SOFTWARE IMPLEMENTATION OF QUANTUM ERROR-CORRECTION

An Undergraduate Research Scholars Thesis

by

AHMED AL-SHEMMERY[1], MUHAMMAD GHASEF PARACHA[2], AND GIBIN GEORGE[3]

Submitted to the LAUNCH: Undergraduate Research office at
Texas A&M University
in partial fulfillment of the requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by
Faculty Research Advisor: Dr. Joseph J. Boutros

May 2022

Major: Electrical Engineering[1,2,3]

RESEARCH COMPLIANCE CERTIFICATION

Research activities involving the use of human subjects, vertebrate animals, and/or biohazards must be reviewed and approved by the appropriate Texas A&M University regulatory research committee (i.e., IRB, IACUC, IBC) before the activity can commence. This requirement applies to activities conducted at Texas A&M and to activities conducted at non-Texas A&M facilities or institutions. In both cases, students are responsible for working with the relevant Texas A&M research compliance program to ensure and document that all Texas A&M compliance obligations are met before the study begins.

We, Ahmed Al-Shemmery[1], Muhammad Ghasef Paracha[2], and Gibin George[3], certify that all research compliance requirements related to this Undergraduate Research Scholars thesis have been addressed with our Research Faculty Advisor prior to the collection of any data used in this final thesis submission.

This project did not require approval from the Texas A&M University Research Compliance & Biosafety office.

TABLE OF CONTENTS

ABSTRACT

Software Implementation of Quantum Error-Correction

Ahmed Al-Shemmery[1], Muhammad Ghasef Paracha[2], and Gibin George[3]
Electrical and Computer Engineering Department[1,2,3]
Texas A&M University at Qatar

Research Faculty Advisor: Dr. Joseph J. Boutros
Electrical and Computer Engineering Department
Texas A&M University at Qatar

The transmission of information is prone to errors that alter the information to an extent depending on the channel. To reduce errors, error-correcting protocols have been developed to increase the reliability of the communication and preserve the information. In quantum communication, quantum bits (qubits) are used to transmit information, and due to the nature of qubits, they offer tremendous advantages over their classical counterparts, the bits. However, their nature also makes them extremely fragile and susceptible to three types of errors: bit-flip, phase-flip, and bit and phase flip. The field of quantum error-correction (QEC) is concerned with the development of protocols to protect qubits and establish reliable communication in quantum channels. An important class of quantum error-correcting codes (QECCs) is the Calderbank-Shor-Steane (CSS) family of codes that use classical error-correcting codes to construct quantum codes. This project is concerned with the evaluation of the performance of CSS codes via conducting Monte Carlo simulations and calculating the probability of error per word for the code in relations to the probability of error in a quantum channel using a Python script. Simulations were conducted to evaluate the performance of the $[[7,\ 1]]$ and the $[[31,\ 11]]$ CSS codes. It was found that each code performs better than the other in a given region of values of channel error probability and perform equally at channel error probability about 5×10^{-3}.

DEDICATION

To my mother, my sister, my uncle Ahmed, and Yasir Abdulrahman. I would not be where I am without your support, love, and sacrifices.
And, To my homeland and the hope of a better tomorrow.

- Ahmed Al-Shemmery

I dedicate this thesis to my parents, Zahida and Khalil. I thank them for their love and support throughout my career and I hope this achievement will complete their dreams of providing the best education.

- Muhammad Ghasef Paracha

To my parents, and my sister thank you for your constant moral, spiritual, emotional, and financial support. To the ones who pushed me forward when I wanted to give up.
To God Almighty, for your guidance and strength.

- Gibin George

ACKNOWLEDGMENTS

Contributors

We would like to sincerely thank our research advisor, Dr. Joseph Boutros, for believing in us and for all the time he invested in our education. We thank Dr. Gilles Zémor (Institut de Mathématiques de Bordeaux, University of Bordeaux) for his tutorial on quantum error-correction. Finally, we thank our friends and families for their encouragement, patience, and support given throughout the duration of the project.

All work conducted for the thesis was completed by the students independently with the aid of the research advisor.

Funding Sources

The work presented did not require any funding.

NOMENCLATURE

BCH Code	Bose–Chaudhuri–Hocquenghem Code
LDPC Code	Low-Density Parity-Check Code
Qubit	Quantum Bit
QEC	Quantum Error Correction
QECC	Quantum Error Correcting Codes
CNOT	Controlled-Not
CLU	Control Logic Unit
CSS Code	Calerbank-Steane-Shor Code

1. INTRODUCTION

In all communication systems, information is transmitted through a medium (channel) over a certain distance. The channel alters the information transmitted, adding errors whose severity depends on the channel itself. To reduce the probability of error, many error-correcting schemes have been developed to increase the reliability of communication between the transmitter and the receiver. In digital communications, *classical error-correcting codes* are used to encode information stored in bits to a larger number of bits to add redundancy and permit correction.

In quantum communications, quantum bits (qubits) are used to transfer information over quantum channels. Qubits are probabilistic, delicate, and susceptible to three different types of errors, complicating the process of protecting quantum information. The field of *quantum error-correction* (QEC) — emerged in the mid 90s — is concerned with the construction of error-correcting schemes to establish reliable communication through quantum channels.

This project is concerned with conducting Monte Carlo simulations of a family of quantum error-correcting codes (QECC), the Calderbank-Shor-Steane (CSS) codes via Python programming to evaluate the performance of the codes for different channel error probability. This chapter explains the theory of classical error-correction in Section 1.1 and the theory of QEC in Section 1.2.

1.1 Classical Error Correction

A linear classical error-correcting code C of length n is defined as a subspace of dimension k of the vector space $\mathbb{F}_2^n$ containing all binary row vectors (the *words*) of length n, where $\mathbb{F}_2 = \{0, 1\}$ is the smallest possible mathematical field [1]. Hence, C is a set of specified words (the *codewords*) such that C is an Abelian additive group that always contains the all-zero array. A code C of length n with dim(C) $= k$ is denoted as $[n,\ k]_2$, where the 2 indicates that it is a binary code, it is dropped since this paper is focused solely on binary codes. k is the number of *information bits* — the bits containing the information to be transmitted — and $n - k$ is the number of *parity bits* used to add *redundancy* which allows error-correction and recovery of the information bits. The cardinality of $C\ [n,\ k]$, i.e. the number of codewords in the code, is 2^k. The *coding rate* of C, R, is defined as the ratio of the number of information bits to the total number of transmitted bits:

$$R = \frac{k}{n}. \qquad \text{(Eq. 1.1)}$$

Therefore, the increase in the coding rate signifies the increase of the number of information bits transmitted and the decrease of the correction capacity of the code and vice versa.

In linear classical error-correcting codes, the number of nonzero elements in a codeword $\mathbf{c} \in C$ is named the *Hamming weight* $w(\mathbf{c})$ of the codeword. The *minimum Hamming weight*, equivalently the *minimum Hamming distance* d_{min}, of the code is:

$$d_{min} = w_{min} = \min_{\forall \mathbf{c} \in C, \mathbf{c} \neq \mathbf{0}} w(\mathbf{c}), \qquad \text{(Eq. 1.2)}$$

where d_{min} defines the error-correcting capabilities of C through the following inequality:

$$d_{min} \geq 2t + 1, \qquad \text{(Eq. 1.3)}$$

where the maximum integer value of t satisfying Eq. 1.3 is the maximum number of errors C is capable of correcting. Figure 1.1 shows the error-correcting capabilities of codes of different d_{min} values between $c, c' \in C$. The red line indicates the midpoint between c and c' that divides the line connecting c to c' into two decision regions, one for c and the other for c'. If c is transmitted, then the correction of the erroneous word received depends on locating the closest codeword (finding in which decision region the erroneous word lies) and acting accordingly.

In addition, for a $[n,\ k]$ code, t is bound by:

$$\sum_{i=0}^{t} \binom{n}{i} \leq 2^{n-k}, \qquad \text{(Eq. 1.4)}$$

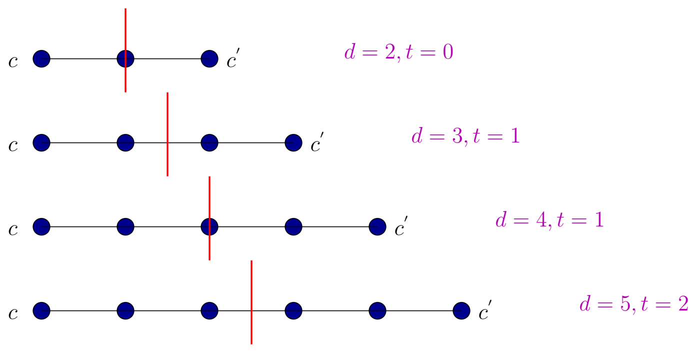

Figure 1.1: Illustration of minimum distance and error-correcting capabilities of codes.

named the *Hamming Bound*, a necessary but not sufficient condition for the error-correcting capabilities of a code, i.e. any t that does not satisfy the bound is not correctable, but a t that does is not necessarily correctable. Codes that achieve equality in the Hamming bound are named *Perfect codes*.

A code $C\ [n,\ k]$ can be represented by two matrices: The $k \times n$ *generator matrix* G and the $(n-k) \times n$ *parity-check matrix* H. The generator matrix G is constructed using k linearly independent codewords as its rows such that the vector space spanned by its rows is the code C itself. G is used to generate all codewords of C as $1 \times n$ vectors:

$$\mathbf{c} = \mathbf{b} \cdot G,\ \mathbf{b} \in \mathbb{F}_2^k, \tag{Eq. 1.5}$$

such that G maps (encodes) a word $\mathbf{b}$ in vector space $\mathbb{F}_2^k$ to a codeword in $C \subset \mathbb{F}_2^n$. Usually, G is written in *systematic form* as:

$$G = \left[\begin{array}{c|c} I_k & P \end{array}\right], \tag{Eq. 1.6}$$

where I_k is the $k \times k$ identity matrix and P is a $k \times (n-k)$ matrix.

The parity-check matrix H is derived from G as defined in Eq. 1.6 as:

$$H = \left[\begin{array}{c|c} P^T & I_{n-k} \end{array}\right], \tag{Eq. 1.7}$$

such that H can be used to verify that a word in $\mathbb{F}_2^n$ is a codeword $\mathbf{c} \in C$ through the *check equation* defined as:

$$\mathbf{c}H^T = \mathbf{0}, \text{ iff } \mathbf{c} \in C. \quad \text{(Eq. 1.8)}$$

In addition. the parity-check equation H can be translated to a set of $n - k$ equations, the *parity-check equations*, that show the the constraints between the different bits of the codewords in C. For $\mathbf{c} \in C,\ \mathbf{c} = (\alpha_1, \alpha_2, ..., \alpha_n)$, the i^{th} column of H represents the coefficients of the bit α_i in each parity-check equation. The parity-check matrix H can also be translated into a *Tanner graph*, a special type of bipartite graphs, that provides a graphical representation of the code. In the Tanner graph, each information bit is represented by a node, the *bit node*, and each parity bit is represented by a different type of node, a *check node* (also called a *subcode node*), such that the bit nodes are connected to the check nodes by *edges* according to the constraints among them defined by H. The degree of the bit nodes d_b is defined as the maximum number of edges connected to a single bit node. The degree of the check nodes d_c is defined similarly.

Every code C has an orthogonal compliment, denoted $C^\perp$, which is a set that contains all words in $\mathbb{F}_2^n$ that are orthogonal to C. $C^\perp$ is in itself a code named the *dual code* of C. The generator matrix of $C^\perp$, $G(C^\perp)$, is the parity-check matrix of C, $H(C)$, and vice versa, meaning that for $C\ [n,\ k]$, its dual code is $C^\perp\ [n,\ n-k]$.

There are many types of linear classical codes such as *repetition codes*, *Hamming codes*, *Bose–Chaudhuri–Hocquenghem (BCH) codes*, *Reed-Solomon codes*, and *low-density parity-check (LDPC) codes* that differ in their construction, properties, and capabilities. However, the following subsection will briefly explain selected codes.

1.1.1 Repetition Codes

Repetition codes are basic classical error-correcting codes that encode by copying the value of a single bit into multiple bits — $\mathbb{F}_2 \mapsto \mathbb{F}_2^n$ — and perform *majority decision* at the decoder to perform error-correction [2]. The smallest repetition code capable of correcting an error is the 3-bit repetition code: $[3,\ 1,\ 3]$ code with $t = 1$, $R = \frac{1}{3}$, and the following G and H:

$$G = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \text{ and } H = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}. \qquad \text{(Eq. 1.9)}$$

Hence, using Eq. 1.5, the code's encoding maps $0 \mapsto 000$ and $1 \mapsto 111$, meaning that the codewords are $\{000, 111\}$.

Assuming that 000 was the codeword transmitted and the decoder receives 010, the decoder will perform majority decision and since the majority of the bits are 0's, the code will correct by flipping 1 to 0, concluding that 000 was the codeword transmitted. However, if the decoder receives 101, the decoder will correct to 111 and, therefore, add to the error. Such a phenomena is caused by the occurrence of an error beyond the capabilities of the code, which the code misidentifies and maps to another codeword. Such a scenario, is not unique to repetition codes.

1.1.2 Hamming Codes

Hamming codes are a family of linear classical error-correcting codes created by *R. W. Hamming* in 1950 [3]. Hamming codes are characterized by $[n = 2^m - 1,\ k = 2^m - 1 - m,\ d_{min} = 3]$, i.e. with $t = 1$ and $R = \frac{2^m-1-m}{2^m-1} = 1 - \frac{m}{2^m-1}$ [4].

An example of a Hamming code, for $m = 3$, is the $[7, 4]$ code with

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \text{ and } H = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}. \qquad \text{(Eq. 1.10)}$$

Hence, the code's parity-check equations are:

$$\begin{aligned} \alpha_1 + \alpha_2 + \alpha_4 &= \alpha_5 \\ \alpha_1 + \alpha_3 + \alpha_4 &= \alpha_6 \\ \alpha_2 + \alpha_3 + \alpha_4 &= \alpha_7 \end{aligned} \qquad \text{(Eq. 1.11)}$$

and the Tanner graph of the code is shown in Figure 1.2, where the square blocks represent the bit nodes and the circular blocks represent the check nodes. The degree of the bit nodes and of the check nodes are both 3.

The $[7,\ 4]$ Hamming code's dual is the $[7,\ 3]$ code. The $[7,\ 4]$ has $2^4 = 16$ codewords and its dual has $2^3 = 8$ codewords, the codewords are shown in Table 1.1 and Table 1.2 respectively.

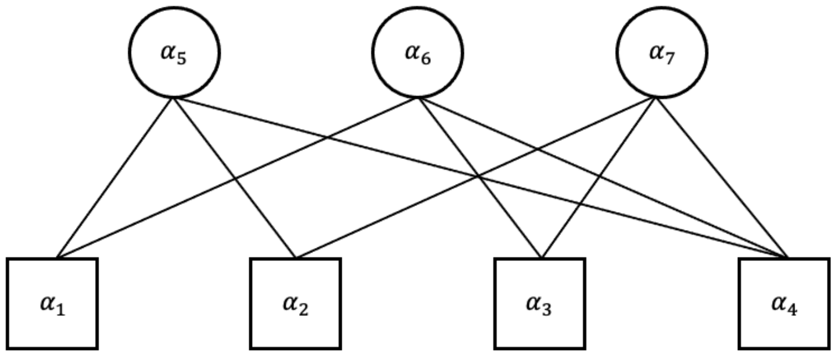

Figure 1.2: Tanner graph of the $[7,\ 4]$ Hamming code.

Table 1.1: Codewords of the $[7,\ 4]$ Hamming code.

0 0 0 0 0 0 0	1 0 0 0 0 1 1
0 0 0 1 1 1 1	1 0 0 1 1 0 0
0 0 1 0 1 1 0	1 0 1 0 1 0 1
0 0 1 1 0 0 1	1 0 1 1 0 1 0
0 1 0 0 1 0 1	1 1 0 0 1 1 0
0 1 0 1 0 1 0	1 1 0 1 0 0 1
0 1 1 0 0 1 1	1 1 1 0 0 0 0
0 1 1 1 1 0 0	1 1 1 1 1 1 1

Table 1.2: Codewords of the $[7,\ 3]$ code.

0 0 0 0 0 0 0	0 0 0 1 1 1 1
0 1 1 0 0 1 1	0 1 1 1 1 0 0
1 0 1 0 1 0 1	1 0 1 1 0 1 0
1 1 0 0 1 1 0	1 1 0 1 0 0 1

1.1.3 Low-Density Parity-Check Codes

LDPC codes are a class of linear classical codes that were first developed in 1963 by Robert Gallager [5]. LDPC codes are characterized by the sparseness of their parity-check matrices, i.e. with few ones and many zeros. Due to its applications to large number of bits, LDPC codes are usually defined, illustrated, and often constructed using Tanner graphs. For a regular $[n, k]$ LDPC, its coding rate is bounded by:

$$R = \frac{k}{n} \geq (1 - \frac{d_b}{d_c}) \qquad \text{(Eq. 1.12)}$$

For example, the Tanner graph of a $[1000,\ 500]$ LDPC code with coding rate $R = \frac{1}{2}$ is shown in Figure 1.3, where the code is constructed by randomly creating 3000 edges (the constraints among the bits) connecting bit nodes to check nodes such that $d_b = 3$ and $d_c = 6$. A parity-check matrix of the $[100, 50]$ LDPC code is shown in Figure 1.4 where a red dot represents a 1 and a white dot represents a 0. The sparseness of the matrix is obvious, however; the sparseness is lost when the parity-check matrix is written in systematic form.

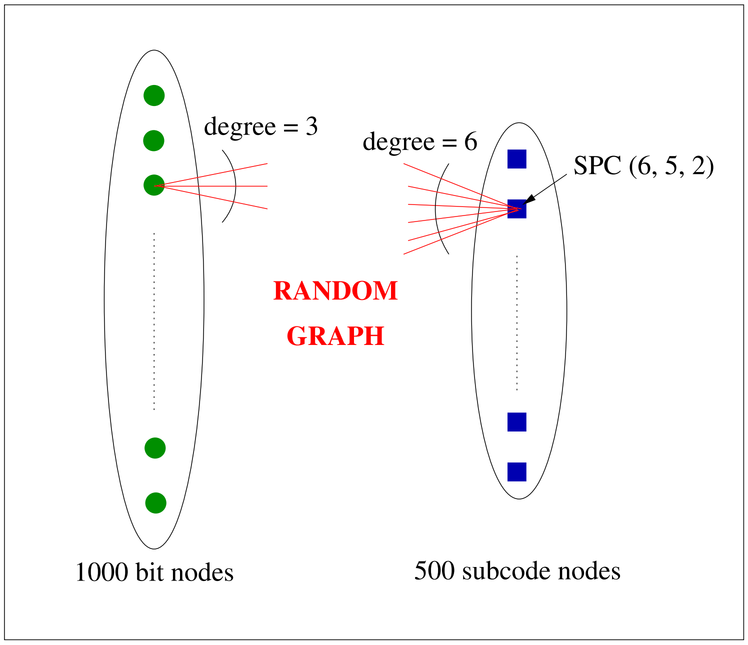

Figure 1.3: Tanner graph of $[1000,\ 500]$ LDPC code with $d_b = 3$ and $d_c = 6$.

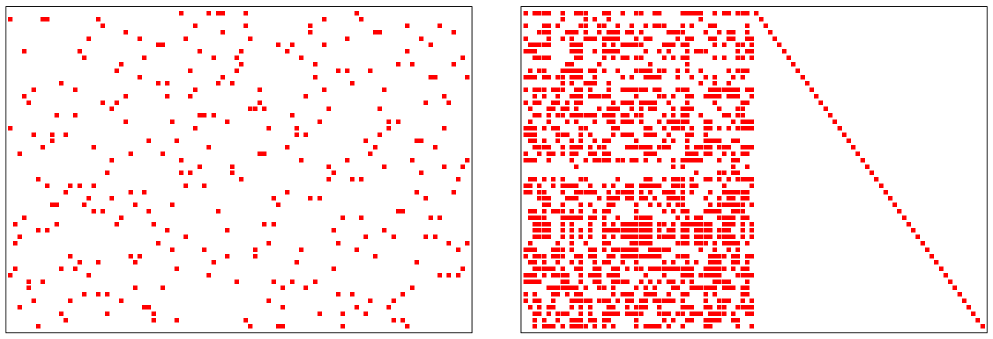
Figure 1.4: Illustration of the parity-check matrix of the $[100,\ 50]$ LDPC code illustrating sparsity and systematic form.

The decoding process of LDPC codes uses iterative belief propagation techniques. To illustrate the decoding process, let us consider an ergodic binary erasure channel, which contains only erasures (where the bit's value is completely erased) and no errors. The channel's input is binary, and the output is ternary, shown in Figure 1.5.

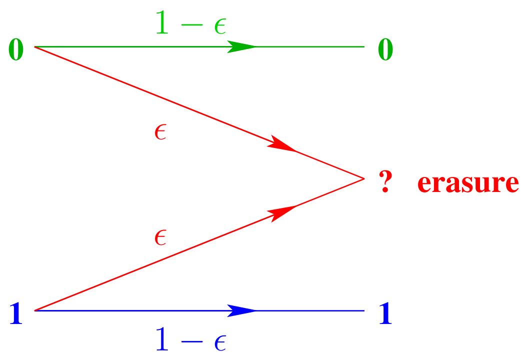

Figure 1.5: Diagram illustrating the behavior of the Binary Erasure Channel.

A LDPC $[n, k]$ code is required to fill erasures on the binary erasure channel. The iterative non probabilistic decoding algorithm works as follows.

1. Count the number of erased bits, μ, connected to check node j.

2. If the number of erased bits is 1, $\mu = 1$, then replace the erased bit with summation of the other bits.

3. Increment j. If the current number of check nodes is higher than total number of check nodes, $j > L$, then increment the number of iterations and set $j = 1$.

4. If the number of Iterations, $Iter$, has exceeded number of max iterations, $Iter > MaxIter$, then loop back to step 1.

1.2 Quantum Error Correction

Analogous to the bit, the *qubit* is the basis of quantum information. Qubits are extremely prone to error caused by the inability to completely isolate a qubit from its environment leading to undesired entanglement with its surrounding — what is known as *decoherence* — and caused by the processing and control of qubits, *gate errors*. Due to the fragility of qubits, any technology in quantum computation or quantum information based on any available physical qubit technology cannot be realized and will remain purely theoretical without proper error detection and correction protocols, namely QEC [6].

A QECC Q $[[n,\ k]]$ is a k-dimensional subset of the Hilbert space, $\mathcal{H} := \mathbb{C}^{2^n}$, where the double brackets are used to differentiate between quantum and classical codes. Hence, a QECC is the set of all states of the system $|\psi\rangle$ (the codewords) that satisfy the construction of the code, i.e. $Q = \{|\psi\rangle\}$. The coding rate of the code remains as defined in Eq. 1.1. A QECC's error-correcting capabilities are bound by the *quantum Hamming bound* [7] — analogous to the Hamming bound defined in Eq. 1.4 — defined as:

$$\sum_{i=0}^{t} 3^i \binom{n}{i} \leq 2^{n-k}, \tag{Eq. 1.13}$$

where n is the number of qubits, k is the number of information qubits, and t is weight of the errors. The additional 3^i expression is due to the three types of errors qubits are susceptible to. The quantum Hamming bound is a necessary but not sufficient condition, i.e. any weight t not satisfying the bound is beyond the capabilities of the code, but a weight t that satisfies it is not necessarily within its capabilities.

1.2.1 Qubits as Abstract Mathematical Objects

A classical bit's value is a deterministic value in $\mathbb{F}_2$ — 0 or 1 — that can be determined through examination. Qubits also have a state, the basis states:

$$|0\rangle = \begin{bmatrix} 1 & 0 \end{bmatrix}^T \text{ and } |1\rangle = \begin{bmatrix} 0 & 1 \end{bmatrix}^T, \quad |0\rangle, |1\rangle \in \mathbb{C}^2, \tag{Eq. 1.14}$$

written in the *Dirac notation*. A qubit's state is represented as a linear combination of its basis states as:

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle, \tag{Eq. 1.15}$$

where α_0 and α_1 are complex numbers called the amplitudes of the state. Therefore, the state of a qubit is represented as a vector in $\mathcal{H} := \mathbb{C}^2$, where the basis states $|0\rangle$ and $|1\rangle$ are the orthonormal bases spanning $\mathcal{H}$ [8].

A qubit exists in a continuum of states between the basis states, and when an examination is conducted to determine the value of the state of a qubit, the examination yields $|0\rangle$ with probability $|\alpha_0|^2$ or yields $|1\rangle$ with probability $|\alpha_1|^2$, where $|\alpha_1|^2 + |\alpha_1|^2 = 1$, meaning that $|\psi\rangle$ in Eq. 1.15 is a unit vector [7].

The probabilistic behavior of qubits complicates the process of understanding and manipulating their behavior. Yet, it is not impossible. There exist methods with which their states can be influenced to yield a desirable output.

Generally, multi-qubit systems are extremely complex. The state of an n-qubit system is a vector in $\mathbb{C}^{2^n}$ Hilbert space spanned by 2^n basis vectors. This shows that the complexity of the system increases exponentially as the number of qubits in the system increases. To put this into perspective, a 100-qubit system has 2^{100}, approximately 10^{30}, basis states and 2^{100} complex variables, the amplitudes. Therefore, any Monte Carlo simulation of the system will have exponential complexity and large execution time. For instance, the general state of a system of 2 qubits is represented by the vector

$$|\psi\rangle = \alpha_{00}\,|00\rangle + \alpha_{01}\,|01\rangle + \alpha_{10}\,|10\rangle + \alpha_{11}\,|11\rangle\,. \qquad \text{(Eq. 1.16)}$$

The basis states shown in Eq. 1.16 are analogous to the 4 possible states of a classical two-bit system. The basis states in n-qubit systems are obtained through the *Kronecker product* "$\otimes$" on the single qubit basis states in Eq. 1.14, for example, the state $|01\rangle$ can be written as:

$$|01\rangle = |0\rangle \otimes |1\rangle = \begin{bmatrix} 1 & 0 \end{bmatrix}^T \otimes \begin{bmatrix} 0 & 1 \end{bmatrix}^T = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}^T. \qquad \text{(Eq. 1.17)}$$

1.2.2 Errors and Operations on Qubits

Qubits are susceptible to bit-flip errors, phase-flip error, or combination of both represented by the 2×2 *Pauli matrices* I, X, Z, and Y defined as:

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},\ Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},\ \text{and}\ Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = iXZ, \qquad \text{(Eq. 1.18)}$$

where I is the 2×2 identity matrix representing the absence of error, X is the bit-flip error, Z is the phase-flip error, and Y is the bit-flip and phase-flip error.

All qubit errors are multiplicative, i.e. are represented by multiplying the matrices in Eq. 1.18 by the state of a qubit. In addition, all Pauli matrices:

1. are involutory matrices — they are their own inverses — for example, $X^2 = I$.
2. either commute or anti-commute amongst themselves, where they commute only with themselves and the identity ($[X,\ X] = 0$) and anti-commute with the rest ($\{X,\ Y\} = 0$).

The first property allows the correction of an error by multiplying the erroneous state by the same error. The second property plays a crucial role in the construction and formalization of QECCs [9].

In general, an error operator E can be constructed to represent any error possible to occur on a qubit as a linear combination of I, X, Y, and Z as:

$$E = \alpha_I I + \alpha_X X + \alpha_Y Y + \alpha_Z Z = \alpha_I I + \alpha_X X + \alpha_{XZ} XZ + \alpha_Z Z. \qquad \text{(Eq. 1.19)}$$

Therefore, to detect and correct errors, one needs only to detect and correct X and Z errors! The effect of the Pauli matrices, errors, on the basis states $|0\rangle$ and $|1\rangle$ is illustrated in Table 1.3.

Table 1.3: Effect of errors on individual basis states of a single qubit.

Error	Effect on $\|0\rangle$	Effect on $\|1\rangle$
I	$\|0\rangle$	$\|1\rangle$
X	$\|1\rangle$	$\|0\rangle$
Z	$\|0\rangle$	$-\|1\rangle$
Y	$i\|1\rangle$	$-i\|0\rangle$

In n-qubit systems, error operators are represented by the Kronecker product of I, X, Y, and Z n-times. Hence, all operators representing errors on the system are elements of the *Pauli group* $\mathcal{P}_n$, a multiplicative group defined as:

$$\mathcal{P}_n := \{\gamma \cdot \{I, X, Y, Z\}^{\otimes n} : \gamma \in \{\pm 1, \pm i\}\}. \qquad \text{(Eq. 1.20)}$$

where elements in $\mathcal{P}_n$ are $2^n \times 2^n$ square matrices. For instance, in a 5-qubit system, one arbitrarily chosen error operator is $E = X \otimes I \otimes Z \otimes X \otimes Y \in \mathcal{P}_5$ (in the future the "$\otimes$" is dropped for convenience). If E is applied on the state vector of the system, X is applied on the first qubit, I on the second, Z on the third, and so on. Another notation for E which will be used throughout the paper, is $E = X_{10011}Z_{00101}$ where the subscripts illustrate which qubits the errors X and Z act on, where 1 in position m means that the error acts on the m-th qubit and 0 means it does not. For example, $X_{10011} = X\ I\ I\ X\ X$. Note that a Y error, as defined in Eq. 1.18, is represented by XZ and hence, is represented by having 1 in both subscripts in the corresponding position. Following the same notation, an error $E = X_u Z_v$ can be represented as a vector $e = [\ v \mid u\ ]$ [7].

Regardless of the number of qubits in the system, Pauli operators either commute or anti-commute among themselves regardless of dimensions. Two Pauli operators commute (respt. anti-commute) if and only if there is an even (respt. odd) number of Pauli matrices in corresponding positions that anti-commute. For example, let us consider two Pauli operators $O_1 = X\ Z\ I\ Y$ and $O_2 = Z\ X\ I\ Y$. In positions 1 and 2, the matrices anti-commute. In positions 3 and 4, the matrices commute. Hence, since the number of anti-commuting matrices is even, $[O_1,\ O_2] = 0$. Now, defining $O_3 = X\ I\ X\ Z$ and comparing it with O_1 and O_2, we find that $\{O_3,\ O_1\} = 0$ and $[O_3,\ O_2] = 0$.

The weight of a Pauli operator is the number of non-identity elements in the operator. For $O_1 = X\ Z\ I\ Y$, the weight $w(O_1) = 3$. The weight of a Pauli operator representing an error is an important parameter when considering the code's ability to correct an error.

There are many operations performed on qubits — in the form of gates — that are used to manipulate, protect, and correct qubits to reliably transmit information and perform computations, including the X, Y, and Z operations, the Pauli gates. However, for the purposes of this paper, only the Hadamard gate, H, and the controlled-not (CNOT) — controlled-X or CNOT — are discussed.

The Hadamard gate introduces superposition in the a qubit's state if the qubit's initial state was deterministic, i.e. a pure basis state $|0\rangle$ or $|1\rangle$. The Hadamard gate is represented by the matrix

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{Eq. 1.21}$$

Therefore, $H|0\rangle = \frac{|0\rangle+|1\rangle}{2} = |+\rangle$ and $H|1\rangle = \frac{|0\rangle-|1\rangle}{2} = |-\rangle$ where $|+\rangle$ and $|-\rangle$ can be used as new computational bases in lieu of $|0\rangle$ and $|1\rangle$ if it provided easement in correcting errors. *H* is a unitary and Hermitian matrix, i.e. $H = H^\dagger$ and $HH^\dagger = H^\dagger H = H^2 = I$. Another useful property of the Hadamard operation is $HZH = X$ — or equivalently $HXH = Z$ — which allows us to manipulate and change the behavior of errors occurring on qubits [8].

The CNOT gate introduces entanglement between qubits and, hence, the duplication of the state of a qubit into another without violating the *No-Cloning theorem* [10]. The CNOT gate is a

2-qubit gate — 2 inputs and 2 outputs — where it uses the control qubit's state $|\psi_1\rangle$ to determine whether to flip the state of the targeted qubit $|\psi_2\rangle$ [9]. The diagram of the CNOT gate is seen in Figure 1.6. For example, if $|\psi_1\rangle = |0\rangle$ then $|\psi_2\rangle$ remains unchanged. However, if $|\psi_2\rangle = |1\rangle$, then $|\psi_2\rangle$ is negated ($|0\rangle \iff |1\rangle$), i.e. the output of the targeted qubit is analogous to the classical XOR "$\oplus$" operation on the targeted qubit.

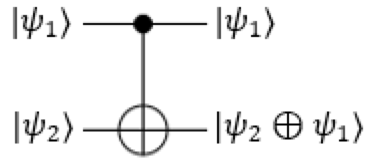

Figure 1.6: Circuit diagram of the CNOT gate.

1.2.3 General Challenges of Quantum Error Correction

In general, an error-correcting code cannot correct every possible error on the system. Instead, it offers the ability to correct a certain set of errors that are more likely to occur in the channel considered [11]. Hence, the inherent capability restrictions of a QECC must be overcome and effectiveness must be increased through the design and construction of the code.

Aside from that, there are three main challenges in the construction of QECCs [6]. First, wavefunction collapse: any measurements or observation conducted to determine the state of a qubit alters the qubit's state. Therefore, quantum codes must carefully conduct measurements on qubits as to not disturb the encoded states or use ancilla qubits to conduct measurements. Second, qubits are susceptible to bit-flip (X) and phase-flip (Z) errors, both of whom must be detectable and correctable by the code. Third and final, the *No-Cloning theorem*: the theorem forbids the creation of an operator that clones the state of qubit onto another qubit, thus forbidding repetition and greatly complicating the manipulation of qubits and the construction of quantum codes [10].

1.2.4 3-Qubit Bit-Flip and Phase-Flip Codes

The simplest QECCs are the 3-qubit codes capable of correcting a single bit-flip error (3-qubit bit-flip code) or a single phase-flip error (3-qubit phase-flip code) [7]. The 3-qubit codes are the quantum equivalent of the 3-bit repetition code discussed in section 1.1.1.

The 3-qubit bit-flip code, whose quantum circuit is shown in Figure 1.7, assumes a *bit-flip quantum channel* on each individual qubit independently, where the probability of an error is p_X and the probability of no error is $p_I = 1 - p_X$. The diagram showing the behaviour of the quantum channel is shown in Figure 1.8.

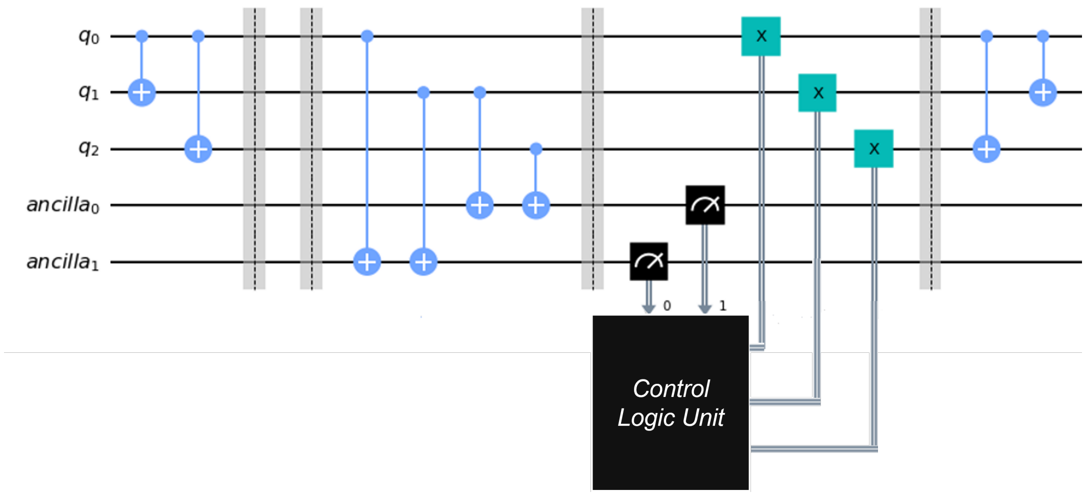

Figure 1.7: Quantum circuit of the 3-qubit bit-flip quantum code.

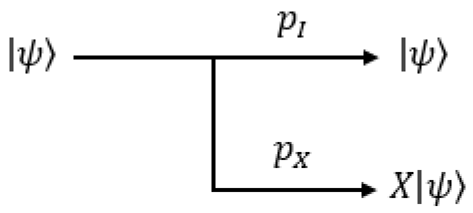

Figure 1.8: Diagram of the action of the bit-flip quantum channel.

The code encodes a single qubit to three qubits by introducing entanglement among the information qubit and the two parity qubits via two CNOT gates such that the state of the information qubit is repeated. Assuming that the information qubit's state is $\alpha|0\rangle + \beta|1\rangle$, the encoding works as such:

$$\alpha|0\rangle + \beta|1\rangle \longmapsto \alpha|000\rangle + \beta|111\rangle \qquad \text{(Eq. 1.22)}$$

After encoding and passing through the channel, the qubits' states are copied into two ancilla qubits to conduct measurement to determine the error. This stage is named *syndrome measurement*. The results of syndrome measurement are passed onto a *control logic unit* (CLU) that determines the necessary action to correct the error present according to the *truth table* shown in Table 1.4. Regardless of whether the action taken corrects or adds to the error, the resultant state of the system will always be in the set of codewords of the code defined earlier.

Table 1.4: Truth table of CLU for the 3-qubit bit-flip code.

ancilla$_0$	ancilla$_1$	Identified Error
0	0	X_{000}
0	1	X_{001}
1	0	X_{100}
1	1	X_{010}

After the appropriate action is taken, the qubits are passed through two CNOT gates in the reverse order of those in the encoding stage to disentangle the states of the qubits.

An example of the behavior of the code is if $\alpha\,|0\rangle + \beta\,|1\rangle$ is transmitted and the channel adds the error X_{001} such that the received state of the system is $\alpha\,|001\rangle + \beta\,|110\rangle$. After the syndrome measurement, 01, the CLU will correctly identify the error as X_{001} and correct it. Hence, the state of the system after correction is the originally transmitted state. However, if the channel adds the error X_{101}, the state received after the channel is $\alpha\,|101\rangle + \beta\,|010\rangle$. The syndrome measurement will yield 11, mistakenly identifying the error as X_{010} and "correcting" to $\alpha\,|111\rangle + \beta\,|000\rangle$.

On the other hand, the 3-qubit phase-flip code assumes that the quantum channel is the *phase-flip channel* that adds phase-flip errors, Z, to each individual qubit independently such that the probability of error on a qubit is p_Z and the probability of no error is $p_I = 1 - p_Z$. The diagram illustrating the behavior of the channel is shown in Figure 1.9.

The 3-qubit phase-flip code works identically as the bit-flip code except for its utilization of the $HZH = X$ relationship to change the behavior of the channel from adding Z errors to adding

X errors [7]. This is done physically by adding a Hadamard gate before the channel and another after the channel. The diagram of the quantum circuit of the code is shown in Figure 1.10.

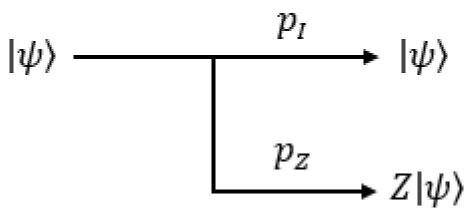

Figure 1.9: Diagram of the action of the phase-flip quantum channel.

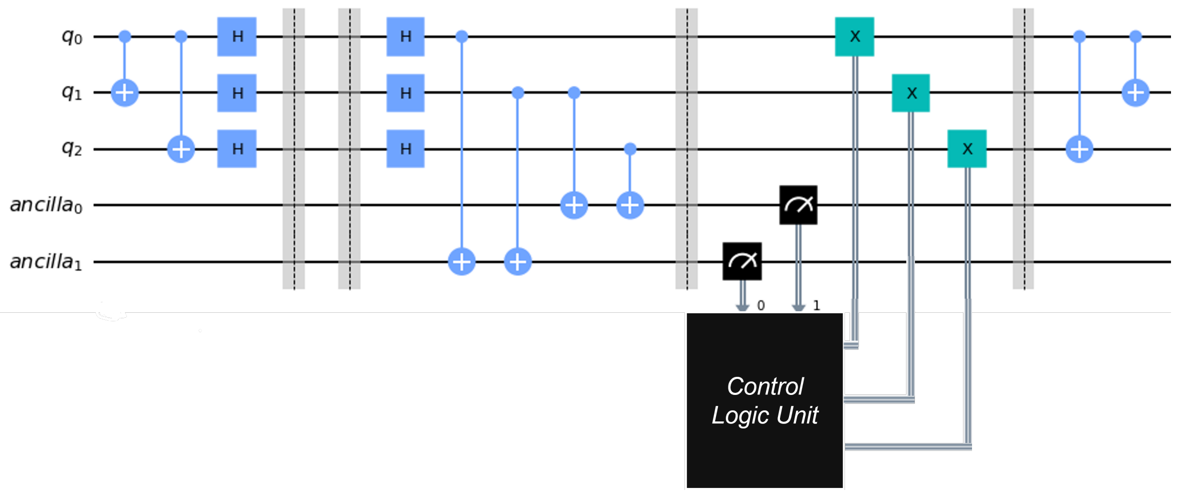

Figure 1.10: Quantum circuit of the 3-qubit bit-flip quantum code.

Peter Shor developed the first QECC, the *9-qubit code*, that uses a total of 9-qubits to correct errors of weight 1. The code is a concatenation of both 3-qubit codes [12]. To read more about the Shor's 9-qubit code refer to [7]-[9].

1.2.5 Stabilizer Codes

A $[[n,\ k]]$ quantum code Q constructed using the *Stabilizer formalism* [13] [14] is defined as:

$$Q = \{|\psi\rangle : S_j\,|\psi\rangle = |\psi\rangle\,, \forall S_j \in \mathcal{S}\},\ \mathcal{S} = \{S_j\} \subset \mathcal{P}_n, \qquad \text{(Eq. 1.23)}$$

where $-I^{\otimes n} \notin \mathcal{S}$, $|\mathcal{S}| = n - k$, and $\mathcal{S}$ is an Abelian multiplicative subgroup of $\mathcal{P}_n$ referred to as the *Stabilizer Group*. All members of $\mathcal{S}$, the *stabilizers*, are Pauli operators that act as an identity when multiplied by the codewords $|\psi\rangle \in Q$, i.e. $|\psi\rangle$ are $+1$ eigenkets of the stabilizers. Thus, defining $\mathcal{S}$ defines all $|\psi\rangle$ and, consequently, Q.

Stabilizer codes detect errors through *syndrome measurement*. The *syndrome* s is a binary vector ($s \in \mathbb{F}_2^{n-k}$) obtained by multiplying the stabilizers by the erroneous codeword and checking whether the each stabilizer commutes or anti-commutes with the error. If the codeword $|\psi\rangle$ is transmitted and the channel adds the error E such that the received erroneous codeword is $|\phi\rangle = E\,|\psi\rangle$, then the syndrome will be:

$$s = \begin{bmatrix} \lambda_1 & \lambda_2 & \dots & \lambda_{n-k} \end{bmatrix}, \qquad \text{(Eq. 1.24)}$$

where

$$\lambda_i = \begin{cases} 0, & [S_i,\ E] = 0 \\ 1, & \{S_i,\ E\} = 0 \end{cases},\ \text{for } S_i \in \mathcal{S},\ i = 1, \dots, n-k. \qquad \text{(Eq. 1.25)}$$

or, in another form

$$\lambda_i = \begin{cases} 0, & [S_i,\ |\phi\rangle] = 0 \\ 1, & \{S_i,\ |\phi\rangle\} = 0 \end{cases},\ \text{for } S_i \in \mathcal{S},\ i = 1, \dots, n-k. \qquad \text{(Eq. 1.26)}$$

Each correctable error — of weight less than or equal to the maximum weight the code is capable of correcting — has a unique corresponding syndrome such that the code is capable of identifying the error and correcting it. However, syndrome measurement will yield a syndrome associated with a correctable error if the actual error that occurred is beyond the capabilities of the code.

Writing the stabilizer as $S_i = X_{u_i} Z_{v_i}$, allows us to define the *quantum-check matrix* A as:

$$A = \begin{bmatrix} u_1 & | & v_1 \\ \vdots & | & \vdots \\ u_{n-k} & | & v_{n-k} \end{bmatrix}, \qquad \text{(Eq. 1.27)}$$

such that the syndrome of an error represented as $e = X_a Z_b = \begin{bmatrix} b & | & a \end{bmatrix}$ can be calculated by:

$$s = eA^T. \qquad \text{(Eq. 1.28)}$$

An example of stabilizer codes is the $[[5,\ 1]]$ code developed by Peter Shor and David DiVincenzo [14][15], which is the smallest QECC capable of correcting any error on a single qubit. The stabilizer group of the code with cardinality $5 - 1 = 4$ is:

$$\mathcal{S} = \{X\ Z\ Z\ X\ I,\ I\ X\ Z\ Z\ X,\ X\ I\ X\ Z\ Z,\ Z\ X\ I\ X\ Z\},$$

therefore, its quantum-check matrix is:

$$A = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & | & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & | & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & | & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & | & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

To illustrate syndrome measurement using Eq. 1.28, let $E = X\ I\ I\ I\ I$, then the syndrome is:

$$
\begin{aligned}
s &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & | & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & | & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & | & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & | & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & | & 1 & 0 & 0 & 0 & 1 \end{bmatrix}^T \\
&= \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix},
\end{aligned}
$$

however, the error $E = X\ Z\ I\ X\ X$ returns an identical syndrome.

1.2.6 Calderbank-Shor-Steane Codes

Calderbank-Shor-Steane (CSS) codes are a special case, a subclass, of stabilizer codes that use classical linear codes to construct a QECC, acting as a link between classical and quantum coding [16][17][18]. A CSS code Q_{CSS} is constructed using two classical linear codes C_1 with $[n,\ k_1]$ and parity-check matrix H_1 and C_2 with $[n,\ k_2]$ and generator matrix H_2. C_1 and C_2 must satisfy the conditions [7]: $C_2 \subset C_1$, both C_1 and $C_2^{\perp}$ are capable of correcting errors of weight t or less, and

$$H_1 G_2^T = \mathbf{0}, \tag{Eq. 1.29}$$

such that Q_{CSS} is capable of correcting errors of weight t and less.

The quantum-check matrix of the code is:

$$A = \begin{bmatrix} H_1 & | & \mathbf{0} \\ \mathbf{0} & | & G_2 \end{bmatrix}, \tag{Eq. 1.30}$$

where the **0** are the zero matrices. The dimensions of A are $(n-k_1+k_2)\times 2n$ where the total number of the qubits in the code is n, the number of information qubits is $k_2 - k_1$, and $R = (k_2 - k_1)/n$ [8]. Hence, the CSS code is a Q_{CSS} $[[n,\ k_2 - k_1]]$ code capable of correcting weight t errors with $\dim(Q_{CSS}) = \dim(C_2/C_1) = \dim(C_2) - \dim(C_1)$. The code Q_{CSS} has $2^{k_2-k_1}$ codewords $|\psi_i\rangle$

that are linear combinations of 2^{k_2} states:

$$|\psi_i\rangle = |x + C_2\rangle = \frac{1}{\sqrt{|C_2|}} \sum_{y \in C_2} |x + y\rangle \text{ for } x \in C_1 \text{ and } i = 1,\ ...,\ 2^{k_2 - k_1}. \qquad \text{(Eq. 1.31)}$$

where $|C_2| = 2^{k_2}$ and x is a coset leader such that each $x + C_2$ is a distinct coset in the quotient group C_1/C_2.

If H_1 and G_2 are represented as:

$$H_1 = \begin{bmatrix} h_1 \\ \vdots \\ h_{n-k_1} \end{bmatrix} \text{ and } G_2 = \begin{bmatrix} g_1 \\ \vdots \\ g_{k_2} \end{bmatrix},$$

where $h_i,\ g_j \in \mathbb{F}_2^n$ are $1 \times n$ binary vectors, then through *A*, the generators of the CSS code are:

$$S_i = \begin{cases} X_{h_i}, & 1 \le i \le n - k_1 \\ Z_{g_{i-n-k_1}}, & n - k_1 < i \le n - k_1 + k_2, \end{cases},\ S_i \in \mathcal{S}, \qquad \text{(Eq. 1.32)}$$

hence, all the stabilizer are composed of X only or Z only, which is one of the main characteristics of the CSS construction. Therefore, C_1 is used to detect X errors and C_2 is used to detects Z errors [9].

An example of a CSS code is the $[[7,\ 1]]$ *Steane Code* [19] constructed using the $[7,\ 4]$ Hamming code (C_1) and its dual the $[7,\ 3]$ code ($C_2 = C_1^{\perp}$) with $H_1 = G_2 = H$ and $H_2 = G_1 = G$ as defined in Eq. 1.10 in Section 1.1.2. The $[[7,\ 1]]$ code has 1 information qubit, is capable of correcting any error on any single qubit ($t = 1$), has coding rate $1/7$, and has 2 codewords where each codeword is a superposition of $2^3 = 8$ states. The quantum-check matrix of the code, according to Eq. 1.30, is:

$$A = \left[\begin{array}{ccccccc|ccccccc} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right],$$

such that

$$\mathcal{S} = \{X_{1101100},\ X_{1011010},\ X_{0111001},\ Z_{1101100},\ Z_{1011010},\ Z_{0111001}\},$$

is the stabilizer group of the code. The *Steane code*, per Eq. 1.31, has the codewords:

$$\begin{aligned} |\psi_1\rangle &= |0000000 + C_2\rangle = |0000000\rangle + |0110011\rangle + |1010101\rangle + |1100110\rangle \\ &\quad + |0111100\rangle + |1011010\rangle + |1101001\rangle \\ |\psi_2\rangle &= |1111111 + C_2\rangle = |1111111\rangle + |1001100\rangle + |0101010\rangle + |0011001\rangle \\ &\quad + |1000011\rangle + |0100101\rangle + |0010110\rangle \end{aligned}$$

where the codewords of C_1 and C_2 are as obtained in Table 1.1 and Table 1.2 respectively.

2. METHODOLOGY

The objective of this research project is to evaluate the performance of CSS codes by conducting Monte Carlo simulations via Python programming and obtaining the probability of error per word P_{ew} of the code for different channel error probability p. The script written is divided into multiple modules: code construction, CLU initialization, codeword transmission, quantum channel, syndrome measurement, CLU, and error-correction. In addition, the script operates in two modes of operation: the *demo mode* and the *general mode*. The block diagram of the script is shown in Figure 2.1.

For the script, the main user inputs are the parity-check matrix H_1 of the first classical code C_1 $[n,\ k_1]$ and the generator matrix G_2 of the second classical code C_2 $[n,\ k_2]$, both used to construct the CSS code $[[n,\ k_1 - k_2]]$ according to the theory explained in Section 1.2.6. Given that H_1 and G_2 satisfy the conditions of CSS code construction, the code is constructed, the CLU is initialized, a codeword is chosen for transmission, the quantum channel adds an error to the transmitted codeword, syndrome measurement is conducted, syndrome is passed to the CLU which returns the appropriate action, and, finally, the action is carried out to correct the error if possible.

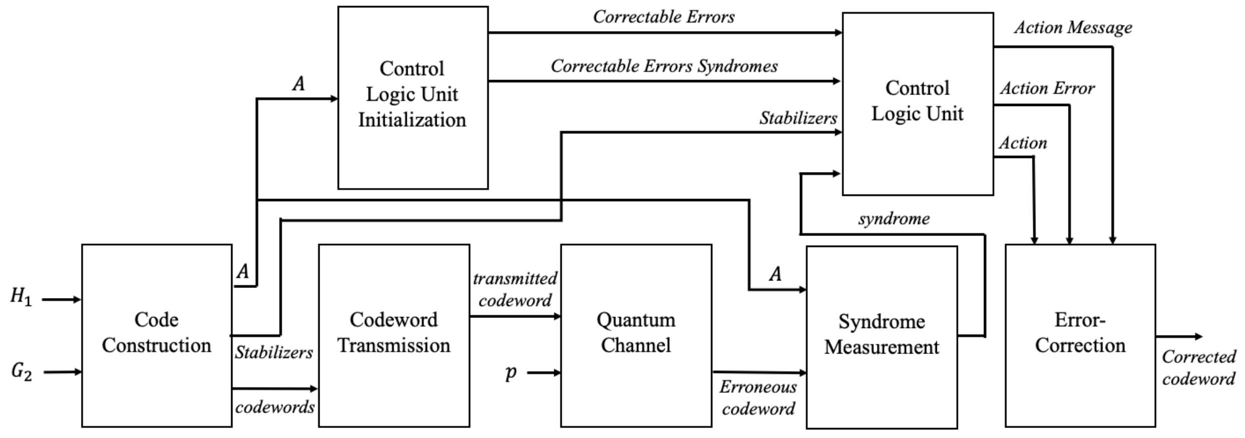

Figure 2.1: Block diagram of the python script.

The script — as explained in the following sections — was used to simulate and evaluate the performance of the $[[7,\ 1]]$ *Steane Code* and the $[[31,\ 11]$ CSS code constructed using the $[31,\ 21]$ BCH classical code and its dual, the $[31,\ 10]$ code. The code performance evaluation is done via obtaining P_{ew} for each value of p, where p is varied from 0 (good channel) to 1 (bad channel).

2.1 Code Construction

The *Code Construction* module is responsible for obtaining the quantum-check matrix A as defined in Eq. 1.30, the stabilizers of the code as defined in Eq. 1.32, and the codewords of the quantum code using Eq. 1.31. The module depends on the user inputted H_1 and G_2. However, the construction is conducted if and only if H_1 and G_2 satisfy the following requirements:

1. The codes C_1 and C_2 are of length n.

2. The codes C_1 and C_2 have a different number of information bits ($k_1 \neq k_2$).

3. The parity-check matrix H_1 of C_1 and the generator matrix G_2 of C_2 satisfy Eq. 1.29:

$$H_1 G_2^T = \mathbf{0}.$$

2.2 CLU Initialization

The *CLU* module is responsible for identifying the error the channel introduced by comparing the measured syndrome with the list of syndromes of correctable errors. The *CLU Initialization* module is responsible for obtaining the complete list of correctable errors and their corresponding syndromes. The module uses the quantum Hamming bound defined in Eq. 1.13 to determine the maximum possible weight the code is capable of correcting, and then calculates syndromes and obtains the list of correctable errors via comparing syndromes and keeping the errors of lowest weight that have unique syndromes. The input to the module is the quantum-check matrix, and the outputs are the list of the correctable errors and their corresponding syndromes.

2.3 Codeword Transmission

The *Codeword Transmission* module selects one of the quantum code's codewords to transmit either at random or per user input depending on the mode of operation. The module's input is the list of the CSS code's codewords, and outputs the transmitted codeword.

2.4 Quantum Channel

The *Quantum Channel* module simulates a quantum channel that acts independently on each qubit. In addition, X, Y, and Z have an equal probability distribution ($p/3$) such that the probability of error on a single qubit is p. The diagram representing the behavior of the quantum channel is shown in Figure 2.2. However, in demo mode, the user has the option of inputting the error added by the channel. The inputs to the module are the transmitted codeword and p, and the outputs are the erroneous transmitted codeword and the channel error that is used only for display in the demo mode.

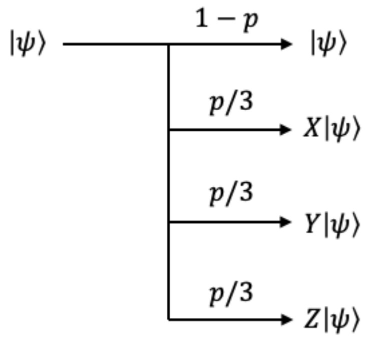

Figure 2.2: Diagram illustrating the action of the quantum channel on a single qubit.

2.5 Syndrome Measurement

The *Syndrome Measurement* module calculates the syndrome of the erroneous codeword using Eq. 1.24 and Eq. 1.26. The inputs of the module are the erroneous codeword and the stabilizers, and the output is the syndrome measured.

2.6 Control Logic Unit

The *CLU* module compares the measured syndrome to the list of the syndromes of the correctable errors of the CSS code to identify the error added by the quantum channel. The module's inputs are the measured syndrome, the stabilizers, the list of correctable errors, and their corresponding syndromes. The module outputs the control signal to activate the error-correction module, the action message containing what the module found, and the action error that the error-correction module should apply. There are two possible findings of the module:

1. Measured syndrome is in the syndrome list: The error-correction module is activated, the action error is equal to the identified error, and the message indicates the error identified.
2. Measured syndrome is not in the syndrome list: The error-correction module is not activated, the action error is equal to the identity, and the message indicates that an error was detected but it is beyond the capabilities of the code.

The error-correction module operations depends solely on the possible findings of this module.

2.7 Error-Correction

The *Error-Correction* module's operation depends on the findings of the *CLU*. If activated, the module applies the error identified on the erroneous codeword to correct and retrieve the originally transmitted codeword.

2.8 Modes of Operation

As mentioned earlier, the script has two modes of operations: the *demo mode* and the *general mode*. The demo mode allows the user to control most modules of the script, while in general mode, the modules are fully automated. the full automation of the modules to acquire plots of probability of error (code failure) of the code over a range of p. Thus, providing an evaluation of the performance of the code with the given quantum channel behavior.

2.8.1 Demo Mode

In the demo mode, after the user inputs H_1 and G_2 to construct the CSS code — which prints the parameters of the code and the codewords and initialize the CLU — and after the user selects the demo mode, the user manually selects the codeword to be transmitted or choose for it to be selected at random. After that, the user manually inputs the error to be added by the quantum channel or select for the channel to apply a random error where the user must select the value of p. Finally, the script measures and displays the syndrome, the identified error, and the action taken.

By allowing the user to select the parameters of the modules, the demo mode provides the capability to experiment with the code and test its performance on a case-by-case basis that is more interactive and educational.

2.8.2 General Mode

After the user inputs H_1 and G_2 and selects the general mode, the fully automated process begins, where the codeword to be transmitted and the channel error are random. The script runs through the modules transmitting different codewords iteratively for a constant number of iterations for each p in the range between 0 and 1.

The general mode evaluates the performance of the code by obtaining the probability of error per word $P_{ew}(p)$ for each value of probability of error in the quantum channel p. Theoretically, $P_{ew}(p)$ For any $[[n,\ k]]$ quantum code capable of correcting weight t errors, operating with the quantum channel assumed, the probability of error per word is given by:

$$P_{ew}(p) = 1 - \mathbb{P}(w(error) \leq t) = 1 - \sum_{i=0}^{t} \binom{n}{i} p^i (1-p)^{n-i}. \qquad \text{(Eq. 2.1)}$$

In the simulation, the script compares the correct codeword with the originally transmitted and takes note of the number of times the code fails to correct the erroneous codeword to the originally transmitted codeword. For each value of p, the script obtains P_{ew} by finding the ratio of code failure to total number of transmitted words. Finally, after the script is done with all values of p.

3. RESULTS

The general mode of the Python script explained in Chapter 2 was used conduct Monte Carlo simulations to evaluate the performance of the $[[7,\ 1]]$ Steane code and the $[[31,\ 11]]$ CSS code. Their performance were evaluated via obtaining the probability of error per word P_{ew} for different probabilities of quantum channel error p for the Monte Carlo simulations and theoretically using Eq. 2.1. The following subsections present the plots of P_{ew} for different values of p for the two quantum codes.

3.1 $[[\mathbf{7}, \mathbf{1}]]$ Steane Code

Figure 3.1 shows the plot of P_{ew} for the code versus p in logarithmic scale for the Monte Carlo simulation of the $[[7,\ 1]]$ code and the theoretical values derived from Eq. 2.1 as:

$$P_{ew}(p) = 1 - \sum_{i=0}^{1} \binom{7}{i} p^i (1-p)^{7-i} = 1 - (1-p)^7 - 7p(1-p)^6, \qquad \text{(Eq. 3.1)}$$

which can be approximated as

$$P_{ew}(p) = 21p^2 + o(p^2) \approx 21p^2. \qquad \text{(Eq. 3.2)}$$

3.2 $[[\mathbf{31}, \mathbf{11}]]$ CSS Code

Figure 3.2 shows the plot of P_{ew} for the code versus p in logarithmic scale for the Monte Carlo simulation of the $[[31,\ 11]]$ code and the theoretical values obtained derived from Eq. 2.1 as:

$$\begin{aligned} P_{ew}(p) &= 1 - \textstyle\sum_{i=0}^{2} \binom{31}{i} p^i (1-p)^{31-i} \\ &= 1 - (1-p)^{31} - 31p(1-p)^{30} - 465p^2(1-p)^{29}, \end{aligned} \qquad \text{(Eq. 3.3)}$$

which can be approximated as

$$P_{ew}(p) = 4495p^3 + o(p^3) \approx 4495p^3. \qquad \text{(Eq. 3.4)}$$

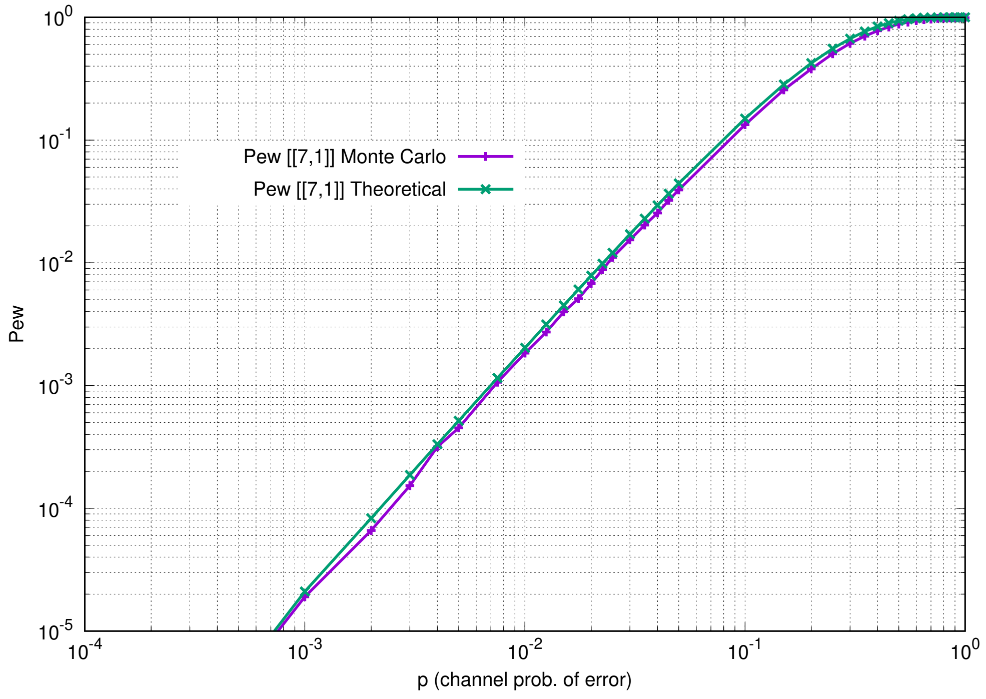

Figure 3.1: Plot of the probability of error per word P_{ew} versus the quantum channel probability of error p of the $[[7,\ 1]]$ code for Monte Carlo simulation and theoretical values in logarithmic scales.

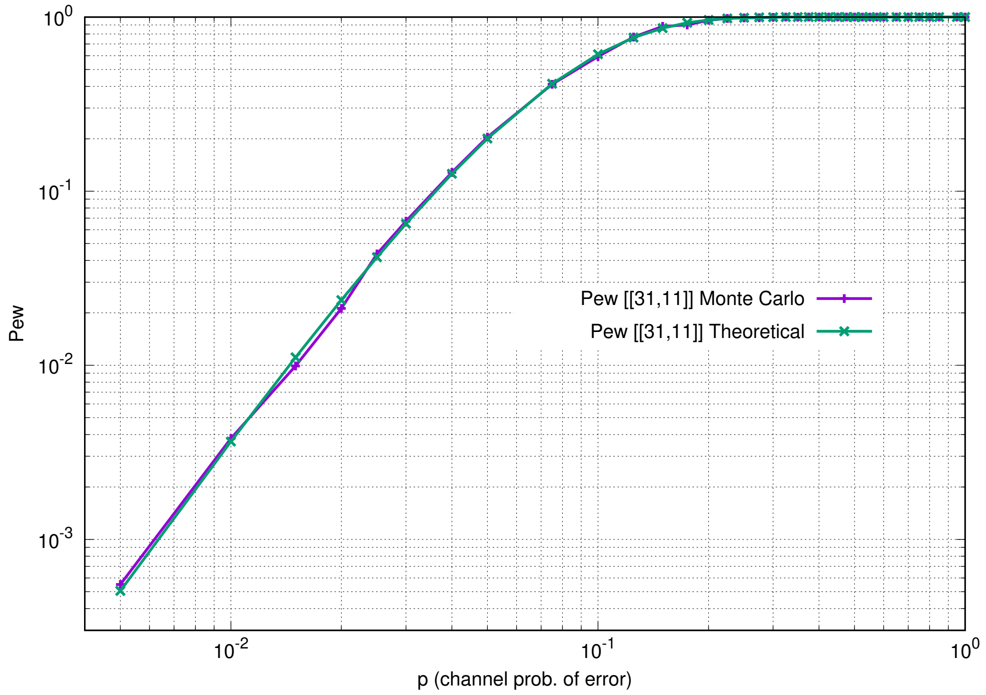

Figure 3.2: Plot of the probability of error per word P_{ew} versus the quantum channel probability of error p of the $[[31,\ 11]]$ code for Monte Carlo simulation and theoretical values in logarithmic scales.

4. CONCLUSIONS

4.1 Evaluation of Performance

From the results displayed in Figure 3.1 and Figure 3.2 presented in Chapter 3, it is evident that the Monte Carlo simulations were verified by the theoretical values of the $[[7,\ 1]]$ and $[[31,\ 11]]$ codes obtained via Eq. 3.1 and Eq. 3.3 respectively.

In addition, the $[[7,\ 1]]$ code performs better (has less probability of error per word P_{ew}) for higher values of channel error probability p than the $[[31,\ 11]]$ code, and $[[31,\ 11]]$ performs better for lower values of p. This is a result of the large coefficient of p^3 in Eq. 3.4 relative to the coefficient of p^2 in Eq. 3.2. This behaviour can also be observed in Figure 4.1 where the performance of both codes are displayed simultaneously.

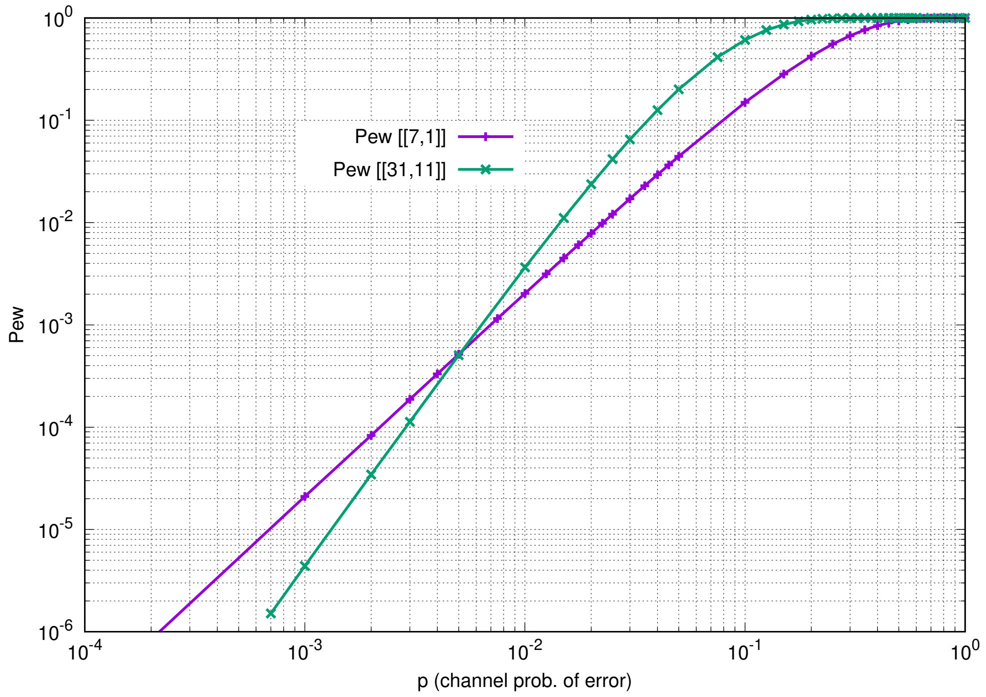

Figure 4.1: Plot of the probability of error per word P_{ew} versus quantum channel's probability of error p of the $[[7,\ 1]]$ and $[[31,\ 11]]$ codes in logarithmic scales.

This region of p in which the $[[7,\ 1]]$ code outperforms $[[31,\ 11]]$ code can be derived mathematically using Eq. 3.2 and Eq. 3.4 as:

$$21p^2 < 4495p^3,$$
$$\tfrac{21}{4495} = 4.671 \times 10^{-3} < p.$$

The $[[31,\ 11]]$ code is better when $4.671 \times 10^{-3} > p$. The point of intersection at $p \approx 5 \times 10^{-3}$ in Figure 4.1 illustrates and verifies this comparison between the performance of the two codes.

4.2 Python Script

As evident by the results obtained, the Python script proved to be an accurate and reliable tool in simulating and evaluating the performance of CSS codes. However, due to the exponential complexity of conducting Monte Carlo simulations of QECCs, the Python script's execution time is considerably long, imposing limits on the possible simulations to be conducted. It is possible, that translating the script to another programming language such as C may reduce the execution time considerably and increase the capabilities of the script.

REFERENCES

[1] R. E. Blahut, *Algebraic Codes for Data Transmission*. Cambridge University Press, 2003.

[2] F. MacWilliams and N. Sloane, *The Theory of Error-Correcting Codes*. North-holland Publishing Company, 2nd ed., 1978.

[3] R. W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 29, pp. 147–160, Apr. 1950.

[4] S. Lin and D. J. Costello, *Error control coding: fundamentals and applications*. Upper Saddle River, NJ: Pearson/Prentice Hall, 2nd ed., 2004.

[5] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, pp. 21–28, Jan. 1962.

[6] J. Roffe, "Quantum error correction: an introductory guide," *Contemporary Physics*, vol. 60, p. 226–245, Jul. 2019.

[7] I. Djordjevic, *Quantum Information Processing, Quantum Computing, and Quantum Error Correction: An Engineering Approach*. Elsevier Inc, 2nd ed., 2012.

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

[9] D. Lidar and T. Brun, *Quantum Error Correction*. Cambridge University Press, 1st ed., 2013.

[10] W. K. Wootters and W. H. Zurek, "A single quantum cannot be cloned," *Nature*, vol. 299, pp. 802–803, Oct. 1982.

[11] A. Steane, "A tutorial on quantum error correction," *Quantum Computers, Algorithms and Chaos*, vol. 162, pp. 1–32, Jan. 2006.

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

[13] D. Gottesman, “Class of quantum error-correcting codes saturating the quantum hamming bound,” *Physical Review A*, vol. 54, pp. 1862–1868, Sep. 1996.

[14] D. Gottesman, “Stabilizer codes and quantum error correction,” 1997.

[15] D. P. DiVincenzo and P. W. Shor, “Fault-tolerant error correction with efficient quantum codes,” *Physical Review Letters*, vol. 77, pp. 3260–3263, Oct. 1996.

[16] A. R. Calderbank and P. W. Shor, “Good quantum error-correcting codes exist,” *Physical Review A*, vol. 54, pp. 1098–1105, Aug. 1996.

[17] A. M. Steane, “Error correcting codes in quantum theory,” *Phys. Rev. Lett.*, vol. 77, pp. 793–797, Jul. 1996.

[18] A. M. Steane, “Simple quantum error-correcting codes,” *Phys. Rev. A*, vol. 54, pp. 4741–4751, Dec. 1996.

[19] A. M. Steane, “Multiple-particle interference and quantum error correction,” *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 452, pp. 2551–2577, Nov. 1996.