [40]. Its implementation on ion-trap computers has been performed to compare against Steane's Fault Tolerance method [41].

Fault-tolerant error correction is the stepping stone to fault-tolerant quantum computation in which encoded gates are used to perform operations and multiple rounds of QEC are applied at regular intervals. With the least overhead, it is possible to perform QEC on NISQ devices and laboratory-scale models of quantum computers. Bare Ancillary Fault

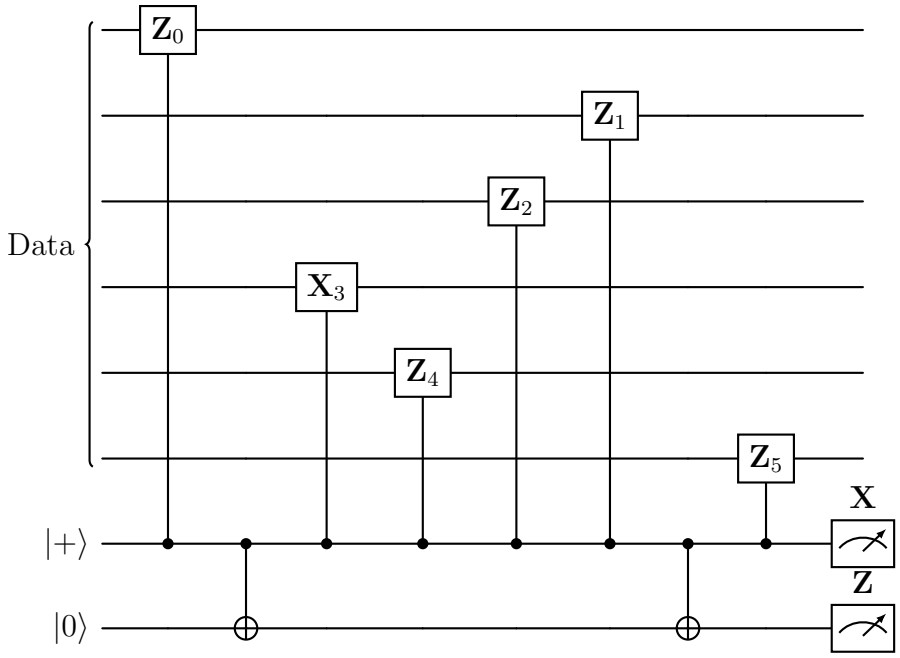


Figure 3.7: Measurement of $\mathbf{Z}_0\mathbf{Z}_1\mathbf{Z}_2\mathbf{X}_3\mathbf{Z}_4\mathbf{Z}_5$ stabilizer generator of bare ancillary [[7,1,3]] code as $\mathbf{Z}_0\mathbf{X}_3\mathbf{Z}_4\mathbf{Z}_2\mathbf{Z}_1\mathbf{Z}_5$ after reordering for flag fault tolerance.

Tolerance technique in certain QECCs promises scalability of error-corrected codeblocks and thus noise-free quantum computation in the near future.

In the next chapter, we will deal with this approach, analyze its performance on noise models inspired by physical systems, and discuss the merits of fault tolerance using a single ancillary qubit.

Chapter 4

Fault Tolerant Simulations

In this chapter, we put to test the Bare Ancillary Fault Tolerance technique described in Section 3.2. Following Brown *et. al* [24], we simulated the [[7,1,3]] non-CSS code and obtained its error rates and pseudo-thresholds. We then simulate a non-CSS [[8,1,3]] code, which we discovered through analysis based on *measurement based model* of quantum computation [42], and made it fault-tolerant using single ancillary qubit. The [[8,1,3]] code suits well to fault-tolerant methods presented in [24] as well as the modified method described in Section 3.2.2.

The code performance is compared in two noise models - *Standard Depolarizing Noise Model* and *Anisotropic Noise Model* and two simulation schemes as described below in Section 4.1.

4.1 Simulation Scheme

We designed the complete QEC simulator using qiskit [43] following the scheme described in Fig. 2.2 with modifications required for the bare ancillary fault-tolerant method. The following steps outline the sub-circuits used to create the complete Bare Ancillary Fault-Tolerant QEC circuit,

1. Consider $|\psi_i\rangle = |0\rangle$ and initialize state $|0\rangle^{\otimes n}$, we apply the Encoder circuit which provides the logical $|\bar{0}\rangle$ state.

2. The Detector circuit is then applied twice. The classical outputs of the two Detector circuits are compared, if they are the same we proceed with the Corrector, else we apply the Detector circuit again and feed its output to the Corrector. We limit our simulation to a maximum of three Detector applications.

3. The Corrector sub-circuit applies the exact error operator given the syndrome value. It includes both the errors correctable by the code (single-qubit errors for $d = 3$) and the propagated errors.

4. (Encoder)† is applied on the error-corrected state, and the data qubits are measured.

We then generate noise models as described below and act on the gates used in the Detector sub-circuit. For our purpose, the Encoder, the Corrector, and (Encoder)† are kept noise-free [17].

4.1.1 Circuit-Level Noise Models

Standard Depolarizing Noise Model

It is the most applicable noise model to test physical systems, in which each gate is followed by a symmetric depolarization with some probability.

1. With probability p_s, each single qubit gate is followed by a single qubit Pauli error drawn uniformly and independently from $\{\mathbf{X}, \mathbf{Y}, \mathbf{Z}\}$.

2. With probability p_t, each two-qubit gate is followed by a two-qubit Pauli error drawn uniformly and independently from $\{\mathbf{I}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}\} \otimes \{\mathbf{I}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}\} \setminus \{\mathbf{I} \otimes \mathbf{I}\}$.

3. With probability p_{meas}, each single qubit measurement has its outcome flipped.

4. With probability p_{prep}, each preparation of $|0\rangle$ is flipped to $|1\rangle$ and each preparation of $|+\rangle$ is flipped to $|-\rangle$.

Anisotropic Noise Model

It is a suitable noise model that mimics the over or under-rotation of gates in ion-trap qubits. It considers noise due to gate coupling and thus two qubit errors are only the ones aligned with gates [24].

1. Single qubit error, measurement error, and preparation error are applied with probability p_s, p_{meas}, and p_{prep} respectively as defined for depolarizing noise model.
2. With probability p_t, every two qubit controlled-$\mathbf{P}$ gate is first followed by $\mathbf{Z} \otimes \mathbf{P}$ error where $\mathbf{P} \in \{\mathbf{X}, \mathbf{Y}, \mathbf{Z}\}$. Each of the two qubits involved is then acted upon by a single qubit error with probability p_s.

For our purpose, we have taken all probabilities to be the same, i.e., $p_s = p_t = p_{meas} = p_{prep}$, in both the noise model.

4.1.2 Error Rates and Thresholds

We analyze two metrics for the code on each noise model.

1. Logical error rate: Probability of the error corrected state to remain in the codespace but different from the encoded state.
2. Total error rate: Probability of the corrected state being different from the encoded state.

On measuring data qubits, $|\psi_d\rangle = |0\rangle^{\otimes(n-1)} |1\rangle$ contributes to logical error, while $|\psi_d\rangle \neq |0\rangle^{\otimes n}$ contributes to total error. For various values of physical error rate, p, we run the simulation circuit N times and note down the logical and total error rate given by (number of errors / N). We repeat the simulation cycle for m rounds to obtain a distribution of error rate values at each p. The mean of m rounds along with minimum and maximum values is plotted against the physical error rate.

Once we have obtained the metrics for a range of values of p, we fit a polynomial curve to judge the pseudo-thresholds. The error correcting scheme is expected to remove any linear

dependence of logical and total error rates on p. It is encouraged to fit a curve of the form

$$f(p) = a_0 p^2 + a_1 p^3 + \cdots + a_{n-2} p^n, \tag{4.1}$$

where a_0 is the leading order coefficient. Since we initialize the state in logical $|\bar{0}\rangle$ state, the effect of logical $\bar{\mathbf{Z}}$ operator will not be noticed and only logical $\bar{\mathbf{X}}$ and logical $\bar{\mathbf{Y}}$ will cause an observable logical error. Thus we compare the obtained error rates to unencoded error rates of $\frac{2}{3}p$ instead of p.

The pseudo-threshold is the value at which the error rate after error correction crosses the unencoded error rate ($\frac{2}{3}p$). It denotes the value of p below which the code would be suitable for quantum computation. Comparing two codes, a code with a higher pseudo-threshold will be preferable. The leading coefficient, a_0, is another metric to compare codes. The lower the coefficient, the better performing the code [17].

4.1.3 Modified Simulation

The above method is more realistic for physical devices. Nonetheless, it does not account for all correctable errors. QEC simulations often involve a noise-free round of projecting the error-corrected state to the codespace and comparing it to the encoded state. This involves additional noise-free error Detector and Corrector circuit before $(\text{Encoder})^\dagger$ operation. The modified approach leads to a state containing only those errors not correctable by the code. This leads to an increase in the logical error rate and a decrease in the total error rate [17].

Here onwards, We call the simulation run without the noise-free projection step described above as the *Practical Method*, and the simulation run with the noise-free detection-correction step as the *Modified Method*.

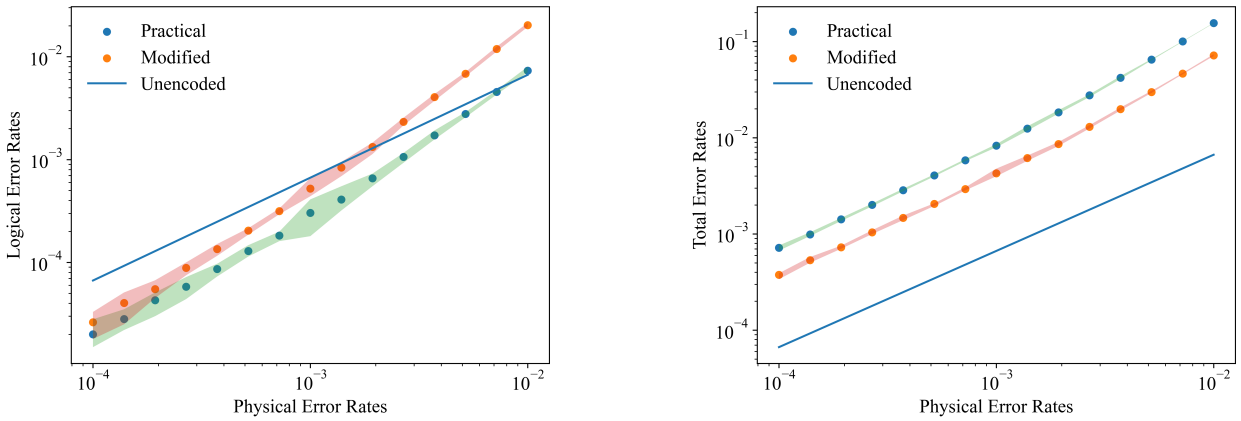
4.2 Bare Ancillary [[7,1,3]] Code

We followed the original bare ancillary fault tolerance approach and applied it to the code described in Table 3.1. The reordering scheme presented in Section 3.2.1 and demonstrated in Figure 3.4 was used and the circuit thus generated was used to simulate fault tolerant

QEC procedure. Further details are provided in [24]. We obtained logical error rates and total error rates for practical and modified simulation methods.

Figure 4.1 presents the code performance on the standard depolarizing noise model. It seems as if the logical error rate drops below the unencoded error rate and thus indicates a high pseudo-threshold value. The total error rate, however, is strictly higher and the curvature shows the error rate to be increasing with lower values of p. Fitting Equation 4.1 onto obtained results does not give an expected curve thus the best fit does not follow a quadratic form as expected.

Figure 4.3a also validates that the logical error rate which initially is below the unencoded error rate is moving upward and does not saturate. We can conclude that the method does not suit for standard depolarizing noise model and thus fails in providing a pseudo-threshold.



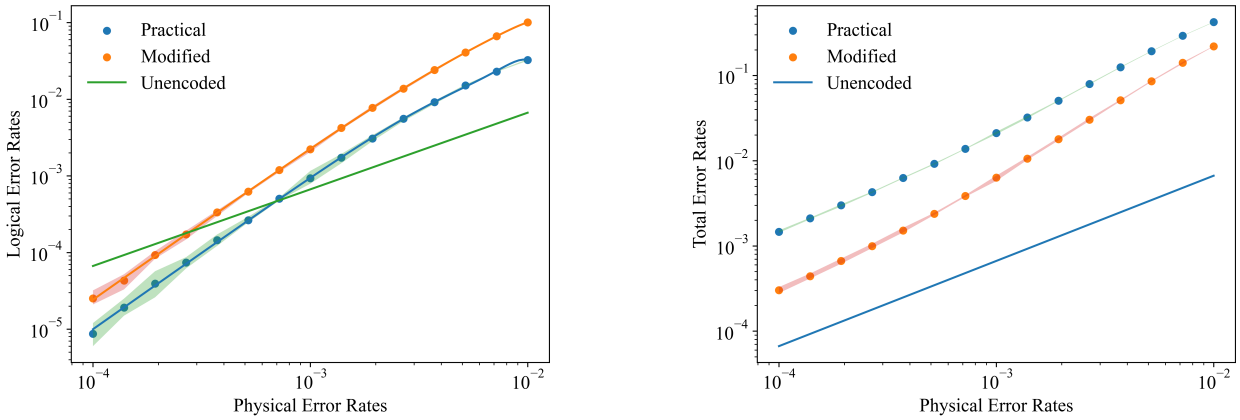
(a) Logical vs Physical Error Rate

(b) Total vs Physical Error Rate

Figure 4.1: Error rates of bare ancillary [[7,1,3]] code under standard depolarizing noise model. Logical errors seem to cross the unencoded error rate while total errors are always higher than unencoded. The best-fit curve does not trace the obtained results and thus is not shown.

Simulating the circuit on the anisotropic noise model results in a descent as shown in Figure 4.2. It also provides pseudothreshold values of 6.91×10^{-4} and 2.79×10^{-4} for practical and modified simulation schemes respectively. Figure 4.3a shows a constant leading order of 1001 and 2444 for the anisotropic noise model in practical and modified methods respectively. We can thus conclude that the bare ancillary [[7,1,3]] code is fault tolerant

under the anisotropic noise model.



(a) Logical vs Physical Error Rate

(b) Total vs Physical Error Rate

Figure 4.2: Error rates of bare ancillary [[7,1,3]] code under anisotropic noise model. There is a descent in both (a) and (b). The best-fit curve is shown for (a) giving us the required pseudo-threshold.

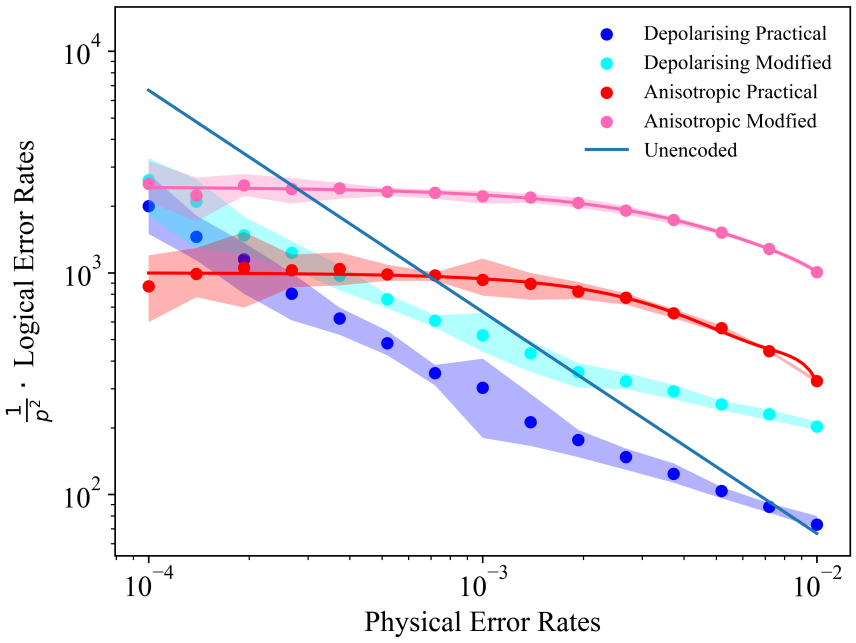
Leading Order of error rates are presented in Figure 4.3. The total error rates of the fault-tolerant circuit are always higher than the unencoded error rates. We also notice that the modified scheme always has higher values for logical error rates and lower values for total error rates than the practical simulation method.

Our simulation results are similar to that obtained in [24] showing a failure for depolarizing errors owing to uncorrectable hook errors and success in reducing error rates and obtaining a pseudo-threshold for anisotropic noise. The next section now presents the application of the modified bare ancillary method on a non-CSS [[8,1,3]] code.

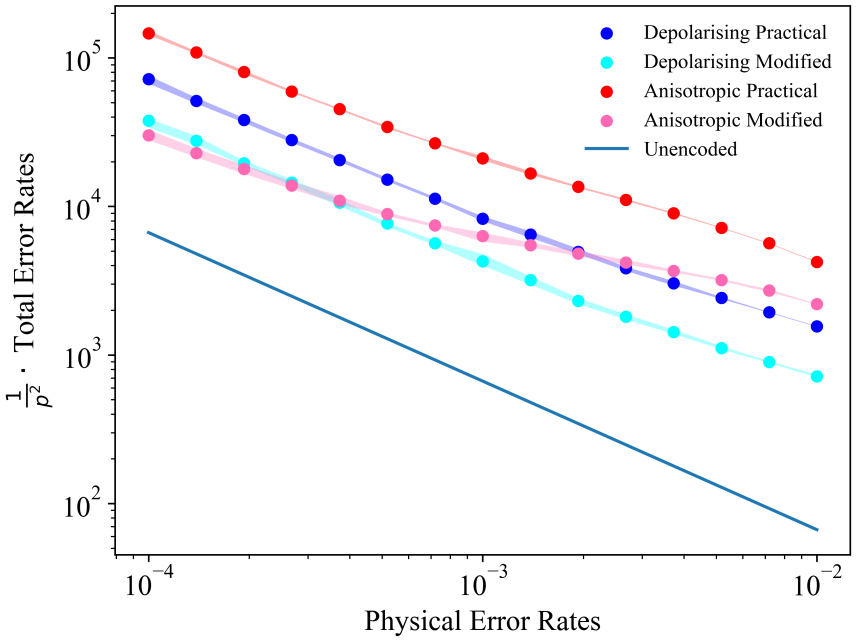
4.3 Bare Ancillary [[8,1,3]] Code

We use the non-CSS code construction method outlined in [42] to obtain stabilizer generators for a $[[8, 1, 3]]$ code. The code is modified to reduce the weight of the stabilizer generators such that it shows fault-tolerant properties requiring a single ancillary qubit. Table 4.1 presents the stabilizer generators and the logical $\bar{\mathbf{X}}$ and $\bar{\mathbf{Z}}$ operators of the code [17].

We analyzed the single ancilla fault-tolerant properties of the code and found that certain



(a) Leading Orders of Logical Rates



(b) Leading Orders of Total Rates

Figure 4.3: Leading order of error rates with both simulation methods and noise models for bare ancillary [[7,1,3]] code. The distribution shows the minimum to maximum range of error rates around the plotted mean value. Logical error rates plotted in (a) indicate success in obtaining a pseudo-threshold for the anisotropic model while total error rates in (b) validate that the code performs better in anisotropic noise.

Stabilizer Generators	Logical Operators
$\mathbf{Z_0X_1Z_2Z_4}$	
$\mathbf{Y_0Y_2Z_3Z_4}$	
$\mathbf{Z_0Z_1X_4Z_6}$	$\bar{\mathbf{X}} = \mathbf{Z_0Z_1X_2Z_5}$
$\mathbf{X_1Z_3X_5X_6}$	$\bar{\mathbf{Z}} = \mathbf{Z_0Z_2Z_5Z_6}$
$\mathbf{Z_3Z_5Z_6X_7}$	
$\mathbf{Z_0X_3Z_6Z_7}$	
$\mathbf{Z_1X_2X_3Z_4X_6X_7}$	

Table 4.1: Stabilizer generators and logical operators for the bare ancillary fault tolerant [[8,1,3]] code.

propagated errors have similar syndrome values. For example, while measuring the syndrome for $\mathbf{Z_0X_3Z_6Z_7}$ stabilizer, an $\mathbf{X}$ error on the ancilla after $\mathbf{X}_3$ gate would propagate to $\mathbf{Z_6Z_7}$ error on the data qubits. Its syndrome is the same as $\mathbf{Y}_5$ error and after applying correction, we will end up with $\mathbf{Y_5Z_6Z_7}$ error which is the logical $\bar{\mathbf{Y}}$ error.

We propose to reorder certain stabilizers of [[8,1,3]] code in the following manner:

$$\mathbf{Z_3Z_5Z_6X_7} \longrightarrow \mathbf{Z_5Z_3Z_6X_7},$$
$$\mathbf{Z_0X_3Z_6Z_7} \longrightarrow \mathbf{Z_7X_3Z_0Z_6},$$
$$\mathbf{Z_1X_2X_3Z_4X_6X_7} \longrightarrow \mathbf{Z_1X_3Z_4X_2X_6X_7}.$$

Such a reordering scheme for syndrome measurement leads to distinct syndromes for all possible propagated errors shown in column 2 (Errors) of Table 4.2. The exact syndrome values corresponding to correctable errors and the errors propagated from faults during detection using a reordered scheme are given in Section 4.3.1. We notice that all the errors have distinct syndrome values up to a stabilizer. Thus, we can achieve *fault-tolerance* in the $[[8, 1, 3]]$ code without requiring extra ancilla qubits.

The [[8,1,3]] code presented is capable of correcting any single qubit error. The above reordering scheme makes it fault-tolerant without requiring extra ancilla qubits other than the one used while syndrome extraction. We expect a single fault in the procedure should not lead to a logical error. The code achieves the same as follows:

- All $\mathbf{P} \otimes \mathbf{X}$ errors are distinguishable by syndromes, up to a stabilizer and a phase, and thus our reordering scheme tackles all hook errors, considering the lookup table

Stabilizers	Errors	
$\mathbf{Z_0X_1Z_2Z_4}$	$\mathbf{Z_0P_1}$	$\mathbf{P_2Z_4}$
$\mathbf{Y_0Y_2Z_3Z_4}$	$\mathbf{Y_0P_2}$	$\mathbf{P_3Z_4}$
$\mathbf{Z_0Z_1X_4Z_6}$	$\mathbf{Z_0P_1}$	$\mathbf{P_4Z_6}$
$\mathbf{X_1Z_3X_5X_6}$	$\mathbf{X_1P_3}$	$\mathbf{P_5X_6}$
$\mathbf{Z_5Z_3Z_6X_7}$	$\mathbf{Z_5P_3}$	$\mathbf{P_6X_7}$
$\mathbf{Z_7X_3Z_0Z_6}$	$\mathbf{Z_7P_3}$	$\mathbf{P_0Z_6}$
$\mathbf{Z_1X_3Z_4X_2X_6X_7}$	$\mathbf{Z_1P_3}$ $\mathbf{P_2X_6X_7}$	$\mathbf{Z_1X_3P_4}$ $\mathbf{P_6X_7}$

Table 4.2: Errors on data qubits when $\mathbf{P} \otimes \mathbf{X}$ error occurs after a noisy gate in Detector circuit. Here, $\mathbf{P} \in \{\mathbf{X}, \mathbf{Y}, \mathbf{Z}\}$

decoder.

- Any $\mathbf{P} \otimes \mathbf{Z}$ kind of error leads to a single qubit error in data while also flipping the syndrome value. We perform a minimum of two rounds of syndrome extraction and compare the measurement results. The same results ensure no $\mathbf{Z}$ error on ancilla while different outcomes require another round of syndrome extraction. When two consecutive syndrome extraction rounds give the same outcomes, we correct the errors accordingly.

- Errors of kind $\mathbf{P} \otimes \mathbf{Y}$ are equal to $(\mathbf{P} \otimes \mathbf{X})(\mathbf{I} \otimes \mathbf{Z})$. Repeated application of the Detector circuit gets rid of $\mathbf{Z}$ on ancilla and the syndrome outcome tells the propagated error to be corrected.

4.3.1 Errors and Syndrome Values

All single-qubit errors are correctable by the [[8,1,3]] code with each error having distinct syndromes given in table 4.3. Errors arising due to faults in detection mainly accumulate through error propagation. Table 4.4 presents syndrome values corresponding to higher weight errors. We observe that the syndrome corresponding to $\mathbf{Z_1X_3X_4}$ is the same as $\mathbf{Z_7}$ error as $\mathbf{Z_1X_3X_4Z_7}$ is a stabilizer.

Single Qubit Errors		
$\mathbf{X}_0 : 1110010$	$\mathbf{Y}_0 : 1010010$	$\mathbf{Z}_0 : 0100000$
$\mathbf{X}_1 : 0010001$	$\mathbf{Y}_1 : 1011001$	$\mathbf{Z}_1 : 1001000$
$\mathbf{X}_2 : 1100000$	$\mathbf{Y}_2 : 1000001$	$\mathbf{Z}_2 : 0100001$
$\mathbf{X}_3 : 0101100$	$\mathbf{Y}_3 : 0101111$	$\mathbf{Z}_3 : 0000011$
$\mathbf{X}_4 : 1100001$	$\mathbf{Y}_4 : 1110001$	$\mathbf{Z}_4 : 0010000$
$\mathbf{X}_5 : 0000100$	$\mathbf{Y}_5 : 0001100$	$\mathbf{Z}_5 : 0001000$
$\mathbf{X}_6 : 0010110$	$\mathbf{Y}_6 : 0011111$	$\mathbf{Z}_6 : 0001001$
$\mathbf{X}_7 : 0000010$	$\mathbf{Y}_7 : 0000111$	$\mathbf{Z}_7 : 0000101$

Table 4.3: Syndrome values of single qubit Pauli errors in bare ancillary fault-tolerant [[8,1,3]] code.

4.3.2 Quantum Circuits

We use the methods presented in Section 2.2.2 to create an efficient unitary Encoder for the chosen stabilizers. The Encoder circuit given in Figure 4.4 utilizes the enhanced standard form mechanism for non-CSS codes without any non-essential gates. We notice the last step involves certain **SWAP** gates, which exchange the qubit values. The purpose of these gates is to map relabeled qubits to the original order of qubits.

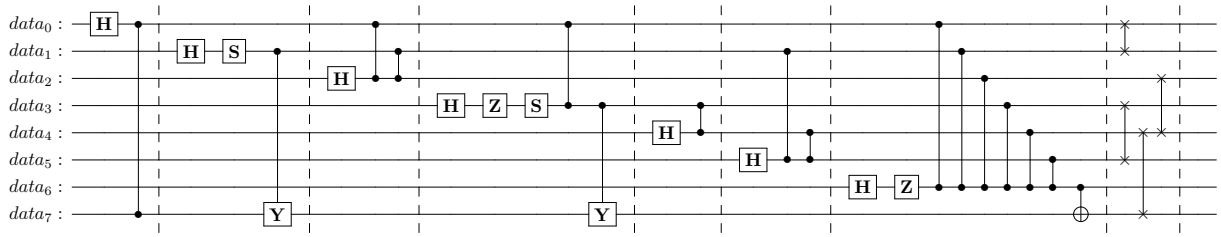


Figure 4.4: Encoder circuit for bare [[8,1,3]] code. The last $k = 1$ qubit is initialized in $|\phi\rangle$ state. No extra gates are required for the $\bar{\mathbf{X}}$ operator.

The syndrome measurement circuit, *Detector*, is built following the reordering scheme presented above. We utilize fast reset of qubits to use a single ancilla for syndrome measurements. It enhances the simulation speed exponentially as fewer space resources are required to store quantum states in classical systems. The complete circuit is shown in Figure 4.5.

The Corrector circuit is a classical-quantum hybrid circuit that applies certain fixed operations on the data qubits given the syndrome values. We note that we are not using a standard lookup table decoder that contains the most likely error configuration for every

Errors due to Faults			
$\mathbf{Z_0X_1Z_2Z_4}$			
$\mathbf{Z_0X_1}$	0110001	$\mathbf{X_2Z_4}$	1110000
$\mathbf{Z_0Y_1}$	1111001	$\mathbf{Y_2Z_4}$	1010001
$\mathbf{Z_0Z_1}$	1101000	$\mathbf{Z_2Z_4}$	0110001
$\mathbf{Y_0Y_2Z_3Z_4}$			
$\mathbf{Y_0X_2}$	0110010	$\mathbf{X_3Z_4}$	0111100
$\mathbf{Y_0Y_2}$	0010011	$\mathbf{Y_3Z_4}$	0111111
$\mathbf{Y_0Z_2}$	1110011	$\mathbf{Z_3Z_4}$	0010011
$\mathbf{Z_0Z_1X_4Z_6}$			
$\mathbf{Z_0X_1}$	0110001	$\mathbf{X_4Z_6}$	1101000
$\mathbf{Z_0Y_1}$	1111001	$\mathbf{Y_4Z_6}$	1111000
$\mathbf{Z_0Z_1}$	1101000	$\mathbf{Z_4Z_6}$	0011001
$\mathbf{X_1Z_3X_5X_6}$			
$\mathbf{X_1X_3}$	0111101	$\mathbf{X_5X_6}$	0010010
$\mathbf{X_1Y_3}$	0111110	$\mathbf{Y_5X_6}$	0011010
$\mathbf{X_1Z_3}$	0010010	$\mathbf{Z_5X_6}$	0011110
$\mathbf{Z_5Z_3Z_6X_7}$			
$\mathbf{Z_5X_3}$	0100100	$\mathbf{X_6X_7}$	0010100
$\mathbf{Z_5Y_3}$	0100111	$\mathbf{Y_6X_7}$	0011101
$\mathbf{Z_5Z_3}$	0001011	$\mathbf{Z_6X_7}$	0001011
$\mathbf{Z_7X_3Z_0Z_6}$			
$\mathbf{Z_7X_3}$	0101001	$\mathbf{X_0Z_6}$	1111011
$\mathbf{Z_7Y_3}$	0101010	$\mathbf{Y_0Z_6}$	1011011
$\mathbf{Z_7Z_3}$	0000110	$\mathbf{Z_0Z_6}$	0101001
$\mathbf{Z_1X_3Z_4X_2X_6X_7}$			
$\mathbf{Z_1X_3}$	1100100	$\mathbf{X_6X_7}$	0010100
$\mathbf{Z_1Y_3}$	1100111	$\mathbf{Y_6X_7}$	0011101
$\mathbf{Z_1Z_3}$	1001011	$\mathbf{Z_6X_7}$	0001011
$\mathbf{Z_1X_3X_4}$	0000101	$\mathbf{X_2X_6X_7}$	1110100
$\mathbf{Z_1X_3Y_4}$	0010101	$\mathbf{Y_2X_6X_7}$	1010101
$\mathbf{Z_1X_3Z_4}$	1110100	$\mathbf{Z_2X_6X_7}$	0110101

Table 4.4: Syndrome values of errors on data qubits propagated from noisy gates in Detector circuit of the bare ancillary fault-tolerant [[8,1,3]] code.

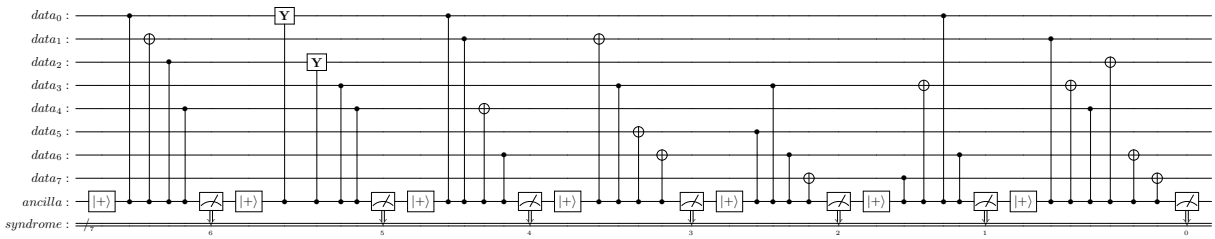


Figure 4.5: Detector circuit for syndrome measurements, incorporating reordering scheme for [[8,1,3]] code. All measurements are in **X** basis followed by resetting the qubit to $|+\rangle$ state.

possible syndrome value. Instead, our lookup table only consists of syndrome and error pairs as given in Section 4.3.1. Following the simulation scheme in Section 4.1, error rates and pseudothreshold values are calculated as described in the next Section.

4.3.3 Performance of [[8,1,3]] Code

We obtain a pseudo-threshold for the code in both noise models. We also obtain the *leading order terms* from the best-fit curve [17].

We first analyze the performance of the code in a standard depolarizing noise model. Figure 4.6 presents the descent of logical and total error rates compared among the two simulation methods. We obtain leading order terms for the practical and modified simulation runs with respective logical error threshold values of 0.31% and 0.12%. The practical method could not provide a pseudo-threshold for total error rates, thus showing a linear relation to unencoded error rates. The modified approach nonetheless provides a threshold of 0.034%.

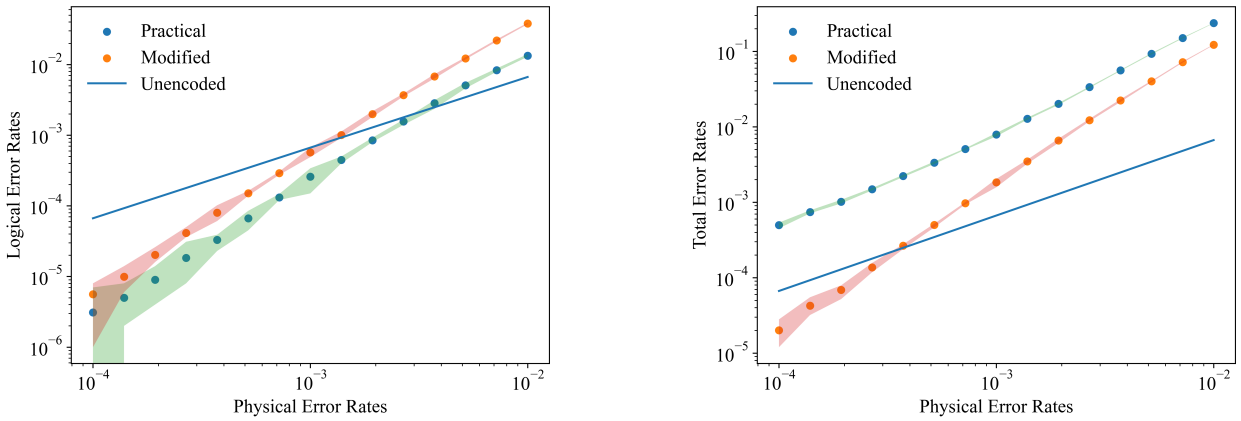
Error Rates		Pseudo-threshold	Leading Order
Logical Error Rates	Practical	0.003145	270
	Modified	0.001212	550
Total Error Rates	Practical	-	-
	Modified	0.0003426	2016

Table 4.5: Performance of bare [[8,1,3]] code under depolarizing noise model.

For the anisotropic noise model, the descent of logical and total error rates is presented in Figure 4.7. The threshold values for the practical and modified methods are found to be

Error Rates		pseudo-threshold	Leading Order
Logical Error Rates	Practical	0.0004239	1689
	Modified	0.0001796	3767
Total Error Rates	Practical	-	-
	Modified	4.95×10^{-5}	13540

Table 4.6: Performance of bare [[8,1,3]] code under anisotropic noise model.



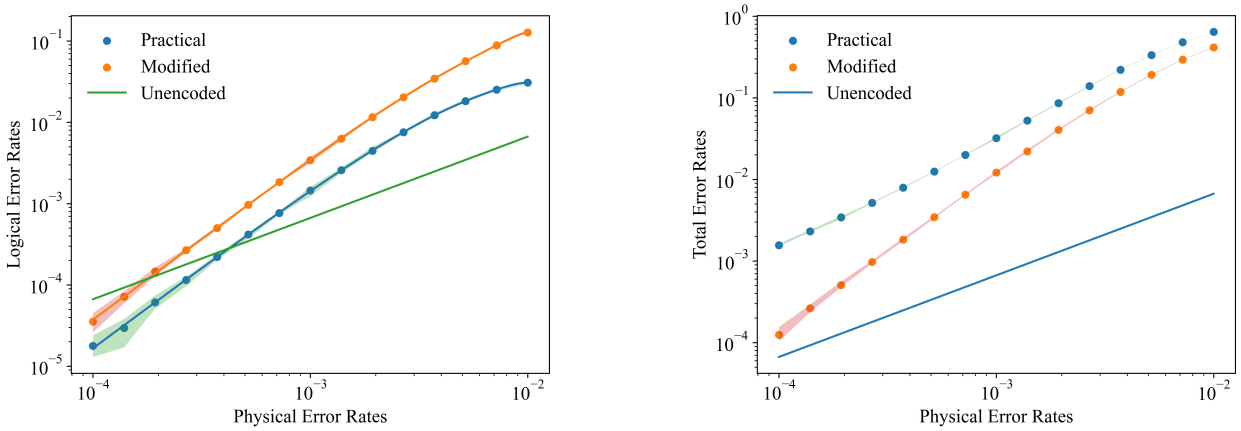
(a) Logical vs Physical Error Rate

(b) Total vs Physical Error Rate

Figure 4.6: Error rates of [[8,1,3]] code under standard depolarizing noise model. Logical error rates cross unencoded error rates for both simulation schemes while total error rates achieve a pseudo-threshold with the modified scheme only.

0.042% and 0.018% respectively for logical error rates. The practical method fails again with total error rates, but the modified method provides a threshold of 0.0049% on extrapolating the best-fit curve.

Figure 4.8 presents the error rates divided by p^2 to reveal the leading order of the best-fit curve. Logical error rates cross the unencoded error rates for both noise models and both simulation methods. Total error rates show the dependence of practical method results on physical error rates while the modified method leads to a distinctive convergence for the leading coefficient of p^2.



(a) Logical vs Physical Error Rate

(b) Total vs Physical Error Rate

Figure 4.7: Error rates of [[8,1,3]] code under anisotropic noise model. Logical error rates cross unencoded error rates for both simulation schemes while total error rates achieve a pseudo-threshold with the modified scheme on extrapolating the best fit curve.

4.4 Discussion of Results

Bare [[7,1,3]] code successfully obtains a pseudo-threshold in the anisotropic noise model while failing in the standard depolarizing noise model owing to its capability to only correct single-qubit errors propagated through the syndrome ancilla. Bare [[8,1,3]] achieves a pseudo-threshold in both noise models due to its ability to tackle errors of form $\mathbf{P} \otimes \mathbf{P}$, where $\mathbf{P}$ is any Pauli operator.

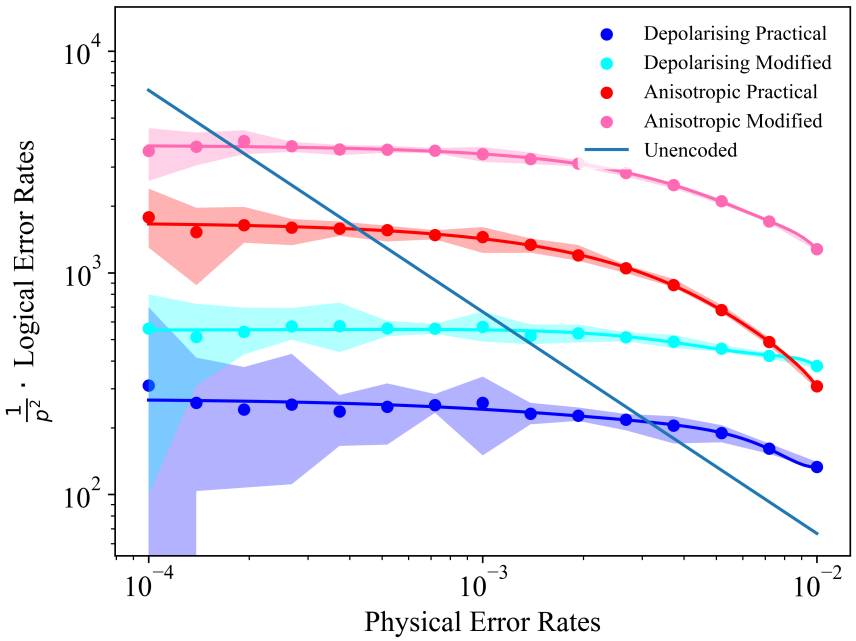
We notice that despite failing in the standard depolarizing model, the pseudo-threshold of bare [[7,1,3]] code in anisotropic model is higher than that of bare [[8,1,3]] code. It might be concerned with the weight of stabilizers in the codes. Bare [[7,1,3]] code contains stabilizer generators of weight two, four, and six, totaling 18 operators corresponding to 18 controlled gates, while bare [[8,1,3]] code contains a total of 30 controlled gates. The more the operations, the higher the possible errorful locations. Thus, it is best to obtain the least weight stabilizers for fault-tolerant error correction.

For [[8,1,3]] code, the code performance is superior in the standard depolarizing model than in the anisotropic model. This can be attributed to the error correction capabilities of the code as well as the noise in the model. The anisotropic noise has a two-qubit error followed by single-qubit errors increasing the chance of $\mathbf{Z}$ error on the syndrome which can

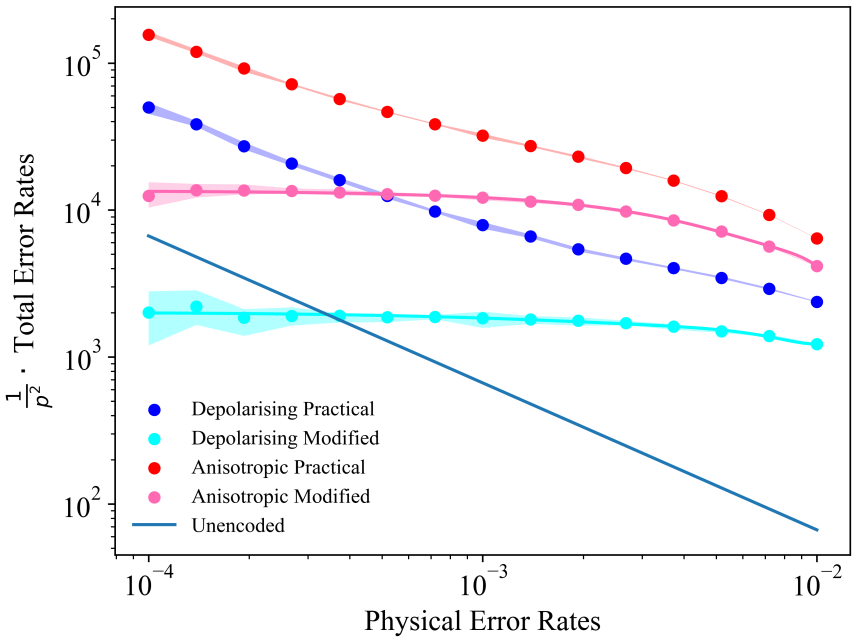
only be corrected by consecutive rounds of error detection. The constraint of three rounds allows many $\mathbf{Z}$ errors to go uncorrected. On the other hand, the code is well structured to identify any $\mathbf{P} \otimes \mathbf{X}$ error, getting rid of hook errors and increasing the pseudo-threshold.

Concerning both codes, while the practical method of simulation is more appropriate to realistic settings, the modified method is more suitable to characterize code performance and provide benchmarks. The modified method leaves the code with only uncorrectable errors, either logical errors or higher weight errors, thus it puts a bar on the efficiency of the code. It is thus advised to benchmark codes using the modified simulation method. We further observe that the leading order of p^2 is also an appropriate metric to compare the performance of a code. The lower the leading order coefficient, the higher the pseudo-threshold value [17].

Our code outperforms the non-CSS $[[7, 1, 3]]$ code by giving a pseudo-threshold in both noise models. In depolarizing model, it also outshines the better-performing flag qubit method through higher pseudo-thresholds despite the same overhead value of $9/1 = 9$ [24]. Our code has lower error rates than fault tolerant non-CSS $[[5, 1, 3]]$ code requiring seven qubits for fault tolerance and CSS $[[7, 1, 3]]$ Steane Code requiring nine qubits for fault tolerance using the flag qubit method of fault tolerance. It provides better results than the Steane's fault tolerance method for CSS $[[7, 1, 3]]$ code with an overhead of 35, as well as distance three surface code with an overhead of 17 [39]. The modified bare ancillary method along with the $[[8, 1, 3]]$ code presented is promising practical quantum computation and is suitable to be tested on NISQ devices.



(a) Leading Orders of Logical Rates



(b) Leading Orders of Total Rates

Figure 4.8: Leading order of error rates of $[[8, 1, 3]]$ code under both simulation methods and noise models. The distribution shows the minimum to maximum range of error rates around the plotted mean value. Best fit curves are shown wherever they trace a quadratic curve of error rates.

Chapter 5

Conclusions

Quantum Computation promises a future where computing capabilities transcend current limitations, unlocking unparalleled potential for solving some of humanity's most challenging problems. However, before this groundbreaking advancement becomes a reality, quantum computation must address one of its most daunting challenges: noise-free computation.

Following the expansive field of error correction in classical computation and communication, the pioneers of quantum information theory presented QECCs in the last decade of the 20^{th} century. From simple repetition codes to a concatenated nine-qubit code to an efficient seven-qubit code, Shor and Steane demonstrated that a solution to this challenge does indeed exist [44, 45].

Daniel Gottesman's breakthrough revealed a structure within QECCs, leading to the discovery of a vast family of codes known as *Quantum Stabilizer Codes*. In our study, we delve into the stabilizer formalism for error correction. Stabilizer codes harness a 2^k-dimensional subspace of a 2^n-dimensional Hilbert space of n qubits to encode all possible states of k qubits. Employing an approach rooted in the Heisenberg picture of quantum physics, we can describe an n-qubit state using n operators, each of which could comprise an n-fold tensor product of Pauli matrices.

We delve into the structure of the *Stabilizer Subgroup*, *Normalizer Subgroup*, and the set of correctable errors, charting out the coset structure among them. Additionally, we map the operators' picture to state vectors, illustrating how a state evolves upon the application

of a particular operator and how errors can be detected and corrected. We compile a review of the stabilizer formalism that is both comprehensive and easy to understand.

We explore quantum circuits for implementing QECCs, given their stabilizer generators. Building upon Gottesman's unitary encoding scheme, we introduce an enhancement that enables encoding both CSS and Non-CSS codes alike, as detailed in [17]. The resulting encoded state, referred to as the logical state, undergoes DiVincenzo-Shor gadgets to extract syndrome values. These syndromes effectively capture any errors that may have occurred in the logical state. Leveraging fast qubit reset techniques, we efficiently simulate syndrome extraction requiring only a single ancilla qubit. Subsequently, based on the syndrome value obtained, we apply appropriate correction operators. Our approach involves constructing a lookup table specifically tailored for single-qubit errors in distance three codes, streamlining the error correction process.

We have developed a framework in Qiskit [43] capable of generating circuits to implement QEC procedures for any stabilizer code. By introducing a layer of Pauli errors on the encoded state and simulating the circuit, we derive error rates for various codes and calculate pseudo-threshold values. Notably, our obtained results align closely with the expected pseudo-threshold values derived through analytical methods. This validation underscores the efficacy of our error correction circuit and the procedural approach we employ to analyze the performance of any given code. Significant effort has been dedicated to creating this comprehensive pipeline, which encompasses all components of the QEC procedure - Encoder, Detector, Corrector, and Decoder and provides error rates for any stabilizer code.

In practical settings, implementing QEC introduces the potential for errors due to the additional operations, which themselves may be noisy. This highlights the critical need for fault tolerance in error correction procedures. We delve into pioneering approaches in fault tolerance, which often necessitate entire ancillary code blocks containing encoded logical states prepared with high fidelity. A primary challenge lies in reducing the overhead associated with fault tolerance techniques. Our study primarily focuses on two significant methods: the bare ancillary fault tolerance method proposed by Brown *et. al* and the flag qubit method proposed by Chao and Reichardt [24, 31]. We thoroughly review both techniques and design quantum circuits to implement them, aiming to understand better their efficacy and potential for reducing overhead in fault tolerance.

Focusing on the bare ancillary method, we've successfully modified it to address limita-

tions present in Brown's original method [17]. By generating codes from graph states, we've uncovered a [[8,1,3]] non-CSS code and thoroughly examined its fault-tolerant properties using our upgraded bare ancillary fault-tolerant procedure. In our analysis, we've tested the non-CSS [[7,1,3]] code previously presented by Brown, comparing it against our newly discovered [[8,1,3]] code. Interestingly, while the [[7,1,3]] code fails to achieve fault tolerance under standard depolarizing noise models, it does successfully attain a pseudo-threshold value of 2.8×10^{-4} under anisotropic noise models.

In alignment with our goals, our method applied to the [[8,1,3]] code showcases superior performance, demonstrating fault tolerance under both noise models. Specifically, we obtain a logical error pseudo-threshold value of 1.2×10^{-3} for standard depolarizing noise and 1.7×10^{-4} for the anisotropic model. Additionally, we achieve pseudo-threshold values for total error rates indicating a constant leading order for both noise models. These results highlight the effectiveness of our upgraded bare ancillary method in achieving robust fault tolerance for QECCs.

Our fault-tolerant scheme demonstrates particular suitability to standard depolarizing noise, yielding higher pseudo-threshold values compared to widely studied fault-tolerant techniques implemented on various codes [39]. In our analysis, we contrast the practical error correction procedure with traditional simulation runs, highlighting the merits of each approach. Notably, we emphasize the necessity of a noise-free projection to the encoded state for reliable simulation results.

The proposed bare ancillary method shows promise for application to codes of higher distance, potentially offering improved error rates. Its compatibility with the standard depolarizing noise model suggests its efficacy, warranting further exploration across other prominent noise models such as phase and amplitude damping. Furthermore, the model's performance can be benchmarked through various QECCs and various parameters, facilitating a robust comparison of its strengths and weaknesses [46]. This comprehensive analysis would shed light on the method's potential and guide its optimization for diverse quantum computing applications.

An improved decoder could significantly enhance the performance of the code, and multiple rounds of error correction could effectively address errors induced by Pauli $\mathbf{Z}$ on the syndrome ancilla. Additionally, the code may exhibit suitability for surface codes, potentially achieving high code performance without excessive depth or as numerous rounds, as

typically required [47]. While implementations of Steane's fault tolerance method and the flag method have been realized on physical devices [41], and successful endeavors have been made to realize logical qubits using atom arrays [48], our fault tolerance scheme prioritizes high fidelity with minimal overhead, and it is imperative to test it on modern-day quantum computing architectures.

References

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Review*, vol. 41, no. 2, pp. 303–332, 1999. [Online]. Available: https://doi.org/10.1137/S0036144598347011

[2] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '96. New York, NY, USA: Association for Computing Machinery, 1996, p. 212–219. [Online]. Available: https://doi.org/10.1145/237814.237866

[3] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, p. 505–510, oct 2019. [Online]. Available: http://dx.doi.org/10.1038/s41586-019-1666-5

[4] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," *Theoretical Computer Science*, vol. 560, p. 7–11, dec 2014. [Online]. Available: http://dx.doi.org/10.1016/j.tcs.2014.05.025

[5] A. Aspuru-Guzik, A. D. Dutoi, P. J. Love, and M. Head-Gordon, "Simulated quantum computation of molecular energies," *Science*, vol. 309, no. 5741, pp. 1704–1707, 2005. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.1113479

[6] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[7] D. Gottesman, "Stabilizer codes and quantum error correction," Ph.D. dissertation, California Institute of Technology, California, 1997.

[8] M. Steffen, "What's the difference between error suppression, error mitigation, and error correction?" Oct 2022. [Online]. Available: https://www.ibm.com/quantum/blog/quantum-error-suppression-mitigation-correction

[9] R. W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.

[10] F. Gaitan, *Quantum Error Correction and Fault Tolerant Quantum Computing*. CRC Press, 2013.

[11] M. M. Wilde, "Quantum coding with entanglement," 2008.

[12] V. V. Albert and P. Faist, "Perfect quantum code," in *The Error Correction Zoo*, 2022. [Online]. Available: https://errorcorrectionzoo.org/c/quantum_perfect

[13] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Phys. Rev. A*, vol. 54, pp. 1098–1105, Aug 1996. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.54.1098

[14] A. M. Steane, "Simple quantum error-correcting codes," *Phys. Rev. A*, vol. 54, pp. 4741–4751, Dec 1996. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.54.4741

[15] P. Padmanabhan, A. Chowdhury, F. Sugino, M. G. Majumdar, and K. K. Sabapathy, "Non-css color codes on 2d lattices : Models and topological properties," 12 2021.

[16] D. Poulin, "Stabilizer formalism for operator quantum error correction," *Phys. Rev. Lett.*, vol. 95, p. 230504, Dec 2005. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.95.230504

[17] P. Maheshwari and A. Raina, "Fault-tolerance of the [[8,1,3]] non-css code," 2024. [Online]. Available: https://arxiv.org/abs/2402.19389

[18] D. Aharonov and M. Ben-Or, "Fault-tolerant quantum computation with constant error rate," *SIAM Journal on Computing*, vol. 38, no. 4, pp. 1207–1282, 2008. [Online]. Available: https://doi.org/10.1137/S0097539799359385

[19] R. Chao, "Quantum fault tolerance with flag qubits," 2021. [Online]. Available: https://www.youtube.com/watch?v=etA9l2NUCXI&list=PLgKuh-lKre11rEOFaO62MJCzBXfjZSDqA

[20] A. Kay, "Tutorial on the quantikz package," 2023.

[21] D. Gottesman, "The overhead of fault-tolerant quantum computing," http://meetings.aps.org/link/BAPS.2014.MAR.L35.4, 2014.

[22] P. Shor, "Fault-tolerant quantum computation," in *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. Los Alamitos, CA, USA: IEEE Computer Society, oct 1996. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SFCS.1996.548464

[23] A. M. Steane, "Active stabilization, quantum computation, and quantum state synthesis," *Phys. Rev. Lett.*, vol. 78, pp. 2252–2255, Mar 1997. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.78.2252

[24] M. Li, M. Guti'errez, S. E. David, A. Hernandez, and K. R. Brown, "Fault tolerance with bare ancillary qubits for a [[7,1,3]] code," *Phys. Rev. A*, vol. 96, p. 032341, 9 2017.

[25] R. Chao and B. W. Reichardt, "Flag fault-tolerant error correction for any stabilizer code," *PRX Quantum*, vol. 1, p. 010302, 9 2020.

[26] D. Bacon, "Cse 599d - quantum computing," 2006. [Online]. Available: https://courses.cs.washington.edu/courses/cse599d/06wi/

[27] R. Cleve and D. Gottesman, "Efficient computations of encodings for quantum error correction," *Phys. Rev. A*, vol. 56, pp. 76–82, 7 1997.

[28] "Errata in stabilizer codes and quantum error correction." [Online]. Available: https://www2.perimeterinstitute.ca/personal/dgottesman/thesis-errata.html

[29] M. Grassl, "Variations on encoding circuits for stabilizer quantum codes," in *Coding and Cryptology*. Springer Berlin Heidelberg, 2011, pp. 142–158.

[30] ——, "Bounds on the minimum distance of linear codes and quantum codes," Online available at http://www.codetables.de, 2007, accessed on 2024-03-15.

[31] R. Chao and B. W. Reichardt, "Quantum error correction with only two extra qubits," *Phys. Rev. Lett.*, vol. 121, p. 050502, 8 2018.

[32] D. P. DiVincenzo and P. W. Shor, "Fault-tolerant error correction with efficient quantum codes," *Phys. Rev. Lett.*, vol. 77, pp. 3260–3263, Oct 1996. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.77.3260

[33] E. Knill, "Quantum computing with realistically noisy devices," *Nature*, vol. 434, no. 7029, p. 39—44, March 2005. [Online]. Available: http://arxiv.org/pdf/quant-ph/0410199

[34] D. P. DiVincenzo and P. Aliferis, "Effective fault-tolerant quantum computation with slow measurements," *Phys. Rev. Lett.*, vol. 98, p. 020501, Jan 2007. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.98.020501

[35] H. Aschauer, W. Dür, and H.-J. Briegel, "Multiparticle entanglement purification for two-colorable graph states," *Phys. Rev. A*, vol. 71, p. 012319, Jan 2005. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.71.012319

[36] K. Fujii, *Quantum Computation with Topological Codes*. Springer Singapore, 2015.

[37] Y. Tomita and K. M. Svore, "Low-distance surface codes under realistic quantum noise," *Phys. Rev. A*, vol. 90, p. 062320, Dec 2014.

[38] R. Chao and B. W. Reichardt, "Fault-tolerant quantum computation with few qubits," *npj Quantum Information*, 2018.

[39] C. Chamberland and M. E. Beverland, "Flag fault-tolerant error correction with arbitrary distance codes," *Quantum*, vol. 2, p. 53, feb 2018. [Online]. Available: http://dx.doi.org/10.22331/q-2018-02-08-53

[40] C. Chamberland, A. Kubica, T. J. Yoder, and G. Zhu, "Triangular color codes on trivalent graphs with flag qubits," *New Journal of Physics*, vol. 22, no. 2, p. 023019, 2 2020.

[41] L. Postler, F. Butt, I. Pogorelov, C. D. Marciniak, S. Heußen, R. Blatt, P. Schindler, M. Rispler, M. Müller, and T. Monz, "Demonstration of fault-tolerant steane quantum error correction," 12 2023.

[42] S. Shaw, H. Gupta, S. M. Shah, and A. Raina, "Construction of non-css quantum codes using measurements on cluster states," 2023.

[43] Q. contributors, "Qiskit: An open-source framework for quantum computing," 2023.

[44] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Phys. Rev. A*, vol. 52, pp. R2493–R2496, 10 1995.

[45] A. M. Steane, "Error correcting codes in quantum theory," *Phys. Rev. Lett.*, vol. 77, pp. 793–797, Jul 1996. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.77.793

[46] A. Chatterjee and S. Ghosh, "Magic mirror on the wall, how to benchmark quantum error correction codes, overall ?" 2024.

[47] A. Chatterjee, S. Das, and S. Ghosh, "Q-pandora unboxed: Characterizing noise resilience of quantum error correction codes," 2023.

[48] D. Bluvstein, S. J. Evered, A. A. Geim, S. H. Li, H. Zhou, T. Manovitz, S. Ebadi, M. Cain, M. Kalinowski, D. Hangleiter, J. P. Bonilla Ataides, N. Maskara, I. Cong, X. Gao, P. Sales Rodriguez, T. Karolyshyn, G. Semeghini, M. J. Gullans, M. Greiner, V. Vuletić, and M. D. Lukin, "Logical quantum processor based on reconfigurable atom arrays," *Nature*, vol. 626, no. 7997, pp. 58–65, dec 2023. [Online]. Available: http://dx.doi.org/10.1038/s41586-023-06927-3