TABRIZ UNIVERSITY

DEPARTMENT OF PHYSICS

Basics of Quantum Error Correction and Fault-Tolerant Quantum Computation

Author: Mehdi Rahmani Zadeh

Supervisor: Rahime Soufiani

October 23, 2024

Bachelor's Project

Bachelor of Physics

Contents

English Summary **3**

Preface **4**

1 Introduction **5**

2 Basics and Structure of QEC **7**

2.1 The Nine-Qubit Code 7

2.2 General properties of QEC codes 11

2.3 Stabilizer codes 15

3 QEC codes **18**

3.1 Structure of codes 18

3.2 Codes over GF(4) 21

3.3 Additional codes 22

3.4 The logical Pauli group 23

3.5 The Clifford group 24

4 Fault-Tolerant Gates **27**

4.1 The need for fault tolerance 27

4.2 Definition of fault tolerance 30

4.3 Transversal gates 33

4.4 Fault-tolerant error correction and measurement 35

4.5 Gate teleportation and universal fault-tolerant quantum computation . . 42

4.6 Fault-tolerant state preparation 44

5 Fault-Tolerant Circuits and the Threshold Theorem **46**

5.1 Good and bad extended rectangles 46

5.2 Level reduction 49

References **59**

English Summary

Quantum states are extremely fragile, which means that implementing quantum error correction will likely be essential for constructing reliable quantum computers. Quantum error-correcting codes share some fundamental principles with classical error-correcting codes but also have key distinctions. Many quantum codes are defined using a concept known as the stabilizer, which is a finite Abelian group that helps in clearly outlining the code's error-correcting capabilities. This stabilizer approach also highlights links to classical coding theory, particularly to classical codes based on the finite field GF(4), which has four elements. In addition to error correction, a successful quantum computer requires fault-tolerant computation methods, which specify how to execute quantum gates on qubits encoded within an error-correcting code. According to the threshold theorem, if the error rate for each quantum gate or time step stays below a certain fixed threshold, it is feasible to carry out any quantum computation accurately.

Preface

This report presents my bachelor's project completed in the fourth year of my physics degree. It was prepared using the master's thesis LaTeX template "IT University Copenhagen" by Tony Beltramelli, available on `www.sharelatex.com`.

The content of this report explores the concepts of quantum error correction and fault-tolerant quantum computation, which are essential for developing reliable quantum computers. To fully appreciate these topics, a background in quantum information theory, group theory, and linear algebra is required. These areas provide the necessary tools for understanding quantum states, operations, and the mathematical structure underlying quantum error-correcting codes.

I would like to extend my deepest gratitude to my supervisor, Rahime Soufiani, for his invaluable guidance, insightful answers to my questions, and for the opportunity to deepen my knowledge in this fascinating field. His mentorship has been instrumental in shaping this project and enhancing my understanding of the subject matter.

Chapter 1

Introduction

Quantum computers hold significant promise, but to reach their full potential, they must be shielded from noise. Unlike quantum computers, classical computers don't require error correction. This is partly because classical systems rely on a vast number of electrons, so if a few experience issues, it doesn't heavily impact the computation. In contrast, a quantum computer's qubit may consist of just one or a few particles, which increases the need for error correction. Classical computers are also digital and self-correct by rounding values to 0 or 1 after each step. However, quantum computers operate in a continuous range of states, so they lack this straightforward correction process. For example, an intended state, $\alpha|0\rangle + \beta e^{i\phi}|1\rangle$, might instead become $\alpha|0\rangle + \beta e^{i(\phi+\delta)}|1\rangle$ due to a slight over-rotation, making it close to the desired state but still incorrect. Without correction, these small inaccuracies accumulate and eventually lead to large errors.

Additionally, quantum states are inherently sensitive—observing one will collapse it. A superposition like $\alpha|0\rangle + \beta|1\rangle$ turns into $|0\rangle$ or $|1\rangle$ based on the probabilities $|\alpha|^2$ and $|\beta|^2$. The surrounding environment is constantly interacting with and "observing" the state in a phenomenon known as decoherence. A primary goal of quantum error correction is to prevent such interactions, thereby protecting the quantum data from unwanted environmental disturbances.

Quantum states are inherently fragile; observing a quantum state collapses it. For instance, a superposition like $\alpha|0\rangle + \beta|1\rangle$ will collapse to $|0\rangle$ with a probability of $|\alpha|^2$ and to $|1\rangle$ with a probability of $|\beta|^2$. The environment constantly interacts with quantum states in a process known as decoherence, which acts as a kind of "measurement" by the environment. One purpose of quantum error correction is to shield the quantum data from such unwanted interactions.

While classical error-correcting codes are well-understood, they don't translate directly to quantum systems. Quantum error correction has additional complexities: it must correct errors in both the amplitude (bit-flip errors) and phase of a quantum state. This is not an issue in classical error correction, where a straightforward repetition code

can correct errors by taking the majority value, as in the code:

$$0 \to 000 \quad \text{and} \quad 1 \to 111 \tag{1.1}$$

which corrects a bit-flip error in a state like 010 by changing it back to 000.

If we attempted to apply a similar repetition code in the quantum domain, it would involve cloning a state $|\psi\rangle$ to $|\psi\rangle \otimes |\psi\rangle \otimes |\psi\rangle$. However, such a code is impossible in quantum mechanics due to the No-Cloning theorem, which states that it's impossible to make an identical copy of an arbitrary unknown quantum state.

Theorem 1 (No-Cloning): *It is impossible to create a quantum operation that maps a state $|\psi\rangle$ to $|\psi\rangle \otimes |\psi\rangle$ for all possible states $|\psi\rangle$.*

Proof. This result follows from the linear nature of quantum mechanics. Suppose, for the sake of argument, that such an operation exists, and consider two distinct states $|\psi\rangle$ and $|\phi\rangle$. By the operation's definition, it would follow that:

$$|\psi\rangle \to |\psi\rangle|\psi\rangle \tag{1.2}$$

$$|\phi\rangle \to |\phi\rangle|\phi\rangle \tag{1.3}$$

Applying this operation to the superposition $|\psi\rangle + |\phi\rangle$ should yield:

$$|\psi\rangle + |\phi\rangle \to (|\psi\rangle + |\phi\rangle)(|\psi\rangle + |\phi\rangle) \tag{1.4}$$

However, by linearity, the actual result would be:

$$|\psi\rangle + |\phi\rangle \to |\psi\rangle|\psi\rangle + |\phi\rangle|\phi\rangle \tag{1.5}$$

This differs from the expected outcome by an additional cross-term:

$$|\psi\rangle|\phi\rangle + |\phi\rangle|\psi\rangle \tag{1.6}$$

Thus, due to this inconsistency, such a cloning operation cannot exist in quantum mechanics. □

Chapter 2

Basics and Structure of QEC

The Pauli matrices form the basis of quantum error correction. They include:

- **Identity I:**

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \text{with} \quad I|a\rangle = |a\rangle \tag{2.1}$$

- **Bit Flip X:**

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \text{such that} \quad X|a\rangle = |a \oplus 1\rangle \tag{2.2}$$

- **Phase Flip Z:**

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad \text{where} \quad Z|a\rangle = (-1)^a|a\rangle \tag{2.3}$$

- **Combined Bit and Phase Flip Y:**

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = iXZ \quad \text{such that} \quad Y|a\rangle = i(-1)^a|a \oplus 1\rangle \tag{2.4}$$

2.1 The Nine-Qubit Code

To address both bit-flip and phase-flip errors, a variation of the classical repetition code can be used, known as the nine-qubit code. Here's how it works: - Encoding of $|0\rangle$ and $|1\rangle$:

$$|0\rangle \to |\bar{0}\rangle = (|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle) \tag{2.5}$$

$$|1\rangle \to |\bar{1}\rangle = (|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle) \tag{2.6}$$

These encoded states avoid violating the No-Cloning theorem because an arbitrary superposition $\alpha|0\rangle + \beta|1\rangle$ does not directly clone but rather forms a unique linear combi-

nation:

$$\alpha|0\rangle + \beta|1\rangle \neq [\alpha(|000\rangle + |111\rangle) + \beta(|000\rangle - |111\rangle)]^{\otimes 3} \tag{2.7}$$

The code's structure, organized in three layers, allows it to independently address:
1. Bit-flip errors: Corrected by taking the majority within each triplet of qubits. For instance, if there's a bit-flip error, $|010\rangle$ or $|101\rangle$ is restored to $|000\rangle$ or $|111\rangle$.
2. Phase-flip errors: Managed by taking the majority of the phase signs across the layers, ensuring consistency.

Since these corrections are independent, the nine-qubit code can handle both bit-flip and phase-flip errors simultaneously. Care must be taken during measurements to ensure that only the necessary information is extracted to avoid collapsing the superposition of the code state. This will be elaborated further in section 2.3.

Bit-flip, phase-flip, and combined bit-and-phase-flip errors are central in quantum error correction, so we will briefly review these types. We also include the identity matrix to represent the case where no error has occurred. we defined these four operators, with I as the identity, X for bit-flip, Z for phase-flip, and Y for combined bit-and-phase flip above. The inclusion of an i factor in the definition of Y is largely a convenience; overall phases in quantum mechanics have no physical impact, though they can simplify or complicate calculations depending on the context.

The set of all operators formed by taking tensor products of these four matrices $\{I, X, Y, Z\}$ across multiple qubits forms what is called the Pauli group. These operators have specific commutation properties: X, Y, and Z anticommute with one another, for example:

$$XZ = -ZX \quad \text{or} \quad \{X, Z\} = 0, \tag{2.8}$$

and similarly,

$$\{X, Y\} = 0 \quad \text{and} \quad \{Y, Z\} = 0. \tag{2.9}$$

For an n-qubit system, the n-qubit Pauli group P_n contains all possible combinations of these operators across n qubits, including an overall phase factor of ± 1 or $\pm i$. This results in a total of 4^{n+1} elements.

While the specific phase of each operator isn't generally crucial, it plays a key role in the structure of the group. The Pauli group is non-Abelian, meaning that the order of operations affects the result. Retaining phase distinctions is necessary to preserve this non-commutative property, which is essential in the context of quantum error correction and computation.

The Pauli group P_n is particularly valuable due to its straightforward algebraic properties. For any two elements in P_n, they will either commute or anticommute with each other. Additionally, squaring any element in P_n results in ± 1. For practical purposes, we typically only use the elements with a square of $+1$, which consist of tensor products

of I, X, Y, and Z with an overall sign of ± 1; the inclusion of i simply ensures that P_n meets the requirements to be a group.

The weight of an operator Q in P_n is defined as the number of tensor factors that are not the identity I. For instance, the operator $X \otimes Y \otimes I$ has a weight of 2 because only two of its factors are non-identity.

The Pauli matrices are also crucial because they span the space of 2×2 matrices for single-qubit operators, and the n-qubit Pauli group P_n spans the space of $2^n \times 2^n$ matrices, making them versatile for representing a wide range of quantum operations. For example, consider a general phase error represented by

$$R_{\theta/2} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix} = e^{i\theta/2} \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix} \tag{2.10}$$

where the overall phase factor is irrelevant. This can be rewritten in terms of Pauli matrices as:

$$R_{\theta/2} = \cos\frac{\theta}{2} I - i \sin\frac{\theta}{2} Z \tag{2.11}$$

demonstrating how Pauli matrices like I and Z can be combined to describe various quantum transformations, including phase shifts.

Our earlier error correction method can also handle more general types of errors without requiring additional steps. For example, this approach may use ancilla qubits—extra qubits initialized to $|0\rangle$—to identify the error type. By interacting with the main qubit, the ancilla records the error information, allowing us to correct it by inverting the error detected.

Consider an error represented by Z acting on a state $\alpha|0\rangle + \beta|1\rangle$:

$$Z\left(\alpha|0\rangle + \beta|1\rangle\right) \otimes |0\rangle_{\text{anc}} \to Z\left(\alpha|0\rangle + \beta|1\rangle\right) \otimes |Z\rangle_{\text{anc}} \tag{2.12}$$

After measuring, the ancilla's state indicates the presence of an error. We then apply the correction, returning the system to:

$$\alpha|0\rangle + \beta|1\rangle \otimes |Z\rangle_{\text{anc}} \tag{2.13}$$

If there's no error (represented by the identity I), the process yields:

$$I\left(\alpha|0\rangle + \beta|1\rangle\right) \otimes |0\rangle_{\text{anc}} \to \left(\alpha|0\rangle + \beta|1\rangle\right) \otimes |0\rangle_{\text{anc}} \tag{2.14}$$

In cases where the error is a phase rotation $R_{\theta/2}$, the ancilla will capture a superposition of error states:

$$\cos\frac{\theta}{2} I\left(\alpha|0\rangle + \beta|1\rangle\right) \otimes |\text{no error}\rangle_{\text{anc}} - i \sin\frac{\theta}{2} Z\left(\alpha|0\rangle + \beta|1\rangle\right) \otimes |Z\rangle_{\text{anc}} \tag{2.15}$$

Measuring the ancilla then collapses this superposition. With probability $\sin^2 \frac{\theta}{2}$, the ancilla indicates an error, yielding:

$$Z\left(\alpha|0\rangle + \beta|1\rangle\right) \otimes |Z\rangle_{\text{anc}} \tag{2.16}$$

and with probability $\cos^2 \frac{\theta}{2}$, it indicates no error:

$$I\left(\alpha|0\rangle + \beta|1\rangle\right) \otimes |\text{no error}\rangle_{\text{anc}} \tag{2.17}$$

In either case, we can correct the error based on the ancilla's state, restoring the original state. This logic holds for any linear combination of errors, ensuring a broad error correction capability. Here's a paraphrase of the provided paragraphs on quantum error correction:

Theorem 2: *If a quantum code is capable of correcting errors A and B, it can also correct any linear combination of these errors. Specifically, if the code can correct all weight t Pauli errors, it will correct all t-qubit errors as well.*

Until now, we have only examined single unitary errors affecting the code. However, we can easily extend this to include all potential quantum errors. The most comprehensive quantum operation, which encompasses decoherence, involves interacting the quantum state with additional qubits through a unitary operation and subsequently discarding some qubits. This procedure can transform pure quantum states into mixed states, typically represented using density matrices. The general operation can be expressed as a transformation of density matrices: $\rho \to \sum_i E_i \rho E_i^\dagger$, where the E_i operators are normalized such that $\sum_i E_i^\dagger E_i = I$. Here, the density matrix ρ represents an ensemble of pure quantum states $|\psi\rangle$, each residing within the coding space of the code. This operation then modifies each $|\psi\rangle$ as follows: $|\psi\rangle \to E_i|\psi\rangle$ with a probability of $|E_i|\psi\rangle|^2$. If we can correct each individual error E_i, then we can also correct this broader error. For example, if quantum operations affect only a single qubit within the code, the operators E_i will be contained within the linear combinations of I, X, Y, and Z, enabling correction. Thus, the term "all t-qubit errors" in Theorem 2 indeed refers to all t-qubit errors, not just the unitary types.

Moreover, it's reasonable to assume that each qubit in our nine-qubit code experiences minor errors. For example, if qubit i encounters the error $I + \epsilon E_i$, where E_i is some single-qubit error, the total error can be represented as:

$$\bigotimes(I + \epsilon E_i) = I + \epsilon(E_1 \otimes I^{\otimes 8} + I \otimes E_2 \otimes I^{\otimes 7} + \ldots) + O(\epsilon^2) \tag{2.18}$$

To the first order of ϵ, the overall error consists of the sum of single-qubit errors, which the nine-qubit code is designed to correct. Consequently, after applying the error correction process, the state will be accurate to $O(\epsilon^2)$ (when the contributions from two-qubit

errors start to matter). Although the code cannot fully correct this error, it still offers substantial improvement compared to not implementing any error correction when ϵ is small. A code capable of correcting more errors would yield even greater benefits.

2.2 General properties of QEC codes

Let's explore the essential properties that contribute to the effectiveness of the nine-qubit code and establish criteria for a subspace to function as a quantum error-correcting code (QECC). One important aspect is linearity, which applies to all quantum codes. To correct errors, it suffices to address a basis of errors (specifically, I, X, Y, and Z in the case of a single qubit), as stated in Theorem 2; all other errors will be handled accordingly.

In any quantum code, it is crucial to maintain the distinction between the states $|0\rangle$ and $|1\rangle$, even when errors occur. This means that the action of an error operator E on $|0\rangle$ must remain orthogonal to the action of another error operator F on $|1\rangle$, expressed mathematically as:

$$\langle 0|E^\dagger F|1\rangle = 0. \tag{2.19}$$

It is sufficient to differentiate between the errors E and F when they are applied to the states $|0\rangle$ and $|1\rangle$. If a measurement can accurately identify the specific error, we can implement the necessary correction. This condition can be represented as:

$$\langle 0|E^\dagger F|0\rangle = \langle 1|E^\dagger F|1\rangle = 0 \quad \text{for } E \neq F. \tag{2.20}$$

However, equation(2.20) is not strictly necessary for the nine-qubit code, as we cannot distinguish between Z_1 and Z_2, but that is acceptable since we can correct either one with a single operation. To better understand the necessary conditions, it is useful to examine the operators $F_1 = (Z_1 + Z_2)/2$ and $F_2 = (Z_1 - Z_2)/2$ instead of directly using Z_1 and Z_2. The operators F_1 and F_2 span the same space as Z_1 and Z_2, which confirms that Shor's code can certainly correct them; now let's see how that works. When we consider the F operators as our basis errors, equation (2.20) holds true. This allows us to measure the error and identify it.

We also need to be able to invert the error, which presents a potential issue since F_1 and F_2 are not unitary. Nevertheless, F_1 behaves like Z_1 within the coding space, meaning that $Z_1^\dagger$ is sufficient to invert F_1 for the states of interest. On the other hand, F_2 acts like the zero operator in the coding space. Although we cannot invert this operator, it is unnecessary because F_2 annihilates codewords, ensuring that it does not contribute to the system's actual state.

The requirement for inverting errors leads to a third condition:

$$\langle 0|E^\dagger E|0\rangle = \langle 1|E^\dagger E|1\rangle. \tag{2.21}$$

This value will either be nonzero (as with F_1), indicating that some unitary operator acts the same way as E on the coding space, or it will be zero (as with F_2), meaning E annihilates codewords and cannot occur.

These observations demonstrate that if there exists a basis for the error space where equations (2.19), (2.20), and (2.21) are satisfied, then the states $|0\rangle$ and $|1\rangle$ form a quantum error-correcting code. By manipulating these three equations and extending them to encompass multiple encoded qubits, we can derive the following theorem.

Theorem 3: *If E represents a linear space of errors acting on the Hilbert space H, then a subspace C of H qualifies as a quantum error-correcting code for correcting the errors in E if and only if the following condition holds for all $E \in E$:*

$$\langle\psi|E^\dagger E|\psi\rangle = C(E). \tag{2.22}$$

The function $C(E)$ is independent of the state $|\psi\rangle$.

Proof. Let $\{E_a\}$ be a basis for the error space E and $\{|\psi_i\rangle\}$ be a basis for the code subspace C. By equating E and $|\psi\rangle$ to the basis elements, as well as considering sums and differences of two basis elements (with or without a phase factor i), we can establish that equation (2.22) is equivalent to:

$$\langle\psi_i|E_a^\dagger E_b|\psi_j\rangle = C_{ab}\delta_{ij}, \tag{2.23}$$

where C_{ab} is a Hermitian matrix that does not depend on i and j.

Assuming equation (2.23) holds, we can diagonalize C_{ab}. This process involves selecting a new basis $\{F_a\}$ for the error space E, resulting in the conditions represented by equations (2.19), (2.20), and (2.21). The preceding arguments demonstrate that in this new basis, we can measure the error, uniquely determine it, and invert it within the coding space. Therefore, this leads us to conclude that we indeed have a quantum error-correcting code.

Now, suppose we have an established quantum error-correcting code, and let $|\psi\rangle$ and $|\phi\rangle$ denote two distinct codewords. In this case, we must have:

$$\langle\psi|E^\dagger E|\psi\rangle = \langle\phi|E^\dagger E|\phi\rangle \tag{2.24}$$

for all E. This means that condition (2.22) must hold. If this is not the case, then the error E would alter the relative sizes of $|\psi\rangle$ and $|\phi\rangle$. The states $|\psi\rangle + |\phi\rangle$ and $|\psi\rangle + c|\phi\rangle$ would both still be valid codewords, leading to:

$$E(|\psi\rangle + |\phi\rangle) = N(|\psi\rangle + c|\phi\rangle), \tag{2.25}$$

where N is a normalization factor, and

$$c = \frac{\langle\psi|E^\dagger E|\psi\rangle}{\langle\phi|E^\dagger E|\phi\rangle}. \tag{2.26}$$

The action of the error E would effectively change the encoded state, resulting in a failure of the code unless $c = 1$. □

There is a subtlety in the phrasing of equation (2.22). We require E to represent a linear space of errors, meaning it must be closed under the sums of errors that may act on different qubits. However, for a code that corrects t errors, it is sufficient to consider only E_a and E_b acting on t qubits, as shown in equation (2.23). We can narrow this even further by focusing solely on Pauli operators for E_a and E_b, since these operators will span the space of t-qubit errors. This leads us to an alternative formulation of the condition:

$$\langle\psi|E|\psi\rangle = C'(E), \tag{2.27}$$

where E can be any operator acting on $2t$ qubits (replacing $E_a^\dagger E_b$ in equation (2.23)). This formulation indicates that no measurement on $2t$ qubits can reveal any information about the codeword. In other words, we can detect up to $2t$ errors in the code without necessarily being able to identify what those errors are, distinguishing them only from the identity operation.

If the matrix C_{ab} in equation (2.23) has full rank, the code is termed nondegenerate. Conversely, if it does not, as is the case with the nine-qubit code, it is considered degenerate. In a degenerate code, different errors appear indistinguishable when they act on the coding subspace.

For a nondegenerate code, we can establish a straightforward bound on the code's parameters by simply counting states. Each error E acting on each basis codeword $|\psi_i\rangle$ generates a linearly independent state. All these states must fit within the complete Hilbert space of n qubits, which has a dimension of 2^n. If the code encodes k qubits and can correct errors on up to t qubits, then the following inequality holds:

$$\left(\sum_{j=0}^{t} 3^j \binom{n}{j}\right) 2^k \leq 2^n. \tag{2.28}$$

The expression in parentheses represents the number of errors of weight t or less, which corresponds to the number of tensor products of I, X, Y, and Z that act as the identity on all but t or fewer qubits. This inequality is known as the quantum Hamming bound. While the quantum Hamming bound applies only to nondegenerate codes, no known codes surpass this limit. For $t = 1$ and $k = 1$, the quantum Hamming bound indicates that n must be at least 5. In fact, there exists a code with $n = 5$, which will be discussed later. A code that can correct t errors is defined to have a distance of $2t+1$, as it requires $2t + 1$ single-qubit changes to transition from one codeword to another. We can also

define distance as the minimum weight of an operator E that violates equation (2.27), allowing for the possibility of codes with even distances.

A quantum code that employs n qubits to encode k qubits with a distance d is denoted as an $[[n, k, d]]$ code (the double brackets differentiate it from classical codes). For example, the nine-qubit code is represented as a $[[9, 1, 3]]$ code, while the five-qubit code is a $[[5, 1, 3]]$ code.

Additionally, we can establish a lower bound to determine when codes exist. Although we won't provide a proof here, it is known that an $[[n, k, d]]$ code exists when the following condition is satisfied:

$$\left(\sum_{j=0}^{d-1} 3^j \binom{n}{j}\right) 2^k \leq 2^n, \tag{2.29}$$

which is referred to as the quantum Gilbert-Varshamov bound. This condition differs from the quantum Hamming bound in that the sum extends up to $d-1$ (which is equal to $2t$) instead of stopping at t.

Theorem 4: a quantum $[[n, k, d]]$ code exists if the condition in equation (2.29) is satisfied. Additionally, any nondegenerate $[[n, k, d]]$ code must adhere to equation (2.28). For large n, where $R = \frac{k}{n}$ and $p = \frac{d}{2n}$ are fixed, the best nondegenerate quantum codes fulfill the following inequalities:

$$1 - 2p \log_2 3 - H(2p) \leq R \leq 1 - p \log_2 3 - H(p), \tag{2.30}$$

where $H(x) = -x \log_2 x - (1-x) \log_2 (1-x)$ is the binary entropy function.

Another important bound, known as the Knill-Laflamme bound or the quantum Singleton bound, is applicable even to degenerate quantum codes. For an $[[n, k, d]]$ quantum code, it states that:

$$n - k \geq 2d - 2. \tag{2.31}$$

This implies that the $[[5, 1, 3]]$ code is indeed optimal; a $[[4, 1, 3]]$ code would not satisfy this inequality.

Although I will not prove the general case of this bound, the case for $k = 1$ can be understood through the No-Cloning theorem. Suppose we have r qubits missing from the code. We can replace these missing qubits with $|0\rangle$ states, but this results in r errors on the modified codeword. Although the types of errors are unknown, all possible errors affect the same set of r qubits. Consequently, all products $E_a^\dagger E_b$ in equation (2.30) will have a weight of r or less, meaning this type of error (referred to as an "erasure" error) can be corrected by a code with a distance of $r + 1$.

Now, if we consider an $[[n, 1, d]]$ code with $n \leq 2d - 2$, we could divide the qubits in the code into two groups, each containing at most $d - 1$ qubits. Each group would then be subject to at most $d - 1$ erasure errors and could thus be corrected independently without access to the other group. This would result in two copies of the encoded state,

which contradicts the principles established by the No-Cloning theorem.

2.3 Stabilizer codes

Let's revisit the nine-qubit code and outline the steps necessary for correcting errors. First, we need to verify whether the first three qubits are identical and identify any discrepancies. This can be achieved by measuring the parities of the first two qubits and the second and third qubits. Specifically, we perform the measurements:

$$Z \otimes Z \otimes I \quad \text{and} \quad I \otimes Z \otimes Z. \tag{2.32}$$

The first measurement checks for an X error on qubits one or two, while the second checks for an X error on qubits two or three. It's important to note that the errors detected in both scenarios anticommute with the measured errors. By combining these two measurements, we can precisely locate the error.

We apply the same approach to the remaining two sets of three qubits, resulting in four additional operators to measure. Importantly, measuring $Z \otimes Z$ provides the exact information needed without introducing additional knowledge that could collapse the superpositions used in the code. To achieve this, we introduce an ancilla qubit, initially in the state $|0\rangle + |1\rangle$, and perform controlled-Z operations with the first two qubits of the code:

$$(|0\rangle + |1\rangle) \otimes |abc\rangle \to |abc\rangle \otimes \left(|0\rangle + (-1)^{a \oplus b}|1\rangle\right). \tag{2.33}$$

After this operation, measuring the ancilla in the basis $|0\rangle \pm |1\rangle$ provides the eigenvalue of $Z \otimes Z \otimes I$ without revealing any additional information about the data.

Next, we need to determine whether the signs of the three qubits are the same or different. This is done by measuring:

$$X \otimes X \otimes X \otimes X \otimes X \otimes X \otimes I \otimes I \otimes I \tag{2.34}$$

and

$$I \otimes I \otimes I \otimes X \otimes X \otimes X \otimes X \otimes X \otimes X. \tag{2.35}$$

These measurements total eight operators, with the first six qubits measuring Z errors on the first group and the last six on the second group. As before, it's crucial that the detected error anticommutes with the measured operator. This characteristic is not coincidental; we are measuring an operator M that should yield an eigenvalue of $+1$ for any codeword:

$$M|\psi\rangle = |\psi\rangle. \tag{2.36}$$

If an error E occurs that anticommutes with M, the state changes to $E|\psi\rangle$, leading to:

$$M(E|\psi\rangle) = -EM|\psi\rangle = -E|\psi\rangle. \tag{2.37}$$

Thus, the new state will have an eigenvalue of -1 rather than $+1$. This property is fundamental for correcting errors: each single-qubit error E anticommutes with a specific set of operators $\{M\}$, which enables us to determine the nature of E.

Z	Z	I	I	I	I	I	I	I
I	Z	Z	I	I	I	I	I	I
I	I	I	Z	Z	I	I	I	I
I	I	I	I	Z	Z	I	I	I
I	I	I	I	I	I	Z	Z	I
I	I	I	I	I	I	I	Z	Z
X	X	X	X	X	X	I	I	I
I	I	I	X	X	X	X	X	X

Table 2.1: The stabilizer for the nine-qubit code. Each column represents a different qubit.

In the nine-qubit code, we are unable to precisely identify the error E, but this lack of specificity is not a concern. For example, we cannot differentiate between Z_1 and Z_2 because:

$$Z_1 Z_2|\psi\rangle = |\psi\rangle \Leftrightarrow Z_1|\psi\rangle = Z_2|\psi\rangle. \tag{2.38}$$

This illustrates that the nine-qubit code is degenerate. Table 2 presents a summary of the measured operators. These eight operators form an Abelian group known as the stabilizer of the nine-qubit code. The stabilizer comprises all operators M in the Pauli group such that $M|\psi\rangle = |\psi\rangle$ for every state $|\psi\rangle$ within the code.Additionally, if we have an Abelian subgroup S of the Pauli group P_n (which consists of tensor products of I, X, Y, and Z with an overall phase factor of ± 1 or $\pm i$), we can define a quantum code $T(S)$ as the collection of states $|\psi\rangle$ for which $M|\psi\rangle = |\psi\rangle$ for all $M \in S$. It is essential for S to be Abelian and to exclude -1; otherwise, the code becomes trivial. If $M, N \in S$, we can express:

$$MN|\psi\rangle = M|\psi\rangle = |\psi\rangle, \tag{2.39}$$

$$NM|\psi\rangle = N|\psi\rangle = |\psi\rangle. \tag{2.40}$$

From this, it follows that:

$$[M, N]|\psi\rangle = MN|\psi\rangle - NM|\psi\rangle = 0. \tag{2.41}$$

Since elements of the Pauli group either commute or anticommute, we have $[M, N] = 0$. Clearly, if $M = -1 \in S$, then there exists no nontrivial state $|\psi\rangle$ such that $M|\psi\rangle = |\psi\rangle$. If these conditions are met, there will exist a nontrivial subspace comprising states that are invariant under all elements of the stabilizer. The number of errors that the code can correct can be determined by examining the operators that commute with the stabilizer. Specifically, we can correct errors E and F if either $E^\dagger F \in S$ (indicating that E and F have the same effect on the codewords), or if there exists some $M \in S$ such that $\{M, E^\dagger F\} = 0$. In this latter case, measuring the operator M will allow us to distinguish between errors E and F. If the first condition is satisfied, then the stabilizer code is considered degenerate; otherwise, it is classified as nondegenerate. To formalize this, we can examine the normalizer $N(S)$ of S within the Pauli group, which, in this scenario, coincides with the centralizer comprising Pauli operators that commute with S. The distance d of the code is defined as the minimum weight of any operator in $N(S) \setminus S$.

Theorem 5. *Let S be an Abelian subgroup of order 2^a in the n-qubit Pauli group, with the condition that $-1 \notin S$. Define d as the minimum weight of an operator in $N(S) \setminus S$. Then, the space of states $T(S)$ that is stabilized by all elements of S forms an $[[n, n-a, d]]$ quantum code.*

To correct errors with a weight of $\frac{d-1}{2}$ or less, we simply measure the generators of S. This measurement will yield a list of eigenvalues, referred to as the error syndrome, which indicates whether the error E commutes or anticommutes with each of the generators. The error syndromes for E and F are the same if and only if the error syndrome for $E^\dagger F$ is trivial. In the case of a nondegenerate code, the error syndrome uniquely identifies the error E (up to a trivial overall phase), as the generator that anticommutes with $E^\dagger F$ differentiates E from F. For a degenerate code, the error syndrome is not unique; however, error syndromes are repeated only when $E^\dagger F \in S$, indicating that E and F act identically on the codewords.

If the stabilizer has a generators, then the code encodes $n - a$ qubits. Each generator partitions the allowed Hilbert space into $+1$ and -1 eigenspaces of equal size. To validate this statement, observe that we can identify an element G from the Pauli group that possesses any specified error syndrome (even if G has a weight greater than $\frac{d-1}{2}$ or possibly greater than d). Each G maps $T(S)$ into an orthogonal but isomorphic subspace, leading to 2^a possible error syndromes. Therefore, $T(S)$ can have a dimension of at most $\frac{2^n}{2^a}$. Furthermore, since the Pauli group spans $U(2^n)$, its action on any single state generates a basis for H. Given that every Pauli operator corresponds to some error syndrome, it follows that the dimension of $T(S)$ is exactly 2^{n-a}.

Chapter 3

QEC codes

3.1 Structure of codes

Stabilizers are a useful way to represent quantum codes, especially when starting with classical coding concepts. In classical coding theory, a linear code is defined by a generator matrix or its dual, the parity check matrix. The generator matrix's rows form the fundamental codewords, and all other codewords result from linear combinations of these rows. On the other hand, the parity check matrix lists conditions (or checks) that every codeword must satisfy. In the quantum realm, the stabilizer serves a similar role to the classical parity check matrix, defining a set of conditions that maintain the code's structure. A prominent example in classical coding is the seven-bit Hamming code, which can correct a single error. Its parity check matrix is:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \tag{3.1}$$

This matrix provides the conditions that each codeword in the Hamming code must satisfy, laying a foundation for constructing quantum codes with similar error-correcting properties. To transform the classical Hamming code into a quantum stabilizer code, we

Z	Z	Z	Z	I	I	I
Z	Z	I	I	Z	Z	I
Z	I	Z	I	Z	I	Z
X	X	X	X	I	I	I
X	X	I	I	X	X	I
X	I	X	I	X	I	X

Table 3.1: Stabilizer for the seven-qubit code.

replace each "1" in the parity-check matrix with the Pauli operator Z and each "0" with the identity I. This substitution simply redefines the parity check measurements in terms

of quantum operators. The classical code's ability to correct one error translates here into the property that any single or two-bit flip error will anticommute with one of these three Z-based operators. Next, if we substitute each "1" with X instead of Z, we define three new operators that anticommute with any single or two-qubit Z-error. Thus, by using a stabilizer comprising three Z-operators and three X-operators, as outlined in Table 3, we create a code capable of correcting any single-qubit error. The first three generators detect X-errors, while the last three detect Z-errors. Errors involving Y are identified since they appear in both categories. To finalize this stabilizer code, we must ensure that the stabilizer is Abelian, which can be easily checked. With six generators over seven qubits, the code encodes one logical qubit, thus forming a $[[7, 1, 3]]$ quantum code. In this example, we used the same classical code for generating both the X and Z stabilizers, though this is not necessary. We could instead choose two different classical codes, C_1 and C_2, as long as the X and Z generators commute. This requirement translates to $C_2^\perp \subseteq C_1$, where $C_2^\perp$ is the dual code of C_2, containing words orthogonal to the codewords of C_2.

If C_1 is an $[n, k_1, d_1]$ code and C_2 is an $[n, k_2, d_2]$ code (where single brackets denote classical codes), then the corresponding quantum code becomes an $[[n, k_1 + k_2 - n, \min(d_1, d_2)]]$ code. This construction is known as the CSS construction, named after its creators, Calderbank, Shor, and Steane.

In a CSS code, all codewords must meet the parity conditions of the classical code C_1, meaning each codeword is a superposition of words in C_1. Additionally, the X generators (from C_2) add words from $C_2^\perp$ to the state, so the CSS codewords take the form:

$$\sum_{w \in C_2^\perp} |u + w\rangle, \quad \text{where } u \in C_1. \tag{3.2}$$

Since $C_2^\perp \subseteq C_1$, it follows that $u + w \in C_1$.

Applying a Hadamard transform on each qubit of the code switches the Z- and X-bases and swaps C_1 with C_2. This transformation changes the codewords to:

$$\sum_{w \in C_1^\perp} |u + w\rangle, \quad u \in C_2. \tag{3.3}$$

To detect and correct errors in a CSS code, we measure the parity checks of C_1 in the Z-basis and the parity checks of C_2 in the X-basis.

$$\begin{array}{ccccc} X & Z & Z & X & I \\ I & X & Z & Z & X \\ X & I & X & Z & Z \\ Z & X & I & X & Z \\ Z & Z & X & I & X \end{array} \tag{3.4}$$

Table 4. The stabilizer for the five-qubit code.

The [[5, 1, 3]] code, which is smaller than many other quantum codes, has its stabilizer given in Table 4. You can confirm that it commutes and indeed has a code distance of 3. The codewords can be found by noting that the sum $\sum_{M \in S} M|\phi\rangle$ lies within the code for any state $|\phi\rangle$. By identifying two such states $|\phi\rangle$ that yield nonzero results for this sum, the code's structure can be derived. This stabilizer representation is efficient, summarizing the coding subspace without explicitly listing each basis codeword.

A practical way to represent stabilizers is through a pair of binary matrices written side-by-side. In the first matrix, a 1 marks the position of any X or Y operators in the stabilizer, while a 0 indicates no X component. The second matrix has a 1 where Y or Z operators appear, with 0 elsewhere. Pauli operator multiplication corresponds to adding rows in both matrices. Two operators M and N will commute if their binary vector representations, say $(a_1|b_1)$ and $(a_2|b_2)$, are orthogonal with respect to a symplectic inner product, $a_1 b_2 + b_1 a_2 = 0$. For instance, the stabilizer matrix for the five-qubit code can be represented as follows:

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

This matrix format provides a clear and compact way to work with stabilizer codes, illustrating their structure and symmetries. As an illustration of how this representation can be utilized, let's demonstrate a fact that was referenced earlier:

Lemma 1. *For any stabilizer S, it is always possible to find at least one error corresponding to any given error syndrome.*

Proof. Assume that S has a generators. The error syndrome of a Pauli operator E can be expressed as an a-component binary vector. Each entry in this vector indicates whether E commutes with (0) or anticommutes with (1) the i-th generator of S. Denote the binary vector representing the i-th generator of S as x_i and the binary vector representing E as e. Then, we say that E has an error syndrome v if the following condition holds:

$$x_i \odot e = v_i, \tag{3.5}$$

where $\odot$ denotes the symplectic inner product.

The generators of S provide linearly independent binary vectors, yielding a independent linear equations in a $2n$-dimensional binary vector space where $a \leq n$. According to a standard theorem in linear algebra, these equations must always have a non-zero solution. In fact, there exists a subspace of solutions that has a dimension of $(2n - a)$. $\square$

Stabilizers	**GF(4)**
I	0
Y	ω^2
X	ω
Z	1
tensor products	vectors
multiplication	addition
$[M, N] = 0$	$\mathrm{tr}(M \cdot \bar{N}\ ) = 0$
$N(S)$	dual

Table 3.2: Connections between stabilizer codes and codes over GF(4).

3.2 Codes over GF(4)

The CSS construction is advantageous because it leverages the extensive existing knowledge of classical binary codes to create quantum codes. However, CSS codes are not as efficient as the optimal stabilizer codes; for example, there is no [[5, 1, 3]] CSS code. To develop the most general stabilizer codes, it is beneficial to utilize another relationship with classical coding theory. Often, classical coding theorists examine not only binary codes but also codes over larger finite fields. One of the simplest finite fields is GF(4), which consists of four elements: 0, 1, ω, $\omega^{\mathbf{2}}$. This field has a characteristic of 2, and its elements satisfy the relations $\omega^3 = 1$ and $\omega + \omega^2 = 1$.

Additionally, two operations on GF(4) are useful to consider. The first is conjugation, which swaps the two roots of the polynomial $x^2 + x + 1$, yielding the results: 1 remains 1, ω becomes $\omega^{\mathbf{2}}$, and 0 stays 0 while ω^2becomes ω. The second operation is trace. The trace of an element x, denoted as tr(x), represents the trace of the linear transformation associated with "multiplication by x" when considering GF(4) as a vector space over Z_2. The traces for the various elements are as follows: $\mathrm{tr}(0) = 0$, $\mathrm{tr}(1) = 0$, $\mathrm{tr}(\ \omega\ ) = 1$, and $\mathrm{tr}(\omega^2) = 1$. Stabilizer codes rely heavily on the Pauli group P_n. A connection between stabilizer codes and codes over $GF(4)$ can be established by associating the four operators I, X, Y, and Z with the four elements of $GF(4)$, as shown in Table 5. The commutativity requirement within the Pauli group translates to a symplectic inner product involving vectors in $GF(4)$. The Abelian nature of the stabilizer can be expressed in the context of $GF(4)$ by stating that the code must be contained within its dual concerning this inner product. To ascertain the number of errors that the code can correct, we need to analyze vectors that lie in the dual (which corresponds to $N(S)$) but not within the code itself (which corresponds to S).

Establishing this correspondence allows us to tap into a wealth of classical coding theory. Numerous classical codes over $GF(4)$ have been identified, many of which are self-dual with respect to the symplectic inner product, thereby defining quantum codes. For example, the five-qubit code is a notable instance; it is essentially a Hamming code

over $GF(4)$. It is important to note that classical coding theorists typically focus on linear codes, which are closed under addition and scalar multiplication. However, in the quantum context, we consider a somewhat broader category known as additive $GF(4)$ codes—these are codes that are closed under the addition of elements but do not necessarily adhere to the rules of scalar multiplication.

3.3 Additional codes

In addition to stabilizer codes, numerous quantum error-correcting codes exist, and significant research has focused on these alternative types. Typically, establishing and utilizing a code requires a certain level of structural organization. There are several methods to ensure that this necessary structure is present. One promising approach is to explore codes that extend beyond qubits to encompass higher-dimensional registers known as qudits. This allows for a natural generalization of stabilizer codes, leading to the discovery of various qudit stabilizer codes. Another avenue is to relax the stabilizer framework slightly, aiming to identify more efficient qubit codes. A particularly noteworthy development in this area is the introduction of operator quantum error correction, commonly referred to as subsystem codes. This methodology involves disregarding certain degrees of freedom within the code, effectively encoding a state as a linear subspace rather than as another state. While subsystem codes do not enhance the fundamental error correction capabilities discussed thus far, they can be beneficial in the context of fault tolerance, as explored in sections 4 and 5.

Another intriguing area of study involves codes that are entirely degenerate, commonly referred to as decoherence-free subspaces (DFS)35. In scenarios where all potential errors act as the identity on the code subspace, no active correction is necessary; the state remains unchanged regardless of the error that occurs. DFS is typically analyzed in the context of continuous-time errors, where the set of possible errors generates a Lie algebra. In this case, the DFS represents a degeneracy of the trivial representation of the Lie algebra acting on the Hilbert space of n qubits. One significant advantage of a DFS is its robustness against very high levels of noise. However, a notable limitation is that non-trivial DFS can only be constructed for specific noise models, although certain models like collective noise are practically relevant. In contrast, more general quantum error-correcting codes (QECC) can effectively reduce error rates across a diverse array of noise types, provided the initial error rates are sufficiently low. Moreover, beyond quantum error correction, there are alternative strategies for protecting qubits from noise. One such method is dynamical decoupling, which involves executing a series of rapid operations designed to cancel out noise. This technique is effective only when the noise varies slowly compared to the speed of the operations being performed. Dynamical decoupling boasts two primary benefits: similar to DFS, it remains effective even at relatively high error

rates; however, unlike DFS or QECC, it does not necessitate the use of additional qubits.

The problem of eliminating errors from a quantum computer is a difficult one, and we will want to use every tool that we can bring to bear on the problem. Most likely, this will include control techniques like dynamical decoupling as a first layer of defense, perhaps followed by specialized error-correcting codes such as a DFS or a phase QECC to handle certain dominant error types, with a more general quantum error-correcting code as the final protection to deal with any kinds of errors not eliminated by the first two layers. However, everything we do — every qubit we add, every additional gate we perform — will have errors in it too, so additional specialized layers of protection come with a cost, not just in additional overhead, but also in additional errors that will need to be cleaned up by the final QECC. It can become a difficult balancing act to judge precisely which protections are useful and which cause more trouble than they are worth.

3.4 The logical Pauli group

The group $N(S)$ is crucial for understanding a code's error correction capabilities, and it will also play a significant role in discussions about fault-tolerant quantum computation. Thus, it is beneficial to take a moment to examine its structure more closely. The elements of $N(S)$ consist of those Pauli operators that commute with every operator in the stabilizer. When considering how an operator $E \in N(S)$ acts on a codeword from the stabilizer code, let $M \in S$ represent an arbitrary stabilizer element. The relationship can be expressed as follows:

$$M(E|\psi\rangle) = EM|\psi\rangle = E|\psi\rangle. \tag{3.6}$$

This equality holds for all $M \in S$, indicating that $E|\psi\rangle$ remains within the code space $T(S)$. In other words, the action of E maps valid codewords to other valid codewords. If E is itself an element of the stabilizer S, this behavior is expected, as it would simply take codewords to themselves. However, if E is not a member of S, it cannot maintain this behavior—it must map at least one codeword to a different codeword. Such an operation is termed a logical operation because it manipulates the encoded state without disrupting the encoding itself. To differentiate between physical operations and logical operations, I will denote logical operations with a bar over the corresponding symbol. For example, X represents a bit flip on an individual physical qubit, while $\overline{X}$ signifies a logical bit flip that alters the state of an encoded qubit. If we consider $F = EM$ where $M \in S$, it follows that $F \in N(S)$ as well, and we have:

$$F|\psi\rangle = EM|\psi\rangle = E|\psi\rangle \tag{3.7}$$

for all $|\psi\rangle \in T(S)$. This means that two Pauli operators that belong to the same

coset of S in $N(S)$ will produce the same action on the codewords. Therefore, the distinct logical operations within $N(S)$ correspond to the elements of the quotient group $N(S)/S$. Additionally, two Pauli operators E and F are said to share the same error syndrome if they reside in the same coset of $N(S)$ within P_n. Since there is always at least one error associated with any specific error syndrome, we know that the size of the Pauli group is given by $|P_n| = 4^{n+1}$. Consequently, we find that:

$$|N(S)| = 4 \cdot 2^{n+k} \quad \text{and} \quad |N(S)/S| = 4^{k+1} \tag{3.8}$$

for a code characterized as $[[n, k, d]]$. Importantly, we can identify $N(S)/S$ with the logical Pauli group P_k. To represent the logical Z operators, you can select any maximal Abelian subgroup R of $N(S)/S$, which includes tensor products of Z and I. The size of such a maximal Abelian subgroup is $4 \cdot 2^k$, since an Abelian subgroup of $N(S)/S$ corresponds to a larger Abelian subgroup of P_n by a factor equal to $|S| = 2^{n-k}$. By selecting elements from $N(S)/S$ that correspond to various syndromes with respect to R, you can also identify the logical X operators. Of course, proper execution of this framework requires appropriate selection of the basis codewords. For example, the encoded state $|0 \ldots 0\rangle$ should be a $+1$-eigenstate of every element within the group R.

3.5 The Clifford group

In the context of stabilizer codes, a particular group of quantum gates frequently arises. These gates are essential for encoding and decoding stabilizer codes, and they are pivotal in fault-tolerance theory. This group is typically referred to as the Clifford group, though its connection to Clifford algebras is somewhat limited. It is formally defined as:

$$C_n = \{U \in U(2^n) \mid UPU^\dagger \in P_n \quad \forall P \in P_n\} \tag{3.9}$$

In other words, the Clifford group consists of unitary operators that normalize the Pauli group P_n within the unitary group $U(2^n)$. Beyond its significance in stabilizer code theory, the Clifford group is noteworthy on its own. It includes several fundamental quantum gates, such as the Hadamard transform H, the $\pi/4$ phase rotation P, and the CNOT gate, represented as follows:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad P = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \quad CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{3.10}$$

To understand how these gates interact with the Pauli group, we can analyze their effects via conjugation. For example, the Hadamard gate transforms Pauli operators as follows:

$$X \to Z, \quad Z \to X. \tag{3.11}$$

The action of H on Y does not need to be explicitly stated, as conjugation is a homomorphism, and since $Y = iXZ$, we find that:

$$Y \to iZX = -Y. \tag{3.12}$$

Notably, the Clifford group is generated by the gates H, P, and $CNOT$. This foundational role in quantum computation highlights the importance of the Clifford group in the broader landscape of quantum error correction and fault-tolerant protocols. In general, specifying a Clifford group operator U can be accomplished by detailing its action on a generating set for the Pauli group, such as X and Z acting on each of the n qubits. This approach works because the Pauli group serves as a basis for the $2^n \times 2^n$ matrices, enabling us to infer the action of U on any projector. However, one ambiguity remains: if $U = e^{i\theta}U'$, then U and U' exhibit identical actions through conjugation. Since this type of global phase carries no physical significance, this ambiguity is relatively inconsequential. The Clifford group can also be represented using binary matrices, similar to stabilizers. Based on the previous argument, we can define a Clifford group element (up to a global phase) by specifying its action on the binary vectors associated with the Pauli operators. Since elements of the Clifford group preserve the commutation and anti-commutation relations, they correspond to symplectic matrices within the $2n$-dimensional binary vector space.

Furthermore, the Pauli group P_n is defined as a normal subgroup of C_n. Given that Pauli operators either commute or anticommute, those within C_n correspond to the identity symplectic matrix. The center $Z(C_n)$ consists solely of diagonal matrices of the form $e^{i\phi}I$, which also correspond to the identity symplectic matrix. We can define $P'_n = Z(C_n)P_n$—the Pauli group augmented by arbitrary phases rather than just ± 1 and $\pm i$. Consequently, P'_n represents the kernel of the mapping from the Clifford group to the group $\mathrm{Sp}(2n, \mathbb{Z}_2)$ of $2n \times 2n$ binary symplectic matrices. Thus, we have the relationship:

$$C_n/P'_n \cong \mathrm{Sp}(2n, \mathbb{Z}_2), \tag{3.13}$$

indicating that if we focus only on the action of the Clifford group modulo phases, the Clifford group effectively behaves as the group of symplectic matrices. As a result of this equivalence, any circuit composed of Clifford group operators acting on an initial stabilizer state, followed by final measurements in the Pauli basis, can be efficiently simulated classically. Although the overall effect of the circuit constitutes a unitary transformation on a 2^n-dimensional Hilbert space, each individual step can be represented as a $2^n \times 2^n$ binary matrix. This allows us to quickly compute the total action of the circuit by

simply multiplying these matrices together. This efficiency extends to scenarios where the circuit incorporates not only unitary gates from the Clifford group but also measurements of Pauli operators throughout the process. Subsequent gates can be influenced by the results of these measurements, yet the classical simulation remains feasible under these conditions.

Chapter 4

Fault-Tolerant Gates

4.1 The need for fault tolerance

A significant challenge remains before we can achieve quantum computers that are robust against errors. It is crucial to learn how to execute operations on a state encoded within a quantum code while maintaining the code's error-correcting capabilities. Additionally, we need to ensure that error correction can be carried out safely, even when the gates involved are themselves prone to noise. A protocol designed to accomplish these objectives is referred to as fault-tolerant (FT). The first fault-tolerant quantum computation protocols were introduced by Shor, but there have been many substantial advancements since then. Current understanding indicates that if the physical error rate for each gate and time step stays below a certain constant threshold, it is possible to execute the desired logical quantum computation with an accuracy that approaches correctness. Furthermore, this can be achieved with overhead that grows polylogarithmically in relation to the duration of the computation. Our objective is to develop protocols that consistently yield the correct output, even when any single component of the circuit experiences failure. The fundamental components required to create a universal quantum computer include:

1. Preparation: This involves operations that prepare a new qubit in a standard state. It suffices to have a single type of preparation that creates a $|0\rangle$ state, although multiple prepared states may be utilized.

2. Quantum Gates: A universal set of quantum gates is essential. It is sufficient to use the Hadamard gate (H), the Controlled-NOT gate (CNOT), and the $\pi/8$ phase rotation $R_{\pi/8} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$. This set of gates generates a group that is dense in $U(2^n)$.

3. Measurement: The capability to measure qubits is required, specifically the ability to measure individual qubits in the standard basis $|0\rangle$ and $|1\rangle$.

4. Wait: To synchronize the operations of gates, it may be necessary for qubits to remain idle without any operations being performed on them.

The physical qubits that compose our quantum error-correcting code are referred to

as physical qubits. Each of the aforementioned actions constitutes a physical action (e.g., executing a physical gate). Each instantiation of one of these components is termed a physical location (or simply a location). The total number of locations in a circuit is therefore bounded by the total number of qubits utilized multiplied by the total number of time steps involved. However, the actual number of locations is often less than the maximum, as we frequently prepare new qubits during the computation and measure qubits that can be discarded before the computation concludes.

It is important to note that wait steps count as locations; however, operations involving classical data (such as measurement results) do not count, as we assume classical computation is perfect. Depending on the specific model being employed, we may simplify our assumptions by stating that modest amounts of classical computation require no time, though this is not a strict requirement. In a quantum circuit, any location can fail, including those designated as wait steps. When a location fails, it is assumed that an error can affect all qubits involved in that specific action. The nature of the error varies depending on the type of action: For single-qubit actions (like preparation, measurement, or single-qubit gates), the error is confined to that single qubit. For two-qubit gates, such as the CNOT gate, we allow for an arbitrary error affecting both qubits involved. This error can include actions that entangle the two qubits.

To model this, the actual error resulting from a failure is considered to be the action of the failed component combined with the inverse of the intended component for that location. For example, if the goal was to apply the Z gate but instead the Y gate was executed, the error can be expressed as:

$$E = Y \cdot Z = iX \tag{4.1}$$

The objective of fault tolerance is to modify a quantum circuit originally designed to operate without errors, transforming it into a new circuit capable of producing the same output as the original circuit, but under the weaker assumption that the number of failed locations is manageable. The specific rules regarding the probability of location failures and the types of errors resulting from such failures will be discussed in section 5. For clarity, any location experiencing an error will sometimes be referred to as a "faulty location." A major challenge in creating a fault-tolerant protocol is managing error propagation. Even if the gates we apply are perfect, their action on a state can exacerbate any existing errors, causing them to spread. This relationship can be expressed as:

$$UE|\psi\rangle = (UEU^\dagger)U|\psi\rangle \tag{4.2}$$

Here, a pre-existing error E on a state $|\psi\rangle$ followed by a correct gate U results in a correct state $U|\psi\rangle$ but with an error transformed to $UEU^\dagger$.

When U is a single-qubit gate, this error propagation is less concerning, as the error's

weight (the number of qubits affected) remains the same, though the error type may change. For instance, an X error may transform into a Z error through a Hadamard gate. The situation becomes problematic with two-qubit gates. A single-qubit error can become a two-qubit error after applying such gates. For example, consider the action of a CNOT gate:

$$\text{CNOT:} \quad X \otimes I \to X \otimes X, \tag{4.3}$$

$$I \otimes Z \to Z \otimes Z. \tag{4.4}$$

In these cases, the CNOT gate can propagate an X error from the first qubit to the second and can similarly propagate a Z error from the second qubit to the first. This error propagation is critical because it can increase the weight of an error. For instance, if a distance-3 code is employed, it can effectively handle a single-qubit error. However, applying a CNOT gate—even if it functions correctly—can convert this single-qubit error into a two-qubit error, which the distance-3 code may not be capable of correcting. Thus, to achieve a universal quantum computer, it is essential to exercise caution when utilizing two-qubit gates, as their interactions can significantly impact the overall fault tolerance of the quantum circuit. The challenges of error propagation in quantum circuits can indeed be addressed through the design of fault-tolerant circuits. In the upcoming sections, I will elaborate on how these circuits can be constructed to ensure that error propagation remains manageable. While errors may still occur and spread, we can effectively correct the resulting errors as long as the initial error rates are kept within certain bounds.

The ultimate objective is to develop fault-tolerant versions for all types of physical operations. I will refer to each of these constructions as a gadget, tailored to perform specific operations. For example, we will create fault-tolerant gates for each member of a universal set of quantum gates. Each gadget will simulate the behavior of its corresponding non-fault-tolerant operation, but instead of acting directly on one or two physical qubits, it will execute the operation on logical qubits that are encoded using a quantum error-correcting code (QECC). When presented with a quantum circuit that needs to be executed, we replace each operation in the original circuit with its corresponding fault-tolerant gadget. Typically, we assume that the original circuit does not take any input directly; instead, all qubits utilized within it must be prepared through dedicated preparation locations. This approach enables us to perform any arbitrary quantum computation, as we can adjust the quantum circuit according to the classical description of the problem we intend to solve. For instance, if our goal is to factor a number N, we can customize the quantum circuit to specifically address the computation related to N.

Finally, the measurement gadgets at the end of the fault-tolerant circuit will generate classical information that, provided the fault-tolerant mechanisms have functioned

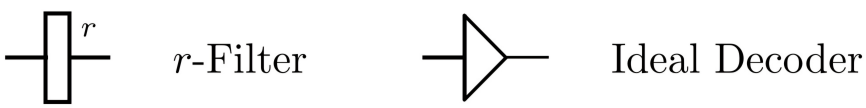
correctly, should yield outcomes equivalent to those of the original circuit if it could be executed without errors. This ensures that the overall computation remains reliable, even in the presence of inherent faults in the physical qubits.

4.2 Definition of fault tolerance

It's important to clarify what we mean by fault tolerance. A fault-tolerant gadget must possess two essential characteristics: First, if the input state to the gadget has a limited number of errors and there are also few errors in the gadget's physical locations, the output state should similarly have a manageable number of errors. Second, if there are not many errors in the input state or during the gadget's operation, the gadget should execute the correct logical operation on the encoded state. To define these characteristics precisely, we need to introduce the concepts of an r-filter and an ideal decoder.

Definition 1: An r-filter is a projector that spans the subspace formed by all states of the type $Q|\psi>$, where $|\psi>$ is any codeword and Q represents a Pauli error with a weight of at most r. An ideal decoder is a transformation that takes the input state and applies a decoding process (including error correction) using a circuit that has no faulty locations. The r-filter projects onto states with a maximum of r errors. However, the r-filter cannot identify the correct codeword during computation, meaning that even a 0-filter may mistakenly project onto an incorrect state. The key point is that only states potentially generated from a valid codeword with up to r single-qubit errors can pass through the r-filter. The ideal decoder processes the encoded state, corrects any errors, and produces an unencoded state. This decoder allows us to describe the logical state of the quantum computer at any given moment during the computation, while the r-filter clarifies the concept of a state having "at most r errors."

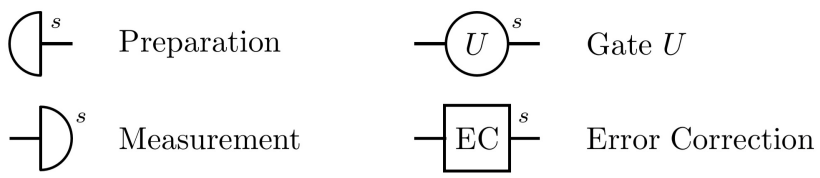
It is convenient to use a graphical notation to represent these objects, as follows: The



horizontal lines represent a single block of a QECC, except for the one on the right end of the ideal decoder symbol, which is a single unencoded qubit. We will focus on the case where the code we use is an [[n, 1, 2t + 1]] code. That is, there is just one encoded qubit per block, the code can correct t errors, and the thick horizontal lines in the diagrams represent n qubits. It is also possible to achieve fault-tolerance with multiple qubits encoded per block , but matters are somewhat more complicated then.

We will require fault-tolerant gadgets for state preparation, measurement, and gates.

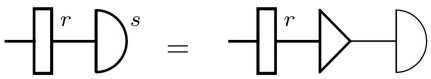
The fault-tolerant "wait" gadget simply involves having all the encoded qubits remain idle. Additionally, we must implement error correction throughout the computation to prevent errors from accumulating to unacceptable levels. Naturally, the error correction step itself must be fault tolerant; otherwise, the process could introduce more errors than it resolves. While high error rates can still lead to problems, a well-designed error correction step gives us a better chance of mitigating errors. We will represent all these



gadgets visually. As previously mentioned, the thick horizontal lines depict a block of an $[[n, 1, 2t+1]]$ quantum error-correcting code (QECC). For an encoded gate, if it involves a two-qubit logical operation, the horizontal lines represent two blocks of the QECC, each corresponding to one logical qubit involved in the operation. The "s" in each diagram indicates the maximum number of faulty locations that may be present in the circuit depicted. For simplicity, we will focus on cases where the errors associated with each fault are Pauli operators. A slight extension of Theorem 2 will allow us to consider other types of errors by analyzing linear combinations of diagrams with specific Pauli errors. If I create a similar diagram using thin lines and without specifying the number of errors, it signifies an idealized, unencoded version of the same operation.

We can now define fault tolerance for these gadgets more precisely. The following definitions will incorporate t, which represents the number of errors the code can correct, as well as the ideal decoder for the code. We only need to ensure that the system behaves correctly when the total number of errors is less than t, as we anticipate that the construction will fail if there are more errors than the code can handle.

Definition 2 (Fault-Tolerant Measurement): A measurement gadget is considered fault tolerant if it meets the following criterion:

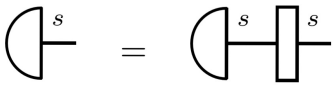
Meas: 

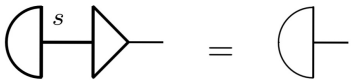
when $r + s \leq t$.

If the total number of errors present in the incoming state and the measurement gadget is at most t, then the outcome from the actual gadget should match the result we would obtain by performing ideal decoding on the incoming state and measuring the decoded qubit. By "the same result," we mean that not only should the probabilities of the various measurement outcomes be identical in both scenarios, but also that the remaining state of the computer, conditioned on the measurement outcome, should remain in the

same relative state in either case. Essentially, we are comparing two operations, each of which transforms the quantum state of the entire computer into a new quantum state representing the computer minus one encoded block, along with a classical measurement outcome. When the measurement gadget is fault tolerant, these two operations will be equivalent.

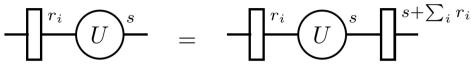
Definition 3 (Fault-Tolerant Preparation): A preparation gadget is fault tolerant if it satisfies the following two properties: That is, a fault-tolerant preparation step with

Prep A:  when $s \leq t$.

Prep B:  when $s \leq t$.

$s \leq t$ errors should output a state that is within s errors of a properly encoded state, and that furthermore, the state should decode to the correct state under an ideal decoder. In the above diagram equation for Prep A, and in many of the equations below, when we have a fault-tolerant gadget on both the left and right side of the equation, assume the faults on both sides are in the same locations and of the same type. The definitions for a fault-tolerant gate are slightly more complicated, but of much the same form:

Definition 4: A gate gadget is fault tolerant if it satisfies the following two properties: In all of these diagrams, a separate filter is applied to each input block when U is a

Gate A:  when $s + \sum_i r_i \leq t$.

Gate B: when $s + \sum_i r_i \leq t$.

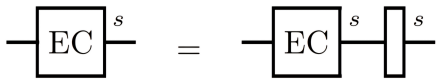
multiple-qubit gate. Input block i gets an r_i-filter. In property Gate A, a separate filter is applied to each output block, but in all cases it is an $s + \sum_i r_i$-filter. In property Gate B, an ideal decoder is applied separately to each block.

Property Gate A asserts that while some error propagation between blocks is permissible (and even expected), errors should not spread within a single block. Consequently, the total number of errors in the outgoing state of each block must not exceed the sum of the errors present in the incoming states and any errors introduced during the gate operation. This condition only needs to be satisfied when the total number of errors is below the threshold t.

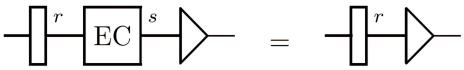
Property Gate B stipulates that if the combined errors in the incoming blocks and the gate gadget remain within acceptable limits, then the fault-tolerant gate gadget must correctly implement the intended encoded gate. In effect, this implies the possibility of forming a commutative diagram involving the ideal decoder, the fault-tolerant (FT) gate

gadget, and the ideal unencoded gate gadget. However, this commutation only needs to hold true when the incoming states have a limited total number of errors.

Finally, we introduce the definition for **fault-tolerant error correction**: **Definition 5:** An error correction (EC) gadget is deemed fault tolerant if it meets the following two criteria:

EC A: 

when $s \leq t$.

EC B: 

when $r + s \leq t$.

In essence, after performing an error correction step with no more than s faulty locations, the resulting state should be at most s errors away from a valid encoded state. It's important to note that this holds true regardless of the number of errors present in the incoming state. While this criterion ensures that the final state is close to a codeword, it does not guarantee that the errors were handled appropriately; the state might correspond to an incorrect codeword, yet it must still be a valid one.

Property EC B states that if the combined total of errors from the incoming state and those incurred during the fault-tolerant error correction (EC) process remains below the threshold t, then the logical state after the correction will match the logical state prior to the error correction step.

4.3 Transversal gates

We now turn our attention to finding constructions that satisfy the definitions of fault tolerance, specifically focusing on gate gadgets. We have previously encountered a method for constructing fault-tolerant gates: Consider an $[[n, k, d]]$ stabilizer code with stabilizer S, where $N(S)/S \cong P_k$ represents the logical Pauli group acting on the k encoded qubits. In the absence of errors, we can execute a logical Z operation by selecting a representative Q from the appropriate coset in $N(S)/S$. Since Q belongs to P_n, performing it entails applying a corresponding Pauli matrix (or the identity) to each of the n physical qubits in the code. This construction inherently satisfies both properties of fault tolerance: **Gate A** and **Gate B**. Property Gate A is upheld because we are implementing single-qubit gates, which prevents errors from spreading to other qubits, even if the number of errors exceeds the threshold t. Property Gate B also holds true, but it is crucial to ensure that the number of errors remains at most t. To formally demonstrate that these properties

are satisfied, the essential observation is that we can rearrange the errors to occur after the logical gate operation. Although moving errors through the Pauli operator Q may alter their identity—due to conjugation by Q—it does not affect their weight, as Q is comprised of a tensor product of single-qubit gates.

Additionally, for many codes, we can identify further gates that can be implemented in a fault-tolerant manner. For example, in the case of the 7-qubit code, it can be verified that applying the Hadamard gate to all 7 qubits (denoted as $H^{\otimes 7}$) effectively performs the logical Hadamard gate H. This method of applying H is inherently fault-tolerant since conjugating an error with H does not alter the weight of the error. Similarly, for the 7-qubit code, one can also perform the $\pi/4$ rotation P fault-tolerantly by executing $P^\dagger$ on each of the 7 physical qubits in the code. If we want to extend our approach to include multiple-qubit gates—which is necessary for achieving universal quantum computation—we can no longer rely on the assumption that conjugating by U will keep the error weight unchanged. However, we can ensure that any increase in the number of errors is distributed across multiple blocks of the code, preventing any single block from becoming overwhelmed.

Specifically, consider a gate that acts on m blocks of the code, constructed as a tensor product $U = \bigotimes U_i$, where U_i operates on the ith qubit of each block. The U_i can perform any operations on the m qubits they affect, and there is no requirement for all U_i to be identical. Such a gate is termed a transversal gate, and it inherently exhibits fault tolerance. When U_i conjugates any number of errors on the m ith qubits, it results in an error that affects only those m qubits. Specifically, U_i will propagate a single error to at most one qubit per block.

Consequently, both properties Gate A and Gate B will also be satisfied for transversal gates. It's important to note that the definition of transversal gates specifies that the ith qubit of one block must only interact with the ith qubit of another block, not with a different qubit in the second block. If this condition were relaxed, the gate could still be fault tolerant if each qubit in one block interacted with only a single qubit in a different block. However, this could lead to issues when combining two such gates. For example, if we have a gate that couples qubit i of block 1 with qubit $j \neq i$ of block 2, and then follow it with a traditional transversal gate, an initial error on qubit i of block 1 could propagate to qubit j of block 2 and subsequently back to qubit j of block 1. This scenario would result in two errors in the first block.

The product of two transversal gates remains transversal and thus fault tolerant, but this would not hold if we modified the definition of transversal gates to allow for permutations of the qubits within a block. To illustrate a transversal two-qubit gate, let's revisit the 7-qubit code. The operation CNOT $\otimes$ 7—a transversal CNOT acting on corresponding qubits of two blocks—realizes the logical CNOT gate. We often refer to this as "transversal U" for the gate $U \otimes n$. Although the term "transversal" does not

necessarily imply that all tensor components are identical, many common examples do exhibit this property, making the phrase a convenient shorthand.

Importantly, the transversal CNOT gate performs the logical CNOT operation for any CSS code, not just the 7-qubit code. This characteristic contributes to the appeal of CSS codes in the context of fault tolerance, which we will explore further in the next section. The 7-qubit code stands out because it supports transversal implementations of the Hadamard (H), the phase gate (P), and the CNOT gate. Since the product of transversal gates remains transversal, it follows that the entire logical Clifford group can also be executed transversally. However, the Clifford group alone does not constitute a universal set for quantum computation—it can be efficiently simulated by classical computers. Unfortunately, the 7-qubit code does not offer any additional transversal logical gates, necessitating alternative constructions to achieve a universal set of logical gates. While one might hope to discover a more efficient code, it is important to note that no code exists that provides a universal set of transversal gates.

4.4 Fault-tolerant error correction and measurement

Before delving into how to achieve a universal set of logical gates, we need to explore another crucial aspect of fault-tolerant protocols: fault-tolerant error correction. To begin, let's clarify the process of non-fault-tolerant error correction.

In the context of an $[[n, k, d]]$ stabilizer code, the goal of error correction is to measure the eigenvalues of each of the $n - k$ generators of the stabilizer. Each eigenvalue corresponds to one bit of the error syndrome. To measure the eigenvalue of a unitary operator U, we can employ a standard technique known as the phase kickback trick (see Figure 4.1).

This technique involves adding a single ancilla qubit initialized in the state $|0\rangle + |1\rangle$ and performing a controlled U operation from the ancilla qubit to the data qubits we want to measure. If the data is in an eigenstate of U with eigenvalue $e^{i\phi}$, the ancilla will remain unentangled with the data and will transition to the state $|0\rangle + e^{i\phi}|1\rangle$. However, if the data is not in an eigenstate of U, the ancilla will become entangled with the data, leading to decoherence of the data in the eigenbasis of U.

In scenarios where the eigenvalues of U are ± 1, which applies when $U \in P_n$, we can measure ϕ by performing a Hadamard transform on the ancilla qubit and then measuring its state.

This construction, however, presents the risk of runaway error propagation, even when U is simply a multiple-qubit Pauli operator. The controlled-U operation is realized through a sequence of controlled-Q_i gates, where $Q_i \in P_1$ acts on the i-th data qubit. Each controlled-Q_i gate has the potential to propagate errors in both directions between the ancilla qubit and the i-th data qubit. As a result, a single error occurring early in

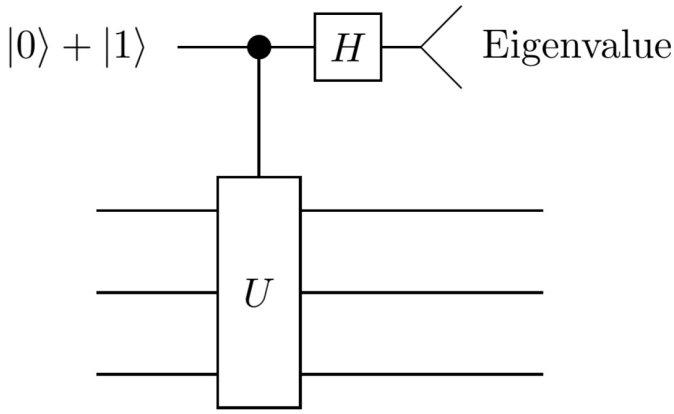


Figure 4.1: Figure 1. A non-fault-tolerant implementation of the measurement of U, which has eigenvalues ± 1

this process can easily spread to multiple qubits within the data block.

To mitigate this issue, we want to distribute the ancilla qubit in such a way that the controlled-U gate behaves more like a transversal gate. Our goal is to ensure that either all the Q_i operators act on the data or none at all. This can be achieved by using an ancilla in the n-qubit "cat" state $|00\ldots0\rangle + |11\ldots1\rangle$, as illustrated in Figure 4.2. After interacting with the data, we need to distinguish between the states $|00\ldots0\rangle \pm |11\ldots1\rangle$.

The simplest method to accomplish this is to recognize that:

$$H^{\otimes n}(|00\ldots0\rangle + |11\ldots1\rangle) = \sum_{\mathrm{wt}(x)\ \mathrm{even}} |x\rangle \tag{4.5}$$

$$H^{\otimes n}(|00\ldots0\rangle - |11\ldots1\rangle) = \sum_{\mathrm{wt}(x)\ \mathrm{odd}} |x\rangle. \tag{4.6}$$

By measuring each qubit of the ancilla in the Hadamard basis and checking the parity, we can extract the eigenvalue of U. Importantly, this approach ensures that the i-th qubit of the data only interacts with the i-th qubit of the ancilla, thereby eliminating the risk of catastrophic error propagation. Errors can only transfer between a single qubit of the ancilla and a single qubit of the data. While I have described the cat state as an n-qubit state for simplicity, it actually needs to be only as large as the weight of U, since any additional qubits in the ancilla will not interact with the data.

However, this construction does not yet provide a fault-tolerant error correction gadget, as there are two outstanding issues. First, we have not yet defined how to create the ancilla cat state. Second, if even a single qubit of the ancilla experiences an error, the measurement result could yield the wrong parity, resulting in an incorrect syndrome bit.

The solution to the second problem, while somewhat tedious, is straightforward: after measuring each bit of the error syndrome to obtain a candidate error syndrome, we can

repeat the measurement process. By repeating the measurements multiple times, and ensuring that the number of errors during this process remains manageable, we can eventually gain confidence in the results. This is because a single faulty location can only lead to one erroneous measurement outcome.

There are additional complexities to consider, as errors may also occur in the data qubits during the correction process, which could alter the actual error syndrome. However, these complications can also be addressed by repeating the measurement sufficiently. Creating a cat state requires careful consideration, as the straightforward circuits for do-

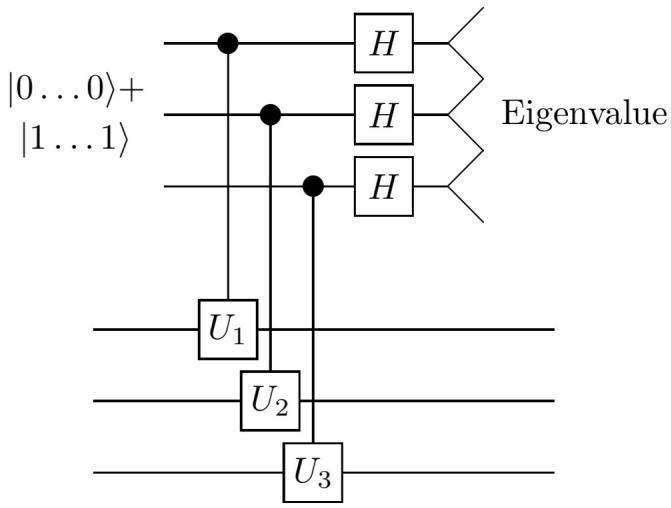


Figure 4.2: A component of a fault-tolerant implementatin of the measurement of $U = \bigotimes U_i$, which has eigenvalues ± 1

ing so are not fault-tolerant. For example, one might initialize a single qubit in the state $|0\rangle + |1\rangle$ and then perform CNOT gates to entangle it with the other $n-1$ qubits, all initially in the $|0\rangle$ state. However, if an error occurs midway through this process, the resulting state could be something like $|0011\rangle + |1100\rangle$, which effectively contains two bit flip errors. When interacting with the data, these two errors would propagate into the data block, leading to a state with two errors.

To prevent this, we must verify the cat state after its creation. One way to accomplish this is to take pairs of qubits from the cat state and perform CNOT operations on them, targeting an additional ancilla qubit initialized to $|0\rangle$. After this interaction, we measure the additional ancilla. If it measures $|1\rangle$, we know that the two qubits tested are different, indicating a problem, and we can discard the cat state and try again.

Although this verification process itself is not transversal, it allows for fault-tolerant verification due to the nature of error propagation in a CNOT operation. Since the ancilla qubit is the target for both CNOT operations, only phase errors can propagate from it into the cat state. A single phase error can already compromise our cat state, leading to an incorrect syndrome bit as an outcome, which is why repeated measurements are

required. By performing enough verification steps, we can confidently conclude that either the ancilla cat state is correct, potentially with s errors, or that there were more than s faulty locations during the preparation and verification of the cat state. This assurance is sufficient to ensure that the overall error correction procedure is fault-tolerant.

In summary, we must first create many cat states. We do this via some non-fault-tolerant procedure followed by verifying pairs of qubits within the cat state to see if they are the same. We use each cat state to measure one bit of the error syndrome, and repeat the measurement of the full syndrome a number of times. In the end, we deduce a consensus error syndrome and from it an error which we believe occurred on the data, and then correct that error. The above procedure is known as Shor error correction, and it works for any stabilizer code. A similar procedure can be used for measuring a logical Pauli operation on a stabilizer code, although in that case, we must be careful to perform error correction before repeating our measurement attempt, as an uncorrected error on the data can cause the measurement outcome to be incorrect even if the measurement procedure itself is completely free of errors. I have been somewhat cavalier about some of the details above partially because they are complicated and partly because Shor error correction is rather laborious, requiring many extra gates and a low physical error rate to work well. There are some much more efficient schemes for fault-tolerant error correction available, so Shor error correction is rarely used.

In the context of fault-tolerant measurement using CSS codes, the presence of faults in the physical measurement locations leads to a measurement outcome of $u+w+e+f$. Here, e denotes any pre-existing bit flip errors that were present in the quantum state before measurement, while f represents errors introduced by faulty measurement locations.

The crucial point is that the measurement is performed transversally, which means that each measurement interacts with only a single qubit of the encoded state. Therefore, even if there are multiple faulty measurement locations, each failure will only affect one bit of the output. This ensures that the weight of the errors introduced by faulty measurements satisfies condition $\mathrm{wt}(f) \leq s$, where s is the number of faulty measurement locations.

If the total weight of the combined errors $e + f$ is such that $\mathrm{wt}(e + f) \leq t_1$ (where t_1 is the error correction capability of the classical code C_1), we can successfully deduce the values of u, w, and $e + f$. However, it is important to note that we cannot distinguish between e and f individually. While this lack of distinction is somewhat inconvenient, it is not particularly detrimental if our only goal is to measure the state.

In this scenario, the value of u effectively gives us the outcome of the measurement, and as long as we can ascertain this result, the overall measurement procedure remains fault-tolerant. Therefore, even with the potential for errors introduced by faulty measurement locations, the system is designed to recover meaningful measurement outcomes, making it a reliable approach for fault-tolerant quantum computation within the framework of

CSS codes.

We have not yet achieved fault-tolerant error correction. While directly measuring the qubits reveals bit flip errors, it does so at the expense of damaging the code block, which is undesirable. We also need to identify any phase flip errors present in the state. To accomplish this, we will again use ancilla qubits. As previously mentioned, transversal CNOT performs the logical CNOT operation for any CSS code. Thus, we should create an ancilla block in a codeword of the CSS code we are employing. Next, we apply a transversal CNOT from the data block to the ancilla block. This ensures that both the data and ancilla blocks remain in valid codeword states. Any bit flip errors from the data block will be transferred to the corresponding locations in the ancilla block through the CNOT operation. After measuring each qubit in the ancilla block, we obtain a value of $e + f$, which represents a combination of bit flip errors in the data block, existing bit flip errors in the ancilla, and errors that occurred during the transversal CNOT or measurement. Although there are several additional errors to consider, we have gained some insight into the errors affecting the data block.

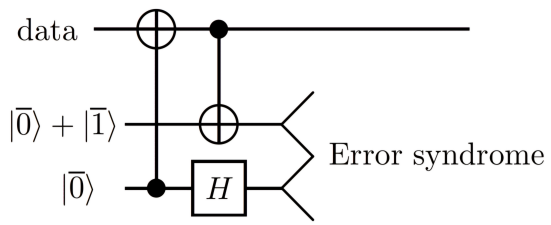


Figure 4.3: Steane Error Correction. Each horizontal line represents a full n-qubit block of the code, and each gate or measurement represents a transversal implementation of that operation.

We need to be cautious with this approach. Although we use a transversal CNOT to transfer errors from the data block to the ancilla, we don't actually want to apply a logical CNOT gate, as error correction should leave the encoded data itself unchanged. To ensure this, we prepare the ancilla block not in just any codeword state, but in the encoded state $|0\rangle + |1\rangle$, an eigenstate of CNOT when used as the target. This prevents the encoded data's state from altering during the CNOT operation. Consequently, measuring u from the ancilla reveals no information about the data's encoded state, as u is random, like w. We must also watch for error propagation: while bit flip errors are transferred from the data block to the ancilla, phase errors move from the ancilla back to the data block. Therefore, it's essential to ensure the ancilla creation process does not introduce excessive errors. Since the ancilla is simply an encoded $|0\rangle + |1\rangle$ state, creating this state requires the same fault-tolerant preparation method as other cases. For now, assume we have a method that, if the encoding circuit has at most s faulty locations, results in a state with no more than s errors. Further details on this preparation will be covered in

section 4.6.

To correct bit flip errors in a CSS code, we start by creating an ancilla block encoded in the same code in the state $|0\rangle + |1\rangle$ and apply a transversal CNOT from the data block to this ancilla block. Afterward, we measure all ancilla qubits and interpret the measurement result as a classical codeword for C_1. Using this result, we identify the error locations and correct them in the data block. For phase error correction, we follow a similar approach: we prepare an ancilla block in the $|0\rangle$ state and perform a transversal CNOT with the ancilla as control and the data block as target. This copies phase errors from the data block to the ancilla, allowing us to measure each ancilla qubit in the Hadamard basis. In this basis, the code is represented as a superposition of codewords from C_2, so we treat the measurement as a classical codeword for C_2 with some errors, which we locate and correct using the classical decoding procedure for C_2.

This entire process is summarized in figure 4.3. Next, we verify that Steane error correction (EC) meets properties EC A and EC B. Assume the incoming data block has errors indicated by a vector e, where e consists of e_P (phase errors) and e_B (bit flip errors). A "1" in the i-th position of these vectors marks an error on the i-th qubit. If both e_P and e_B have a "1" in the same location, a Y-error exists there. By linearity (as stated in Theorem 2), we assume all errors in the incoming block are Pauli errors.

Let's consider the error setup in this scenario. Suppose, after preparing the phase ancilla, there are errors in its qubits indicated by a binary vector f_P, and similarly, errors in the bit flip ancilla are indicated by a binary vector f_B. We denote the faults occurring in the CNOT gates by g_P and g_B. It's important to note that any faulty CNOT gate could affect both the data qubit and the corresponding ancilla qubit, though not necessarily in the same way. Additionally, suppose that faults occurring during the Hadamard rotation and measurement on the phase ancilla are described by h_P, and faults in the measurement of the bit flip ancilla are indicated by h_B. For h_P and h_B, only errors that affect the measurement result are counted, as they cannot otherwise propagate back to the data block.

For errors outside the incoming data block, no restrictions are placed on the type of error, and we exclude errors from the final correction step, where any corrections based on syndrome measurements are made transversally. Although errors can occur at this step, each one only affects a single qubit, preventing any failure of EC A or EC B.

To prove EC A, assume that the combined weight $s = \mathrm{wt}(f_P) + \mathrm{wt}(f_B) + \mathrm{wt}(g_P) + \mathrm{wt}(g_B) + \mathrm{wt}(h_P) + \mathrm{wt}(h_B) \leq t$. We aim to demonstrate that, after correcting the detected error, the resulting state will be no more than s errors away from a valid codeword. The claim is as follows:

Claim 1: *After error correction, any errors left in the final state will only be located in positions corresponding to errors in the ancilla or data block that occurred during the error correction (EC) circuit. Specifically, errors will only be present at qubit locations*

represented by the bitwise OR of the vectors f_P, f_B, g_P, g_B, h_P, and h_B, and there will be no more than s of these.

Proof. Let's analyze the sources of errors. In the proof, subsets of f_x, g_x, and h_x (where $x = P$ for phase and $x = B$ for bit flip) will be denoted with superscripts, such as $f_P^{(1)}$, $g_P^{(1)}$, etc., without particular meaning for the indices, which just label these subsets.

When correcting phase errors, the observed phase error will be a combination $e_P + f_P^{(1)} + g_P^{(1)} + h_P^{(1)}$. Since there's no restriction on $\mathrm{wt}(e_P)$, this could result in more errors than can be corrected by the phase code C_2. If that happens, the classical decoding of C_2 may deduce an error d different from the actual error combination but having the same syndrome as $e_P + f_P^{(1)} + g_P^{(1)} + h_P^{(1)}$. Consequently, $v = d + e_P + f_P^{(1)} + g_P^{(1)} + h_P^{(1)} \in C_2$. Because C_2's X-type generators are in $C_2^{\perp}$, the operator Z_v, which applies Z operators at qubits indicated by $v \in C_2$, commutes with every X generator. Thus, Z_v is in the normalizer $N(S)$ and maps one logical codeword to another.

Before any error correction, the data block, after interacting with the ancilla blocks, will have phase errors in positions described by $e_P + g_P^{(2)} + g_B^{(2)} + f_B^{(2)}$. When we apply phase error correction, it targets qubits in positions given by d, yielding a net phase error of $d + e_P + g_P^{(2)} + g_B^{(2)} + f_B^{(2)}$. Given that $d + e_P = v + f_P^{(1)} + g_P^{(1)} + h_P^{(1)}$, we can rewrite the net phase error as $v + f_P + f_B + g_P^{(1)} + g_P^{(2)} + g_B^{(2)} + h_P^{(1)}$.

Since $Z_v \in N(S)$, applying Z-type errors at locations defined by v results in a different logical codeword without impacting correctness. Thus, the final state post-correction is equivalent to a valid codeword with phase errors indicated by $f_P^{(1)} + f_B^{(2)} + g_P^{(1)} + g_P^{(2)} + g_B^{(2)} + h_P^{(1)}$, which is a subset of $f_P \vee f_B \vee g_P \vee g_B \vee h_P \vee h_B$. A similar approach shows that bit flip errors in the final state are also restricted to this subset. □

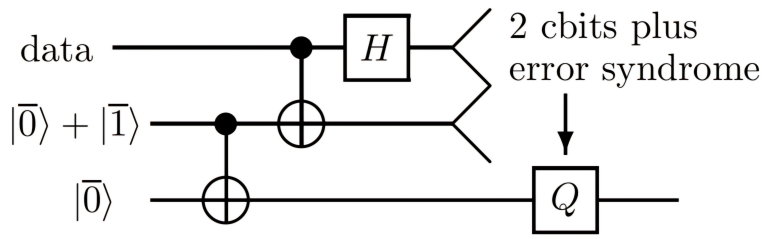


Figure 4.4: Knill EC. Each horizontal line represents a full n-qubit block of the code, and each gate or measurement represents a transversal implementation of that operation. Q is a Pauli operator and contains a correction for both the teleportation outcome and error correction

Proving property EC B becomes more straightforward with the assumption that the total error weight, $s + \mathrm{wt}(e)$, is at most t. Under this assumption, final errors will still only affect qubits in positions covered by the combined vectors $f_P \vee f_B \vee g_P \vee g_B \vee h_P \vee h_B$. Since there is no shift by an element of the normalizer $N(S)$, an ideal decoder will yield

identical results whether applied before or after the error correction (EC) step. Thus, the initial error e can be recovered directly without alteration.

Knill Error Correction (Knill EC) offers an alternative method of fault-tolerant error correction. Similar to Steane's approach, Knill EC also relies on an ancilla state encoded with the same code as the data block, but it is applicable to any stabilizer code. Knill's method utilizes quantum teleportation to transfer the encoded data state to a new code block. During teleportation, the Bell measurement extracts more than just the necessary teleportation data; it also provides the error syndrome for the combined errors on the data and ancilla blocks, enabling error correction.

As depicted in Figure 4.4, Knill EC involves two blocks measured in the teleportation process: one to reconstruct the X-type stabilizers and the other to reconstruct the Z-type stabilizers. For a complete explanation of the mechanism behind Knill EC.

Knill EC can also be adapted for fault-tolerant measurement. By replacing the encoded EPR pair with an ancilla in the state $|0\rangle$ in the error correction circuit, the encoded Bell measurement provides not only the error syndrome but also the outcome of a logical measurement on the data block in the standard basis, rather than yielding two random bits as in traditional teleportation.

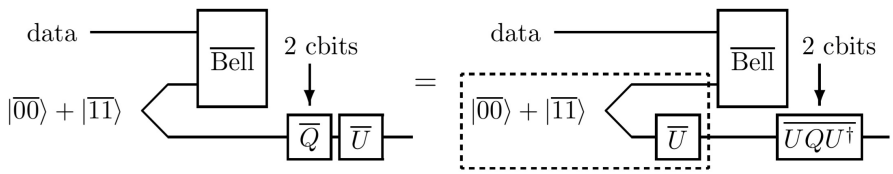


Figure 4.5: Gate teleportation of $\bar{U}$. The process of teleporting the state followed by $\bar{U}$ is the same as teleporting the state through a special ancilla with an appropriately modified correction operation

Both **Steane** and **Knill error correction (EC)** methods are based on shifting most of the error-correction workload to the preparation of a specific ancilla state. This approach has two key benefits, First, it reduces the operations needed directly on the data qubits, so they aren't held in place while EC is applied, minimizing their exposure to additional errors. Second, because the ancilla states are prepared in known configurations, they can be thoroughly verified before use, a level of scrutiny that isn't feasible for the unknown state of the data qubits.

4.5 Gate teleportation and universal fault-tolerant quantum computation

Gate teleportation and universal fault-tolerant quantum computation aim to achieve a universal set of fault-tolerant gates, which is not possible to implement transversally.

Therefore, an additional component—measurement—is required. The overall approach resembles the Steane or Knill error correction methods, where significant effort is focused on creating a specific ancilla state, which then briefly interacts with the data to effectively implement the desired gate. This technique allows for certain non-Clifford group gates to be performed, and when combined with Clifford group gates, it provides a universal gate set.

The process can be explained using quantum teleportation on encoded states, beginning with an idealized scenario that ignores fault tolerance. To implement a gate U on an encoded state $|\psi>$, consider performing quantum teleportation followed by U. This would result in the output state $U|\psi>$. Now, imagine that Bob, responsible for the output block, performs U prematurely, before receiving Alice's measurement outcome. When Alice later completes the logical Bell measurement, she sends Bob two classical bits that correspond to a Pauli operator Q. To correctly finalize the teleportation, Bob now needs to undo his initial U, apply Q, and then reapply U, resulting in the gate $UQU^\dagger$.

This technique might not initially seem beneficial, but certain gates U make it useful. The state $(I \otimes U)(|00> + |11>)$ can act as a specialized ancilla state, essentially replacing the standard EPR pair in teleportation. Because it's a fixed ancilla state, independent of the data, special methods can be applied to prepare it efficiently.

To perform a gate like $UQU^\dagger$, we can't apply it in advance on the ancilla state because Q depends on the result of a logical Bell measurement on the data block. However, $UQU^\dagger$ is often simpler than the original gate U. For example, when $U \in C_1$ (the Clifford group), $UQU^\dagger \in P_1$ (the Pauli group), which is a defining property of the Clifford group. Moreover, for some non-Clifford gates, like the $R_{\pi/8}$ rotation, we also get that $UQU^\dagger$ remains within the Clifford group:

For instance,

$$R_{\pi/8} X R_{\pi/8}^\dagger = \begin{pmatrix} 0 & e^{-i\pi/4} \\ e^{-i\pi/4} & 0 \end{pmatrix} = e^{i\pi/4} X P^\dagger \tag{4.7}$$

$$R_{\pi/8} Z R_{\pi/8}^\dagger = Z. \tag{4.8}$$

Operators that conjugate Pauli operators into Clifford operators form a set called C_3. Generally, we define sets C_k where $C_k = \{U | UQU^\dagger \in C_{k-1} \, \forall Q \in C_1\}$. The teleportation construction shows how, with suitable ancillas, we can perform a C_k gate if we have access to gates in C_{k-1}. However, C_k is not closed under multiplication.

This framework suggests a path to achieving a universal fault-tolerant gate set. For example, the 7-qubit code supports logical Clifford group operations. Since a Bell measurement is a Clifford group operation (along with measurement, which can also be performed fault-tolerantly), and we know that $R_{\pi/8} Q R_{\pi/8}^\dagger$ is a Clifford operation for $Q \in P$, we can implement a fault-tolerant $R_{\pi/8}$ gate if we prepare the ancilla state $(I \otimes R_{\pi/8})(|00> + |11>)$ fault-tolerantly. This, combined with the Clifford group gates,

provides a universal fault-tolerant gate set for the 7-qubit code.

The universal set of gates in fault-tolerant quantum computation, despite being finite for a fixed number of logical qubits, generates an infinite group because the gates do not commute. Although this group is infinite, it doesn't encompass all unitary transformations (as the unitary group is uncountable). However, we can approximate any unitary gate to an arbitrary precision using this universal set, leveraging the Solovay-Kitaev theorem. This theorem states that to approximate a gate U to within a distance ϵ, we need only a number of gates scaling as polylog$(1/\epsilon)$, which is efficient.

The method described for fault-tolerant single-qubit gates using encoded EPR pairs can be extended to multi-qubit gates. For example, a fault-tolerant CNOT gate can be implemented with a four-qubit ancilla. Stabilizer codes with stabilizer S allow us to perform logical Clifford group operations using Knill error correction and the Bell measurement, as $N(S)$ provides transversal implementations of the logical Pauli group. Thus, we achieve fault-tolerant implementations for all Clifford gates through gate teleportation with an appropriate ancilla state. From there, using gate teleportation and the Clifford group plus the $R_{\pi/8}$ gate, we get a universal gate set for any stabilizer code.

Additionally, some gates permit simplified constructions. For example, the $R_{\pi/8}$ gate, a diagonal gate, can be implemented with a more efficient setup that only requires a single encoded block as an ancilla, as shown in figure 4.6. This streamlined approach is particularly beneficial, as it reduces the complexity of ancilla preparation.

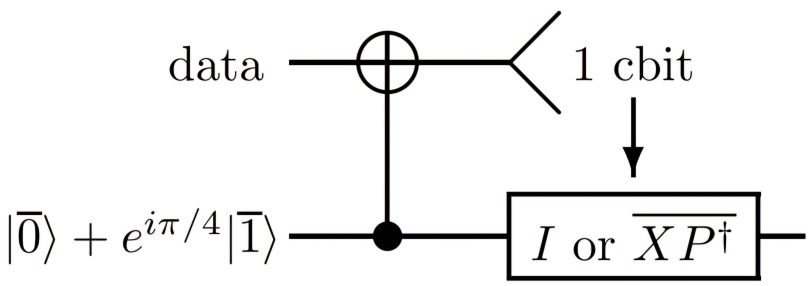


Figure 4.6: More efficient gate teleportation of $\bar{R}_{\pi/8}$. The CNOT represents a logical $C\bar{N}OT$, performed transversally.

4.6 Fault-tolerant state preparation

The methods discussed in the previous sections depend on the capability to prepare specific states in a fault-tolerant manner. While it may appear that we've merely delayed the challenges associated with executing non-Clifford group gates and ensuring fault-tolerant error correction, this is somewhat accurate. However, because the ancillas in these processes are prepared in predetermined states, we have greater flexibility in how to generate

them compared to performing operations, like the $\pi/8$ rotation, on an unknown data state. To implement Steane or Knill error correction, we need to accurately create encoded $|0\rangle$ and encoded $|0\rangle + |1\rangle$ states with minimal errors. Additionally, we require encoded $|0\rangle$ states to initialize fault-tolerant quantum computations, with a specific focus on avoiding correlated errors in these states. While it is possible to generate the desired states using non-fault-tolerant encoding circuits, these circuits involve interactions between pairs of qubits, meaning that a single error during encoding could lead to correlated errors across multiple qubits in the new block. This would likely violate property **Prep A** and could also compromise **Prep B**. One effective approach for creating these states is through a variant of Shor error correction. By introducing a stabilizer generator that corresponds to Z (for $|0\rangle$) or X (for $|0\rangle + |1\rangle$), we define a code with only one state specified by the new stabilizer, which is the state we aim to create. Utilizing Shor error correction to adjust any arbitrary state to this unique codeword results in a fault-tolerant state preparation method.

Steane and Knill error correction methods do not directly provide state preparation techniques, as they require ancillas to be prepared for the specific state we want to create. However, they do guide us in the right direction. If we were to apply Steane error correction to the created state without first verifying the ancilla states, the outcome would be unreliable due to the potential for multiple correlated errors arising from a single error during the encoding process. Instead, if we modify our approach and use the error detection procedure without correcting based on the error syndrome, we can identify the presence of errors. In this case, if any errors are detected, we would discard both the main state we aim to create and the measured ancillas. For a correlated error to go unnoticed by this method, it would have to occur in both the data block we are verifying and one or both of the ancilla blocks, and those errors would need to cancel out in the error syndrome. Since the data and ancilla blocks are prepared independently, this scenario would require two separate faults: one during the creation of the data block and another during the creation of the ancilla block. Although we might need to repeat this procedure using ancillas that have already passed prior screenings, with enough iterations, we can achieve fault-tolerant state preparation even for codes with large distances.

Chapter 5

Fault-Tolerant Circuits and the Threshold Theorem

5.1 Good and bad extended rectangles

We have successfully demonstrated the existence of gadgets for fault-tolerant state preparation, measurement, error correction, and a universal set of gates. With this foundation, we can now explore the implications of applying these gadgets. I will first establish that, as long as errors do not occur too frequently, we can assemble a series of fault-tolerant gadgets to construct a circuit that yields the same output as an ideal, unencoded circuit.

Definition 6: Let C represent a quantum circuit composed of various locations C_i, where each C_i can be a preparation location, a measurement location, a gate location, or a wait location. Preparation locations introduce new qubits into the circuit, while measurement locations remove qubits from it. We assume that C can be organized into time steps, such that at each time step, every qubit (excluding those to be added in later time steps or removed in earlier ones) is involved in exactly one location. The initial location for each qubit must be a preparation location, and the final location for every qubit must be a measurement location. Two locations are considered consecutive if they occur at adjacent time steps and share a qubit (if both are two-qubit gates, they only need to share one qubit).

The fault-tolerant protocol for C (using a specific $[[n, 1, 2t + 1]]$ quantum error correction code) results in a quantum circuit C' that is constructed by substituting each location C_i with its corresponding fault-tolerant gadget and adding fault-tolerant error correction gadgets between any two consecutive locations. C' is termed a fault-tolerant circuit for C or a fault-tolerant simulation of C. However, proving properties directly about the entire circuit can be cumbersome. Thus, it is more practical to concentrate on smaller units. We will break down the fault-tolerant circuit into segments that contain only a single location from the original circuit C.

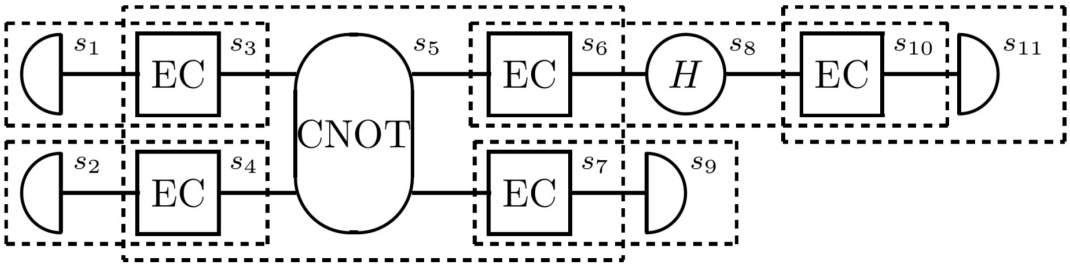


Figure 5.1: A schematic representation of a sample fault-tolerant protocol, with extended rectangles marked.

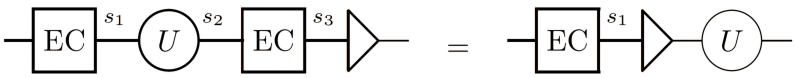
Definition 7: An extended rectangle (abbreviated as ExRec) in the fault-tolerant circuit for C consists of the fault-tolerant gadget corresponding to a specific location C_i in the original circuit, along with all the fault-tolerant error correction (EC) gadgets that are situated between C_i and its adjacent locations. An ExRec can be characterized by the type of the location C_i; for instance, if C_i is a preparation location, the corresponding extended rectangle is termed a preparation ExRec. The EC steps that occur before the gadget for C_i are referred to as leading EC steps, while those following the gadget for C_i are termed trailing EC steps. Figure 5.1 illustrates an example of a fault-tolerant circuit, highlighting the ExRecs with dashed lines.

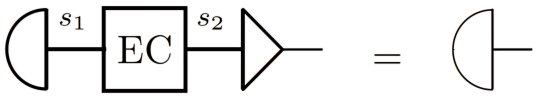
It is important to note that a preparation ExRec or a measurement ExRec contains only a single EC step, while a gate (or wait) ExRec encompasses two EC steps for each qubit involved in the gate. All ExRecs, except for a preparation ExRec, begin with EC steps. Additionally, adjacent extended rectangles overlap, which complicates matters; however, this is easier to manage than having non-overlapping rectangles. The reason for this is that we want to establish a notion that a rectangle "behaves correctly" if it does not have too many faulty locations. If the rectangles do not overlap, there could be an accumulation of errors at the end of one rectangle and at the beginning of the next. Since there is no time available to correct errors between these two rectangles, the errors would combine, leading to a greater total error count than might be anticipated by merely counting the faulty locations within one rectangle. Although this situation can be addressed, it means that analyzing the behavior of a single rectangle becomes dependent on the behavior of the preceding rectangle, preventing us from fully leveraging the quantum error correction code's (QECC) capacity to correct errors.

To say that an extended rectangle (ExRec) behaves correctly means that the encoded state after the ExRec matches the encoded state before it, with the appropriate logical operation applied to the state. However, there is a complication: errors may occur just before the start of an ExRec and then additional errors may arise at the beginning of the ExRec. This situation could result in the state before the ExRec being correctable,

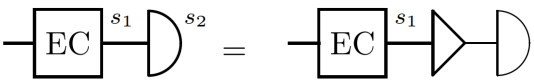
but once the ExRec begins, before error correction has a chance to take effect, it may no longer be correctable. To address this issue, instead of assessing the state prior to the ExRec, we should evaluate the state after the leading EC steps of the ExRec but before the main gadget for the location C_i. At this stage, any errors from the previous rectangle will have either been corrected or combined with errors that occurred during the leading EC steps. Thus, we can establish the following definitions:

Definition 8: A gate or wait ExRec is considered correct for specific fault arrangements if A preparation ExRec is *correct* for some particular arrangements of faults if A





measurement ExRec is *correct* for some particular arrangement of faults if We expect

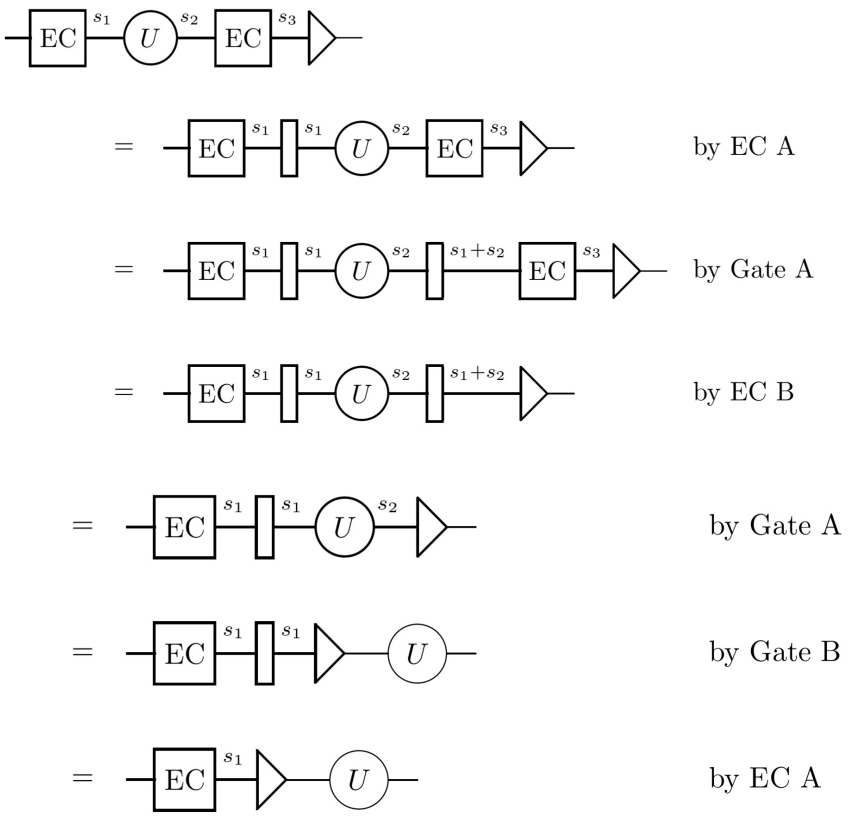


that an ExRec should be correct if it contains no more errors than the code can correct. Thus, we make the following definition:

Definition 9: A full ExRec is termed good if it contains no more than t faults, where the quantum error correction code (QECC) in use is designed to correct t errors. Conversely, a full ExRec is classified as bad if it does not meet this criterion. The term "full ExRec" is used for a technical reason that will be explained later, but for now, it can be set aside. The rationale for defining a good ExRec is supported by the following theorem:

Theorem 6 (Good implies correct): A good ExRec is correct.

Proof: The proof follows directly from the definitions of fault-tolerant gadgets. I will illustrate this for a gate ExRec, noting that measurement and preparation ExRecs are similar in nature. Given that the ExRec is good, in the following diagrams, we have the relationship $s_1 + s_2 + s_3 \leq t$.



□

5.2 Level reduction

Now, let's examine a complete fault-tolerant circuit. If all the extended rectangles (ExRecs) in the circuit are classified as good, we can apply Theorem 6 repeatedly. This allows us to construct ideal decoders based on the correctness property of the measurement locations. We can then push these ideal decoders to the left, utilizing the correctness of the gate and wait locations, and finally eliminate them using the correctness property of the preparation locations. As a result, we obtain the original circuit C, which the fault-tolerant circuit aims to simulate.

Theorem 7: If a fault-tolerant circuit for C consists solely of good extended rectangles, then the output distribution of the fault-tolerant protocol matches that of C.

This is the outcome we desire; however, it only holds true when all extended rectangles are good. As we will discuss more concretely about error models shortly, this result indicates that a fault-tolerant protocol indeed provides protection against errors. Specifically, for sufficiently low error rates, the probability of having only good ExRecs in the complete fault-tolerant circuit is greater than the probability of executing the original circuit C without any errors.

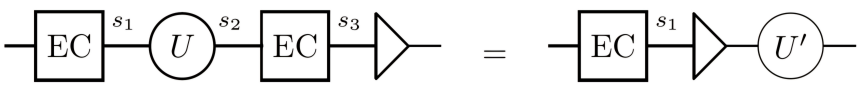
Unfortunately, in large circuits C, it remains unlikely that we can traverse a fault-tolerant circuit entirely without encountering a bad extended rectangle, unless we employ codes capable of correcting many errors. However, if we opt for codes with a large t, we

must select a specific family of codes, as otherwise, generating the large ancillas required for fault-tolerant gate operations and error correction becomes challenging. Fortunately, two families of codes exist that address this issue, allowing for a threshold theorem. One such family is based on topological constructions. we will focus on the other family, which is somewhat easier to analyze: concatenated codes.

The concept of a concatenated code is quite intuitive. We have established (pending clarification on the appropriate error model) that we can take any circuit C and construct a fault-tolerant circuit C' that yields the same result but is more reliable than C. This raises the question: why not take C' and create another fault-tolerant circuit C'' that simulates it? This should, in theory, enhance reliability even further. If this is insufficient, we can repeat the process multiple times until the circuit achieves sufficient robustness to produce accurate results, even if the original circuit C is large.

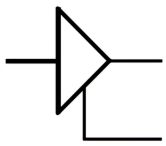
Indeed, this approach is effective. We can generate a sequence of circuits $C(k)$ for $k = 1, \ldots, L$, where each $C(k)$ serves as a fault-tolerant simulation of $C(k-1)$, starting with $C(0) = C$. Although it is unlikely that $C(L)$ contains only good extended rectangles when C is large, it will be true that $C(L)$ simulates $C(L-1)$, which, with high probability (assuming the error rate is sufficiently low), contains fewer bad extended rectangles than $C(L)$. This pattern continues until we reach $C(1)$, which, with very high probability, will contain only good extended rectangles, thereby providing a nearly perfect simulation of C.

To prove this, we must first comprehend the behavior of a bad extended rectangle when we push an ideal decoder through it from right to left. Our goal is to establish that a bad extended rectangle simulates a faulty location in the original circuit, which is nearly accurate. Specifically, we want the following to hold true when $s_1 + s_2 + s_3 > t$ for some erroneous gate U': Unfortunately, making such a statement is complex. The issue



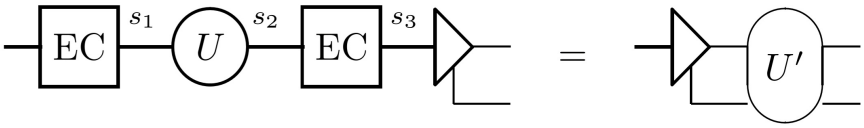
arises because the error occurring on the physical gate after the ideal decoder can depend not only on the faults present in the gate gadget and the trailing error correction (EC) but also on the faults in the leading EC and even the carryover error from the preceding extended rectangle. This interdependence complicates our ability to define U' without considering the broader context.

To address this challenge, we introduce an alternative ideal decoder, denoted as the *-decoder, which tracks the error syndrome. The information regarding the error syndrome of the state input on the left is contained in a line that drops out from the bottom of the *-decoder. As a result, the *-decoder can be treated as a unitary operation. To replace



*-Decoder

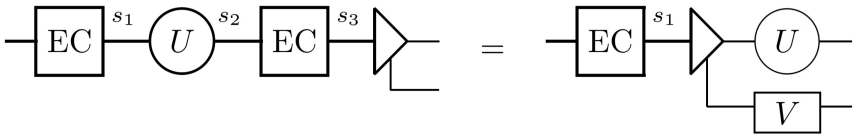
the previous representation, we can express it as follows: The erroneous gate U' utilizes



the error syndrome as a control mechanism to determine the specific error to apply to the decoded qubit. Importantly, we have managed to push the *-decoder all the way through the extended rectangle, effectively eliminating both the leading EC and the gate gadget, as well as the trailing EC. This step is crucial to ensure that bad extended rectangles do not overlap, as such overlaps could introduce correlated errors in the simulated circuit. we will return to this important point shortly.

Correctness for *-decoders is defined in a manner similar to that of regular ideal decoders, with the primary distinction being the need to consider the behavior of the error syndrome. When we move a *-decoder through a gadget that has no faults, the resulting error syndrome will typically change to reflect the operations performed on the erroneous physical qubits. For example, executing a transversal Hadamard will convert pre-existing physical X errors into physical Z errors, leading to a different error syndrome.

As a result, we can define correctness for *-decoders using the same diagrams employed for regular ideal decoders, but we must adjust for the error syndrome in an unspecified manner. Specifically, a gate extended rectangle (ExRec) is considered correct for *-decoders if it satisfies:



where V represents some operation related to the error syndrome, which depends on the errors present in the original ExRec as well as the gate U. The definitions for preparation and measurement ExRecs follow a similar pattern; for the preparation ExRec, the error syndrome is introduced as a separate input on the right-hand side of the definition,

while for the measurement ExRec, the error syndrome is discarded immediately after being generated.

From this, we derive the following lemma:

Lemma 2. *If an ExRec is correct for an ideal decoder, it is also correct for a *-decoder.*

Proof. It is important to note that the ideal decoder is simply the *-decoder with the error syndrome discarded. The correctness definition for an ideal decoder must hold true for all input states, including superpositions and components of entangled states. Therefore, if the error syndrome output from the *-decoder were to interact further with the data block after the *-decoder's operation, it would contradict this definition. Thus, the lemma is established. □

We have established that good extended rectangles (ExRecs) are correct for *-decoders, which sets the stage for us to refine Theorem 7 by considering both good and bad ExRecs. However, a further complication arises when we want to replace ideal decoders with *-decoders in the proof of Theorem 7 and push these *-decoders through the entire circuit. The crux of the issue lies in the fact that if we push a *-decoder through a bad ExRec, the preceding ExRec will lose its trailing error correction (EC) step, thereby ceasing to qualify as an ExRec. This brings us to the following definition:

Definition 10. An ExRec that is missing one or more trailing EC steps (which is particularly relevant in the case of multiple-qubit gate ExRecs) is termed a truncated ExRec. A truncated ExRec is considered good if it contains at most t faulty locations (where t is the error correction capability of the quantum error-correcting code (QECC)) and bad if it exceeds this threshold. To determine whether an ExRec within a larger circuit should be classified as good or bad, as well as whether it is full or truncated, we can follow this procedure:

1. Start from the End: Begin the evaluation from the output of the circuit, focusing first on the measurement ExRecs. Assess whether they are good or bad based on their fault counts.

2. Truncation Process: For each subsequent ExRec in the circuit, assess if it should be truncated. An ExRec should be truncated by removing a trailing EC step if that particular EC step is involved as a leading EC step in a bad ExRec (whether that bad ExRec is full or truncated).

3. Assess Good or Bad Status: Once the truncation process has been applied, determine whether each ExRec is good or bad based on its remaining faulty locations.

4. Recursive Analysis: Continue this assessment recursively throughout the circuit to understand the status of all ExRecs, enabling a comprehensive view of their integrity.

By following this systematic approach, we can effectively classify all ExRecs in the fault-tolerant circuit, taking into account the implications of bad ExRecs and their in-

fluence on the overall correctness of the circuit. Defining correctness for both standard ideal decoders and *-decoders in the context of truncated extended rectangles (ExRecs) is straightforward; it simply involves omitting the trailing error correction (EC) step from each diagram.

Lemma 3: *A good truncated ExRec is valid for both ideal decoders and *-decoders.*

Proof. Since the ideal decoder is designed to provide perfect error correction, we can insert an EC gadget with zero errors before any ideal decoder without affecting the output. (This cannot be done before a *-decoder because the perfect EC gadget would eliminate the error syndrome.) Thus, the correctness diagram for ideal decoders remains unchanged for both full and truncated ExRecs, indicating that a good truncated ExRec is correct for ideal decoders. Furthermore, using Lemma 2—which also applies to truncated ExRecs—we conclude that a good truncated ExRec is correct for *-decoders as well. □

This leads us to an enhancement of Theorem 7:

Theorem 8: Suppose we have a fault-tolerant circuit C' for C. By classifying the extended rectangles in C' as good or bad, we can create a new circuit $\tilde{C}$ in the following way: If the ExRec for C_i is good, we keep C_i unchanged in $\tilde{C}$. If the ExRec for C_i is bad, we replace C_i with the erroneous gate U' from Representation(5.2). The circuit $\tilde{C}$ employs ancilla registers to manage the types of U' errors. Consequently, the output distribution of C' matches that of $\tilde{C}$.

In other words, C' simulates a version of C with errors substituting for the bad extended rectangles. We prove this theorem similarly to Theorem 7, by establishing *-decoders at the right end of the circuit and pushing them left using the correctness conditions and Representation(5.2). The ancilla registers that determine the types of errors are the error syndrome registers produced by the *-decoders.

Now, we can discuss a specific error model and demonstrate that fault tolerance reduces the error rate.

Definition 11: An uncorrelated error model posits that each location C_i is faulty with probability p_i, independently across different locations. If there is an error at location C_i, it is selected from a distribution E_i that is independent of other locations. In an uncorrelated Pauli error model, if a location is faulty, there is a probability p_{Q_i} of a Pauli error Q occurring on the qubit(s) associated with that location (compared to the correct operation). Typically, we assume that p_i and p_{Q_i} depend solely on the type of location C_i, without regard to other factors, and often assume $p_i = p$ for all i.

An uncorrelated depolarizing error model is a specific instance of the uncorrelated Pauli error model where p_{Q_i} does not vary based on Q (except for $Q = I$, where no error may have a different probability). Uncorrelated Pauli error models, especially uncorrelated depolarizing error models, are very useful for simulating fault-tolerant protocols to assess their behavior and error tolerance. One reason for this is that error correction

(EC) steps for CSS and general stabilizer codes can be executed using only Clifford group gates, allowing the behavior of Pauli errors in a fault-tolerant circuit with only logical Clifford group gates to be efficiently simulated on a classical computer. In contrast, under more general error conditions, simulations quickly become complex after just a few steps.

However, an uncorrelated error model alone is insufficient for our needs. The reason is that, in Theorem 8, even if the errors in C are uncorrelated, the placement of good and bad extended rectangles (ExRecs) exhibits some correlation. This correlation arises because the ExRecs overlap, and whether an ExRec is truncated depends on whether the subsequent ExRec(s) are good or bad. Additionally, the error types in $\tilde{C}$ are even more correlated and potentially entangled, since the error type at a location $\tilde{C}_i$ in $\tilde{C}$ relies on the persistent error syndrome register generated by the *-decoder. Consequently, even if C' follows a straightforward uncorrelated error model, whether Pauli or otherwise, $\tilde{C}$ will not.

To address this, we must adopt a more complex error model.

Definition 12: Consider a complete circuit C where, with probability p_S, faults occur precisely at the set S of locations. In this case, the error at those locations can be any quantum operation consistent with the causal structure of S (i.e., if location C_i precedes C_j chronologically, the error must act on C_i before C_j). Let $p_i < 1$ be a fixed probability for each location C_i. For any set R of locations in C, if the total probability of faults occurring at every location in R (and possibly at additional locations outside R) is at most $\prod_{i \in R} p_i$, we define this as a local stochastic error model.

As before, we typically consider scenarios where p_i depends solely on the type of location C_i, often simplifying to the case where $p_i = p$ for all i. In this situation, the probability of errors occurring at all locations within a specified set of r locations is at most p^r. It's important to note that this condition does not impose any requirements on the behavior of locations outside the set R; some may exhibit errors while others may not. Furthermore, we place no restrictions on the type of errors occurring at any specific location C_i for $i \in R$.

Often, we envision an adversary selecting the type of error with the intent of complicating matters as much as possible. This adversary can choose errors that create entanglements between different faulty locations and may even opt to disable errors at some locations if that would result in greater complications than leaving them active. However, the adversary must adhere to a strategy where the probability of error across any given set of locations decreases exponentially with the size of that set, reflecting the "local" nature of the error model.

An additional crucial limitation is implied in the definition above: we have assumed that the fault locations are chosen randomly, justifying the designation of a "stochastic" error model. This does not encompass the most general scenario, as one could have a superposition of faults across various location sets, potentially entangled with envi-

ronmental qubits that retain persistent memory. While it is possible to generalize the threshold theorem (Theorem 10) to such cases, the specifics can become quite intricate.

An uncorrelated error model is inherently a local stochastic error model, maintaining the same p_i. However, as previously mentioned, if C' operates under an uncorrelated error model, it does not necessarily imply that $\tilde{C}$ will also follow an uncorrelated error model. This is where the benefit of extending to a local stochastic error model becomes evident: if C' is subjected to a local stochastic error model, then $\tilde{C}$ will also adhere to a local stochastic error model.

Theorem 9 (Level Reduction): Suppose we have a fault-tolerant circuit C' for C, and let C' be subjected to a local stochastic error model characterized by error bounds p_i and the probability p_S of errors occurring at precisely the set S of locations. For any particular set S, we define $\tilde{C}_S$ as described in Theorem 8, and we establish an error model for C by substituting C with $\tilde{C}_S$ with probability p_S. This constitutes a local stochastic error model. The error bounds p'_i for C are given by the inequality:

$$p'_i \leq \sum_R \prod_{i \in R} p_i, \tag{5.1}$$

where the sum over R includes sets of locations contained in the extended rectangle (ExRec) for the location C_i and has $|R| = t + 1$ (with t being the number of errors correctable by the quantum error correcting code (QECC)). In the specific scenario where $p_i \leq p$ for all i, we have:

$$p'_i \leq Ap^{t+1}, \tag{5.2}$$

where A denotes the maximum number of sets consisting of exactly $t + 1$ locations within the ExRec across all types.

Proof. According to Theorem 8, the error model for C takes on a stochastic form, meaning that for any set of locations S', there exists a corresponding probability $p'_{S'}$ for having faults specifically at the locations in S'. To derive an upper bound on the probability of errors occurring at a set R of locations in C (and potentially others), we sum the probabilities p_S for each set S of locations in C' that results in bad ExRecs for the locations C_i, where $i \in R$.

For a single extended rectangle (ExRec), specifically for the location C_i, we can establish an upper limit on the probability that it is bad by considering all sets S that include a subset R of locations present in the ExRec, where $|R| = t+1$. Any set of locations with $|R| > t + 1$ contains as a subset multiple combinations with $|R| = t + 1$, thus ensuring that these combinations are included in our sum. Utilizing the union bound along with the definition of a local stochastic error model, we conclude that the probability of any single ExRec being bad is at most p'_i, as expressed in equation (5.1).

For a collection of ExRecs associated with the locations C_i where $i \in R'$, we sum over sets S that encompass subsets of $t+1$ locations within each ExRec in the group. In cases where the ExRecs do not overlap, this yields precisely the bound $\sum_{i \in R'} p'_i$ by considering S as comprising sets R_i with $|R_i| = t+1$, where R_i is a subset of the ExRec for C_i.

However, when the ExRecs overlap, we must exercise greater caution since the earlier ExRec will be truncated if the subsequent ExRec is deemed bad. In this scenario, we sum over R_i that are contained within the truncated ExRec. To revert to the expression $\sum_{i \in R'} p'_i$, we need to incorporate additional sets of locations that include R_i intersecting with the truncated error correction (EC) step; this addition can only enhance the bound, which is permissible. □

This theorem hinges on the need to truncate extended rectangles (ExRecs). When two ExRecs are entirely separate, for both to be deemed bad, there must be $t+1$ errors in each. Consequently, the probability of both being bad is at most $p^{2(t+1)} = (p'_i)^2$ when $p_i \leq p$. However, for overlapping rectangles, if we do not truncate, only $t+1$ errors are required for both to fail, provided all errors occur within the shared error correction (EC) step. In this case, the probability of both failing would be $O(p'_i)$, not $O((p'_i)^2)$.

By applying additional computational techniques, we can actually refine the upper bound on p'_i. While we define an ExRec as bad when it has $t+1$ or more faulty locations, some configurations of $t+1$ locations (or more) may still result in a correct ExRec. A malignant set of locations R is defined as a set for which the ExRec is incorrect given some error configuration at R. Thus, the sum in equation (5.1) can be restricted to the minimal malignant sets of errors.

Notably, when $p_i = p$, the error rate p'_i for circuit C is less than the physical error rate p for C' if $Ap^t < 1$. In this scenario, we can leverage concatenated coding to minimize the logical error rate as much as desired. The threshold theorem then follows directly from theorem 9:

Theorem 10 (Threshold Theorem): There exists a threshold error rate p_T. Given a local stochastic error model with $p_i \leq p < p_T$, for any ideal circuit C and any $\epsilon > 0$, there exists a fault-tolerant circuit C' that, under the error model, yields an output with a statistical distance of at most ϵ from the output of C. The circuit C' will require a number of qubits and timesteps that are at most polylog$(|C|/\epsilon)$ times greater than those in C, where $|C|$ is the total number of locations in C.

Proof. We utilize concatenated codes to construct a series of circuits $C(k)$ for $k = 1, \ldots, L$, where each $C(k)$ serves as a fault-tolerant simulation of $C(k-1)$, starting with $C(0) = C$. The final circuit $C' = C(L)$ operates under a local stochastic error model, and by theorem 9, this holds true for all $C(k)$ with $k < L$. The local stochastic error model for $C(k)$ has an error bound $p^i_{(L-k)} \leq p_{(L-k)}$, where $p(0) = p$ and:

$$p(j) \leq A\left(p(j-1)\right)^{t+1} = p(j-1)\left(\frac{p(j-1)}{p_T}\right)^t \tag{5.3}$$

with $p_T = 1/A^{1/t}$. Thus, it follows that

$$\frac{p(1)}{p_T} \leq \left(\frac{p}{p_T}\right)^{t+1}, \quad \frac{p(2)}{p_T} \leq \left(\frac{p}{p_T}\right)^{(t+1)2}, \quad \text{and} \quad \frac{p(j)}{p_T} \leq \left(\frac{p}{p_T}\right)^{(t+1)j}. \tag{5.4}$$

After L levels of encoding, the logical error rate decreases double exponentially when $p < p_T$. To achieve a final error rate of ϵ_1 per location in C, we select:

$$L = \left\lceil \log_{t+1}\left(\frac{\log(\epsilon_1/p_T)}{\log(p/p_T)}\right)\right\rceil. \tag{5.5}$$

This choice ensures that C experiences a local stochastic error model with an error bound $p_i \leq \epsilon_1$, thus giving an upper limit on the probability of faults at location i. Consequently, the overall probability of an error occurring at any location in C is at most $\epsilon = |C|\epsilon_1$. With a probability of $1-\epsilon$, there are no faults in the simulated circuit C, leading to an overall statistical difference in the output of at most ϵ from the correct output distribution.

The size of C' is determined by L. The total number of qubits, including ancillas, utilized in a fault-tolerant gadget is bounded by a constant G, as is the total number of time steps. Thus, the qubits and time steps involved in $C(k)$ are at most G times those in $C(k-1)$, meaning the number of qubits and time steps in $C(L)$ is at most G^L times the original count in C. By selecting L as in equation (5.5), we establish that

$$G^L \leq G\left(\frac{\log(\epsilon_1/p_T)}{\log(p/p_T)}\right)^{\frac{\log G}{\log(t+1)}}. \tag{5.6}$$

□

To clarify further, when selecting a specific [[n, 1, 2t + 1]] quantum error-correcting code (QECC) and a corresponding fault-tolerant protocol, we can explicitly compute the constant A from equation (5.2). The threshold p_T for that protocol can then be defined as:

$$p_T = \frac{1}{A^{1/t}}. \tag{5.7}$$

Different codes or protocols, or alternative analytical approaches, can yield varying thresholds, with the true threshold p_T being the supremum across all possible selections. For instance, when using the 7-qubit code and meticulously accounting for malignant error sets, researchers have established that:

$$p_T \geq 2.73 \times 10^{-5}. \tag{5.8}$$

Various studies have explored a range of codes and fault-tolerant protocols, with significant results emerging from simulations. Notably, Knill has reported the highest claimed thresholds through intricate ancilla preparation techniques, achieving a threshold p_T as high as 5%, contingent on the specifics of the error model utilized. Currently, the most robust rigorous lower bound on the threshold, employing similar techniques, establishes a threshold of:

$$p_T \geq 10^{-3}. \tag{5.9}$$

These findings underscore the ongoing research and advancements in quantum error correction, highlighting the importance of both code selection and fault-tolerant protocols in optimizing error thresholds. The threshold theorem has potential for enhancement in several ways. A related theorem can be established considering more complex noise arising from weak interactions between the quantum computer and a non-Markovian environment, which encompasses systematic errors. For example, if each time a phase rotation R_θ is executed, it is consistently over-rotated by a small angle, this can be accommodated in the model.

The gadgets described in this paper presuppose the ability to perform gates between any pairs of qubits. However, it is also feasible to create gadgets that operate solely with nearest-neighbor interactions, applicable in one or two dimensions. While most quantum gates tend to mix different types of Pauli errors, careful design can lead to fault-tolerant protocols that leverage a significant asymmetry between X and Z errors. Optimizing the threshold generally incurs a substantial overhead cost, and while there has been some investigation into the balance between overhead and threshold values, further research in this area is warranted. Recent findings suggest that thresholds can be enhanced by not treating each concatenation level in isolation. Instead, allowing the error correction (EC) gadgets to utilize error information produced by lower-level error correction processes can lead to better outcomes.

Theorem 10 is built on certain fundamental assumptions that are indispensable. Firstly, as long as there are non-zero error rates at the physical wait locations, we cannot afford to leave any qubit uncorrected for extended periods. This necessitates the ability to perform operations in parallel; otherwise, as the quantum computer scales, we would only be able to correct errors on an increasingly negligible fraction of the qubits at any moment. Additionally, we need the capacity to prepare qubits during the computation itself. This is crucial because ancilla qubits, which are essential for error correction, would accumulate significant error rates by the time they are used in an error correction (EC) gadget if they were only prepared at the start.

Moreover, we must assume that errors do not permanently remove qubits from our system. An error should either transition us to a valid physical computational state

(even if it is incorrect) or lead to a state that can be restored to a valid computational state through appropriate corrective actions. Lastly, we require some constraints on the correlations within the error model. While the local stochastic model accommodates strong correlations among errors, it at least presumes that the probability of an error occurring in a large set of qubits decreases exponentially. This is generally necessary; the non-Markovian model relaxes this assumption to suggest that the amplitude of errors affecting many qubits diminishes exponentially. If we were to encounter merely polynomial decay in the amplitude of such errors, there would be a polynomially small probability that all qubits in the computer could fail simultaneously, making recovery from such a catastrophic failure impossible.

Bibliography

[1] D. Aharonov and M. Ben-Or, "Fault-tolerant quantum computation with constant error," Proc. 29th Ann. ACM Symp. on Theory of Computation, ACM, 1998, pp. 176–188; arXiv:quant-ph/9611025.

[2] D. Aharonov, M. Ben-Or, R. Impagliazzo, and N. Nisan, "Limitations of Noisy Reversible Computation," arXiv:quant-ph/9611028, 1996.

[3] P. Aliferis, D. Gottesman, and J. Preskill, "Quantum accuracy threshold for concatenated distance-3 codes," Quant. Info. Comp., vol. 6, 2006, pp. 97–165; arXiv:quant-ph/0504218.

[4] C. H. Bennett, D. P. DiVincenzo, J. A. Smolin, and W. K. Wootters, "Mixed state entanglement and quantum error correction," Phys. Rev. A, vol. 54, 1996, pp. 3824–3851; arXiv:quant-ph/9604024.

[5] A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane, "Quantum error correction and orthogonal geometry," Phys. Rev. Lett., vol. 78, 1997, pp. 405–408; arXiv:quant-ph/9605005.

[6] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," Phys. Rev. A, vol. 54, 1996, pp. 1098–1105; arXiv:quant-ph/9512032.

[7] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, "Topological quantum memory," J. Math. Phys., vol. 43, 2002, pp. 4452–4505; arXiv:quant-ph/0110143.

[8] D. Gottesman, "Class of quantum error-correcting codes saturating the quantum Hamming bound," Phys. Rev. A, vol. 54, 1996, pp. 1862–1868; arXiv:quant-ph/9604038.

[9] D. Gottesman, "Stabilizer codes and quantum error correction," Ph.D. dissertation, California Institute of Technology, Pasadena, CA, 1997; arXiv:quant-ph/9705052.

[10] D. Gottesman, "Theory of fault-tolerant quantum computation," Phys. Rev. A, vol. 57, 1998, pp. 127–137; arXiv:quant-ph/9702029.

[11] D. Gottesman, "An Introduction to Quantum Error Correction and Fault-Tolerant Quantum Computation," arXiv:quant-ph/0004072.

[12] E. Knill and R. Laflamme, "Theory of quantum error-correcting codes," Phys. Rev. A, vol. 55, 1997, pp. 900–911; arXiv:quant-ph/9604034.

[13] E. Knill, "Quantum Computing with Realistically Noisy Devices," Nature, vol. 434, 2005, pp. 39–44; arXiv:quant-ph/0402171.

[14] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," Phys. Rev. A, vol. 52, 1995, pp. 2493–2496.

[15] A. M. Steane, "Error correcting codes in quantum theory," Phys. Rev. Lett., vol. 77, 1996, pp. 793–797.

[16] A. M. Steane, "Active stabilization, quantum computation, and quantum state synthesis," Phys. Rev. Lett., vol. 78, 1997, p. 2252; arXiv:quan-ph/9611027.

[17] B. M. Terhal and G. Burkard, "Fault-tolerant quantum computation for local non-Markovian noise," Phys. Rev. A, vol. 71, 2005, 012336; arXiv:quant-ph/0402104.

[18] W. K. Wootters and W. H. Zurek, "A single quantum cannot be cloned," Nature, vol. 299, 1982, p. 802.