الجـــمـهـوريـــــة الجـــــزائـريـة الديمقـــــراطـيـــــة الـشـــــعـبـيـــــة

DEMOCRATIC AND POPULAR REPUBLIC OF ALGERIA

وزارة التـــــعـليـــم العـــــــالي والبحـــــث العلـــــــمي

Ministry of Higher Education and Scientific Research

جـــــــــــامعة أبي بكـر بلـــــــقـايد – تـلمســـــــــــــان –

University Abou Bekr Belkaïd – Tlemcen –

Faculty of TECHNOLOGY

Thesis

Presented for the **MASTER**'s degree

In : Telecommunication

Major: Optical Telecommunication

Par :

Dra Rihab Aya

Fidouh Souhila

Subject

Harnessing Quantum Communications Techniques in Multiple Access Networks

Argued, on / / 2024, before the jury composed of:

Dr. ABDELMALEK Abdelhafid	MCA	Tlemcen. Univ	President
Dr. SEDJELMACI Amina Nadjet	MCB	Tlemcen. Univ	Expert
Dr. BENDIMERAD Mohammed Yassine	MCA	Tlemcen. Univ	Supervisor

Academic Year 2023/2024

In the name of Allah the most merciful,
to whom I owe everything.

إهداء

قال تعالى: (قل اعملوا فسيرى الله عملكم ورسوله والمؤمنون)

إلهي لا يطيب الليل إلا بشكرك ولا يطيب النهار الا لطاعتك

ولا تطيب اللحظات إلا بذكرك ولا تطيب الآخرة الا بعفوك

ولا تطيب الجنة إلا برؤيتك.

الله

أهدي هذه المذكرة

إلى أمي **بن زيان رشيدة** رحمها الله، على الرغم من أنني لم أعرف دفء

أحضانك، ولا سمعت همساتك الحنونة إلا أن ذكراك تنير دربي فأنت

تعيشين في كل خطوة أخطوها وفي كل إبتسامة أبتسمها فإن ذكراك تشع

مضيئة هذا المكان.

إلى أمي **بن زيان عمارية**، موطن الحنان والحب، من كانت سندي في السراء

والضراء، من اجتهدت على نشأتي وتربيتي، من بها أعلو وبها أرتكز،

أنت الملاذ الآمن والسند والينبوع الذي يروي ضمأ روحي.

إلى والديّ **درى جمال** و**عبد اللّاوي عبد العزيز**، انتما العمودان

اللذان أستند عليهما، في كل نجاح أحققه وفي كل تحدي أتغلب عليه

أرى بريق حبكما يلهمني.

إلى أخي وحبيب قلبي **عبد اللاوي محمد نجيب**، فأنت الجبل الذي أسند عليه

نفسي عند الشدائد، أنت الشريان الذي يوصل الدم إلى القلب فقد قال تعالى:

(سنشد عضدك بأخيك).

إلى أختي **عبد اللاوي سارة جازية**، فأنت قوتي وإلهامي، شريكتي في الفرح

والحزن فما الأخت إلا اتكاء روح على روح.

إلى أختي **عبد اللاوي نور الإيمان**، فأنت أنيسة روحي وضوئي وكأنك خلقت

لتكوني ملاكا لقلبي ورضا لأوقاتي.

إلى أخي **درى محمد لخضر**، فأنت أجمل ميراث من أمي وأبي.

إلى خالاتي **بن زيان خيرة** و **بن زيان مليحة** ،من كلماتهم تمنحني الحكمة و في نصائحهم أجد القوة.

إلى أختي **درى دعاء** وأخي **درى آدم** و **بكاي نزيهة**.

إلى من شاركت الضحك والأحلام والمخاوف معهم، **أصدقائي**

إلى من كان له الفضل في تعليمي منذ بداية مسيرتي إلى نهايتها

درى رحاب آية

Acknowledgement

In embarking upon this academic endeavor, I have been fortunate to receive support and guidance from a multitude of individuals whose contributions have been instrumental in shaping the culmination of this work. As I reflect on the journey that has led me to this moment, it is with deep gratitude and humility that I acknowledge the invaluable roles played by those who have stood by my side, offering encouragement, wisdom, and unwavering support. Their influence has not only enriched the process of research and discovery but has also left an indelible mark on my personal and intellectual growth.

To my dearest parents **FIDOUH Habib** and **FIDOUH Houria**, whose unwavering support are heralds of hope and liberty, akin to the beloved constant stars,

In the gentle dawning of each morn and the tranquil hush of night,

Your steadfast devotion has been my guiding beacon,

Leading me through the tempestuous seas of scholarly pursuit.

Your belief hath kindled a fire within,

And through your eyes, I have garnered the fortitude to reach for the heavens.

To my dear brother **FIDOUH Mohammed**, a wellspring of inspiration and support,

Your presence, has been a guiding light throughout this journey.

In your unwavering belief in my abilities, I found strength,

And in your quiet encouragement, a source of motivation.

Your resilience and dedication have been a constant reminder of the importance of perseverance,

And for this, I am eternally grateful.

To my cherished friends, the bearers of mirth and fellowship,

With warmth, the wellspring of joy, whose laughter is a wondrous melody,

Serenity's smile, within I found a sanctuary of solace,

Steadfast belief in me, a sanctuary of support.

Cherished companionship, a pillar through the challenges,

Camaraderie that comforted my darkest hours,

Nurturing nature, a testament to nurturing aspirations.

Radiant spirit, a reflection of unwavering solidarity,

Steadfast presence, an enduring source of grace.

Hearts intertwined with mine, providing hope and harmony.

Through the tumultuous tides of this academic journey, you stood as stalwart pillars,

Your friendship a melodious symphony that played through my darkest hours,

A testament to the sublime beauty of shared dreams and steadfast camaraderie.

And to myself, the tireless wayfarer,

In the echoing silence of countless sleepless nights and the dawning light of myriad challenges,

Thou embraced resilience, transforming whispers of doubt into resounding triumphs.

In the mirror's reflection, I see a soul that hath dared greatly,

A heart that hath endured, and a mind that hath sought wisdom’s embrace with fervor.

To the sweat, the tears, and the relentless quest for knowledge,

I extend a hand of gratitude, for thou art the master of this journey.

This thesis is a rich tapestry, woven with threads of love, friendship, and unwavering effort.

It stands as a monument to the power of steadfast support and intrinsic resolve,

A symphony of gratitude to those who walked beside me,

And a humble salute to the tireless spirit within.

Fidouh Souhila

Gratitude

To my esteemed supervisor Mr BENDIMERAD Yassine professor at the university of Tlemcen,

In the vast expanse of academia, you stood as a beacon of wisdom and guidance, illuminating the path of scholarly pursuit with your profound expertise and unwavering support. Your mentorship has been invaluable, shaping not only the trajectory of my academic journey but also fostering personal and intellectual growth beyond measure.

Beyond the realm of academia, your kindness, encouragement, and genuine interest in my success have touched my heart profoundly. Your unwavering belief in my abilities has instilled in me the confidence to overcome challenges and strive for excellence in all my endeavors.

As I embark on the next chapter of my academic and professional journey, I carry with me the invaluable lessons learned under your guidance. Your mentorship has not only shaped my scholarly pursuits but has also inspired me to pay it forward, nurturing future generations of scholars with the same dedication and compassion that you have shown me. With deepest gratitude and utmost respect.

Content Table

Figure List

Table List

General Introduction

The primary objective of this master thesis is to explore the potential of harnessing quantum communication in multiple access networks, coupled with quantum error correction techniques, to address the growing demand for secure and efficient data transmission in the era of burgeoning digital connectivity. This research aims to elucidate the feasibility and performance implications of deploying quantum technologies in real-world communication infrastructures. Additionally, the thesis seeks to delve into the intricacies of quantum error correction, a pivotal aspect of quantum communication systems that mitigates the adverse effects of noise and decoherence, thereby preserving the integrity of transmitted quantum information. Through a comprehensive analysis of existing quantum error correction codes and their applicability to multiple access scenarios. This research endeavors to pave the way for the development of robust, high-performance communication technologies capable of meeting the evolving demands of modern society.

The thesis plan consists of Four chapters, each focusing on a distinct aspect of communication and error correction. Chapter One provides an extensive overview of quantum communication, tracing its evolution from classical systems and highlighting its emergence as a solution to the limitations of classical communication. It explores fundamental concepts such as qubits, superposition, and entanglement, elucidating their significance in quantum information processing. Additionally, it delves into various applications of quantum communication, including cryptography, error correction, and computing, underscoring its transformative potential in information technology. Chapter Two shifts focus to classical error correction codes, offering a detailed examination of their methods and significance in ensuring reliable data transfer. It covers block codes like Hamming and Low-Density Parity Check codes, cyclic codes such as BCH codes, and convolutional codes with decoding methods like the Viterbi algorithm. Chapter Three delves into the intricate realm of quantum error code correction, emphasizing its critical role in maintaining the integrity of quantum information amidst noise and decoherence. It explores various types of quantum error codes, their unique qualities, and practical implementation challenges. A comparative analysis reveals their relative strengths and limitations, highlighting the importance of ongoing research in improving quantum computing

and information processing paradigms. Finally, Chapter Four concludes with a discussion on performance criteria and practical insights gained from Python simulations.

Chapter 1

Introduction to Quantum Communication

The evolution of communications is a multidimensional journey that spans millennia of human history, from basic early communication modes to sophisticated digital networks. In the early stages, human communication was based on basic pictograms, spoken language, and gestures, which served as the foundation for idea sharing throughout ancient groups. The emergence of writing systems, such as cuneiform and hieroglyphics, was a watershed moment, allowing knowledge to be recorded and passed down through generations. With the invention of the telegraph in the nineteenth century, communication crossed geographical boundaries for the first time, transforming long-distance correspondence and spurring the globalization of information networks. This era saw great developments in electrical and mechanical engineering, as well as the establishment. The emergence of telegraph businesses and worldwide cable networks laid the framework for today's interconnected globe. Mass media technologies, such as radio and television, proliferated in the twentieth century, revolutionizing information dissemination while also shaping worldwide public opinion and cultural standards. The internet's development in the late twentieth century represented a paradigm shift in communication, democratizing access to information and enabling unprecedented levels of connectivity around the world. This digital revolution has resulted in the rise of new communication platforms and social networks, profoundly changing how people connect, consume media, and participate in civic debate.

Furthermore, recent advances in quantum communication technology promise even faster, more secure communication networks, with applications ranging from cryptography to distributed computing. Part of this thesis seeks to elucidate the complex interplay between communication technologies, societal dynamics, and cultural evolution by delving deeply into these historical and technological developments, shedding light on the transformative power of human connectivity in shaping the world we live in today.

1.1 Limitations of Classical Communication Systems

Despite the fundamental use of classical communications nowadays, it faces many challenges and limitations in terms of performance and efficiency. These restrictions and limitations have emerged due to several characteristics and constraints.

1.1.1 Limited Range

One of the most well-known restrictions of classical communication is its limited range. The data can become inaccurate and faulty due to traveling over long distances [1][3][4][5]., and it is caused by:

Attenuation: As the data or information is transmitted over the transmission channel, the signal loses strength, which can be affected by the physical deformation of the transmission medium that increases with distance[3][5].

Noise: Because the signal travels over long distances, it becomes more and more exposed to noise whether it is Thermal, Shot noise or interference which will cause signal to noise ration degrade[3][4].

Dispersion: Intersymbol interference can be caused during transmission in some particular medium, such as optical fibers, which can contain several different frequencies that cause the signal to spread[3].

Nonlinear Affects: The signal can be distorted by several nonlinear effects, for instance, self-phase modulation, four-wave mixing, and stimulated Raman scattering[3].

1.1.2 Bandwidth Constraints

In classical communications, the bandwidth is most important due to its control over the quantity and rapidity of the data transmitted.

Data Transmission Rate: The bandwidth determines how much data can be transmitted, which affects the data transmission rate. The higher the bandwidth, the faster the transmission rates[7][8].

Data Bottlenecks: Another constraint is when the amount of data transferred is unable to be processed by the transmission medium's capacity, which is called "data bottlenecks." [7].

Signal Quality: The limitations of the bandwidth can also result in the limitation of the frequency range, which causes interference, signal distortion, and reduced SNR [8].

1.1.3 Susceptibility to interference

The vulnerability of classical systems to interference has caused a real challenge in this field, and that is due to various sources:

Electromagnetic Interference (EMI): The signal quality can be affected by additive noise, which is conducted by unwanted voltages or currents[9][10]

Frequency mixing: The signals transmitted at several different frequencies can be distorted[11]

Crosstalk: When signals from nearby channels interfere with one another, it can result in signal overlap, which is known as "crosstalk.". This interference is common in high-density communication systems.[11][12][13]

Wireless Interference: Multipath interference and signal deterioration can be the result of external sources of interference, such as rival networks and environmental conditions, which can affect wireless communication systems.[14]

1.1.4 Security Vulnerabilities

The classical communication systems contain many loopholes when it comes to system security, which can compromise sensitive data.

Zero-day vulnerability: This invokes serious threats because it is instant and attackers can exploit it before a security measure is put in place[15].

Remote Code Execution: This vulnerability allows for the change or execution of malicious code directly into the system[15][16].

Weak Authentication: Weak authentication can create opportunities for unauthorized access, all through a weak password[16].

Insecure Data Storage: Lack of encryption can result in stolen data, which can be sensitive[15][16].

1.1.5 Scalability

Classical communication systems are limited by the number of users; the higher the number, the lower the performance and reliability of the transmissions.

Network Congestion: Increased traffic from additional users or devices on the network can result in congestion, which can cause delays, packet loss, and decreased throughput.

To address these limitations, a new technology has emerged called quantum communication.

which aims to overcome these challenges by utilizing quantum mechanics principals.

1.2 Emergence of Quantum Communication

Quantum communication, driven by the revolutionary concepts of quantum physics, has emerged as a significant development in the world of telecommunications [17]. Traditional communication systems face issues with scalability, bandwidth, and interference, making quantum communication a promising alternative [17][18].

The unique properties of quantum physics, such as superposition, entanglement, and quantum indeterminacy, enable quantum communication to achieve unprecedented levels of efficiency and security [17]. Quantum teleportation allows the transfer of quantum states over vast distances, while quantum key distribution (QKD) utilizes quantum entanglement to create secure cryptographic keys [17].

The need for secure communication channels and the rise in cyber security threats have accelerated the development of quantum communication, which holds the promise of safe data transmission and unbreakable encryption [18]. Quantum communication protocols are now widely used in practice due to advancements in quantum networking, quantum information processing, and quantum optics [17][18].

Governments, academic institutions, and business leaders have collaborated to drive quantum communication research and development [18]. This collaboration has fostered innovation and set the stage for the revolutionary advancement of secure communication, quantum computing, and quantum internet infrastructure in the future [18]. However, challenges remain in addressing issues such as scalability, affordability, and integrating quantum devices into existing networks [18].

The application of quantum mechanics to communication in the 1980s and 1990s laid the foundation for quantum communication [17][18]. The proposal of the first quantum key distribution protocol, BB84, by Charles H. Bennett and Gilles Brassard in 1984 marked a significant milestone in secure communication using quantum physics [17]. Subsequent decades witnessed substantial advancements in the theoretical and experimental aspects of quantum communication, with successful demonstrations of quantum networking, quantum cryptography, and quantum teleportation [1].

By 2024, quantum communication has evolved from a theoretical concept to a practical technology with various real-world applications [1]. Recognizing the transformative potential of quantum communication for secure data transmission, governments and business organizations have made substantial investments in its research and development [1]. While challenges persist, the progress in quantum communication has created new opportunities for an increasingly secure and efficient communication environment in the future [1].

1.3 Fundamentals of Quantum Communication

1.3.1 Qubits:

A qubit, or quantum bit, is the basic unit of quantum information, which is fundamental in the field of quantum computing [19]. The qubit is the quantum analog of the bit, which is the basic unit of classical information. In classical computing, a bit can take on one of two values, 0 or 1. Similarly, a qubit has two orthogonal basic states, denoted as $|0\rangle$ and $|1\rangle$ [20]. When a qubit is in a superposition state of $|0\rangle$ and $|1\rangle$, the qubit is written as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$; where α and β are complex numbers, and $|\alpha|^2 + |\beta|^2 = 1$. Measured a qubit in state $|\psi\rangle$, the qubit has a probability of $|\alpha|^2$to collapse to state $|0\rangle$, and of $|\beta|^2$ to collapse to state |1|. Six or seven quantum circuits are needed to prepare the initial state of $\alpha|0\rangle + \beta|1\rangle$.c The collapse is an operation in the quantum theory, turning the quantum states from $|\psi\rangle$ to $|0\rangle$ or $|1\rangle$. When $|\psi\rangle$ collapses, nature actually generates a new universe.

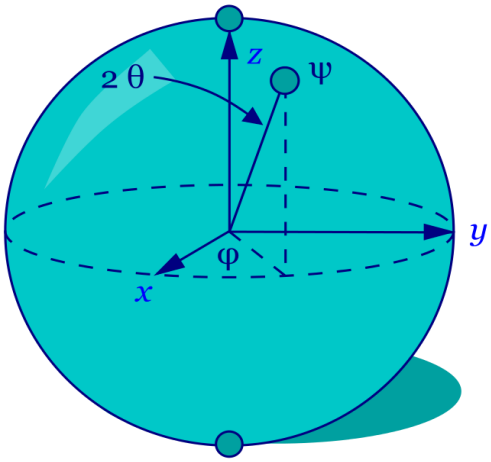

Figure 1.1: A Qubit the basic unit of quantum information.

1.3.1.1 Importance of Qubits in Quantum Computing

Combining two qubits makes a four-dimensional space, with four so-called computational basis states. They are |00⟩, |01⟩, |10⟩, and |11⟩, where the first digit refers to the status of the first qubit, and the second digit refers to the status of the second qubit. Extending this to n qubits, requires 2^n dimensions, which grows very, very quickly as n increases [4]. Quantities you may wish to store in a quantum computer scale exponentially in n. Quantum circuits can operate on qubits to perform a variety of tasks, including computer simulations, factoring, database search, cryptography, and machine learning. Quantum algorithms are highly parallel, meaning they use a lot of qubits at once and therefore will need algorithms to be much faster in order to gain a quantum advantage.

Quantum computers rely on a physical analog to bits which is qubits [20]. In classical information theory, information is stored in bits that are either 0 or 1. In quantum information theory, qubits can take advantage of the superposition principle and exist, not just in the 0 state or the 1 state, or as a random blend of them, but as 0's,1's, and simultaneous combinations of the two in various proportions. Superposition means that quantum states add linearly rather than add as probabilities do. There is a one to one correspondence between complex numbers and qubit states, meaning that qubits can be represented as a vector of complex numbers in a two-dimensional complex Hilbert space, which is frequently written as C^2. Measuring a qubit destroys superposition by forcing it to take one single value. Measurement yields a result of 0 with probability $|\alpha|^2$and a result of 1 with probability$|\beta|^2$. Superposition is the keystone to

quantum information processing and distinguishes quantum from classical computers [19][20][21][22].

1.3.1.2 Physical Realizations of Qubits

Physical realizations of qubits, the building blocks of quantum computers, include a variety of experimental platforms, each with unique strengths and limitations. Here are some typical physical realizations:

- **Photon Qubits:** Photon qubits use the polarization or other degrees of freedom of individual photons as qubits. Photons are manipulated with optical components such as beam splitters and phase shifters. Photon qubits enable long-distance communication over optical fibers and are ideal for quantum communication protocols such as quantum key distribution[23].
- **Trapped Ions:** Trapped ions use ions contained in an electromagnetic trap as qubits. Laser pulses are utilized to control the ions' intrinsic states and induce entanglement between them. Trapped ions have extremely long coherence times and high-fidelity operations, making them ideal for constructing error-corrected qubits. [23][24]
- **Neutral Atoms:** Qubit arrays are created by cooling and trapping neutral atoms with electromagnetic fields. Laser beams are used to modify atoms' internal states, resulting in entanglement. Neutral atoms provide extended coherence periods and high-fidelity operations, with the possibility for scalability via optical lattices [24].

1.3.2 Superposition:

Quantum superposition is a two-particle or many-particle state that is represented as a linear combination. This is another way of saying that it is a way to be in two places at the same time. One of the most famous arguments against the interpretation of superposition as a macroscopic phenomenon was the one you suppose you put a single atom in the box. When its state is described by a superposition, whether the state of the atom can be inferred before the box is opened.

Quantum superposition arises when a quantum particle arrives in an environment and the system plus environment are initially uncorrelated. This also happens when a system interacts with an environment in a way that is not quite, but almost, perfectly reversible. The superposition can be seen in a superposition of two distributions having distinct properties.

Think of an electron passing through a scattering potential and then arriving on an image plane where a measurement is performed. The electron goes into a superposition of wavefront on the image plane. The integral above runs over all possible pairs of points in the image plane that are related by a Fourier pair. The classical distributions are the ones that arise from using the quantum superposition to calculate the intensity as an average of either the position or momentum distribution[25][26].

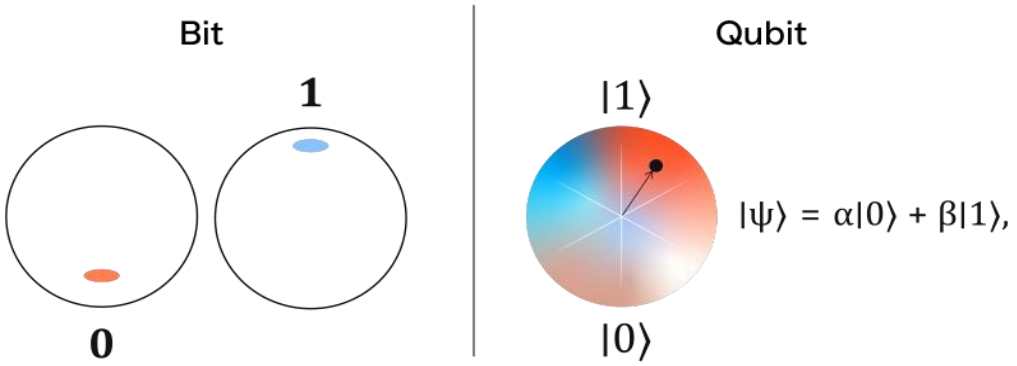

Figure 1.2: Comparison between classical bit and a quantum bit (qubit) showcasing the states $|0\rangle$, $|1\rangle$ and the superposition $|\psi\rangle$.

1.3.2.1 Applications of Superposition

The technical importance of the concept is as striking as its breadth. At the lowest level, superposition can help us make more exacting measurements of a quantity that is to be quantized. Higher measurement accuracy may involve a beam splitter or similar device and subsequent recombination of the fragments. Using photon superposition, we have been able to develop essentially all of nuclear magnetic resonance. The use of a coherent state or its superposition is at the heart of quantum key cryptography, another recent important application. If we can also control individual photons, we can store short elementary logical steps of information in quantum memory, with potential speed-ups of computations that depend on the notorious difficulty of relative searches. These also require controlled photons[26][27].

1.3.2.2 Limitations and Challenges

In the following, we discuss several important issues that illustrate the limitations and challenges of the superposition system. When multiple sources are considered, the superposition principle does not completely constrain the response of the receiver [27].

A) Measurement and Observation

When an unobserved quantum system evolves unitarily, it usually does not remain in a definite state of position or momentum observable. Instead, measurements of position and momentum find random results, consistent with position and momentum probability distributions that are constant in time or, for bound states that change only by phase terms. In either case, the position and momentum probability distributions will describe spread. There are some exceptions — for instance, position measurements find location-organized probability distributions for energy Eigen functions of a system with attractive forces in just one degree of freedom — but those exceptions are rare. For quantum systems to match typical classical systems, quantum states must superpose states that are unequal in position and momentum. A quantum system generally has no deterministic trajectory between measurements[28].

B) Practical Implementations

The main difficulty in implementing photonic superpositions derives from the difficulty to get photon sources in a coherent state without the assistance of a good mean ancilla verification to operate as an ancilla for the unitary system. However, to experimentally incoherent to uncorrelated photons for spectrometric detection and uncorrelated proposals, when superposition states are defined as a single spontaneous process, several optical elements could have specific coherence requirement[27][28].

1.3.3 Entanglement:

Quantum entanglement refers to the collection of correlations between the quantum observables of distinct subsystems. It provides complete information about the overall state of the system, but does not reveal any information about the individual states of the subsystems, regardless of their separation distance. When the first measurement is made on any subsystem, all subsystems will collapse into their entanglement-induced states, and the overall state of the system will thereafter behave in a classical manner.

1.3.3.1 Schrödinger Cat:

In 1935, a well-known physicist, Erwin Schrodinger, thought of a theory to enhance the answer given by Copenhagen, which states that there is no definitive reality before measurements in quantum theory, which will help later on explain the term “Particle

Entanglement". This theory was proven by an experiment called "Schrodinger's Cat," where a cat was put in a box with a radioactive element, a hammer attached to the radioactive element, and a flask full of poison.

This box is only observed from the outside; what happens on the inside is unknown. In this box, if the radioactive element emits a radioactive particle, then the hammer automatically breaks the poison flask, and the cat dies. On the contrary, the radioactive element does not emit, and therefore the cat does not die. This concludes that the fate of the cat is connected to time radioactive elements, and this particular experiment explains fundamentally the idea of entanglement, where it is merely a connection or correlation between two particles or more, so that if a measurement is performed on only one particle, the other one will automatically collapse.

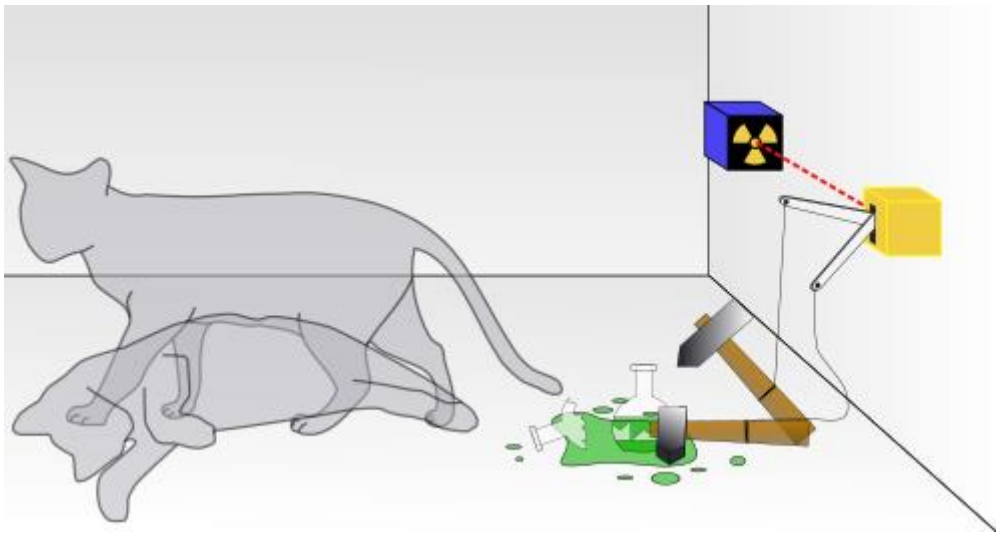

Figure 1.3: Representational figure of the Schrodinger's Cat experiment.

1.3.3.2 Bell Inequalities

In 1964, the physicist John Bell formulated Bell inequalities as part of his study on quantum mechanics, Bell inequalities are mathematical constraints that challenges local hidden variable theories, which explains probabilistic nature of quantum mechanics by adding unnoticed variables that detects entanglement. Where if a system violets the Bell inequalities then the system is in an entangled state [29][30].

The most common Bell inequalities are Bell-CHSH [30] [31] given by:

$$tr\ (\rho \mathrm{B})\ \leq 2 \tag{1.1}$$

Where:

ρ: the qubit state

B: Bell operator

$$\mathbf{B} = \vec{\boldsymbol{a}} \cdot \vec{\boldsymbol{\sigma}} \otimes \left(\vec{\boldsymbol{b}} + \overrightarrow{\boldsymbol{b'}}\right) \cdot \vec{\boldsymbol{\sigma}} + \overrightarrow{\boldsymbol{a'}} . \vec{\boldsymbol{\sigma}} \otimes \left(\vec{\boldsymbol{b}} - \overrightarrow{\boldsymbol{b'}}\right) \cdot \vec{\boldsymbol{\sigma}} \quad (1.2)$$

1.3.3.3 Generation of the entangled state

Researchers have demonstrated various complicated methods to generate entangled photons. These methods are advanced techniques such as nonlinear metasurfaces, which enables the full control over the direction and entanglement of the photons [32]. The techniques used generally in creating the entangled state are Atomic Sources and Parametric down-conversion.[4]

A) Atomic Source

Photon qubits, namely polarization qubits and time-slot qubits, are essential for researching different entangled states, including the analysis of Bell's inequality. These qubits are mostly used in quantum communication technologies, such as quantum cryptography, because they are compatible with transmission by fiber optics[32][33]. The conventional approach to generate entangled photons employs the atomic cascade process in calcium, resulting in the emission of a pair of intertwined photons, as described by the equation:

$$|\psi >= \frac{1}{\sqrt{2}} * (|LL > + |RR >) \quad (1.3)$$

This equation represents the entangled state of two photons, with L and R denoting the left and right circular polarization conditions, respectively. The entangled state can be expressed in terms of a linear polarization basis as follows:

$$|\psi >= \frac{1}{\sqrt{2}} * (|HH > + |VV >) \quad (1.4)$$

In this context, the symbols H and V denote the states of horizontal and vertical linear polarization, respectively[32][33]. While this approach of producing entangled photons is innovative, it is difficult to successfully capture the emitted pairs due to the random discharge of photons at a complete three-dimensional angle. Furthermore, the manipulation of individual atoms is not feasible for extensive speculative inquiries. Therefore, although atomic cascade is

the first source of entangled photons, it has not been widely exploited in quantum information communication practices[33].

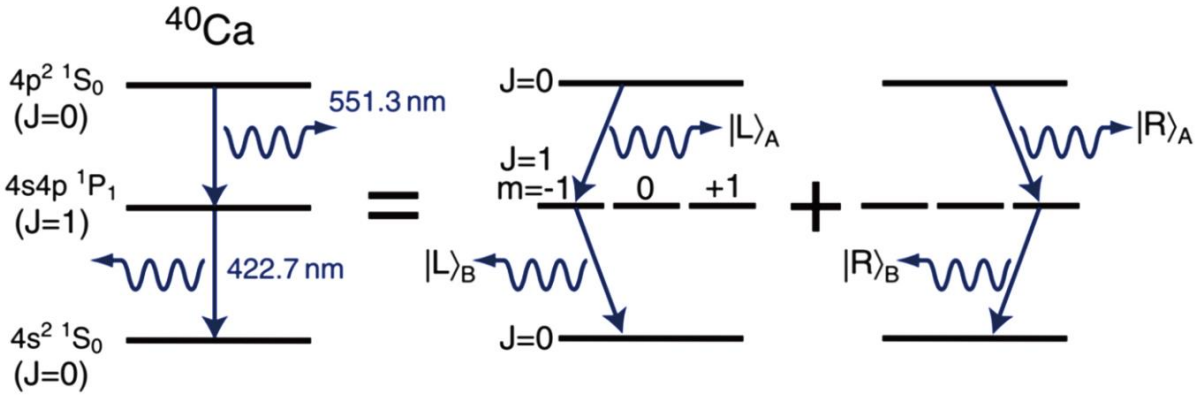

Figure 1.4: Representational design of the knotted photon set discharge via atomic cascade in calcium

B) Parametric down-conversion

The principal technique for producing entangled photons is the nonlinear optical process known as Spontaneous Parametric Down-Conversion (SPDC).

In SPDC, a pump photon splits into two lower-energy photons, referred to as the signal and idler photons, when a high-intensity laser beam travels through a nonlinear crystal. The conservation of energy and momentum governs this process, guaranteeing that the idler photons and the signal have linked features, such as their polarization states [34]. The following equation may be used to characterize the entangled photons generated by SPDC:

$$|\psi >= \frac{1}{\sqrt{2}} * (|HV > + |VH >) \quad (1.5)$$

The entangled state of the state is represented by this equation. the two photons' entangled state, where H and V stand for the corresponding horizontal and vertical polarization states. A crucial component of the formation of entangled photons is the photons polarization state, and depending on the phase matching method employed, SPDC can yield photons with correlated polarization states[33][34].

In SPDC, there are two kinds of phase matching conditions that lead to various polarization correlations between idler photons and the signal. Photons with parallel polarization are produced by Type I phase matching, whereas photons with orthogonal

polarization are produced by Type II phase matching. Phase matching conditions are selected based on the intended characteristics of the entangled photons and the particular application [34].

To sum up, SPDC is a dependable technique for producing entangled photons, which are necessary for a number of applications of quantum information processing. A crucial component of the formation of entangled photons is the photons' polarization state, and depending on the phase matching method employed, SPDC can yield photons with correlated polarization states [34].

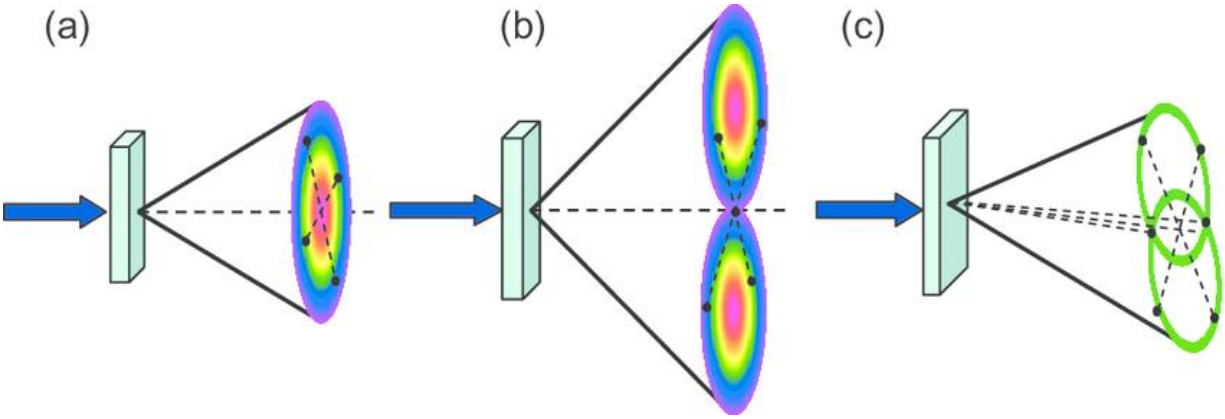

Figure 1.5 : Three different types of parametric down-conversion schemes: (a) type-I phasematching ; (b) collinear degenerate type-II phase-matching, with two cones overlapping along the pump direction; (c) non-collinear degenerate type-II phase-matching—the cones intersect along two symmetric directions

1.3.4 Quantum Gates:

Similar to how classical logic gates work on bits in traditional digital computers, quantum gates are the essential building blocks of quantum computing. They act on qubits, which are the fundamental units of quantum information[35][36]. Contrary to many conventional logic gates, quantum gates have the special quality of reversibility[35]. These gates are essential to quantum circuits because they allow the manipulation and processing of quantum information[36]. They are represented by unitary matrices.

The Hadamard gate, a single-qubit gate represented by a certain matrix, is one prominent example[36][37]. A different kind is the phase shifter gate, which modifies the quantum state of a qubit by adding a phase shift[37][38]. Two qubits are used in controlled gates, such the controlled-U gate, where the first qubit acts as a control for the operation regarding the second qubit[37][38].

Implementing quantum algorithms and carrying out quantum calculations require quantum gates [38]. Complex quantum operations and calculations are made possible by their ability to precisely and reversibly manipulate qubits [39][40] which makes them fundamental components of quantum computing that allow for the remarkably efficient and secure implementation of quantum protocols and the realization of quantum algorithms[39].

1.3.4.1 Single Qubit Gates

Fundamental elements of quantum computing are single qubit gates, which alter the quantum states of individual qubits.

A) The Pauli X gate:

It switches the qubit's state from |0⟩ to |1⟩ or vice versa, much like the NOT gate in classical computing [38].

$$X(\alpha|0\rangle + \beta|1\rangle) = \beta|0\rangle + \alpha|1\rangle \tag{1.6}$$

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

Figure 1.6 :The Pauli X Gate Matrix

B) The Pauli Z gate:

Changes the phase of the |1⟩ state without altering the |0⟩ state [38].

$$Z(\alpha|0\rangle + \beta|1\rangle) = \alpha|1\rangle - \beta|0\rangle \tag{1.7}$$

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

Figure 1.7: The Pauli Z Gate Matrix

C) The Pauli Y gate:

Equivalent to applying both X and Z gates along with a global phase [38].

$$Y(\alpha|0\rangle + \beta|1\rangle) = -i\beta|0\rangle + i\alpha|1\rangle \tag{1.8}$$

$$Y = \begin{bmatrix} i & 0 \\ 0 & -i \end{bmatrix}$$

Figure 1.8: The Pauli Y Gate Matrix

D) The Hadamard gate:

Converts the $|0\rangle$ state into an equal superposition of the $|0\rangle$ and $|1\rangle$ states, resulting in a superposition state and vice versa [40].

$$H|0\rangle = |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{1.9}$$

$$H|1\rangle = |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{1.10}$$

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$$

Figure 1.9: The Hadamard Gate Matrix

1.3.4.2 Multiple Qubit Gates

Quantum gates that operate on two or more qubits at the same time are known as multi-qubit gates. These gates make it possible to create and work with entangled states, which are necessary for carrying out intricate quantum computations.

A) The Controlled Not gate:

A gate consisting of two qubits that operates on the target qubit by performing a NOT operation only while the control qubit is in the state$|1\rangle$ [40].

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

Figure 1.10 : CNOT Gate Matrix

B) The Swap gate:

A two-qubit gate that switches their states [40].

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

Figure 1.11: Swap Gate Matrix

C) The Toffoli gate:

A three-qubit gate that, only when both control qubits are in state$|1\rangle$, operates on the target qubit as a NOT operation. It is also called as CCNOT gate (Controlled-Controlled Not) [40].

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

Figure 1.12: Toffoli Gate Matrix

D) The Fredkin gate:

A three-qubit gate that, only when the control qubit is in state$|1\rangle$, switches the states of two target qubits [41].

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

Figure 1.13: Fredkin Gate Matrix

1.4 Applications of Quantum Communication

Applications for quantum communication are plentiful and use the unique features of quantum technology to improve efficiency, accuracy, and security in a variety of fields. Here are some key applications of quantum communication with more details.

1.4.1 Quantum Cryptography

Quantum cryptographic systems offer unparalleled security by leveraging the principles of quantum mechanics, specifically the non-locality and randomness inherent in quantum entanglement. These systems can generate secret keys securely even if an attacker fully controls

the transmission line and knows the protocol. This security is due to the non-simultaneity in the revelation of classical and quantum information, as illustrated by the EPR paradox. Unlike classical cryptographic systems threatened by quantum computers, quantum cryptography ensures long-term data secrecy through quantum key distribution (QKD), which is based on fundamental laws of nature. This not only guarantees secure communication but also sets limits on an eavesdropper's knowledge and certifies the randomness of the key, making QKD a critical link for secure quantum communication [42][43][44].

1.4.1.1 Quantum Key Distribution

Quantum Key Distribution (QKD) is a leading quantum communication technology that enables two parties, Alice and Bob, to create a shared, secret cryptographic key using a public channel, a quantum channel, and some initial shared secret information, or seed. Alice sends a sequence of quantum particles to Bob, who measures their states. They then perform a Measurements Coincidence Test by publicly announcing some measurement bases. If the coincidences exceed a threshold, they assign the quantum particles as keys, ensuring their data are identical barring system imperfections. While QKD schemes offer unconditional security against passive attacks, active attacks require Alice and Bob to verify key security through classical communication. Physical implementations often only approximate ideal quantum channels, allowing for potential deviations that must be accounted for to ensure security [45][46][47].

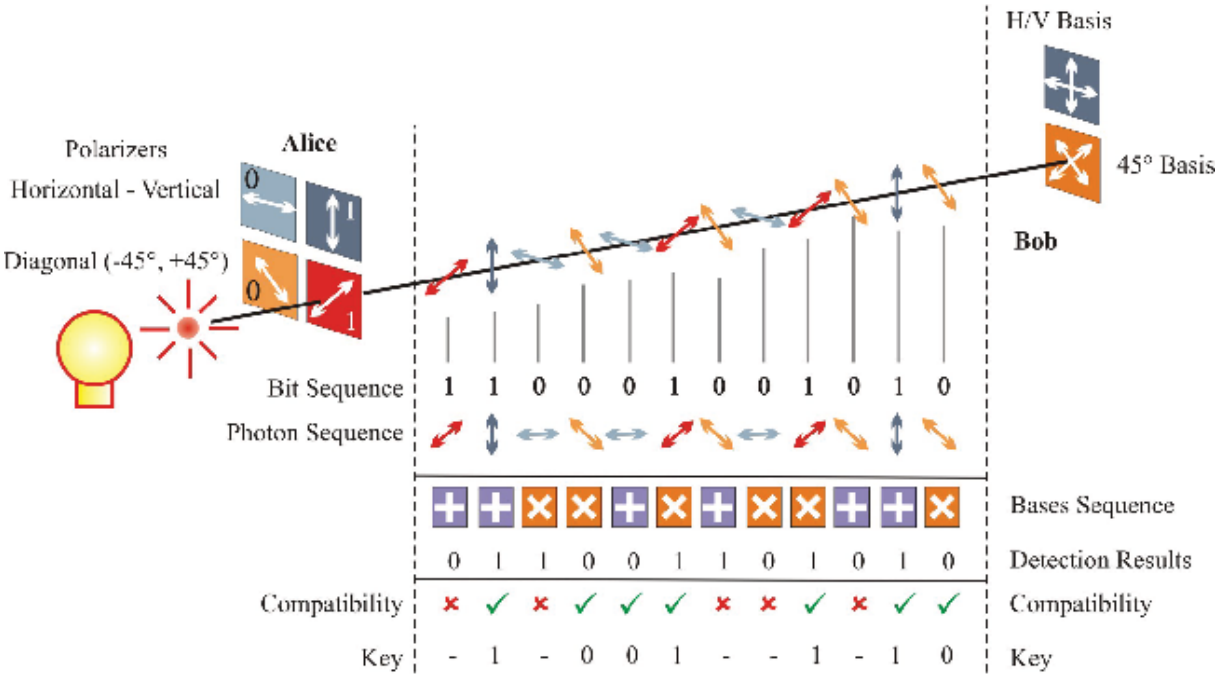

Figure 1.14: The Quantum Key Distribution System using BB84 Protocol

1.4.1.2 E91 Protocol

The E91 protocol, proposed by Artur Ekert in 1991, is a quantum key distribution (QKD) method that uses entangled quantum particles to establish a secure cryptographic key between two parties, typically called Alice and Bob. In this protocol, a source generates pairs of entangled particles, with one particle sent to Alice and the other to Bob. Both parties randomly choose measurement bases to measure their particles' states. After measurement, they publicly compare a subset of their results to check for correlations indicative of entanglement. If these correlations exceed a certain threshold, they confirm the presence of entanglement, ensuring the security of the remaining measurement outcomes, which are used to form the secret key. The security of the E91 protocol relies on the fundamental principles of quantum mechanics, making it robust against eavesdropping [47][48][49].

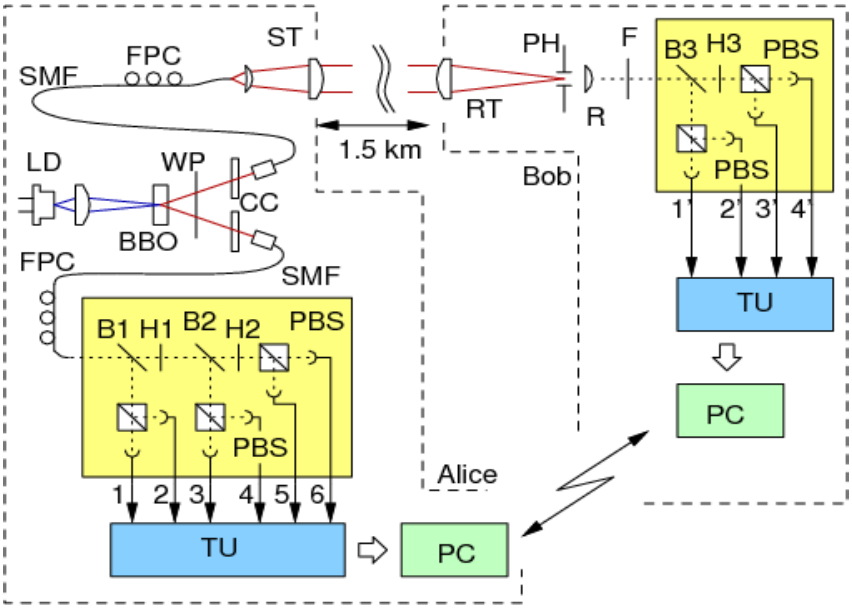

Figure 1.15: The Quantum Key Distribution System using E91 Protocol

1.4.2 Quantum error correction

A key idea in quantum computing is quantum error correction, which deals with the intrinsic susceptibility of quantum information to mistakes brought on by noise, decoherence, and other external circumstances. Quantum error correction codes use the ideas of quantum physics to safeguard quantum states against mistakes without actually measuring them, as measurement might upset the fragile quantum superposition. This is in contrast to classical error correction, which depends on redundancy and majority voting.

The Shor Code, which uses nine qubits to encode a single logical qubit and correct errors using a mix of entanglement and quantum gates, is one of the first quantum error correction codes. Quantum error correction codes find and fix faults without erasing the quantum information by redundantly spreading information across many qubits .

Through the use of syndrome measurements and subsequent error correction procedures based on the identified syndromes, this redundancy makes it possible to identify mistakes.

To achieve fault-tolerant quantum computation—where quantum algorithms may be successfully implemented even in the face of errors—quantum error correction is necessary. Developing effective error-correcting codes, creating error detection systems, and refining error correction protocols are all necessary to implement quantum error correction, which is a challenging endeavor that reduces the impact of mistakes on quantum operations. Current

efforts in quantum error correction research are focused on enhancing the scalability and efficiency of error correcting methods to facilitate the real-world deployment of massive quantum computing systems.

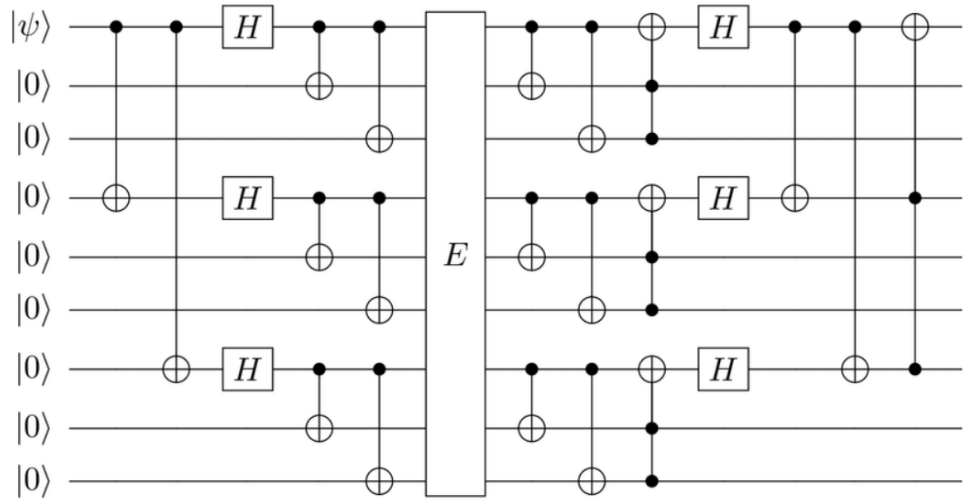

Figure 1.16: An example code of the error correction codes (Shor's nine-qubit error-correction code)

1.4.3 Quantum internet

The goal of the ground-breaking idea known as the "quantum internet" is to establish a network of connected quantum computers that can send, process, and receive data that is encoded in quantum states. The quantum internet, in contrast to the standard internet, will offer new features like quantum cloud computing and cryptography while leaving existing infrastructure in place [50].

The quantum internet's capacity to use quantum technology to provide safe and effective communication exchanges is one of its primary characteristics. One important use of the quantum internet is quantum key distribution, which offers an extremely safe way to send cryptographic keys [50]. It is now in use.

Researchers predict that interstate quantum networks will be formed inside the quantum internet in the future, while the full ramifications of this technology are still being investigated the next ten to fifteen years in the United States. With the use of quantum technology, the quantum internet has the potential to facilitate safe and quick communication exchanges, opening the door for more sophisticated uses of quantum computing and communication. [50][51].

All things considered, the quantum internet is essentially a paradigm leap in communication technology, with the potential to create safe, effective, and unbreakable channels of communication that use the laws of quantum physics to transform information exchange and data transfer [51].

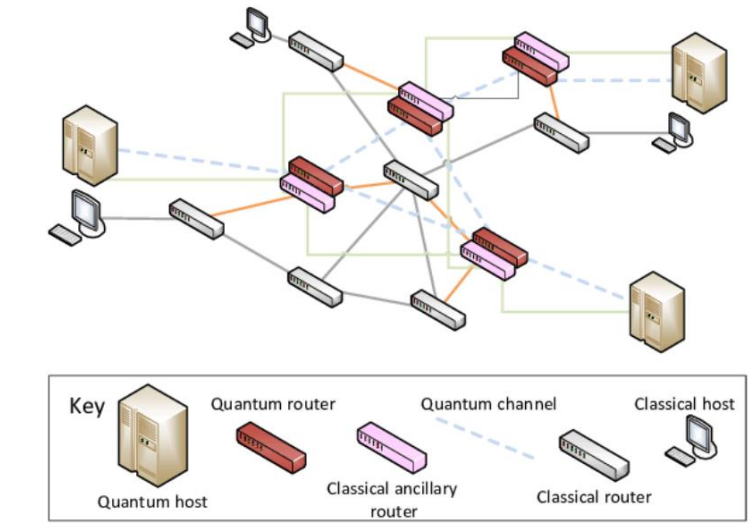

Figure 1.17: The architecture and the materials of a quantum internet link

1.4.4 Quantum Computing

Quantum computing, proposed by Richard Feynman in 1981, leverages principles of quantum theory, such as superposition and entanglement, to perform computations. Unlike classical computers, which process bits in a binary state, quantum computers use qubits that can exist in multiple states simultaneously, enabling parallel calculations and exponential increases in computational speed. Quantum computing shows promise for solving complex problems like integer factorization, breaking RSA encryption, simulating quantum physics, modeling chemical reactions, and optimizing routes, which are challenging for classical computers. Despite being in early development stages, quantum computing research is advancing rapidly, with significant progress in quantum hardware, algorithms, and our understanding of quantum mechanics, heralding potential speed advantages over classical systems for certain tasks [52][53][55].

1.4.4.1 Quantum Computing Principle

Quantum bits, or qubits, are the basic unit of quantum computing being in the state of superposition allows quantum computers to do tasks in simultaneously, significantly improving their computing capacity when compared to classical computers.

Furthermore, qubits show entanglement, a fundamental quantum feature. This effect allows quantum computers to do certain sorts of calculations far more efficiently than traditional computers [53][56].

In practice, qubits can be implemented utilizing a variety of physical systems, including trapped ions, photons, among others. Each of these systems has distinct advantages and challenges in terms of qubit coherence (the length of time a qubit remains in its quantum state), scalability (the ability to reliably manage and control large numbers of qubits), and error correction.

Quantum computing researchers are continuously investigating various qubit implementations and creating novel strategies for qubit manipulation, coherence maintenance, and error correction. As the field advances, realizing the full potential of qubits offers the possibility of transforming computers in a variety of fields, including encryption, optimization, materials research, and drug development [54][57].

1.5 Conclusion

Information exchange undergoes a radical alteration as it moves from the limitations of classical communication to the domain of quantum communication. The complex ideas of superposition, entanglement, qubits, and quantum gates serve as the foundation for quantum communication, which ushers in a new age of dependable, secure, and efficient data transfer. Information security is being revolutionized by quantum communication, which enables unmatched levels of confidentiality and anonymity by utilizing the quantum phenomena of entanglement and superposition. Applications of quantum communication cover many industries, from the internet of things to quantum key distribution, and they promise increased accuracy and robustness.

The next chapters will investigate the similarities and differences between classical and quantum error-correcting codes, illuminating the critical role these codes play in guaranteeing

the security and correctness of data transfer in both the classical and quantum computing domains.

The foundation of error-correcting methods is a strong and elegant theory, which has been the distinctive feature of classical error-correction codes for more than 75 years. These codes, among which binary linear codes are a fundamental component of classical coding theory, encode data to guard against mistakes and noise. However, as quantum error-correcting codes were created less than 30 years ago, they constitute a very recent development. In order to ensure the dependability and integrity of quantum communication systems, which make use of the concepts of quantum physics to shield quantum information from mistakes and flaws.

Chapter 2

Classical Error correction codes

Since the invention of Information theory, the quality of the received data has transmitted data always been prioritized. This data often gets jeopardized while being transmitted over noisy channel which causes errors in the original messages. Therefore, in order to detect these errors and correct them, several methods were invented called error correcting codes. These error correcting codes are a fundamental concept in digital communication which play an indispensable role in ensuring the Correct to and reliable transmission of the data. And to do so, the process of interleaving is applied to these codes where the data bits are rearranged before sending. They add bits of redundancy that will give the original message an assertion when received at the end of the transmission channel and allowing the recovery of the initial data even with interference. There are various types of Error correcting codes such as: Hamming Codes, Trellis diagrams, Low density parity check… which will be the topic of this chapter.

2.1 Channel Coding

Channel coding is a technique used in data transmission to ensure the reliability of the communication link, more efficient and error free transmission. “Claude Shannon” first introduced it in 1948 in his article “The Mathematical Theory of Information”, where it was defined that diagram of a general communication system contains five elements:

An Information Source: this is the step where a message or a sequence is generated to be transmitted (data) to the receiving terminal [58].

The Transmitter : typically generates signals that are sent through a noisy channels , where the message is imprinted into the signal (Information Carrier) [58].

The Channel: It is the pathway where the data is being transmitted through from the source to the destination. It can be either physical (Fiber Optics cables, coaxial cables …) or wireless (Electromagnetic waves …) [58].

The Receiver: a receiver inverses the work of the transmitter, as a result, we retrieve the original message that was generated by the source [58].

The Destination: which is the person or the object that the message was intended to [58].

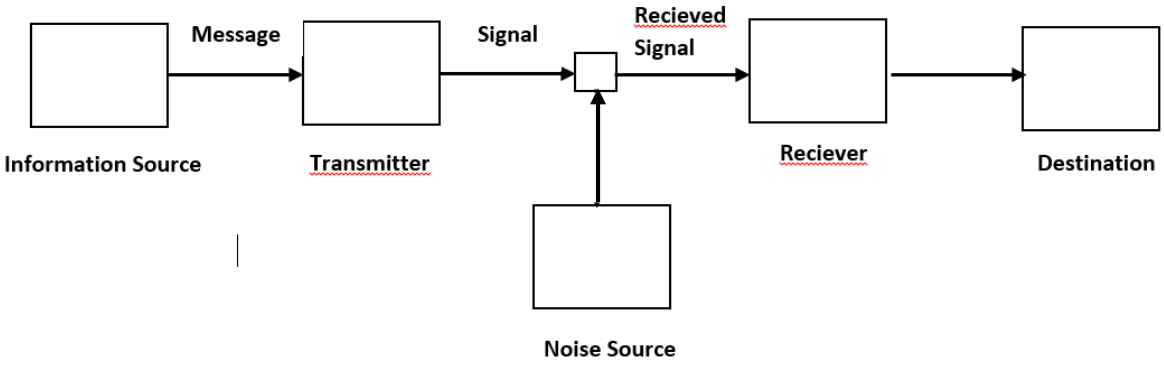

Figure 2.1: Schematic diagram of a general communication system

When the signals goes through a noisy channel errors occur all over the signals and the problem caused was that because the signals are put in numerical data (0s and 1s) , we won't be able to know exactly where did the errors happen hence the creation of "Shannon Limit" or "Channel Capacity " [58] which it states that if the coding rate is lower than the given equation

$$C = B \log_2(1 + \frac{S}{N}) \tag{2.1}$$

C: Channel Capacity, *B :* Bandwidth, $\frac{S}{N}$: Signal to Noise Ratio

Then it is possible that the information transmitted contains very low error rate. Therefore, to correct and detect these errors, a few methods were invented for this specific problem and it is called "Error Correction Codes" such as: Block codes, Cyclic codes, Convolutional codes, etc…

2.2 Classification of error correcting codes:

The classification of error-correcting codes is critical in digital communication systems for maintaining data integrity during transmission and storage. Block codes divide data into different blocks and encode each one individually with predetermined block lengths, making them suitable for error detection and repair. Convolutional codes, on the other hand, treat data as a continuous stream, using sliding windows to operate on bit streams of different length. These codes, which are usually decoded with algorithms such as the Viterbi algorithm, are

widely used in communication systems such as CDMA, GSM, satellite communications, and wireless LANs to improve data transfer reliability and efficiency.

2.2.1 Block codes:

In coding theory, error correcting codes contain large families and one of those is Block codes. Block codes encode and decode data in fixed-length blocks, with each block being considered as a whole entity during the encoding and decoding procedures. They are intended to offer redundancy to sent data, allowing for the identification and correction of errors caused by noise or interference on the communication channel [59].

Block length N: The number of bits in each block.

Block Rate R: The ratio of the information bits to the total bits in each block.

The Message length k: The number of bits in the original message before encoding.

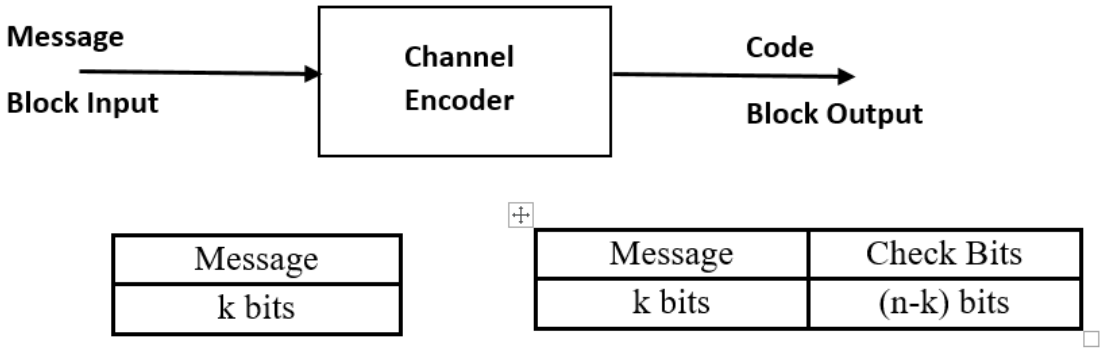

Figure 2.2: Functional diagram of block coder

These Block codes error correction codes are obtained by a process of encoding, decoding and error detection.

A. Encoding process:

In this particular step, the encoding process means the creation of the codeword that is to be transmitted later.

1. Message Victor:

The initial message is altered to a binary vector with ‘’0s ‘’ and ‘’1s’’ of a ‘’k’’ length where each element represents either 0 or 1 [59][60].

$$\boldsymbol{n} = (\boldsymbol{n_1}, \boldsymbol{n_2}, \boldsymbol{n_3}, \dots, \boldsymbol{n_k}) \quad (2.2)$$

2. Generator Matrix G :

In this matrix, every row embody a codeword. A generator matrix G is calculated using the concatenation of two matrices "$\boldsymbol{I_k}$" and "$\boldsymbol{P}$" horizontally to form it [59][60].

$$\boldsymbol{G} = [\boldsymbol{I_k} \quad \boldsymbol{P}] \tag{2.3}$$

$\boldsymbol{I_k}$**:** Is the identity matrix of size $\boldsymbol{k \times k}$

$\boldsymbol{P}$**:** Is a binary matrix of size $\boldsymbol{k} \times (\boldsymbol{N} - \boldsymbol{k})$

$$\boldsymbol{I_k} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \boldsymbol{P} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

The identity Matrix **The binary Matrix**

$$\boldsymbol{G} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{I_k}} \underbrace{\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}}_{\boldsymbol{P}}$$

The Concatenation

Figure 2.3: Generation of G using the identity matrix and the binary matrix in which k=4 and N=8

Therefore, the encoded codeword is obtained by the multiplication of the matrix G with the initial message n

$$C = \boldsymbol{G} \times \boldsymbol{n} \tag{2.4}$$

B. Decoding process:

After the encoded message passes through to transportation channel, the process of decoding starts by:

1. Received Vector:

At the reception, the message may be flawed and inaccurate due to the channel noise where:

$$\boldsymbol{r} = (\boldsymbol{r1}, \boldsymbol{r2}, \dots, \boldsymbol{rk}) \tag{2.5}$$

2. **Syndrome Calculation:**

This step is the most important in the data transmission because this syndrome works as the detector and the corrector of errors occurred in the previous step and it is calculated from the received vector and generator Matrix. If the block code is linear, the syndrome is calculated by multiplying the received vector and the transpose of the parity matrix check [60][61].

$$S = \boldsymbol{r} \times H^T \tag{2.6}$$

S**:** The Syndrome, $\boldsymbol{r}$**:** The message received, H**:** Parity Matrix check , H^T**:** The transpose of H

3. **Error Detection:**

Once the syndrome is calculated, if the result is zero, no error is detected. If it is different than zero, there are errors received in the codeword [60].

4. **Error Correction:**

The syndrome determines the exact pattern of errors in the codeword if the codes supports [60].

2.2.1.1 Hamming codes:

The Hamming code is a linear error-correcting code that can detect and fix single-bit faults in transmitted or stored data, as well as detect (but not correct) double-bit errors. This code, invented by Richard W. Hamming in 1950, is regarded a perfect code since it achieves the greatest achievable rate for codes with its block length and minimum distance of three. Hamming codes add redundant parity bits to the original data bits, the amount of which is determined by precise calculations based on the number of data bits [62].

In order to understand how Hamming codes work, a few notions are crucial to explain :

Redundant bits: these are extra binary bits that will be added to the original message to results in the codeword that is sent through the channel [62].

Parity bits: This method defines if the original (initial) data has an odd or even count of ones [62].

Moreover, there are two types of parity bits:

Even parity bits: if the count of 1's is even, then the parity bits is ''0''

Odd parity bits: if the count of 1's is odd, then the parity bit is ''1''

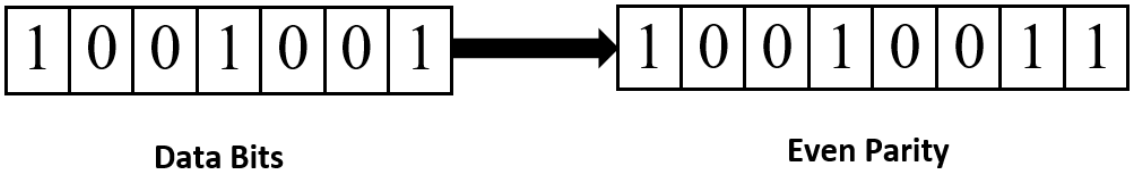

Figure 2.4: Hamming code parity checking

Therefore, to create a hamming code, we use this process:

A) Encoding process:

1. **Identify the number of parity bits:**

Firstly, a number of parity bits have to be determined by the following equation [63]:

$$2^r \geq n + r + 1 \tag{2.7}$$

r : redundant bits, n : number of data bits

2. **Assign each data bit to a position in the codeword :**

The parity bits are placed in positions that are power of 2 " ($\mathbf{2^n}$) "while the data bits are placed in the remaining positions [63].

3. **Calculate the value of each parity bit:**

If the number of ''1'' is even, then the parity bit is ''0''

If the number of ''1'' is odd then the parity bit is ''1''

B) Decoding process:

After the transmission through the noisy cannel, some errors are found in the data codeword and that problem is resolved by decoding Hamming codes using syndrome calculations.

1. **Recalculated the expected parity bits for the received data:**

Using the same formula :

$$2^r \geq n + r + 1$$

We recalculated the parity bits of the received data and compared them with the parity bits calculated of the encoding process. If they are equal, it's a "0" and if they are not equal, it's ''1''. A binary number is formed with these ''0'' and ''1'' then convert it to a decimal number and that is the incorrect bit [63][64][65].

Position= ''0'' ⟶ No errors occurred

Position=''1 '' ⟶ Errors occurred

To correct the error, we flip the value of the incorrect and the parity bits are removed to obtain the initial message (the original).

2.2.1.2 Cyclic Codes:

Cyclic codes are a sub family of block codes known by their linearity and their shifting cycles. So a linear code is called cyclic if the cyclic shift of a codeword is also a codeword [66]

$$(x_0, x_1, \dots, x_{n-1}) \in C \longrightarrow (x_{n-1}, x_n, \dots, x_{n-2}) \in C \tag{2.8}$$

These cyclic codes are characterized by several properties :

- **Unvariate polynomial ring:**

Which is an algebric structure that consists of polynomials with a single variable defined by R(x) where R is a ring and x is the indeterminate or variable [66].

$$\boldsymbol{P} = (p_0, \dots, p_{n-1}) \longleftrightarrow p_0 + p_1 x + p_2 x^2 + \cdots + p_{n-1} x^{n-1} \tag{2.9}$$

- **An ideal:**

An ideal in abstract algebra is a fundamental concept representing a subset of a ring that is closed under addition and multiplication by elements of the ring. In the context of cyclic codes, an ideal plays a crucial role as cyclic codes correspond bijectively to ideals in the quotient ring $F_q[x]/(x^n - 1)$ $F_q[x]/(x^n - 1)$, where F_q is a finite field, n is the length of the cyclic code, and x is a polynomial variable [67].

- **Generator polynomial:**

It exists as unparalleled polynomial g that generates the cyclic code.

Even so, g has to divide $x^n - 1$ and a degree of $n - k$ with k being the dimension of the code C [66][67].

A) Encoding process:

The encoding process of a cyclic code is very simple and straight forward, if we consider a cyclic code C with its generator polynomial, then it is capable of encoding k-length messages with $n - k$ bits [66].

1. Unsystematic encoding:

Assuming our message is of $\boldsymbol{D} = (\boldsymbol{D_0}, \boldsymbol{D_1}, \dots, \boldsymbol{D_{k-1}})$, then in order to gain a codeword, all that is to be done is multiply $D(x)$ with the generator polynomial $g(x)$ [66].

$$C(x) = D(x) \times g(x) \epsilon\, C \tag{2.10}$$

2. Systematic encoding

There exists a more efficient and simpler method to encode a cyclic code and it's called systematic encoding, where if a message is taken into consideration $D(x)$, it will be encoded by multiplying it with x^{n-k} [66].

Then the result is divided by the polynomial $g(x)$

The remainder $r(x)$ is added to the message to obtain the encoded message:

$$C(x) = D(x)x^{n-k} + r(x) \tag{2.11}$$

B) Decoding process :

In order to know if an error was occurred or not, it is sufficient to divide the received word by the generator polynomial to obtain a remainder $r(x)$ [67].

If $(x) = 0$, there are no error found
If $r(x) =$ *' a number of bit '*, there are errors found

1. Cyclic redundancy check

Cyclic redundancy check is a systematic cyclic error detecting code technique

A CRC divides a numerical message by a numerical data generated from the generator polynomial [67].

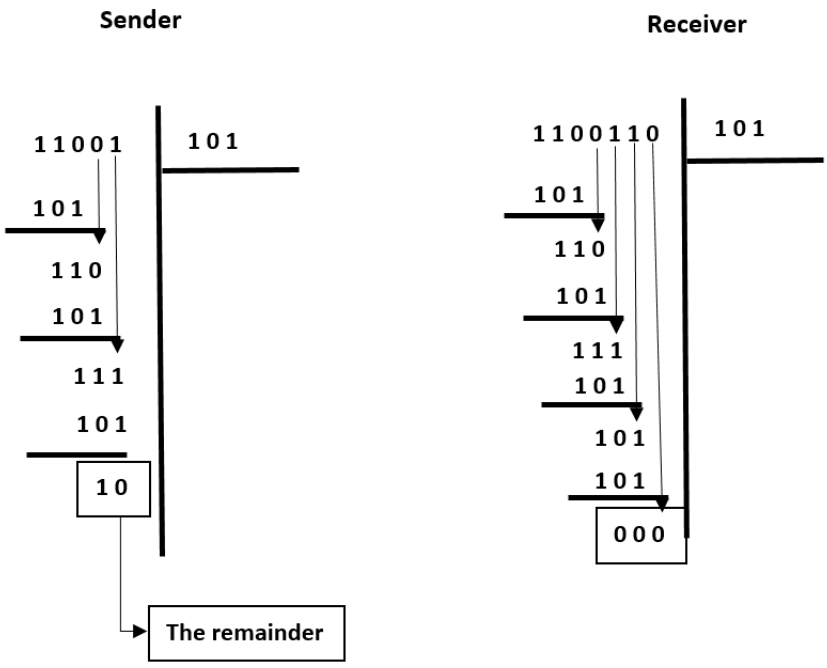

Figure 2.5: Error detection using Cyclic Redundancy Check

$$g(x) = x^2 + 1$$

the numerical data is : $1 \times x^2 + 0 \times x^1 + 1 \times x^0 \Longrightarrow$ data= ‘’101’’

that it is our divisor ‘’101’’

the remainder is the data that shall be added to the initial message

2.2.1.3 LDPC codes (Low Density parity check codes)

Low-Density Parity-Check (LDPC) codes are a class of linear error-correcting codes known for their excellent performance in detecting and correcting errors in digital communication systems.

A) Sparse Parity Check Matrix

LDPC codes are binary linear block codes distinguished by their Sparse parity check matrices which means that the density of the number ‘’1’’ is far less then density of the number ‘’0’’ [68]. An LDPC code parity check matrix is represented by two properties:

w_c: which is the number of ‘’1’’ in a column

w_r : which is the number of ‘’1’’ in a row

and each w_r is a codeword sequence

$$H_{M\times N}\ (w_r, w_c) \tag{2.12}$$

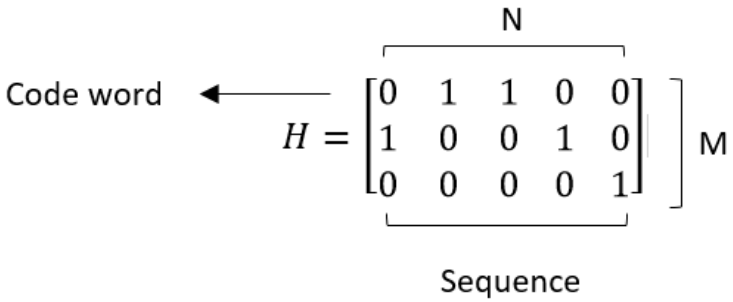

Figure 2.6: Sparse parity check matrix

It exists two types of sparse parity check Matrix

- **Regular Matrix :**

A regular sparse matrix is one in which the majority of the elements are zero and the number on non-zeros are uniform in both rows and columns, resulting in a high sparsity level [68] [69]. They are easier to encode and decode

$$M.w_r = N.w_c \tag{2.13}$$

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

- **Irregular Matrix :**

An irregular Matrix does not contain the same number of ‘’1’’ in each row and column. Where the fraction of columns of weight i is vi and the fraction of rows of weight i by hi [69].

$$M \sum_i h_i \times i = N \sum_i v_i \times i \tag{2.14}$$

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

Figure 2.8: Irregular sparse parity check matrix

If the weight in a row= 2 and h_2=0,3, it means that 30% of the rows in the matrix contain 2 ‘’1’’ entries.

A) Construction of an LDPC code :

In order to obtain an LDPC code, a degree distribution must be achieved which is the distribution of ‘’1’’ across the sparse parity check matrix

1. Gallagers construction :

Gallager invented his LDPC construction of regular codes in 1960 which he states that the parity check matrix is divided into sets, each set contains M/w_c rows where each rows has w_r consecutive ones from left to right. As for the rest of sets, the row weight is scattered randomly through each row [68].

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

Figure 2.9: Representation of Regular Gallagers construction of the sparse parity check matrix for $n = 20$, $w_r = 4$, $w_c = 3$

2. Mackay and Neal construction :

It is a method proven for constructing irregular parity check matrix. The parity matrix columns are added one by one from left to right and the weight in columns are chosen to satisfy the degree distribution, as for the position of ‘’1’’ in the construction is totally random.

The degree distribution of rows is not exact. The process is started again until the correct row degrees agrees are obtained [68].

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

Figure 2. 10 Representation of a Regular Mackay and Neal construction of the sparse parity check matrix for $n = 12$, $w_r = 4$, $w_c = 3$

B) Tanner graph :

A Tanner graph is a bipartite graph used to represent the structure of low-density parity-check (LDPC) codes, consisting of variable nodes (VN) that represent the code bits, with each column of the parity-check matrix H corresponding to one VN, and check nodes (CN) that represent the code constraints, with each row of H corresponding to one CN, connected by edges based on the positions of 1's in H, where a variable node vi is connected to a check node cj if the corresponding element Hij in H is 1, providing a complete graphical representation of LDPC codes and helping describe the iterative decoding algorithms used for LDPC codes, such as belief propagation, with the structure of the Tanner graph, particularly the presence of cycles, affecting the performance of the decoding algorithm [69].

Let’s take the following parity check matrix :

$$H = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & C_7 & C_8 \end{bmatrix}$$

From the matrix, the equations of the parity are deducted:

$$Z1 = C_2 \oplus C_4 \oplus C_5 \oplus C_8$$

$$Z2 = C_1 \oplus C_2 \oplus C_3 \oplus C_6$$

$$Z3 = C_3 \oplus C_6 \oplus C_7 \oplus C_8$$

$$Z4 = C_1 \oplus C_4 \oplus C_5 \oplus C_7$$

The codewords are represented as the variable nodes and the equations as the check nodes.

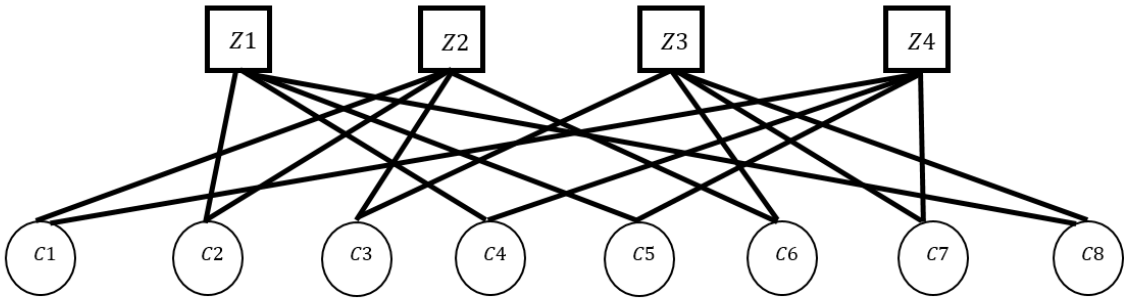

Figure 2.11: Representation of the Tanner graph.

C) Encoding Process :

As mentioned previously, an LDPC code is defined by the sparse parity check matrix H that's has to be constructed either by Gallagher's method or Mackey and Neal method. The main role of this matrix is to designate the structure to the code. Therefore, the generator Matrix G is deducted from the parity check matrix. For the LDPC encoding, the same steps of encoding Block codes are applied in this code, the major difference between these methods is and then the codeword is obtained by multiplying the generator matrix with the original message [68][69].

C) Decoding of an LDPC code :

1. Hard Decision Decoding:

Hard decision decoding is a method for decoding error-correcting codes in which the received signal is compared to a predetermined threshold to decide whether the transmitted bit is 0 or 1. In hard decision decoding, the received signal is quantized to the nearest valid symbol in the code alphabet, which is usually 0 or 1 in binary codes. This method is often used in digital communication systems that encounter noise or interference, resulting in a low signal-to-noise ratio. The decoder makes a judgment based on the threshold voltage, decoding signals above it as 1 and voltages below as 0. Hard decision decoding is simpler and less computationally intensive than soft decision decoding, however there may be restrictions in mistake correction capability.

The following are the important steps in hard decision decoding:

- Receive the encoded codeword r(x), which was sent across the noisy channel.
- Make hard decisions on each received symbol, quantifying it to the next valid symbol in the code alphabet. This generates the hard decision vector ĉ(x).
- Apply the parity-check matrix H to get the syndrome S(x) from the hard decision vector ĉ(x). If S(x) is zero, the codeword is presumed to be error-free.
- Use algebraic approaches, such as the Berlekamp-Massey or Euclidean algorithms, to determine the error locator polynomial Λ(x) from the syndrome .
- Locate the roots of Λ(x) to identify the error sites. The mistakes are supposed to occur at the points corresponding to the roots of Λ(x).
- Correct errors by flipping bits at recognized points in the hard choice vector ĉ(x).
- The adjusted vector represents the final hard decision-decoded codeword.

Hard decision decoding has a limited error correcting capability, as it can only correct up to t errors, where t represents the code's designed error correction capabilities.

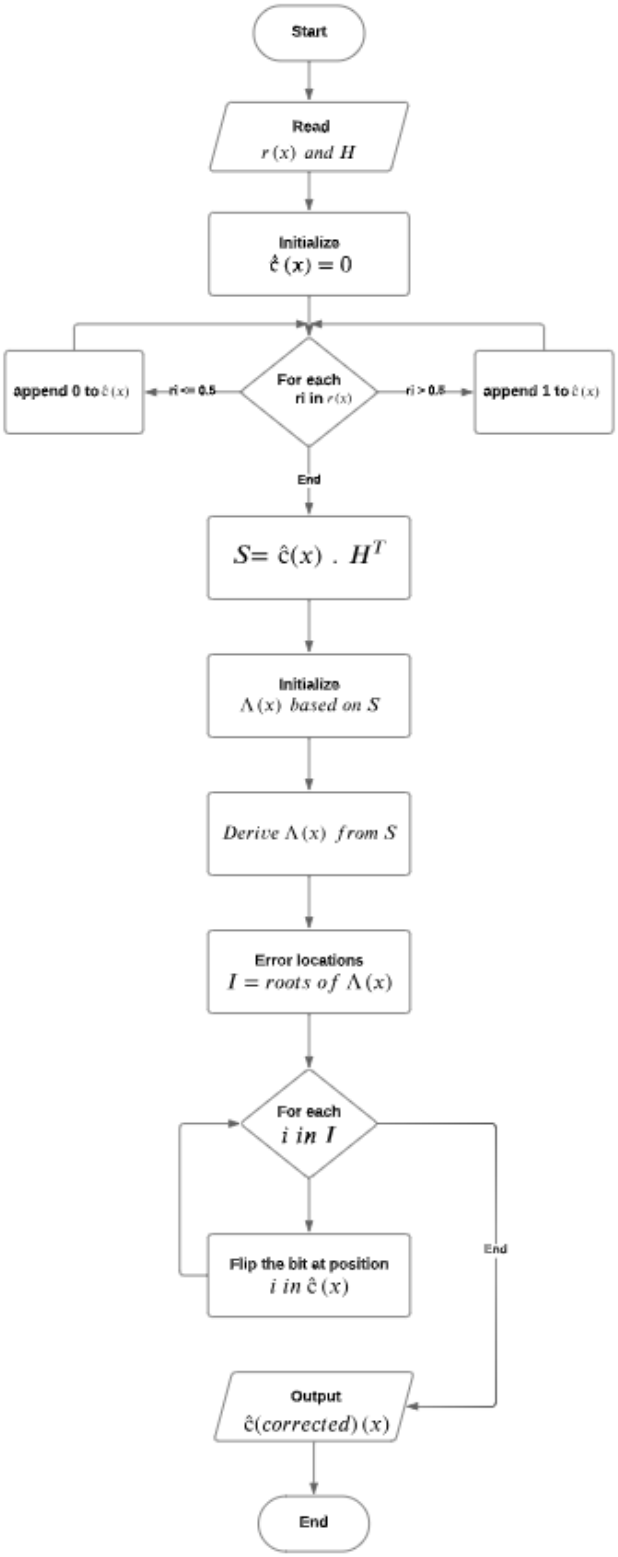

Figure 2. 12:An Algorithm representing Hard Decision Decoding

2. **Message Passing Algorithm:**

Message passing decoding is a powerful iterative technique used to decode low-density parity-check (LDPC) codes and other error-correcting codes. The decoding process involves passing messages between variable nodes (representing code bits) and check nodes (representing parity constraints) in the Tanner graph representation of the LDPC code, with the messages passed between nodes representing beliefs about the values of the code bits [70].

The MPA mainly works on the Marginal probabilities of the codeword and that by updating the messages iteratively and contains two types of messages [70]

For decoding using MPA, a set of steps or rules have to be followed:

- **Graph Setup:**

  Define the graphical model, which includes nodes (variables) and edges (dependencies between variables).

  Each node represents a variable, and each edge represents a probabilistic dependency between the variables.

- **Initialization:**

  Initialize the messages. Messages are usually probabilities or likelihoods that are sent from one node to another.

  Each message is initialized based on the prior knowledge of the network or set to a neutral value (like uniform distribution).

- **Message Computation:**

  For each edge in the graph, compute the message to be sent from one node to another. This computation involves aggregating information from all other incoming messages to the node except from the node to which the message will be sent.

  The formula for message update typically depends on the type of graphical model. For instance, in a Bayesian network, the messages are computed based on the product of incoming messages and the local likelihood function.

- **Message Passing:**

  Pass the computed messages along the edges of the graph. This can be done synchronously (all at once) or asynchronously (one at a time).

  Iteratively update the messages until convergence (when messages do not change significantly between iterations) or for a fixed number of iterations.

- **Belief Update:**

After the final iteration, compute the beliefs at each node. Beliefs are the posterior probabilities of the variables and are computed by combining all incoming messages to a node with the node's initial value (prior).

This step often involves normalization to ensure that the computed beliefs sum to one (probabilistic models).

- **Decision Making:**

Use the beliefs to make decisions or predictions. For example, in a classification task, you might choose the class with the highest posterior probability.

- **Termination:**

The algorithm terminates either after a fixed number of iterations or when the change in message values between consecutive iterations falls below a predetermined threshold.

2.2.2 Convolutional codes

Convolutional codes were firstly introduced by Peter Elias in 1955. They were considered as a great alternative for block codes where unlike them, convolutional codes do not have a finite block length and rather have a memory called ‘’shift register’’ due to the data sequence being transmitted serially rather in block codes. They are primarily defined by their ability to store a memory using ‘’Shift registers’’ which is a type of digital circuits that make the data Shift from one position to the next serial inputs and outputs via sliding bit window. They are represented by the parameters (n, k, m).

Convolutional codes are accredited due to their ability to detect and correct random errors, burst errors or both. Also they tend to be much easier than block codes to encode and decode using polynominal or graphical encoding. During the process, in order to create the encoded message, the bits contained into the original message are integrated using the mathematical operation Mod-2 addition. The binary imput data is being shifted along the shift register k-bit at a time. For every k-bit input, there is an n-bit output. As for the decoding process, convolutional codes can be decoding using either ‘’Sequential decoding’’ or ‘’Viterbi decoding’’.

2.2.2.1 Convolutional codes properties

A) Code Rate R :

It is a measurement of the efficiency of the code given by the equation below :

$$Code\ Rate\ R = \frac{k}{n} = \frac{number\ of\ imput\ bits}{number\ of\ output\ bits} \tag{2.17}$$

Which means that for each k-input bits the encoder provides a n-output bits

B) Constraint length (L):

The constraint length L is known as the amount of impute bits in the shift register that can affect the output bits where:

$$L = k(m-1) \tag{2.18}$$

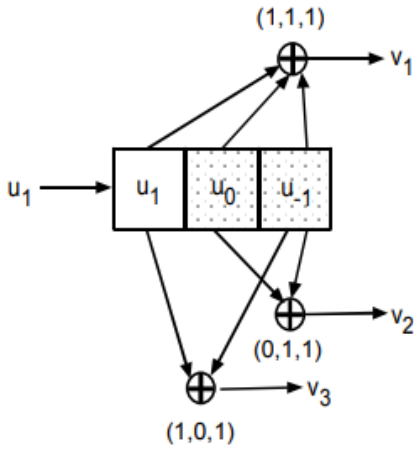

Figure 2.13: This (3,1,3) convolutional code has 3 memory registers, 1 input bit, 3 output bits

2.2.2.2 Convolutional code structure :

Convolutional codes are very simple to implement once the parameters are known. The number of shift registers is the number of the boxes drawn, then n mod-2 adders are added wich represent the encoded data [71].

After successfully constructing the structure, the output data is selected using generator polynomials. These polynomials are and can be selected by the number of the memory registers (m) where exists several list by scientist like ‘’Peterson and Weldon’’, ‘’Busgang’’ or it can be obtained by computer simulation [71].

Constraint Length	G_1	G_2
3	110	111
4	1101	1110
5	11010	11101
6	110101	111011
7	110101	110101
8	110111	1110011
9	110111	111001101
10	110111001	1110011001

Table 2.1: Generator polynomials found by Busgang for good rate 1/2 codes.

2.2.2.3 Encoding of convolutional codes

A) Mathematical encoding:

For a convolutional code, in order to create the encoded message, the initial sequence has to be multiplied by the impulse response.

$$v = u \times g \tag{2.19}$$

The encoded sequence is created by sliding the generator matrix g across the input message and executing matrix multiplication per step. The sliding application of the generator function gives rise to the phrase "convolutional coding" [72].

The generator matrix g can be represented in numerous ways, including the shift register view and the state machine view, which offer alternative viewpoints on the encoding process. For the purpose of obtaining a profound understanding [72], we would take an example of a sequence D=010011 with the convolutional code of (3, 1,3)

At state 0, the encoders register contains only ‘’0’’ bits

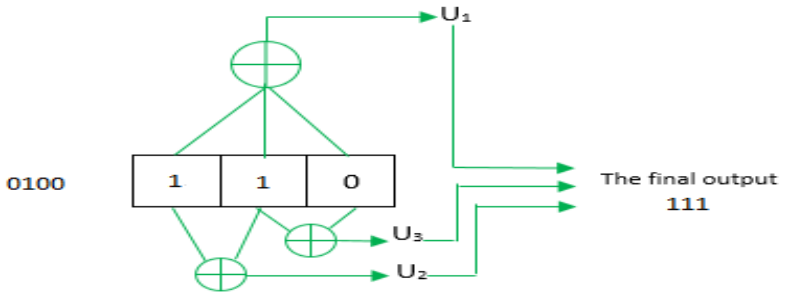

Figure 2.14: Convolution code of (3,1,3) at state 0

At the state 1, the shift moves by 1 register because it is a -1-input bit register.

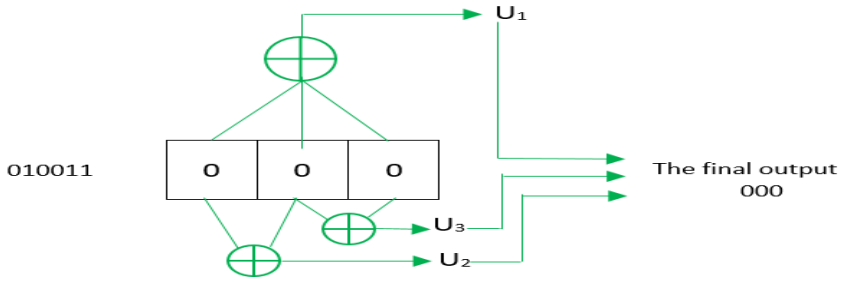

Figure 2. 15 Convolution code of (3,1,3) at state 1

At the state 2,

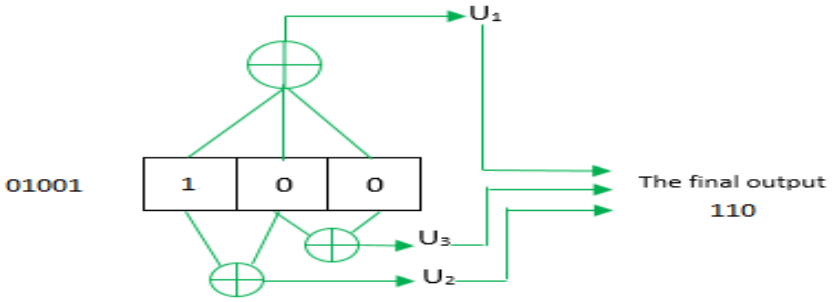

Figure 2.16: Convolution code of (3,1,3) at state 2

At the state 3,

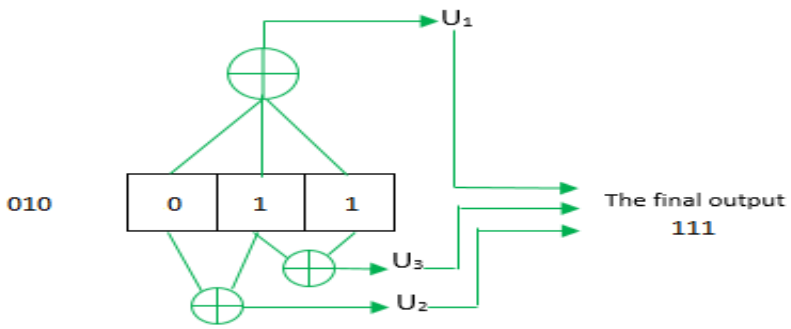

Figure 2.17: Convolution code of (3,1,3) at state 3

At the state 4,

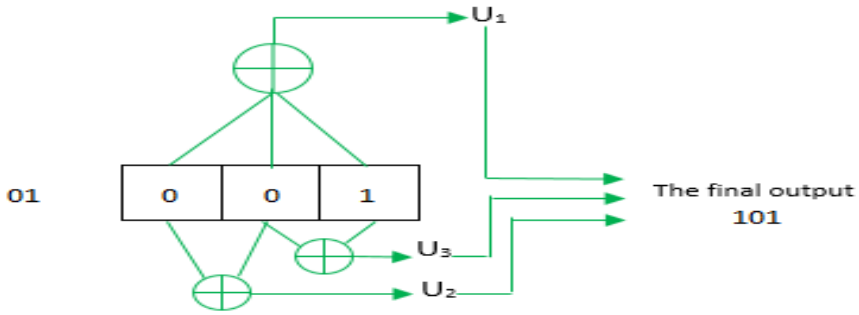

Figure 2.18: Convolution code of (3,1,3) at state 4

At the state 5,

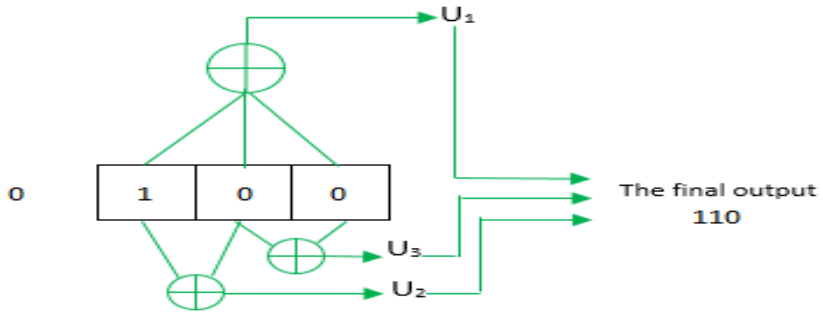

Figure 2.19: Convolution code of (3,1,3) at state 5

At the state 6,

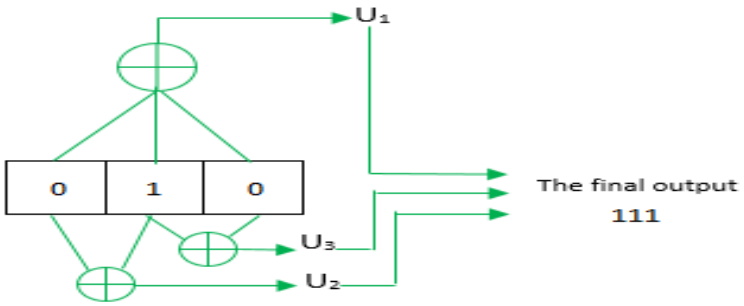

Figure 2.20: Convolution code of (3,1,3) at state 6

In conclusion, an impulse response was created for this encoder: 000 110 111 111 101 110 111 and then this impulse response is multiplied by the initial sequence to obtain the encoded message.

B) Graphical Encoding:

In graphical encoding, the encoder uses a look up table which consists of:

- Input bits
- The state of the encoder
- Output bits
- Output states

Input Bit	Input State				Output Bits			Output State		
I_1	s_1	s_2	s_3		O_1	O_2		s_1	s_2	s_3
0	0	0	0		0	0		0	0	0
1	0	0	0		1	1		1	0	0
0	0	0	1		1	1		0	0	0
1	0	0	1		0	0		1	0	0
0	0	1	0		1	0		0	0	1
1	0	1	0		0	1		1	0	1
0	0	1	1		0	1		0	0	1
1	0	1	1		1	0		1	0	1
0	1	0	0		1	1		0	1	0
1	1	0	0		0	0		1	1	0
0	1	0	1		0	0		0	1	0
1	1	0	1		1	1		1	1	0
0	1	1	0		0	1		0	1	1
1	1	1	0		1	0		1	1	1
0	1	1	1		1	0		0	1	1
1	1	1	1		0	1		1	1	1

Table 2.2: Look up table for the encoder of code (2, 1, 4)

1. **State Diagram :**

A state diagram for a convolutional code is a graphical representation of the convolutional encoder's various states and transitions between those states, providing a compact way to visualize the encoding process and determine the encoder's output for any given input sequence. Each node in the diagram represents a possible state of the convolutional encoder. The number of nodes equals $2^{(L-1)L}$, where L is the constraint length and k is the number of input bits per encoder step . Each node has 2^{L} branches that match to the potential input bits . Each branch is labeled with the input bit (0 or 1) and the associated output bits[73].

Solid lines show transitions for input bit 0, while dashed lines represent transitions for input bit "1", To compute the encoder's output for a particular input sequence using the state diagram, begin with the initial all-zero state and follow the branches corresponding to the input bits (solid for 0, dotted for 1), with the output bits indicated by the labels on the branches. The state diagram, along with the tree diagram and the trellis diagram, is one of the three basic graphical representations of convolutional codes. It gives a straightforward way to understand the encoding process and may be used to study the distance features of the code.

Using the look up table in the section above, The state diagram is structured like the image below [73].

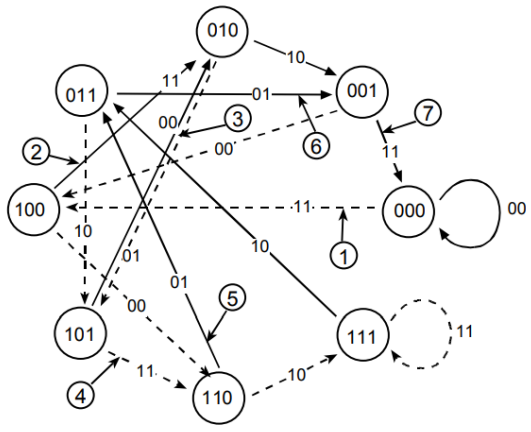

Figure 2.21: State Diagram of (2, 1,4) code

Therefore, if we deconstruct this diagram, we start by :

State 000:

The arrived bit is ''1'' and the output state is ''100'' and ''11'' output bits

State 100:

The arrived bit is ‘’0’’, the output bits ‘’11’’, the output state is ‘’010’’

State 010:

The arrived bit is ‘’1’’, the output bits ‘’01’’, the output state is ‘’101’’

State 101:

The arrived bit is ‘’1’’, the output bits ‘’11’’, the output state is ‘’110’’

But in order to create the complete encoded word the state diagram has to be returned to state 000

State 110:

The arrived bit is ‘’0’’, the output bits ‘’01’’, and the output state is ‘’011’’

State 011:

The arrived bit is ‘’0’’, the output bits ‘’01’’ and the output state is ‘’001’’

And finally state ‘’000’’ with an output bits of ‘’11 ’’ the encoded message is 11 11 01 11 01 01 11

2. Tree diagram :

The tree diagram representation provides a visual depiction of all possible information and encoded sequences for a convolutional encoder, illustrating the encoding process through a tree-like structure where each node corresponds to a potential encoder state, and branches emerging from each node represent the possible input bits (0 or 1), with the branches labeled accordingly with the corresponding output bits or symbols. To encode a message using a tree diagram, start at the root node (leftmost node) and follow the higher branch if the input bit is 0 or the lower branch if the input bit is 1, with the labels on the traveled branches indicating the output bits or symbols[72][73]. This graphical representation allows you to easily visualize the encoding process and calculate the output sequence for a particular input message[73]

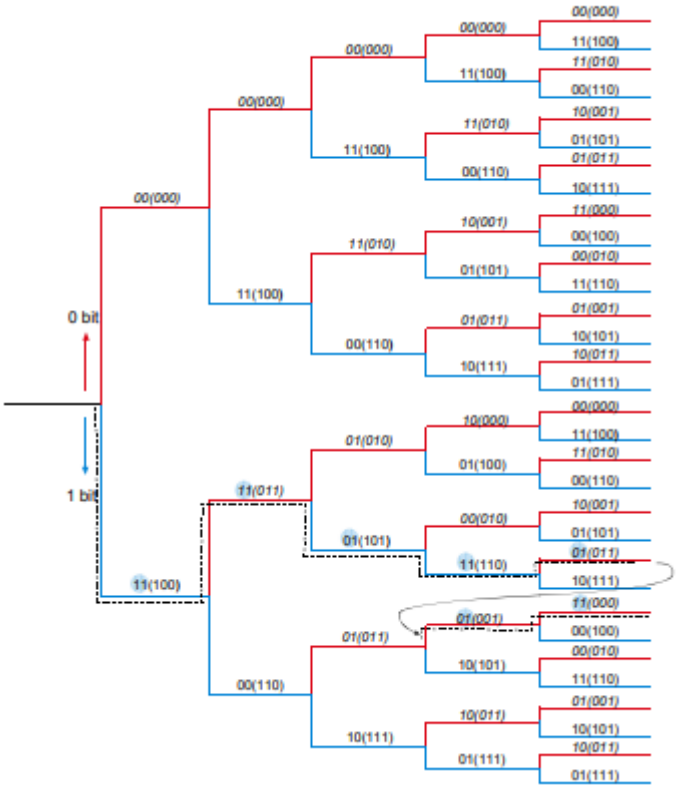

Figure 2.22: Tree diagram of (2, 1, 4)

If we break down the diagram, we can observe that:

At state ‘’000’’:

The arrival bit is ‘’1’’ which means a go downwards on the tree branch with a shaded line. The bits between parentheses represent the state ‘’100’’ and the two left bits represents the output bits ‘’11’’

At state ‘’100’’:

The arrival bit is ‘’0’’ which means we go up the tree branch with a solid line. The output state is ‘’011’’, the output bits are ‘’11’’

At state ‘’011’’:

The arrival bit is ‘’1’’, we go downwards. The output state is ‘’101’’ and the output bits are ‘’01’’

At state ‘’101’’:

The arrival bit is ‘’1’’, we go downwards. The output state is ‘’110’’ and the output bits are ‘’11’’

Regarding tree diagram, flush bits are necessary to be added to the sequence in order to reset the internal state of the encoder.

Therefore, when encoding these flush bits, we return to phase two of the diagram and repeat the same process discussed above

3. Trellis diagram :

The trellis diagram is a graphical representation of the encoding process of a convolutional code, depicting the encoder's different states and transitions between them. It is organized into vertical columns that represent multiple time instants, each with nodes representing potential encoder states. The branches connecting these nodes illustrate state changes, which are labeled with the matching input and output bits or symbols. During the encoding process, the encoder starts in an initial state, moves through the permissible states, and accepts k input bits that are shifted into the shift register. Based on the current state and input bits, the encoder enters a new state and generates n coded bits or modulation symbols, with the output dictated by the labels on the branches. This graphic representation helps to comprehend the dynamic behavior of the convolutional encoder and the encoding process [73]. The following trellis diagram encodes the sequence D=1011 using

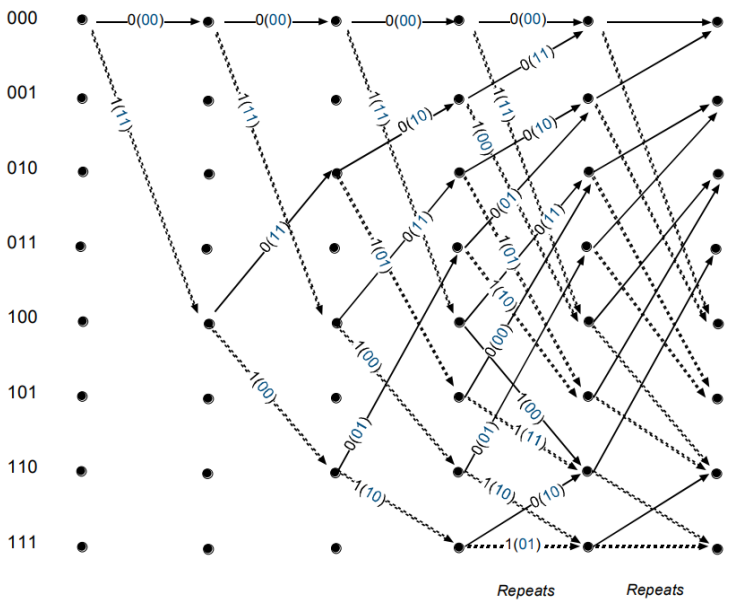

Figure 2.23: Trellis diagram of (2, 1, 4) code

2.2.2.4 Decoding Process :

For decoding convolutional codes there exist several methods designed to efficiently decode these convolutional codes such as : sequential decoding (Fano Algorithm), Maximum likely-hood decoding (Viterbi decoding) and soft/hard decision decoding.

By far, the best method used in convolutional decoding is the maximum likely-hood decoding more specifically the Viterbi decoding. The Viterbi decoding was firstly introduced by Andrew Viterbi in 1967 which uses the Trellis structure in order to obtain the correct and possible states of the convolutional code while decoding and it works by examining the whole sequence. The Viterbi Algorithm has a set of rules or steps that is crucial for obtaining the recovered message [73].

A) Branch Metric Calculation :

The decoder starts by calculating a specific Metric for each path which can be either Hamming Distance or Euclidean distance but in most cases the most use dis Hamming Distance.

The path is chosen based on the results of this metric, if it is lower, it is the right path and if it is higher, the path is discarded.

Bits Received	Valid Codeword 1	Valid Codeword 2	Hamming Metric 1	Hamming Metric 2
00	00	11	2	0
01	10	01	0	2
10	00	11	1	1

Table 2.3: Each branch has a Hamming Metric Depending on what was received and the valid codewords at the state

For better understanding, we will decode the sequence 01 11 01 11 01 01 11 received using the trellis diagram with the Viterbi decoding

Step 01:

The received bits are: 01

According to the trellis diagram made before, at this step exist only 02 paths that can be taken :

At the path: 000 ⟶ 000 :the output bits are 00

the received bits are 01

the path Metric = 1

At the path 000 ⟶ 000: the output bits are 11

the received bits are 01

the path Metric = 1

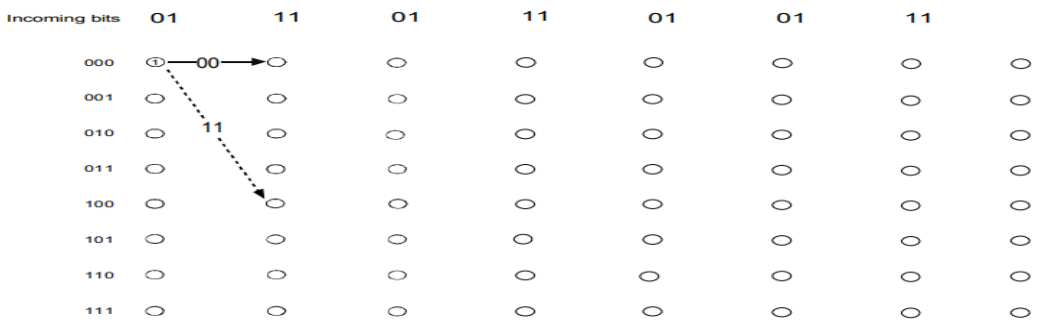

Figure 2.24: Viterbi decoding, Step 1

Step 02:

In step two, the decoder jump from 02 possible states to 04 possible states

At state 000: Following the Trellis diagram

00 ⟶ 000 : the output bits are 00

the received bits are 11

the path Metric = 2

00 ⟶ 100 : the output bits are 11

the received bits are 11

the path Metric = 0

At the state 100 : following the trellis diagram

100 ⟶ 010: the output bits are 11

the received bits are 11

the path Metric = 0

100 ⟶ 110: the output bits are 00

the received bits are 11

the path Metric = 2

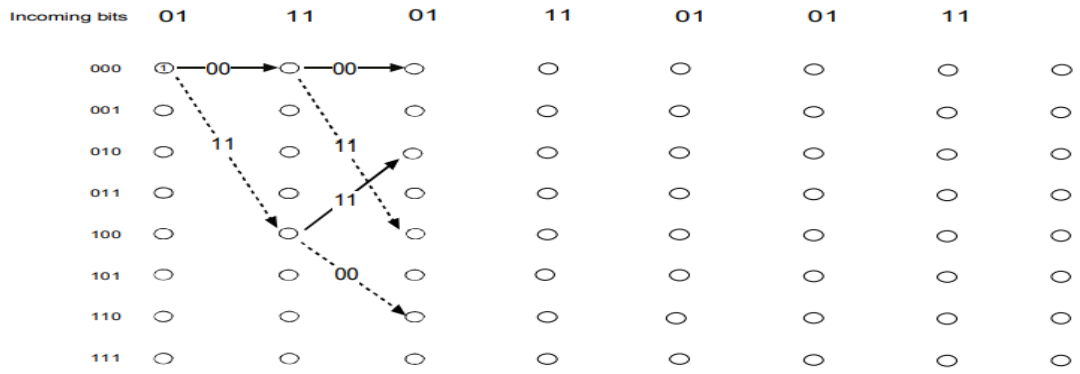

Figure 2.25: Viterbi decoding step 2

Step 03:

The decoder jumps from 4 states to 8 states

At state 000: following Trellis diagram

000 ⟶ 000: the output bits are 00

the received bits are 01

the path Metric = 1

000 ⟶ 100: the output bits are 11

the received bits are 01

the path Metric = 1

At state 010 :

010 ⟶ 001: the output bits are 10

the received bits are 01

the path Metric = 2

010 ⟶ 101: the output bits are 01

the received bits are 01

the path Metric = 0

At state 100 :

100 ⟶ 010: the output bits are 11

the received bits are 01

the path Metric = 1

100 ⟶ 110: the output bits are 00

the received bits are 01

the path Metric = 1

At state 110 :

110 ⟶ 011: the output bits are 01

the received bits are 01

the path Metric = 0

110 ⟶ 111: the output bites are 01

the received bits are 01

the path Metric = 0

As seen in the figure above, all states has at least one path that goes through them which means that the trellis is fully populated

At the following step, two metric will be calculated for each path with the same method used before as it shows in the figure below

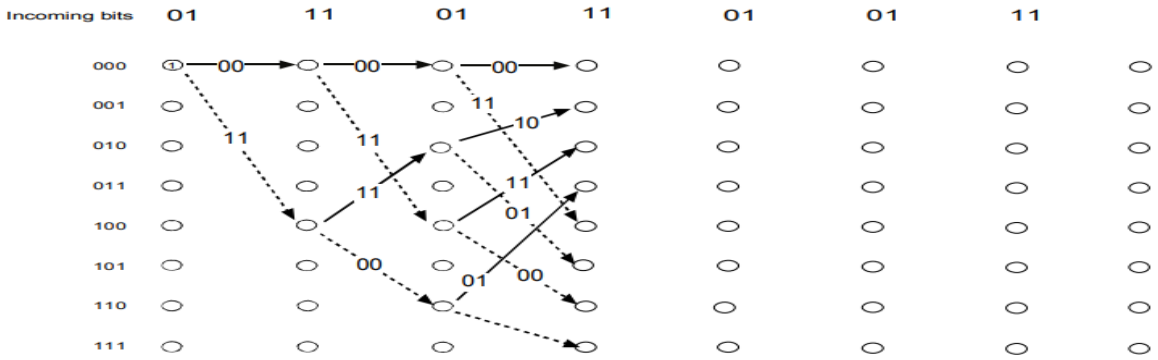

Figure 2.26: Viterbi Decoding step 3

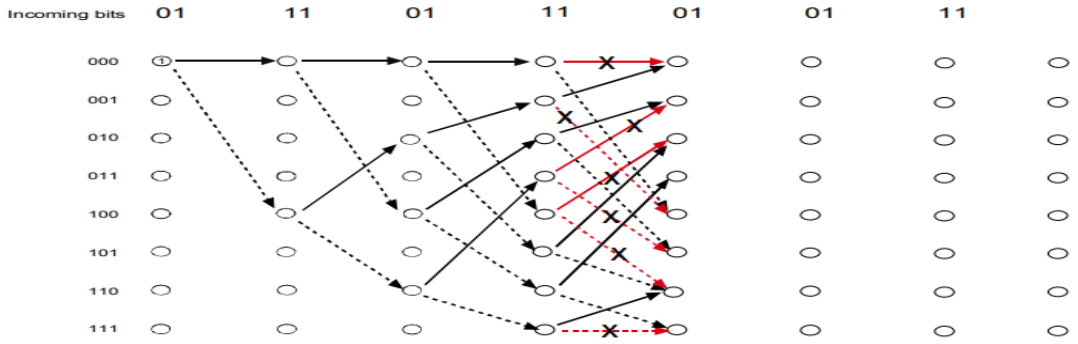

Figure 2.27: Viterbi Decoding step 4

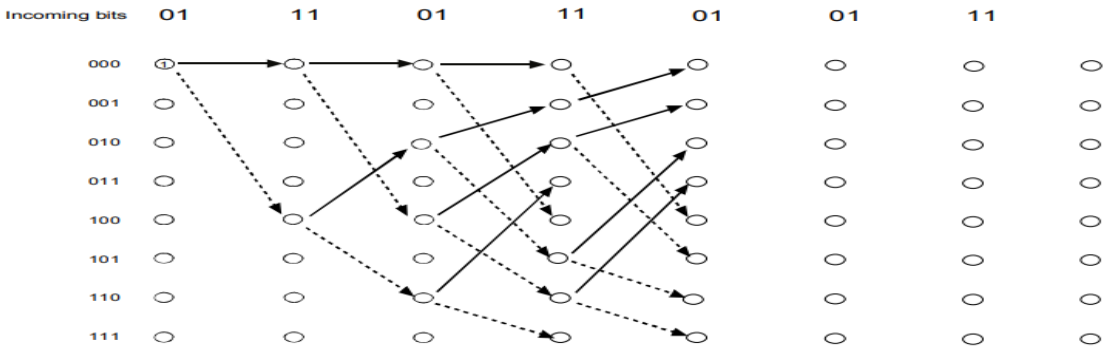

Figure 2.28: Viterbi Decoding step 4 after discarding

Step 05:

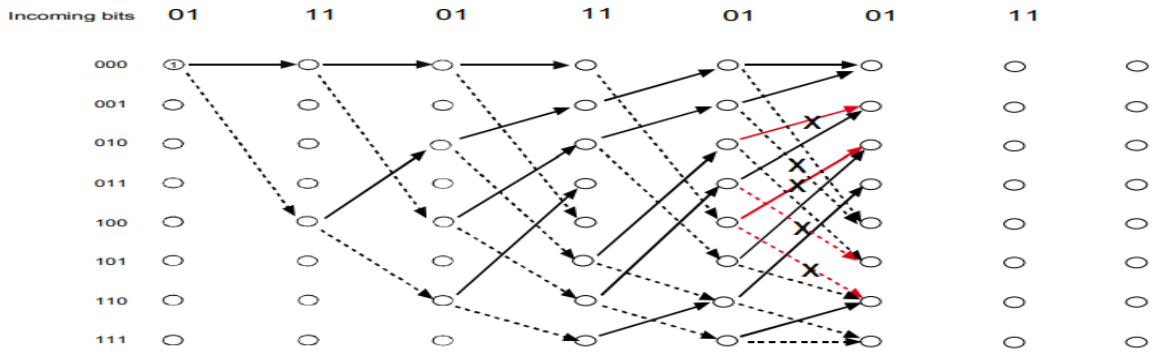

Figure 2.29: Viterbi Decoding step 5

Step 6:

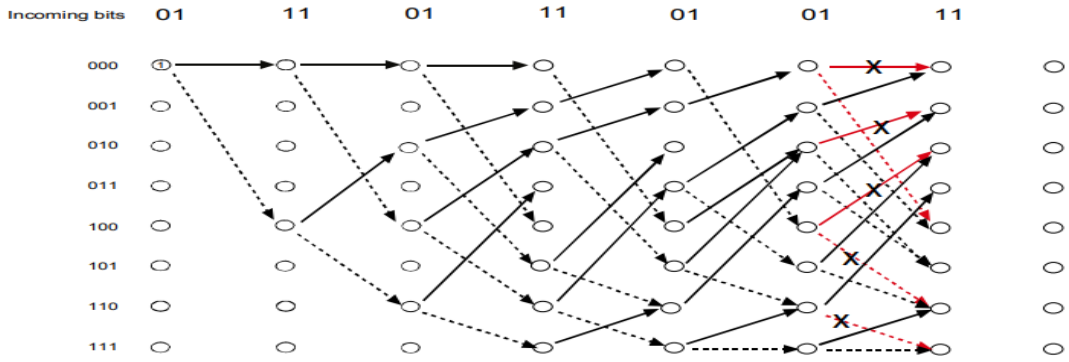

Figure 2.30: Viterbi Decoding step 6

Step 7:

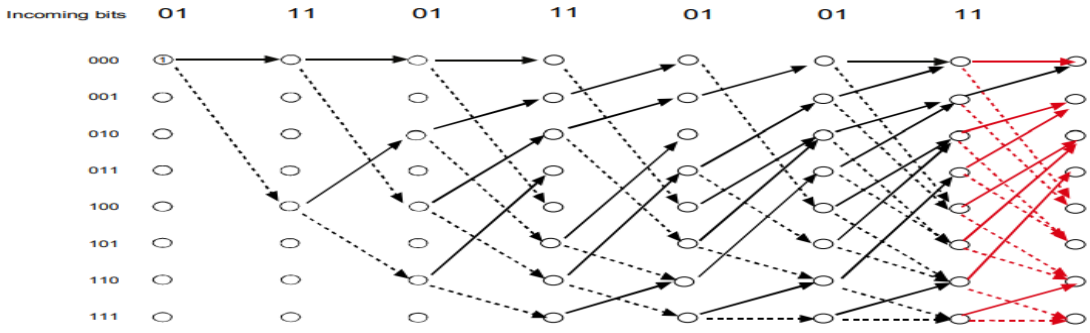

Figure 2.31: Viterbi Decoding step 7

2.3 Error Correcting codes performance Measurement

The performance of error-correcting codes is critical for determining their usefulness in identifying and repairing errors during data transmission and storage. The performance of these codes is measured using a variety of criteria, including error detection rates, error correction capabilities, and efficiency in maintaining data integrity. Error-correcting codes' performance is measured by their capacity to detect and rectify errors accurately and efficiently, hence improving the dependability and robustness of digital communication systems.

2.3.1 Bit Error Rate:

Bit error rate is one of the many methods used to measure the performance, reliability of the channel transmission and the quality of the received data. It is expressed as a ratio or percentage, which represents the probability of the BER.

Once calculated, the lower the BER is, the better the transmission

$$BER = \frac{Errors}{Total\ number\ of\ bits}$$

2.3.2 Coding gain:

Another method of measuring the performance of different coding schemes is the coding gain. Coding gain uses the difference between the SNR (Signal to noise ratio) levels between the coded and the uncoded messages or signals. It is represented using decibels (dB) in order to elevate the reliability of data transmission oven noisy channels.

It is calculated using the equation below:

$$CG = SNR_{coded} - SNR_{uncoded} \tag{2.20}$$

2.3.3 Coding rate:

In order to measure the proportion of the coded and uncoded data is that uses Error correction coding the code rate utilized is the coded rate is expressed using the following equation.

$$R = \frac{k}{n} \tag{2.21}$$

where for every k information bits, the encoder generates a n coded bits adding an efficient coding process results in a lower code rate hence a better error correction.

2.4 Conclusion

In conclusion, classical error correction codes include both block codes and convolutional codes, each with unique advantages and uses. Block codes, such as Hamming and Reed-Solomon codes, offer reliable error correction for fixed-size data blocks, whereas convolutional codes, such as Viterbi and Reed-Muller codes, excel when data is delivered as a continuous stream. These classical coding techniques have helped to improve the reliability and performance of different communication and storage systems.

As we go toward quantum computing, the emphasis switches to quantum error correction codes, which play an important role in minimizing the fragile nature of quantum information. The following chapter will look at the various types of quantum error correction codes, including stabilizer codes, topological codes, and surface codes, among others. These quantum codes use quantum mechanics' unique qualities to detect and fix defects in quantum systems, paving the path for the development of fault-tolerant quantum computers.

The creation of effective and scalable quantum error correcting codes is a significant challenge in the realm of quantum computing. Researchers are currently investigating new coding schemes, optimization approaches, and hardware-software co-design methodologies to improve the reliability and performance of quantum computers. Advances in this sector will have far-reaching repercussions, enabling the practical deployment of quantum technologies and unleashing the transformational potential of quantum computing across numerous domains, from cryptography and simulation to optimization and beyond.

Chapter 3

Quantum Error Correction Codes

Quantum error correction has become one of the most important topics of quantum information theory, since any practical quantum information processing is impossible without efficient methods for error correction. It is defined that quantum error correction certain quantum mechanical operations executed on a physical states can be effectively reversed without looking at the state itself. An error correcting code is defined by the set of reversible operations on the physical state and the logical subspace on which the information is encoded. These codes must be implemented in such a way that the logical information can be recovered with high probability using only the results of error syndrome measurements followed by a correction procedure carried out on the physical state. Correct ability against general (i.e. nonlocal) noise requires that errors occurring in the code space can be detected from their results on the code space alone. This subject is interesting as it is automatically with a fault tolerant quantum computation, since the ability to efficiently correct errors is a necessary and sufficient condition for reliability of a computation. But it will have implications for quantum information processing in general, because error correction will eventually be necessary in any context where one wishes to preserve quantum information for a nontrivial length of time.

3.1 Definition of Quantum Error Code Correction

A quantum error-correcting code detects and corrects errors in a quantum state without measuring the state itself. Error correction for a general quantum code can be done in two ways. The first is by encoding the information qubits into a subsystem of many qubits, a code with large distance properties between the code words can then be used to detect and correct errors that occur and are limited to affecting only a few of the qubits in each code word. The second method is to perform error correction on a noisy quantum channel that is degrading a quantum state. This method is more analogous to error correction in classical information theory, it sets

the foundation for the study of fault-tolerance in quantum computation. An important aspect of quantum error correction that differentiates it from classical error correction is the no-cloning theorem. This theorem states that it is impossible to make an identical copy of an arbitrary unknown quantum state. This has implications on the ways in which we can backup quantum information to protect against data loss.

3.1.1 Importance of Error Code Correction:

Quantum error correction (QEC) is vital for the success of both quantum computations and quantum cryptographic tasks. Understanding the potential advantages of quantum computing necessitates comparing it to the reliability of classical computation methods, which benefit from well-established error correction techniques. Without effective QEC, quantum computing cannot achieve the reliability of classical systems. As large-scale quantum computers become feasible, QEC becomes essential to ensure their reliability, especially in the face of errors arising from decoherence and various noise sources. Traditional error correction methods are insufficient for quantum errors, making effective QEC crucial for building practical quantum computers. QEC detects errors by encoding quantum information and comparing it to predefined patterns, allowing for error detection and potential correction while preserving the integrity of the quantum state. Although QEC may incur overhead, its benefits include increased reliability, improved error detection and correction, and enhanced fault tolerance, all essential for advancing quantum computing capabilities.

3.2 Theoretical Foundations of Quantum Error Code Correction:

Theoretical foundations of quantum error code correction picks up the story at the point where the Dennis and Kitaev conditions, which are necessary and sufficient conditions for transversal universality, apply to quantum error correction. Justifying why quantum error correction is a topic worthy of independent study rather than a direct analogue of a well understood classical theory. By drawing parallels between the classical and quantum theories in this way we obtain greater insight into what is uniquely quantum about a given concept and learning how it must be used differently in the quantum case. Often found useful to resort to the classical case to help understand a concept in quantum error correction. Constructing new codes that makes them more or less powerful or more or less like the classical version of the theory. This understanding is a necessary preliminary to attempting to generalise the theory or

to apply the concepts from quantum error correction to other areas of quantum computation or quantum information theory [74][75][76].

3.2.1 Overview of Quantum Error Code Correction

Error correction is an internationally significant area in computing science in both the classical and quantum fields. Classical error correction deals with the protection and recovery of classical information. This is often done by introducing redundancy and then applying a correction algorithm. When considering quantum information, the situation is more complicated due to the distillation theorem. This shows us that we cannot make use of simple copying and comparison techniques to correct errors in quantum systems. However, we can still take classical codes and apply them to protect quantum information. An example of this would be encoding a logical qubit into a subsystem of many physical qubits using a classical code and then using the syndrome measurements to detect and have some level of correction an error event. A quantum error correcting code is a unitary transformation on k qubits, effectively entangling them with a subsystem of n-k qubits such that the original information can be recovered through measurements on the subsystem without the need for further corrections. The code has to be constructed so that the information can be recovered with a threshold level of accuracy, i.e. we still have the possibility of success greater than 1/2. Given a quantum error correcting code, fault-tolerant quantum computation is possible [74][82][86]. This is a method of computation where we apply gates to our encoded logical qubits that are spread across the n qubit subsystem. The n qubit gates cause a high probability of error due to noise and faults in the quantum computer. Usually, a standard quantum circuit model would apply error correction between each gate by measuring the syndrome and then applying corrections. This is not efficient because the error correction itself is also susceptible to error and the code must be designed so that recovery is still possible at this level. The threshold theorem shows that given a set of faulty quantum gates with an error rate below a certain threshold, it is possible to perform logical operations arbitrarily accurately, provided the physical error rate can be arbitrarily reduced with increasing n [74][82][81].

3.2.2 Error Models in Quantum Systems

$|0\rangle$and $|1\rangle$ are basis vectors, and the error occurs if the state of any basis vector is changed, In the theory of error correction for quantum computers, we recognize three types of errors:

3.2.2.1 Depolarizing Errors

Depolarizing errors are a natural choice for a quantum error model, especially if you are interested in considering the effect of noise in a purely classical context (i.e., the noise affects the measurement process only). Depolarizing errors affect a quantum bit (qubit) by randomly applying one of the three Pauli operators with equal probability. This error model can be extended to a channel which acts on an n-qubit state by having the channel act independently on each qubit. This formulation is called the independent error model, and when considering these errors, it is often assumed that the error rate is small. In this context, the error on an n-qubit state can be expressed as the simultaneous application of m depolarizing errors, where m is the number of qubits, and the total error rate is p = m/N, where N is the total number of degrees of freedom in the system.

3.2.2.2 Bit Flip Errors

Let us now consider a model for a special type of error: the bit-flip error. This error model occurs when a qubit, which can be thought of as a vector in the usual 3-dimensional space models, has its basis states |0⟩ and |1⟩ mixed up. Equivalently, if we represent the qubit state by a point on the Bloch sphere (e.g. |0⟩ = |up>, |1⟩ = |down>), a bit-flip error would cause the point to move in a direction perpendicular to the |0⟩|1⟩-plane. In the absence of other error processes, this means that the qubit undergoes a Pauli operation X, causing a change of |0⟩ to |1⟩ and vice versa. From this, it is clear that the bit-flip error is a Pauli error.

3.2.2.3 Phase Flip Errors

Phase flip errors are sometimes seen as the generalization of bit flip errors. A phase flip error occurs whenever a |1⟩ state changes to a -1 state or a |0⟩ state changes to a 0. Pure phase errors only occur between states of the same energy, an example of this is the states |0⟩ and |1⟩ in a classical bit. If the state has changed energy, then the error can be understood as a combination of a bit flip error and a phase error. When a bit is subjected to a higher magnetic field, a lower energy state is created and so a bit flip occurring at this state would be a phase error on the original bit. Thus a phase error can be seen as a rate of interconversion between bit flip and phase flip errors.

3.3 Types of Quantum Error Codes

3.3.1 Classical based quantum Error Codes

3.3.1.1 Quantum Low Density Parity Check

The development of quantum low-density parity check (QLDPC) codes stems from the need for efficient error correction in quantum information processing. Inspired by classical LDPC codes, QLDPC codes offer capacity-achieving performance and low decoding complexity, making them appealing for various quantum applications. These codes are essential for protecting quantum information against errors induced by decoherence and external noise. By encoding information into a quantum state on a block of physical qubits and applying qubit operations, QLDPC codes enable reversible error correction without excessive resource consumption. This capability is crucial for reliable quantum communication and computation, aligning with the goals of quantum error correction theory. QLDPC codes represent a promising avenue for extending classical error correction concepts to the quantum domain, ensuring the robustness of quantum systems in the face of noise and decoherence [74][75][76].

A) Importance of Quantum Low Density Parity Check

LDPC quantum error correction code is the code derived from the classical LDPC code to a quantum code. LDPC quantum error correction code is proposed by using graph theory, where the qubit is represented by a graph and the check node will be represented by a set of qubits connected to it. LDPC quantum error correction code can correct bit and phase flips with a probability of error of about 7.5%, and furthermore, this code can be implemented with low complexity. LDPC quantum code is expected to achieve error correction performance that approaches the Shannon limit, like the classical LDPC code [79].

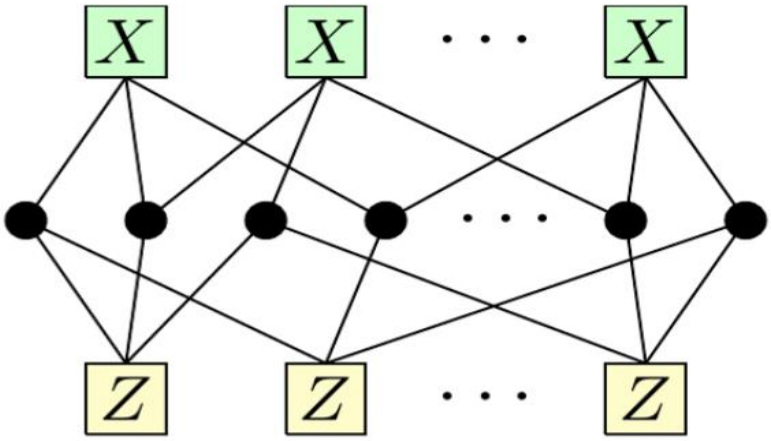

Figure 3.1: The tanner graph of the QLDPC code

B) Principles of Quantum Low Density Parity Check

1. Sparse Parity-Check Matrix:

- QLDPC codes are defined by a sparse parity-check matrix. This sparsity allows for efficient encoding and decoding techniques, as operations requiring sparse matrices are computationally simpler and demand less to perform.
- Because the parity-check matrix is sparse, each qubit is involved in only a few stabilizer checks, decreasing resource overhead and making QLDPC codes viable to implement.

2. Quantum Stabilizer Formalism:

- QLDPC codes are defined using the quantum stabilizer formalism, which depicts the code's stabilizer generators as tensor products of Pauli operators (X, Y, Z) operating on subsets of qubits.

- The stabilizer generators provide the limitations that encoded quantum states must meet, establishing a foundation for error detection and correction operations.

- QLDPC codes can effectively encode and decode quantum information while retaining its stabilizer features by modifying stabilizer generators.

3. Local Encoding and Decoding:

- QLDPC codes frequently use local encoding and decoding methods, which encode and decode data in small, local qubit neighborhoods.
  Local encoding and decoding make it easier to develop QLDPC codes by lowering operational complexity and communication overhead amongst qubits.
  Local encoding and decoding schemes further improve the scalability of QLDPC codes, enabling them to be used on larger quantum devices without dramatically raising processing needs.

4. Error Correction Capability:

- QLDPC codes are intended to detect and fix faults encountered during quantum data processing or communication.

- QLDPC codes repair mistakes resulting from noise, decoherence, and other types of quantum errors by encoding quantum data redundantly and performing error correcting algorithms depending on the code's stabilizer features.

- QLDPC codes' error correcting capability is determined by their parameters, such as code rate, distance, and decoding algorithm, in addition to the quantum system's unique noise model.

5. Fault Tolerance:

- Fault tolerance is an important aspect of QLDPC codes, since it allows them to effectively rectify errors even in the face of noise and flaws in quantum hardware.

- QLDPC codes can reduce the consequences of mistakes and protect the integrity of quantum information throughout computation or transmission by spreading redundancy across encoded quantum states and utilizing the code's stabilizer features.

- The fault tolerance of QLDPC codes is critical for establishing consistent quantum computation and exchange of information in noisy quantum systems.

6. Scalability:

- QLDPC codes are intended to be scalable, which means they can encode larger quantum systems while remaining efficient in encoding and decoding methods.

- Scalability is vital when designing quantum error correction in substantial quantum computing architectures, as the number of qubits and the difficulty of quantum algorithms continue to rise.

- QLDPC codes provide scalability by using local encoding and decoding techniques, minimizing code parameters, and adapting decoding algorithms to higher code sizes.

3.3.1.2 Stabilizer Codes Explained

The fundamental idea is extremely simple. A stabilizer code is a subspace C of n qubits that has been encoded, selected so that the several qubit steps on are in some sense 'simultaneously' corrected by a solitary action. The standard and easiest case is when the action is a Pauli error-correction operation. A Pauli operation is a step on n qubits given by a string of Pauli matrices, that is a tensor product of either I, X, Y, or Z. On a solitary qubit, X is the bit-flip, and Y and Z are distinct phase-flips. Recall that n qubit errors are routinely corrected by a chain of n single-qubit error-correction steps. A Pauli error-correcting code is one that corrects against every AGEEP of n qubits by a particular Pauli operation on the encoded bit, and an error syndrome is given by a sequence of operators that matches the actual error [77][78][79].

Stabilizer codes were perhaps the first discovered class of quantum error-correcting codes. Utilizing sluggish error-correction aspects, they have been exploited practically to put into practice error-correction on rudimentary quantum computers and to verify the mechanism of quantum error-correction. Due to these and further pragmatic features, an extensive scientific idea of stabilizer codes has been supposed for the earlier fourteen years. Nonetheless, a precise and general definition has only lately emerged. This classification formalizes numerous notions and structures that were implicit in earlier work and provides innovative understanding into the character of quantum error-correcting codes in standard [80][81][82].

A stabilizer code is a CSS code (Calderbank-Shor-Steane Code). This code can correct both bit and phase flips. The CSS code can be obtained from an LDPC (Low Density Parity Check) code by splitting the bit flip code and the phase flip code and changing one of them to its dual code. The dual code of a linear binary code C is the binary code that is orthogonal to the code C'. The notation of C' is the linear binary code where each codeword is perpendicular to all codewords of C.

3.3.2 Quantum based Error-Correcting Codes

These codes are a family of codes with a restricted set of stabilizer generators. Stabilizer codes are central to fault tolerant quantum computation as they can be efficiently decoded. A stabilizer is a tensor product of Pauli operators (X, Y, Z tensored over many qubits). These codes are believed to be the most practical for near term quantum computing implementations. One important code of this type is the [[7,1,3]] Steane code. This code encodes 1 logical qubit in 7 physical qubits and can correct any single qubit error model. Topological codes are an exciting prospect for fault tolerant quantum computation as they have a high threshold for error correction (theoretical evidence suggests greater than 1%). These codes are defined on an abstract two dimensional manifold and the code space is a result of many ground states of local Hamiltonians. The logical qubit is encoded in nonlocal observables and this allows for error correction without a direct measurement of the data qubits [83][84][85][86].

3.3.2.1 Topological Codes

Topological codes, like the Surface code, represent a relatively new class of error-correcting codes in quantum computing, characterized by their abstract nature compared to stabilizer codes. While stabilizer codes can be described using lattices formed by stabilizer generators, understanding topological codes requires a deeper exploration due to their more complex structure. Typically, topological codes are defined in terms of graphs or lattices that support the stabilizer code, although they can also be defined solely in terms of stabilizers and logical operators. Unlike stabilizer codes, which rely on a code subspace defined by simultaneous eigenvectors of stabilizer generators, topological codes possess a similar code subspace but with highly entangled stabilizers and a significant degeneracy in the phase space. This built-in redundancy enables topological codes to have substantial error correction potential, leading to large error correction thresholds. While topological codes may present a steep learning curve

due to their abstract nature, they offer promising prospects for robust error correction in quantum computing [87][88].

A) surface codes

Surface codes are two-dimensional arrays of qubits arranged on a square lattice, resembling the surface of a crystal or lattice. They are topological quantum error-correcting codes, meaning they encode quantum information in the non-local properties of the lattice rather than the state of individual qubits. Surface codes are defined by a set of stabilizer generators associated with plaquettes and vertices of the lattice. These codes have the unique property, amongst all known error correction codes to date, that their error threshold can be made arbitrarily low given a large enough system (i.e. a quantum computer with many physical qubits). This property, combined with the fact that they are only a constant factor overhead in qubit usage, makes surface codes the most exciting error correction code known to date for quantum computing [87][89].

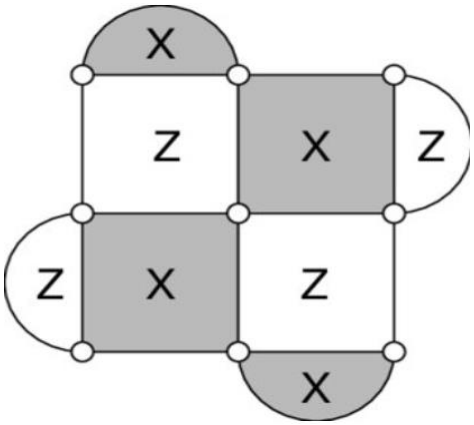

Figure 3.2: The smallest possible surface code consisting of 9 data qubits

B) Color Codes

Color codes are considered highly efficient quantum error-correcting codes, although their full potential is yet to be realized and emphasized in the literature. These codes currently excel in detecting and correcting errors at the level of individual qubit dephasing, aligning well with the capabilities of current experimental technology. Their efficiency stems from the ability to perform error correction using only qubit measurements and single-qubit gate operations, which are more feasible than two-qubit gates with low error rates [90][91]. The local nature of these operations simplifies error correction algorithms, as they involve encoding, error correction, and decoding using a small number of qubits repeatedly. An efficient quantum error-correcting

code encodes k logical qubits into a larger n physical qubit code, where n is not significantly larger than k, and can correct errors on the qubits with high success probability using only local operations [92][93]. In the case of color codes, where k equals n due to the formation of a Planar grid, the focus is on correcting dephasing errors. Despite their efficiency, the significance of color codes in quantum computation warrants further attention and recognition in research literature [94][95][96].

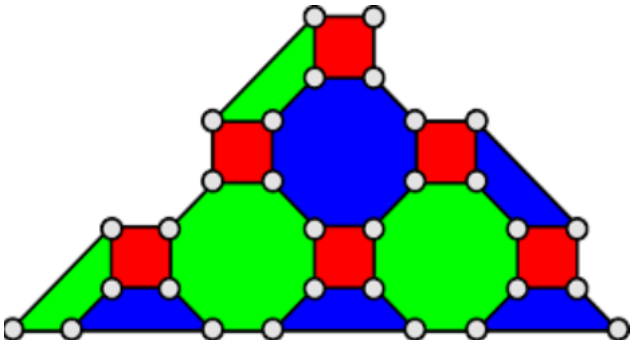

Figure 3.3: Graphical representation of the color code

3.3.3 Encoding and Decoding Quantum Error Codes

In order to use the error correcting properties of a quantum error code, one must first translate the information into a code which has stabilizer generators. This process is known as encoding and must be done carefully, keeping track of the type of quantum error as it will determine how the errors manifest on the encoded data and how they are corrected. This leads to the concept of a transversal error where a single error on a physical qubit gives rise to an error on the corresponding encoded qubit, i.e. a logical error. The conditions for correcting and detecting errors are faults in the measurement of the syndrome and logical errors must be correctable if the code is to be useful.

Encoding can be divided into two methods. The most commonly used is to run the syndrome measurement circuit (detection of errors) and then compute the result as this often requires much less computational effort. An error can then be corrected by running a correction circuit whose output is determined by the error. Since the error will usually only affect a few qubits, it will be most efficient to run the correction circuit only on the data qubits with non-trivial errors. This is known as a non-fault-tolerant (NFT) method as the error correction operations are not themselves protected by the code. The alternative is the fault tolerant method

which incorporates error correction into the original encoding process in a method known as encoding by a logical gate. This will give better error correction as the code can detect and correct more errors, but the process is more complex and is yet to be demonstrated that it can outperform the NFT method.

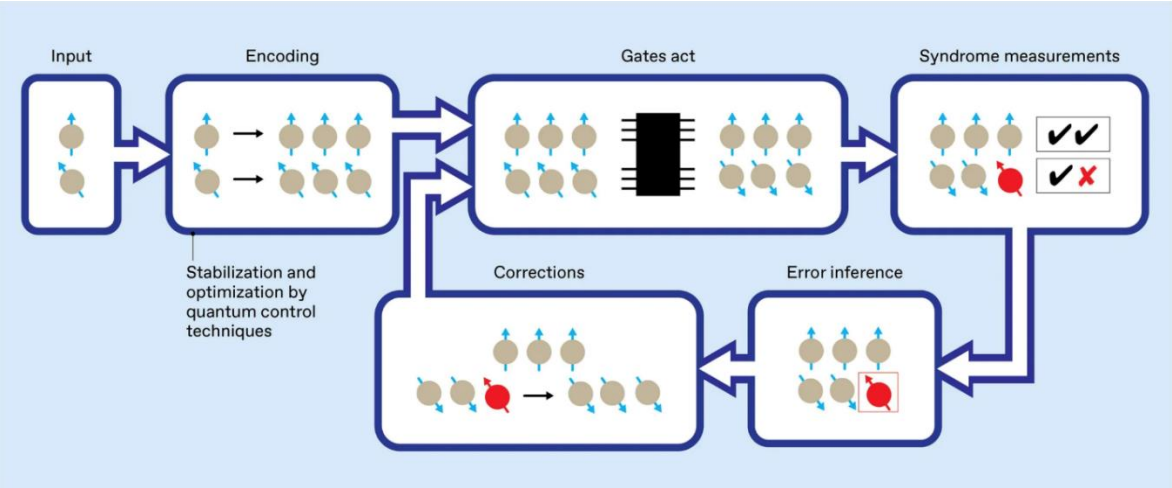

Figure 3.4: Showcasing the process of encoding and decoding quantum information

3.3.3.1 Quantum Low-Density Parity-Check (QLDPC) codes

A) Encoding using Quantum Low-Density Parity-Check (QLDPC) codes:

Encoding with Quantum Low-Density Parity-Check (QLDPC) codes entails creating a quantum state in a precise manner that embeds the information to be secured against errors. Here's a general process for encoding with QLDPC codes.

1. Select a QLDPC Code: Select a QLDPC code depending on required attributes such as code rate, distance, and decoding complexity. The code used is determined by the application needs as well as the quantum system's properties.

A quantum LDPC code with the parameters $[[n,k,d]]_q$ is defined by a pair of classical LDPC codes $C_x, C_z \subseteq F_q^n$ parity-check matrices H_x ,H_z such as $H_x H_z^* \subseteq F_q^n$ and d=min(d_x,d_z),

$$d_x = min_{c \in \frac{C_x}{C_z^\perp}} |C| \tag{3.1}$$

$$d_z = min_{c \in \frac{C_z}{C_x^\perp}} |C|. \tag{3.2}$$

F_q^n : This denotes the n-dimensional vector space over the finite field F_q. This means it consists of all possible n-tuples (vectors) where each entry is an element of F_q a finite field with q elements, where q is typically a prime power. A finite field is a field with a finite number of elements, and F_q is often referred to as GF(q), standing for Galois Field.

2. Define the Parity Check Matrix: Create the parity-check matrix for the chosen QLDPC code. The parity-check matrix defines the links between physical qubits and stabilizer checks in the code. It is often sparse, which means that the majority of its entries are zero.

- An [[n,k]] stabilizer code can be derived from additive code GF(4) with check matrix $S \in GF(4)^{m\times n}$, where $m = n - k$.
- Checks are the rows S_i of S.
- Correspond to the m stabilizer generators S_i , with $0 \to I, 1 \to X, \omega \to Z, \overline{\omega} \to Y$.
- S is self-orthagonal w.r.t trace inner product (commutativity of the stabilizer group S).

$$\langle S, S\rangle = 0 \Leftrightarrow \langle S_i, S_j\rangle = \sum_{x=1}^{n}\langle S_{ix}, S_{jx}\rangle = 0 \quad (3.3)$$
$$\forall i, j \in \{1,2, \dots, m\}.$$

3. Prepare the Initial Quantum State: Starting with a set of physical qubits initialized in a known state, such as |0⟩. The amount of physical qubits required is determined by the size of the QLDPC code and the number of logical qubits to encode.

4. Encoding Gates: Apply a set of quantum gates to the physical qubits in accordance with the QLDPC code's encoding protocol. This process usually entails entangling the qubits in certain patterns determined by the parity-check matrix.

5. Introduce Redundancy: The encoding gates add redundancy to the quantum state, allowing for mistake detection and repair during future decoding. The exact patterns of entanglement produced by the encoding gates dictate how errors are discovered and fixed.

6. Final Encoded State: The physical qubits' final state represents the quantum information encoded and safeguarded by the QLDPC code. This encoded state is prepared for subsequent quantum operations or transmission over a quantum channel.

7. Verify the Encoding: Optionally, check to ensure that the encoded state meets the stabilizer conditions given by the QLDPC code. These tests guarantee that the encoding process was completed correctly and that the encoded state is valid for error repair.

8. Store or Transmit: Once encoded, quantum information can be stored in a quantum memory or sent over a quantum communication channel. The QLDPC code's redundancy enables for error correction during storage or transmission, ensuring that the encoded information remains intact.

Overall, encoding using QLDPC codes entails establishing a quantum state in a precise manner that takes advantage of the code's features to prevent errors. The encoding technique and QLDPC code used are determined by the quantum application's specific requirements as well as the quantum hardware's characteristics.

B) Decoding Quantum Low-Density Parity-Check (QLDPC) codes:

The Quantum Belief Propagation (BP) decoder is a decoding technique used in quantum error correction to decode particular kinds of quantum error-correcting codes, such as Quantum Low-Density Parity-Check (QLDPC). It is modeled after usual belief propagation algorithms used in error correction and inference applications. Here's an explanation of how the Quantum BP decoder operates:

1. Message Passing: The Quantum BP decoder runs by passing messages between the quantum error-correcting code's qubits. These messages indicate the probabilities or likelihoods associated with various error patterns on qubits.

2. Graph Representation: Quantum error-correcting codes, such as QLDPC codes, can be seen as graphs with qubits as nodes and stabilizer checks as edges. The Quantum BP decoder works with this graphical representation of the code.

3. Initialization: The decoding procedure starts with initializing the messages that are sent between qubits and stabilizers. These initial messages usually represent uniform distributions or prior probability for various error patterns.

4.Message Update: The decoder iteratively adapts the messages sent between qubits and stabilizers based on the incoming messages and code structure. Each iteration entails modifying the messages based on specific rules or algorithms.

5. Belief Update: As messages are updated, the decoder calculates the beliefs or probabilities associated with various error patterns on the qubits. These beliefs are used to determine the most likely error configuration during quantum computation or communication.

6. Convergence: The decoding process continues for a set number of repetitions, or until a convergence condition is reached. Convergence usually happens when the messages steady, indicating that the decoder has obtained a reliable estimate of the error pattern.

7. Error Correction: After the decoding process is finished, the decoder uses the estimated error pattern to apply error correction operations to the encoded quantum state. These operations seek to erase the impact of errors while recovering the original quantum information. The Quantum BP decoder is notable for its scalability and efficiency, especially when decoding large-scale quantum error-correcting codes. However, it does not always guarantee optimal or near-optimal performance, and its effectiveness is determined by the quantum error-correcting code's unique properties and the quantum system's noise model. Researchers are still investigating and developing superior decoding algorithms for quantum error correction in order to address these problems and increase the reliability of quantum computing.

3.3.3.2 CSS codes:

A) Encoding using CSS codes:

CSS (Calderbank-Shor-Steane) encoding is creating a quantum state in such a way that the information is redundantly encoded to prevent errors. CSS codes are a sort of stabilizer code that originated from classical linear codes and is commonly utilized in quantum error correction. Here's a general approach for encoding with CSS codes.

1. Select a CSS Code: Choose a CSS code based on the desired parameters, such as code rate, distance, and decoding complexity. CSS codes consist of two traditional linear codes: one for X-errors and one for Z-errors. The codes used are determined by the application needs as well as the quantum system's features. Let $C_1 = [n, k_1]$ and $C_2 = [n, k_2]$ be two linear codes in a way that $C_2 \subset C_1$ and C_1 and $C_2^{\perp}$both correct t errors.

2. Prepare the Initial Quantum State: Begin with a set of physical qubits started in a known state, such as $|0\rangle$. The amount of physical qubits required is determined by the size of the CSS code as well as the number of logical qubits to encode. To generate a $[[n, k_1 - k_2]]$ CSS code of C_1 over C_2, which can correct t qubits, use the encoding method below. Let x represent any codeword that belongs to C_1, $x \in C_1$. So the quantum state $|x + C_2\rangle$ equals to:

$$|x + C_2 > \equiv \frac{1}{\sqrt{|C_2|}}, \sum_{y \in C_2} |x + y\rangle \tag{3.4}$$

If $|x + C_2\rangle$ in Equation (3.4) represents the initial encoded state, the erroneous state is:

$$\frac{1}{\sqrt{|C_2|}} \sum_{y \in C_2} (-1)^{(x+y).e_2} \ |x + y + e_1\rangle \tag{3.5}$$

e_1: bit flip error.

e_2: phase flip error.

3. Use X-Encoding Gates: Apply a series of quantum gates to the physical qubits based on the X-error correcting code. These gates entangle the qubits in patterns similar to the classical X-error correcting code. To detect bit flips, an ancilla state initialized at $|0\rangle$ can be used to hold the error syndrome. Using the parity-check matrix H_1 for code C_1 allows us to effectively determine the state. $|x + y + e_1\rangle_a|0\rangle$ to $|x + y + e_1\rangle|H_1(x + y + e_1)\rangle_a$

$$\frac{1}{\sqrt{|C_2|}} \sum_{y \in C_2} (-1)^{(x+y).e_2} \ |x + y + e_1\rangle|H_1 e_1\rangle \tag{3.6}$$

Since $x + y \in C_1$, the ancilla state $|H_1 e_1\rangle$ only contains the error syndrome e_1. After measuring the ancilla state, the error e_1 can be estimated using the error syndrome $|H_1 e_1\rangle$ The recovery process is carried solely by using Pauli X-gates to the qubits at any place in the error e_1 where a bit flip happened.

Following the removal of all bit flip errors, the recovered state is:

$$\frac{1}{\sqrt{|C_2|}} \sum_{y \in C_2} (-1)^{(x+y).e_2} \ |x + y\rangle \tag{3.7}$$

4. Use Z-Encoding Gates: Apply another set of quantum gates to the physical qubits based on the Z-error correcting code. These gates entangle the qubits in ways similar to the classical Z-error correcting code. To identify a phase error, each qubit in the state (3.7) is converted using Hadamard gates and transferred to the state:

$$\frac{1}{\sqrt{|C_2|2^n}} \sum_z \sum_{y \in C_2} (-1)^{(x+y).(z+e_2)} \ |z\rangle \tag{3.8}$$

where the sum represents all possible values for N bits z. If $z' = z + e2$, the state is corresponds to:

$$\frac{1}{\sqrt{|C_2|2^n}} \sum_z \sum_{y \in C_2} (-1)^{(x+y).z'} \ |z' + e_2\rangle \tag{3.9}$$

If $z' \in C_2^{\perp}$then

$$\sum_{y \in C_2} (-1)^{y.z'} = |C_2| \tag{3.10}$$

Whereas $z' \notin C_2^{\perp}$, implies

$$\sum_{y \in C_2} (-1)^{y.z'} = 0 \tag{3.11}$$

Which means that the transformed state in Equation (3.9) can take the form

$$\frac{1}{\sqrt{\frac{2^n}{|C_2|}}} \sum_{z' \in C_2^{\perp}} (-1)^{(x).z'} \; |z' + e_1\rangle \tag{3.12}$$

This is comparable to the detection of a bit flip error. Using ancilla states and the parity-check matrix H_2 for code $C_2{}^{\perp}$, error syndrome H_2e_2 can be identified and correcte the phase flip error by using Pauli Z-gates. The regained states are

$$\frac{1}{\sqrt{\frac{2^n}{|C_2|}}} \sum_{z' \in C_2^{\perp}} (-1)^{(x).z'} \; |z'\rangle \tag{3.13}$$

To complete error correction, each qubit undergoes another Hadamard transformation again.

5. Introduce Redundancy: The X- and Z-encoding gates add redundancy to the quantum state, allowing for mistake detection and correction during future decoding. The exact patterns of entanglement produced by these gates dictate how faults are discovered and repaired.

6. Final Encoded State: The final state of the physical qubits corresponds to the encoded quantum information that is secured by CSS code. This encoded state is prepared for subsequent quantum operations or communication over a quantum channel.

7. Verify Encoding: Optionally, execute checks to ensure that the encoded state meets the stabilizing requirements provided in the CSS code. These tests guarantee that the encoding process was completed correctly and that the encoded state is valid for error repair.

8. Store or Communicate: Once encoded, quantum information can be stored in a quantum memory or sent over a quantum communication channel. The CSS code's redundancy enables

error correction during storage or transmission, protecting the integrity of the encoded information.

Encoding with CSS codes entails constructing a quantum state in a certain manner that takes advantage of the capabilities of classical error correcting codes to protect against errors. The encoding process and CSS code used are determined by the quantum application's specific requirements as well as the quantum hardware's peculiarities.

B) decoding stabilizer CSS codes:

The pseudo-inverse is important in decoding quantum error-correcting codes, particularly those based on stabilizer formalisms such as CSS (Calderbank-Shor-Steane) . The pseudo-inverse is a mathematical process used in decoding algorithms to efficiently retrieve quantum information from noisy syndromes obtained by erroneous measurements. In the stabilizer formalism, a syndrome is a collection of measurement results that indicate the presence of mistakes in the encoded quantum states.
Here's how the pseudo-inverse works in decoding:
1. Syndrome Measurement: After completing error syndrome measurements on the encoded quantum state, the resulting syndrome is determined. This syndrome describes the faults that happened during quantum computation or transmission.
2. Error Correction: The purpose of decoding is to use syndrome information to identify and repair the most likely error that occurred. The pseudo-inverse operation is employed on the syndrome to return it to the most likely error operator(s) that created it.
3. Error Recovery: After identifying the error operator(s) corresponding to the syndrome using the pseudo-inverse, error recovery operations are performed on the encoded quantum state to fix the mistakes. These operations usually involve applying Pauli operators or other unitary transformations on the quantum state in order to reverse the impact of the errors.
4. Information Extraction: Following error correction, quantum information can be extracted from the corrected state and used for additional processing or measurements. The decoded state should correspond to return to the original quantum state before mistakes were introduced.
The pseudo-inverse operation is especially valuable because it enables efficient decoding of quantum error-correcting codes by mapping syndromes back to error operators without requiring exhaustive search or sophisticated optimization procedures. It allows for quick error correction and recovery, making it an essential component of many decoding algorithms used in quantum error correction.
It is important to note that the implementation and application of the pseudo-inverse may differ

depending on the decoding method and quantum error-correcting code employed. Furthermore, pseudo-inverse operations can be tailored and optimized for specific code families or error models in order to increase decoding performance in practical quantum computing systems.

3.3.3.3 Surface codes :

A) Encoding information using the surface codes:

Surface code encoding entails constructing a bigger 2D grid of physical qubits in a specified manner that embeds the error-protected information. Surface codes are a sort of topological quantum error-correcting code noted for their fault tolerance. The following are the general steps for encoding quantum information using surface codes [97].

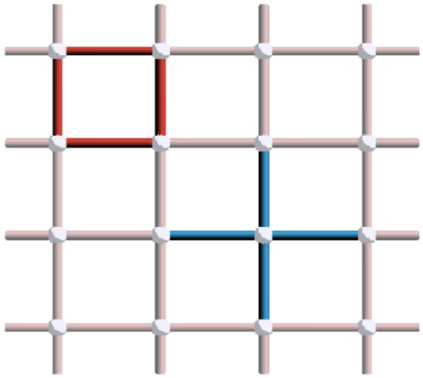

Figure 3.5: Surface code defined on a grid of size 4x4 qubit

1. Select a Surface Code: Select a suitable surface code for encoding. Surface codes are distinguished by their lattice structure, which typically comprises of a 2D grid of physical qubits organized in a square or hexagonal arrangement. The choice of surface code is determined by code distance, computational overhead, and implementation complexity [97].
2. Set up the Qubit Grid: Create a 2D grid of physical qubits to mimic the surface code lattice. Each qubit in the grid represents a physical qubit in the quantum hardware. the **vertex stabilizers** defined on every vertex of the lattice as a cross of four Z operators and the **plaquette stabilizers**, defined on every face as a square of four X Operators Examples of vertex and plaquette stabilizers are shown below, where *red* means X and *blue* mean Z [97][98].
3. Apply Initialization: Set all physical qubits in the grid to a known state, typically $|0\rangle$. This guarantees that the encoded quantum information begins in a well-defined starting state.
4. Use Hadamard Gates: Apply Hadamard gates to specified qubits in the grid using the

surface code's encoding technique. Hadamard gates are placed according to the surface code's lattice structure and encoding rules [98].

5. Measure Ancilla Qubits (Optional): Surface codes frequently employ ancilla qubits for syndrome assessments. If ancilla qubits are present, measure them using the surface code's measurement procedure. The results of these measurements will be used to identify errors in the encoded quantum information [98].

6. Perform Error Detection: Use the results of ancilla qubit measurements to identify errors in the encoded quantum information. Errors are manifested as differences between expected and measured syndrome values [98].

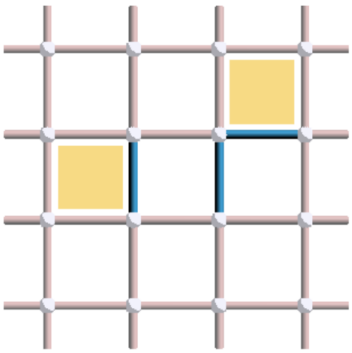

Figure 3.6: showcasing Z errors

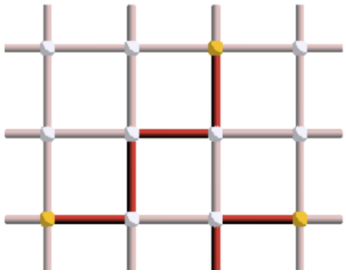

Figure 3.7: Showcasing X errors

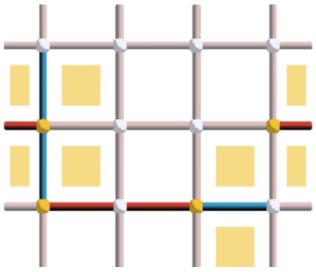

Figure 3.8: Showcasing X and Z errors.

7. Correct Errors (Optional): If errors are discovered during error detection, do the necessary rectification operations on the qubits in the grid. Correction operations often entail applying Pauli operators (X, Y, and Z) to specific qubits based on the identified syndrome [98].

8. Finalize encoding: After error detection and correction, the encoded quantum data is

available for further processing or transmission. The surface code protects against errors and ensures the integrity of the encoded quantum information [98].

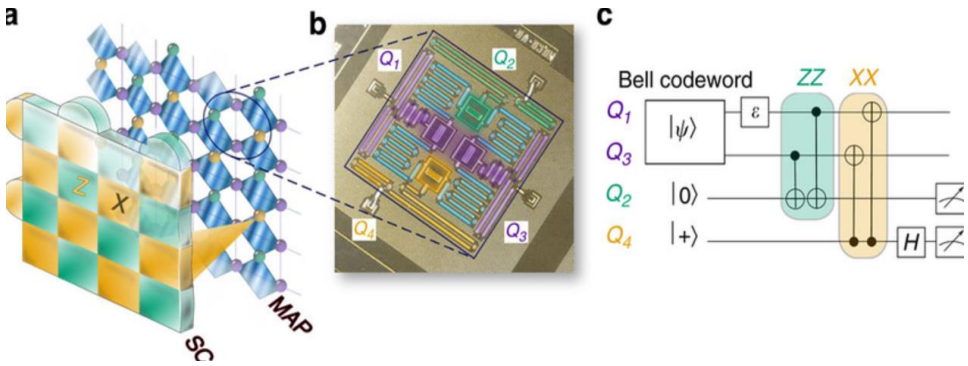

Figure 3.9: Surface code implementation and error detection quantum circuit

3.3.3.4 Color Codes:

Color codes are another type of topological quantum error-correcting code, like the Surface Code, that encode quantum information in two-dimensional lattice structures. The encoding method for color codes entails building a lattice of physical qubits and performing certain operations to create the encoded state of the logical qubits.

A) Encoding and decoding information using color codes :

- To encode information using quantum color codes, the first step is to define a lattice of qubits on a two-dimensional surface, such as a square or hexagonal grid. The qubits are then arranged in a specific pattern, with some qubits serving as data qubits and others serving as ancilla qubits. The data qubits are used to store the quantum information, while the ancilla qubits are used to detect and correct errors.

- Next, a set of stabilizer measurements is defined for the lattice of qubits. These stabilizer measurements are used to detect errors in the quantum state of the qubits. The stabilizer measurements are chosen such that they commute with each other, meaning that they can be measured simultaneously without disturbing the quantum state of the system.

- Once the stabilizer measurements are defined, the quantum information is encoded into the lattice of qubits by initializing the data qubits in a specific quantum state. The state of the data qubits is then manipulated using quantum gates, which are applied in a way that preserves the stabilizer measurements.

- Finally, the quantum information is decoded by measuring the stabilizer measurements and using the results to correct any errors that have occurred. The decoding process

involves identifying the most likely error pattern based on the measurement results and applying a correction operation to the quantum state of the data qubits.

3.4 Characteristics of Quantum Error Codes

3.4.1 Code Distance

A means to measure the "goodness" of a code is the code distance. The code distance is simply the minimum length of the error vector required to turn one code word into another. A code is said to detect t errors and correct up to t errors if and only if for every code word the ball of radius t about that code word (i.e. the set of all words within distance t of that code word) contains no other code word. This concept is easy to visualize when considering classical codes. For example, the repetition code $0 \rightarrow 000$ $1 \rightarrow 111$ has code distance 3. It can then detect any single error and correct any single error since there is only one code word within distance 1 of each code word. A code is said to t-detect and t-correct if the minimum Hamming distance between the code word and all other code words is at least 2t+1. Code distance has a direct effect on the noise threshold for fault-tolerant quantum computation.

3.4.2 Code Rate

Code rate and code distance are the two primary parameters of error-correcting codes. The code rate is defined as the ratio k/n of the number of input symbols k to the code word length n. A quantum error-correcting code encodes k qubits in a block of n qubits. The ratio k/n is the information-theoretic rate of the code. The code rate is a good measure of the efficiency of the code. An error-correcting code allows one to transmit information in a noisy environment with an arbitrarily low probability of error at a rate which is the capacity of the channel for that noise level. If the noise level is small, it may not be efficient to use an error-correcting code if the rate is significantly less than the capacity of the channel, since a higher rate code can be used with a higher code distance at the cost of only slightly higher probability of error.

3.4.3 Fault-Tolerance Threshold

A code can be used to transmit and store information, so there is an energy cost in either encoding or decoding a qubit, and that energy cost may depend on the type of error that occurs. The fault-tolerance threshold for a code is the highest error rate per physical qubit for the depolarizing (i.e., when any error is equally likely) noise model at which the logical qubits

encoded are protected from errors, provided that the energy cost of implementing the error correction code is less than the cost of using the same qubits to simulate the code again and correct the errors. This can be taken as a definition for fault-tolerant quantum computation, and it follows closely the threshold theorem for fault-tolerant classical computation under a certain noise model. If we fix the error rate at a value below the fault-tolerance threshold, and a code has a lower energy cost for implementing the error correction, then it is a better code in the sense that it consumes fewer physical qubits to achieve the same level of reliability for the logical information against errors. It is conjectured that a large-scale quantum computer can be built using a lattice of physical qubits with nearest-neighbor interactions. For this physical system, we have a specific error model, and a code is better if it has a higher fault-tolerance threshold against the specific error model [99][100][101].

3.5 Practical Implementation of Quantum Error Codes

The intrinsic fragility of quantum information presents a daunting obstacle to academics and technologists seeking to exploit the capabilities of quantum computing. Unlike classical bits, which are resilient and stable, quantum bits, or qubits, are extremely vulnerable to errors induced by environmental noise, hardware flaws, and fundamental quantum physics principles. As quantum systems grow to accommodate more qubits and more complicated computations, the requirement for error correction becomes more pressing.

3.5.1 Development of Quantum Error Codes

Quantum error correction is a dynamic and rapidly changing field. Researchers are still looking into new architectures, such as quantum low-density parity-check (QLDPC) codes and surface codes, to reach higher error thresholds and more efficient correction processes. These advancements are critical for moving quantum technology toward fault-tolerant quantum computing, in which systems may work reliably over long periods of time, paving the way for practical quantum applications in areas such as cryptography, optimization, and large-scale simulations. The ongoing research in QEC focuses not only on error correction but also on integrating these codes with future quantum hardware, matching error correction capability with quantum systems' technological limits [102][103].

3.5.2 Limitations and Challenges in Practical Implementation

3.5.2.1 Challenges in Practical Implementation

Decoherence is the most pressing obstacle faced when developing a quantum computer or when it comes to quantum communication, and correspondingly correcting errors is most difficult when information is most fragile.

A) Decoherence and Quantum Noise

When a quantum information is encoded on a quantum memory, it will suffer decoherence through its interaction with the environment. This interaction will introduce errors into the information. Environment-induced errors can be caused by a number of effects. If the error rates in the underlying physical information storage are too high, fault-tolerant error correction procedures will not be possible unless an unrealistically high degree of redundancy is used. For a physical system to be reliably correctable with only a modest degree of redundancy, the error rates must be several orders of magnitude lower than the threshold for the code being used. High threshold error correction codes exist, for example, the Surface Code introduced. In general, environment-induced errors will be a complex mix of error types both within and between different physical systems. This will mean that characterization and correction of errors in a quantum information processing device will be an extremely challenging task [102][103][106].

B) Complexity of Error Correction Algorithms

The construction needs to be fault tolerant before any logical information is encoded and that the error correction capability should not be reliant on the perfect nature of the code. Also, the error correction must be able to disentangle from the environment to a sufficient degree. This is important because while the quantum system is being protected from errors due to unwanted interactions with its surroundings, it is likely that the error correction itself will become corrupted. When this happens, we would like to apply error correction to the error correction. These properties are what it means to be self-correcting, and they characterize the failure of classical codes. Unfortunately, except in some cases of very small degeneracy, not much is known about creating self-correcting quantum codes. The CSS codes are not all self-correcting .

1. **Fault-Tolerant Architectures**

Designing fault-tolerant quantum computing architectures that smoothly integrate error correction is critical for creating scalable and dependable quantum computing systems. Creating fault-tolerant systems that can efficiently handle faults while preserving computational performance is a continuous research challenge [99][100].

2. **Hardware Constraints**

Quantum error correction protocols often require a large number of qubits for encoding quantum information redundantly, increasing the hardware resources needed. However, current quantum hardware technologies face limitations in qubit coherence times, gate fidelities, and qubit connectivity, making it challenging to implement error correction codes efficiently [101].

- **Qubit Coherence Times:**

Qubits are the fundamental building pieces of quantum computing, much like classical bits in classical computing. One important metric influencing qubit performance is coherence time, which determines how long a qubit can hold its quantum state before decohering. Longer coherence periods are preferable for errors correction because they provide a wider window of opportunity to detect and rectify problems. However, coherence times are constrained by a variety of variables, including external noise, interactions with surrounding qubits, and flaws in control algorithms [105][106].

- **Gate Fidelities:**

Quantum gates are operations used on qubits to execute quantum calculations. High gate fidelities are required for successfully implementing quantum error correcting codes because faults in gate operations might propagate and accumulate throughout the calculation. Gate fidelities refer to the precision with which quantum gates carry out their intended functions [104][107]. Imperfections in gate processes, such as gate faults and leakage, can drastically reduce the performance of error correction methods.

- **Qubit Connectivity:**

The connection of qubits in a quantum processor is critical for efficiently implementing error correcting codes. To successfully detect and fix errors, many error correction systems require precise qubit connection topologies. However, contemporary quantum hardware technologies frequently have limits in qubit connectivity due to physical constraints such as qubit layout on a chip and the quantum processor's networking architecture. Limited qubit connection may limit the sorts of error correction codes that may be implemented, necessitating the use of extra qubits for routing and communication purposes [106][107].

- **Resource Overhead:**

Implementing error correction codes generally results in a large resource overhead, such as additional qubits, gates, and computational resources. The overhead involved with error correction might worsen current hardware limits, limiting quantum computing systems' scalability. Balancing the trade-offs between error correction performance and resource utilization is critical for practical implementation, particularly on resource-limited quantum computing platforms.

3.6 Conclusion

As we go toward more powerful quantum computing systems, effective quantum error correction becomes increasingly important. Throughout this chapter, we've looked at why quantum error correction is important, emphasizing its role in preserving the integrity of quantum information in the face of noise, decoherence, and other types of operational errors. We have explored the theoretical foundations that drive quantum error-correcting codes, as well as how these codes are designed to retain quantum superposition and entanglement while reducing the effects of errors.

The study of many types of quantum error codes, ranging from classical error codes to complex quantum error-correcting codes, reveals a wide range of techniques to solve the quantum error problem. The comparison of different codes revealed their individual strengths and weaknesses, allowing us to better grasp the landscape of error correction in quantum computing.

Practical implementation constitutes its own set of obstacles, as research in this area moves quickly. While quantum error correction codes provide potential answers, implementing them in real-world quantum systems necessitates overcoming technical challenges such as scalability, resource needs, and the quantum no-cloning theorem. Addressing these restrictions is critical to creating fault-tolerant quantum computing.

In final analysis, this chapter emphasizes the continuous importance of quantum error correction in the quest for dependable quantum systems. It is a dynamic discipline that requires ongoing innovation to overcome theoretical and practical issues. The findings presented here establish the framework for future research into enhanced quantum error-correcting codes and the creation of scalable, robust quantum structures. Finally, these initiatives will be critical in enabling the widespread adoption of quantum technology and their use to solve complicated issues.

Chapter 4

Simulation of Low Density Parity Check Codes and Steane Codes

This chapter delves into the world of Python coding, examining the unique characteristics of Steane codes and low-density parity check (LDPC) codes. LDPC codes have excellent error-correction capabilities that are essential for current communication systems. They are renowned for their sparse parity-check matrices. In the meanwhile, a key component of quantum computing is the seven-qubit quantum error-correcting code known as the Steane code. We address the encoding, decoding, and performance analysis of these codes using Python implementations, providing readers with a thorough grasp of the importance of LDPC and Steane codes in the fields of classical and quantum information processing.

4.1 Simulation tools

4.1.1 Visual Studio Code:

Visual Studio, a powerful integrated development environment (IDE) developed by Microsoft, provides a wide range of features and tools to improve the coding experience for developers across multiple areas. It has powerful code visualization capabilities, allowing users to generate dynamic diagrams that depict the calling structure of source code in a class diagram-like form while supporting numerous programming languages. Visual Studio also includes powerful debugging tools, such as a visual watch window, which allows users to examine data structures throughout the debugging process and provides detailed visualizations of monitored variables. Furthermore, Visual Studio's modeling capabilities assist users in understanding existing code, describing their applications, and validating high-level system designs using dependency diagrams.

The IDE also supports data science work, with tools such as Jupyter Notebooks and the Interactive Window for data analysis and visualization, IntelliSense support, machine learning model construction and deployment, and a variety of Python data science extensions. Visual Studio's broad range of features and tools enables developers and data scientists to work more efficiently, analyze data effectively, and visualize code structures with ease. supports a wide range of programming languages, including C, C++, C#, F#, Visual Basic, JavaScript, TypeScript, Python, Ruby, and many more, and provides language-specific services that allow the code editor and debugger to interact seamlessly with these languages. The IDE also includes powerful debugging and diagnostics capabilities, such as IntelliTrace, which records program state to aid in issue detection, and Code Map Debugger Integration, which displays the call stack during debugging. Visual Studio also includes a suite of built-in testing tools, such as Live Unit Testing for continuous testing, IntelliTest for generating unit tests, Code Coverage for measuring test coverage, and the Microsoft Fakes framework for enabling unit test isolation, providing developers with a comprehensive set of tools to ensure code quality and reliability.

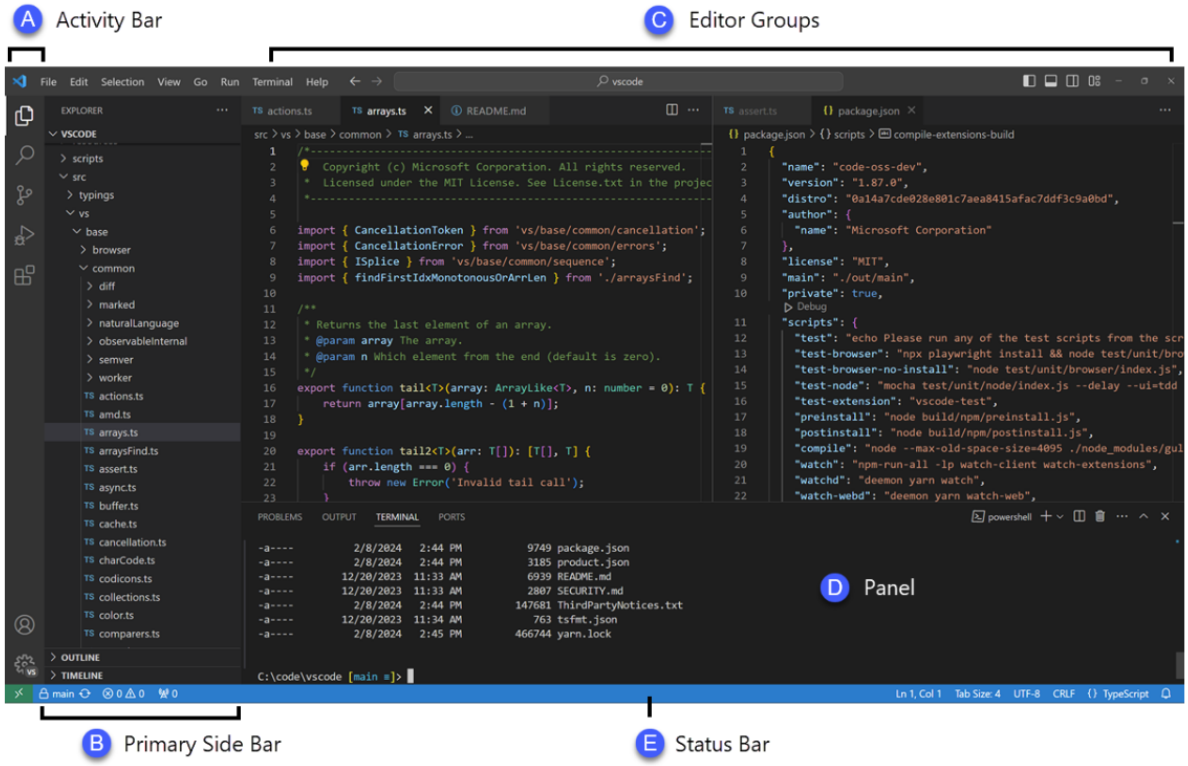

Figure 4.1: Visual Studio Code Interface

4.1.2 Used Language: Python

High-level and general-purpose, Python is renowned for its ease of use, readability, and adaptability. Python is a programming language that was created by Guido van Rossum in the late 1980s as an alternative to ABC. It places a strong emphasis on code readability through extensive indentation. It is a versatile language suitable for a wide range of applications since it supports several programming paradigms, such as object-oriented, structured, and functional programming. Python's large standard library, dynamic typing, and dynamic binding make it a popular choice for scripting and fast application development. The language's straightforward syntax, modules, and packages encourage code reuse and software modularity, which boosts output and lowers maintenance expenses. For all major platforms, Python's standard library and interpreter are freely available, which makes distribution easier and fosters a vibrant development community. The quick edit-test-debug cycle of the language, Because of the language's quick edit-test-debug cycle, debugging easiness, and introspective power, programmers who want to be efficient and productive in software development often choose it.

4.1.2.1 NumPy Library

Large, multi-dimensional arrays and matrices are supported by NumPy, the core package for scientific computing in Python. It is a robust library that offers a wide range of high-level mathematical functions to effectively work with big arrays. It allows users to accomplish things like import data from different file formats, generate random numbers, apply sophisticated mathematical and logical operations, create and manipulate arrays, and integrate C/C++ and Fortran code. NumPy is a vital tool for Python scientific computing, data analysis, and machine learning due to its effective implementation of multidimensional arrays and vectorized operations and it contains the following classes:

- **numpy.ndarray:** a multidimensional, homogeneous array represented by NumPy's fundamental data structure.
- **Numeric types:** NumPy provides a wide range of numeric data types, including `int8`, `int16`, `int32`, `int64`, `uint8`, `uint16`, `uint32`, `uint64`, `float16`, `float32`, `float64`, `complex64`, and `complex128`.
- **numpy.generic:** The foundational class of all scalar NumPy types.
- **numpy.number:** The foundation class for all scalar numerical data types, encompassing both floating-point and integer values.
- **numpy.integer:** The foundational class of all scalar integer types.

- **numpy.signedinteger:** The base class of all scalar types that are signed integers.
- **numpy.unsignedinteger:** The root class of all scalar types that are unsigned integers.
- **numpy.floating:** The foundational class of all scalar floating-point types.
- **numpy.complexfloating:** The fundamental class of all scalar types with complex floating points.
- **numpy.flexible:** The base class of all other scalar types, including void, object, bytes, and str.

4.1.2.2 Matplotlib Library

Matplotlib is a popular Python library that may be used to produce interactive, animated, and static visualizations. Matplotlib, created by John D. Hunter, provides an extensive collection of tools for creating figures suitable for publication in a range of settings and formats. Matplotlib is primarily focused on graph plotting, and it connects with NumPy in a smooth manner to make it easy for users to plot data arrays. Because of its adaptability, which enables modification and plot aesthetic refinement, it is a preferred option among scientific and technical communities looking for superior visual data displays. Matplotlib is a popular library for Python data visualization jobs because of its vast ecosystem of add-on toolkits and platform freedom, despite its verbose syntax and initial learning curve.

Compared to NumPy, Matplotlib does not have the same understanding of classes. Rather, Matplotlib is arranged around these essential elements:

- **Figure:** The top-level plot element container. Subplots, axes, titles, legends, and other creative components are all held together by it.
- **Axes:** The most fundamental and adaptable subplot creation unit. Plots can be placed anywhere in the figure using axes. There might be more than one axis in a given figure[5]. Ticks and labels for a single coordinate system are managed by the axis. The x- and y-axes are represented by the two Axis objects that are commonly found on axes[5].
- **Artist:** The base class of every element in a figure that is visible. The Artist class is the root of several other classes, including axes, text, lines, and figures.
- **Pyplot:** A state-based Matplotlib interface that offers a plotting interface akin to MATLAB. It upholds a set of active axes and the global state.
- **Widgets:** To create interactive applications, Matplotlib offers a collection of GUI-neutral widgets such as text boxes, sliders, and buttons.

4.1.2.3 Networkx Library

A flexible Python program called NetworkX is made for building, examining, and displaying intricate networks. NetworkX, which focuses on graph theory, provides an extensive collection of network analysis methods as well as a rich set of data structures for expressing different kinds of networks, such as graphs, digraphs, and multigraphs. Because of its adaptability, nodes can be represented by any hashable object, opening up a wide range of applications, including the modeling of biological networks and social network analysis. NetworkX is an important tool for academics, data scientists, and network analysts since it allows the creation, reading, and writing of graphs in many formats. With its wide library of graph algorithms and ability to manage big-scale networks, NetworkX is a well-liked option for delving into and comprehending the complicated dynamics and structures of complex networks in python. This library offers a number of classes for displaying various graph kinds. Among these classes are:

- **Graph:** The fundamental class for undirected graphs; it supports key-value attribute pairs for each undirected edge and accepts any hashable object as a node.
- **DiGraph:** A directed graph's base class; think of it as Graph with directed edges instead.
- **MultiGraph:** A class of undirected graphs that permits more than one edge—possibly with distinct attributes—to connect any two nodes.
- **MultiDiGraph:** A kind of directed graphs that permits more than one edge—possibly with distinct attributes—to exist between any two nodes.

4.1.2.4 Pyldpc Library

The 'pyldpc' library is a Python package that works with Low-Density Parity-Check (LDPC) codes. The library offers a number of methods for encoding messages, simulating noisy channels, creating LDPC codes with parameter customization, and decoding received messages via iterative belief propagation techniques. `pyldpc is a useful tool for researchers, engineers, and developers working on improving data transmission reliability in various domains, such as satellite communications, wireless networks, and storage systems. It provides a minimal example implementation that makes it easy for users to incorporate LDPC coding and decoding into their Python applications. This library offers a number of features for interacting with codes that are Low-Density Parity-Checked (LDPC). Among the principal functions of Pyldpc are:

- **make_ldpc:** The LDPC coding and decoding matrices H and G are created by this function. To generate the matrices, it requires parameters like n, d_v, d_c, systematic, sparse, and seed.
- **encode:** This method uses the LDPC code that has been created to encode a message.
- **decode:** Using the LDPC code and a number of decoding techniques, including the belief propagation method, this function decodes a message that has been received.
- **get_message:** After decoding with the LDPC code, this function is used to get the original message from the received and decoded message. By reversing the encoding process, it improves in recovering the original data conveyed across the noisy channel.

4.1.2.5 Qiskit Library:

Qiskit is an open-source software development kit (SDK) for interacting with quantum computers at the circuit, pulse, and algorithm levels. It includes tools for developing and manipulating quantum programs, as well as running them on prototype quantum devices on the IBM Quantum Platform or local computer simulators.

Qiskit's primary version is composed in Python. It is based on the circuit model for universal quantum computation and may be utilized with any quantum hardware that adheres to this paradigm (currently superconducting qubits and trapped ions).It consists of various components that work together to enable quantum computing.

- **Qiskit Terra:** Qiskit's foundation, which provides tools for creating quantum circuits at or near the level of quantum machine code.
- **Qiskit Aer:** A high-performance simulator for quantum circuits with noise models.
- **Qiskit Ignis:** offers tools for calibrating and characterizing quantum devices.
- **Qiskit Aqua:** provides algorithms and applications for quantum computing.
- **Qiskit Nature:** provides tools for employing quantum simulations to solve problems in physics, chemistry, material science, and biology.
- **Qiskit Machine Learning:** provides tools for developing quantum machine learning algorithms.

4.2 Analyzing the performance of LDPC codes

The provided code exemplifies the utilization of the pyldpc library to implement an efficient Low-Density Parity-Check (LDPC) code system. With parameters set at n = 15 for the codeword length, d_v = 4 for the variable node degree, and d_c = 5 for the check node degree,

a systematic and sparse parity-check matrix H is constructed using make_ldpc. By generating a random message v of length k and calculating the code rate CR, the code proceeds to encode the message with encode and introduces Additive White Gaussian Noise (AWGN) to the codeword based on specified SNR levels. The decoding process involves utilizing both hard decision decoding and the Message Passing Algorithm (MPA) decode function to recover the original message v from the noisy codeword y_noisy. Visualizations include comparisons of original and recovered messages for various SNR values, the Tanner graph representation of the parity-check matrix H using networkx, and plots illustrating BER, coding gain, and code rate variations across the SNR range. This comprehensive analysis showcases the robustness and effectiveness of the LDPC code system under different decoding methodologies, providing insights into its performance and reliability in noisy communication environments.

4.2.1 Hard Decision Decoding

The main focus of this code is the hard decision decoding technique, which is implemented with the decode function from pyldpc and the default method='hard' parameter. Hard decision decoding compares the received signal to a specified threshold to decide whether the sent bit is 0 or 1. The decoder then chooses the codeword with the shortest Hamming distance among all possible codewords, attempting to identify the most likely transmitted message.

The recovered message d is received using the hard decision decoding process, while the original message v is recovered via the get_message function. The bit error rate (BER) is calculated using the bit_error_rate function, and the coding gain is derived by comparing the BER of the coded system to the uncoded.

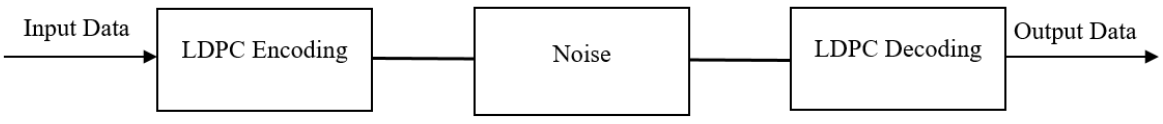

Figure 4.2: LDPC codes transmission chain

4.2.1.1 Result interpretation:

A) Original Message VS Recovered Message:

Regarding the input and the output data, it is observable that at lower SNR values, such as 5.0 and 10.0, the recovered messages differ significantly from the original message's triangular shape. At an SNR of 3.0, severe distortions are observed, suggesting a high level of

noise interference, but at an SNR of 10.0, the recovered message improves. At higher SNR values, such as 20.0 and 30.0, the recovered messages virtually exactly mimic the original triangular shape. At an SNR of 20.0, the recovered message is highly accurate, with little to no departure from the original, and at an SNR of 30.0, the recovery is practically faultless, with the restored message indistinguishable from the original.

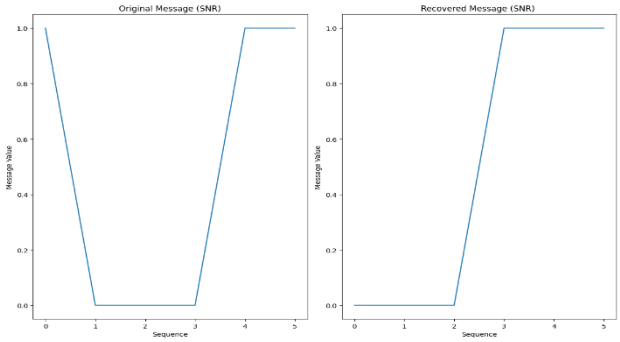

Figure 4.4: The recovered message VS the original message at SNR=0

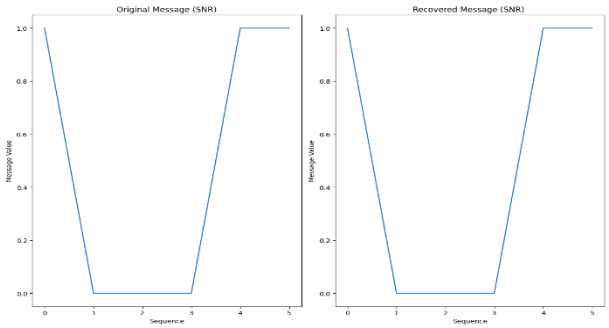

Figure 4.3: The recovered message VS the original message at SNR=10

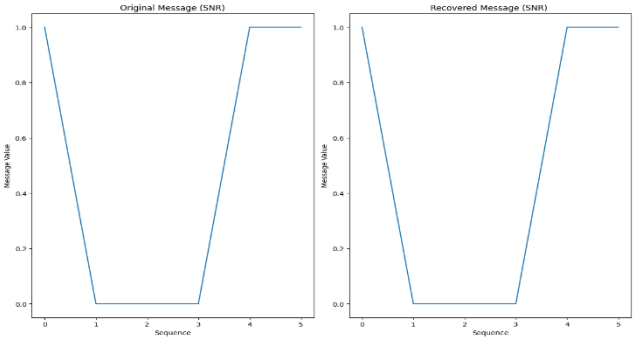

Figure 4.6: The recovered message VS the original message at SNR=20

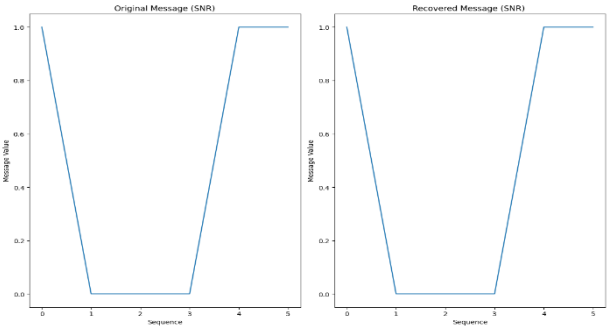

Figure 4.5: The recovered message VS the original message at SNR=30

B) Code Rate:

This conservative strategy offers reliable communication under a variety of situations while sacrificing efficiency when SNR is high. In bad channel circumstances (low SNR), a low code rate (high redundancy) is required to guarantee data integrity. However, if the code rate remains constant at 0.2 even when SNR improves, it means that the system is not dynamically altering the code rate based on SNR. This problem could be caused by algorithmic or system configuration issues that prohibit the code rate from adjusting as anticipated. Possible causes include design limits, software problems, and hardware limitations.

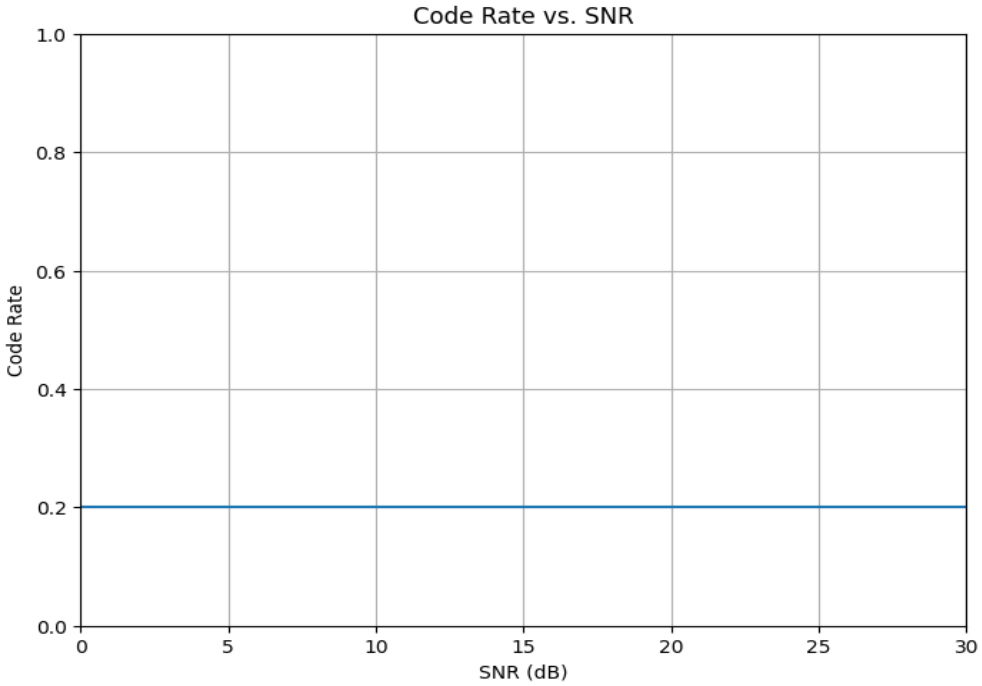

Figure 4.7: The variation of the code rate in function of SNR

C) Coding gain:

Observing the figure below, it is noticeable that with an SNR of -30 dB, the signal strength is 30 decibels lower than the noise level. This describes a situation in which noise overwhelms the signal, making it difficult for error-correcting codes to efficiently distinguish between the transmitted signal and the noise. As a result, the coding gain at such low SNR levels may be small. As the SNR approaches 0 dB, the signal becomes stronger in relation to the noise. This means that the transmitted signal stands out more against the noise, allowing error-correcting codes to more effectively rectify faults introduced during transmission. As the SNR rises, the coding gain increases, resulting in improved error detection and correction performance.

The decrease in coding gain from -30 dB to 0 dB represents a major increase in the communication system's reliability and accuracy. With increased SNR, the frequency of mistakes in the received signal lowers, resulting in a higher probability of correct decoding by the receiver. This improvement in performance can lead to greater service quality, higher data rates, better coverage, and a more enjoyable user experience in communication systems.

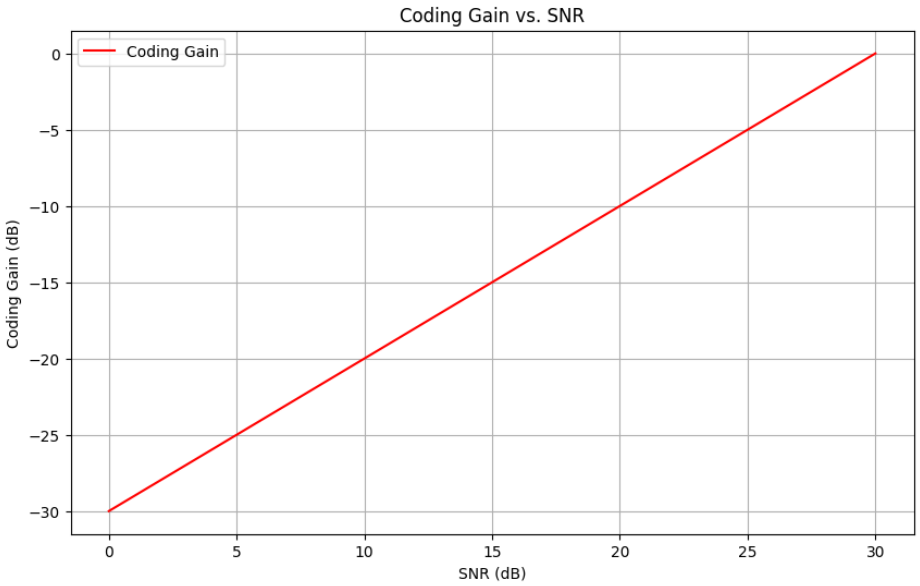

Figure 4.8: The variation of the coding gain in function of the SNR.

D) Tanner graph

Interpreting a Tanner graph offers information on an LDPC code's error-correcting capabilities and structural properties, which are critical for assessing its performance in real-world communication systems. The Tanner graph visually depicts the relationships between variable nodes (representing bits) and check nodes (representing parity-check equations), as well as how these equations connect the bits. Analyzing the graph reveals the code's sparsity, which indicates the level of redundancy and the possibilities for mistake correction. The graph also exposes the code's interconnection, demonstrating how each bit contributes to several parity-check equations, and vice versa. Understanding these structural qualities aids in determining the code's ability to detect and rectify problems effectively.

Furthermore, the Tanner graph helps to optimize LDPC codes by providing insights into their design and decoding algorithms, allowing communication engineers to adjust the codes to specific channel conditions and performance needs. Finally, this understanding contributes to the construction of strong and dependable communication systems capable of effectively conveying data over noisy channels.

The check nodes

The variable nodes

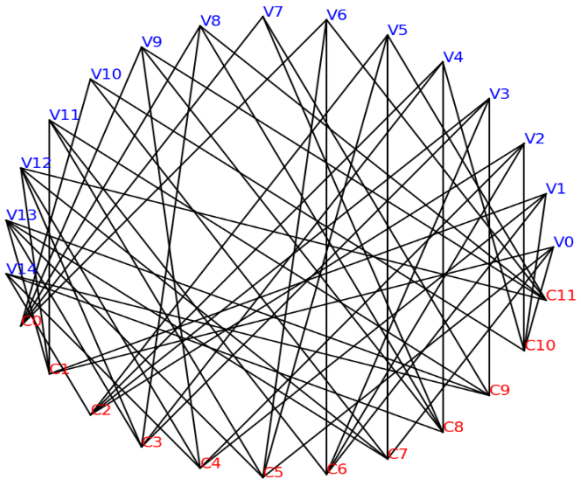

Figure 4.9: Representational Tanner graph of LDPC code.

E) Bit Error Rate:

The graph shows the link between Bit Error Rate (BER) and Signal-to-Noise Ratio (SNR) in a digital communication system. Initially, in the high noise zone on the left side of the graph, the system operates in a noisy environment with poor SNR, resulting in a high BER of 0.85 and unstable data transfer. This describes instances in which the received signal is weak in comparison to the background noise. As SNR grows from left to right, the system reaches a transition phase in which the BER considerably decreases, indicating enhanced data reliability. Engineers optimize system parameters to achieve this balance of SNR and BER.

Beyond the transition region, the curve flattens out, indicating that increasing SNR has a diminishing effect on lowering the BER. At this point, the system is already functioning well, and further SNR gains may not be cost-effective due to practical constraints such as cost and power usage.

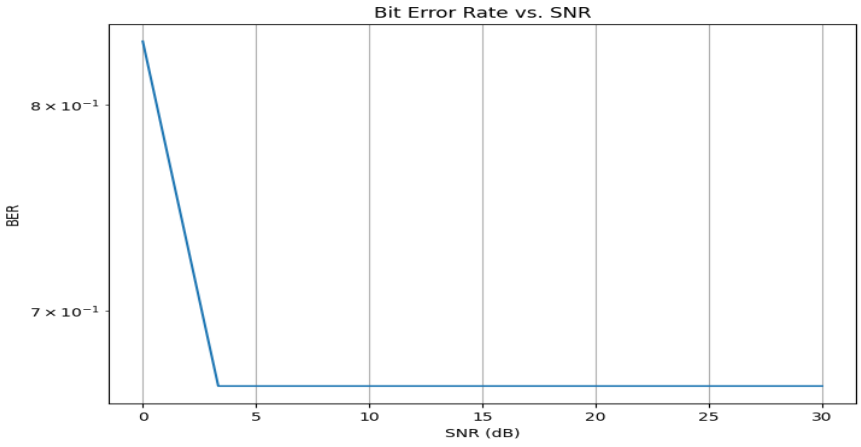

Figure 4.10: The variation of BER in function of SNR when using Hard Decision Decoding

4.2.2 Message Passing Decoding

The primary focus of this code rotates to message passing decoding, which is a basic approach in decoding LDPC codes. This is accomplished by the message passing technique, notably the belief propagation method, which iteratively exchanges messages between variable nodes and check nodes until convergence. Unlike hard decision decoding, message passing decoding is based on probabilities rather than hard decisions, allowing for a more nuanced assessment of the transmitted message.

4.2.2.1 Result Interpretation:

A) Recovered Message VS Original Message

When the Signal-to-Noise Ratio (SNR) is low, the received signal is dominated by noise, making decoding and recovery of the original message a challenging task. This scenario is akin to trying to hear a faint voice in a noisy room - the noise overwhelms the signal, making it difficult to discern the intended message.

As the SNR increases, the likelihood of correctly recovering the original message improves significantly. With a higher SNR, the signal stands out more clearly against the noise, allowing the decoding algorithm to become more effective in extracting the intended information. This is similar to turning up the volume on a radio to hear a distant station more clearly - the increased signal strength relative to the noise makes the desired transmission more intelligible.

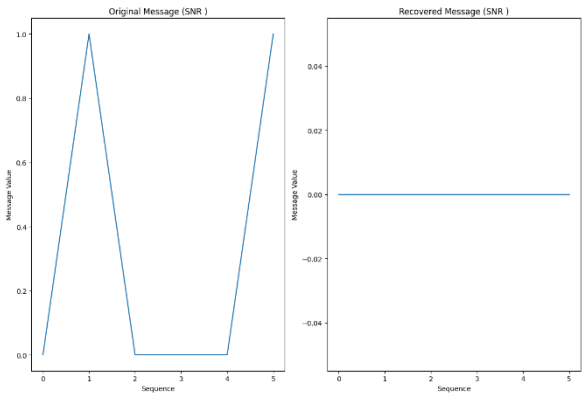

Figure 4.11: The recovered message VS the original message using message passing decoding at SNR=0.

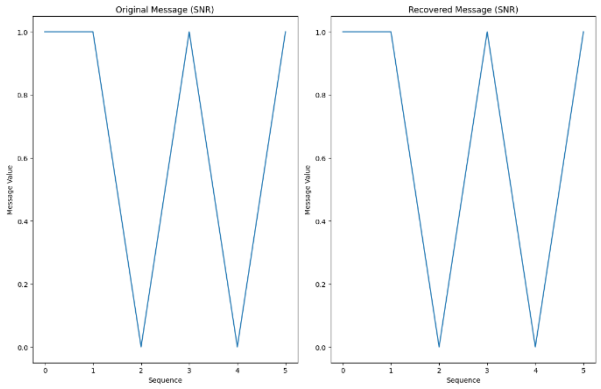

Figure 4.12: The recovered message VS the original message using message passing decoding at SNR=10.

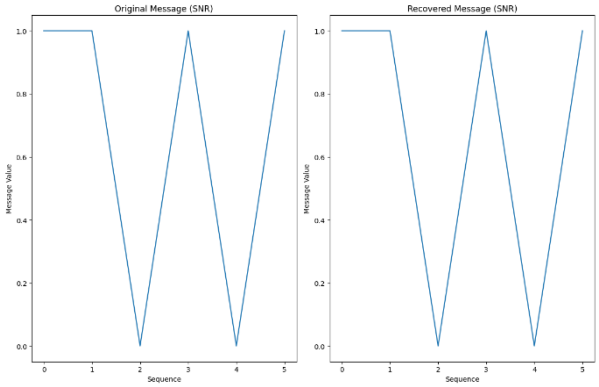

Figure 4.13: The recovered message VS the original message using message passing decoding at SNR=20.

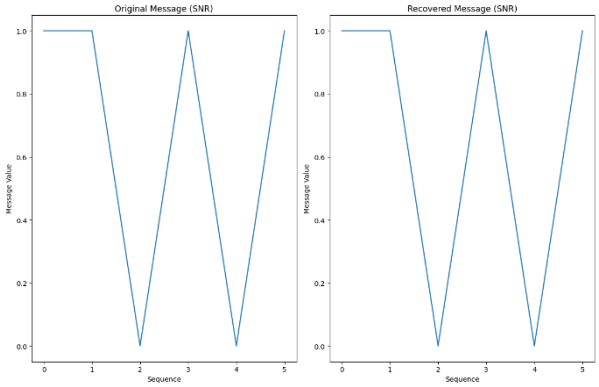

Figure 4.14: The recovered message VS the original message using message passing decoding at SNR=30.

B) Code Rate:

The graph shows a horizontal line that follows a constant trajectory across the whole range of Signal-to-Noise Ratio (SNR) values. This observation implies that SNR fluctuations have little effect on the coding rate. The coding rate looks to be around 0.20, or 20%, over the SNR range shown. This fixed code rate, which is unaffected by SNR fluctuations, simplifies overall system design and maintains a consistent degree of data transmission efficiency. By keeping a steady code rate, the system may run effectively without requiring dynamic modifications in response to changing SNR conditions. This feature is important because it delivers predictable and consistent data throughput, allowing for optimal system planning and resource utilization.

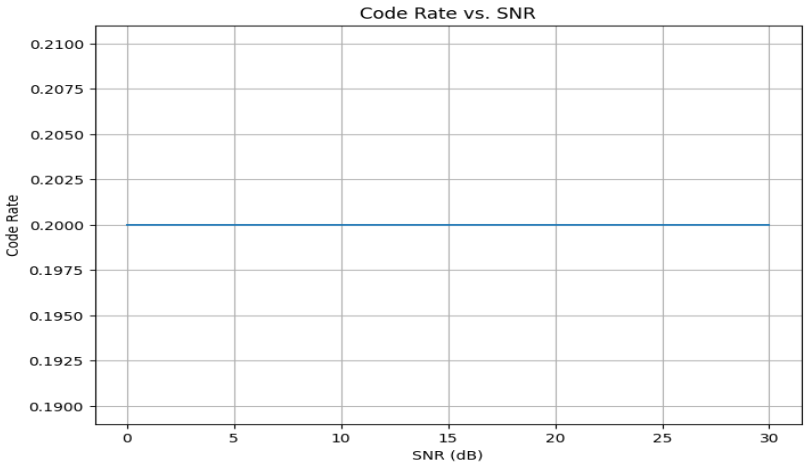

Figure 4.15: The variation of code rate in function of SNR when using Message Passing decoding

C) Coding Gain:

The graph shows a noticeable trend in coding gain in terms of signal-to-noise ratio (SNR). Initially, as SNR increases from 0 dB, the coding gain decreases dramatically. Around an SNR of 5 dB, the coding gain drops to around -25 dB. As SNR increases, the coding gain rises rapidly. As SNR approaches 30 dB, the coding gain gradually levels off around 10 dB. This pattern indicates that at low SNR levels, where noise is dominant, the coding gain is modest due to considerable noise interference. However, as SNR rises, the coding gain increases, showing that the system is more resilient to noise. Beyond a certain SNR threshold additional coding improvements become minimal, implying declining returns as SNR increases

.

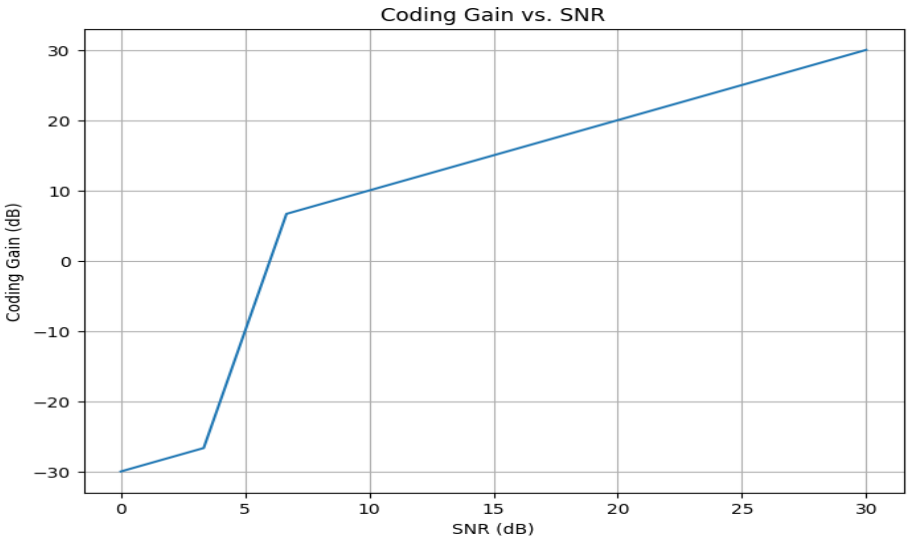

Figure 4.16: The variation of the coding gain in function of SNR when using Message Passing decoding.

D) Bit Error Rate:

The graph illustrates the behavior of Bit Error Rate (BER) as Signal-to-Noise Ratio (SNR) varies. Initially, as SNR increases from left to right on the graph, the BER experiences a rapid decline, indicating improved data reliability with higher SNR levels. However, as SNR continues to increase, the curve flattens out, signifying diminishing returns where additional SNR increments have a reduced impact on reducing the BER. The optimal operating point is where the curve starts to flatten, representing a balance between achieving a low BER and maintaining a reasonable SNR level. In the low SNR region on the left side of the graph, the system faces challenges due to high noise levels, resulting in a high BER and unreliable data transmission. In the transition region, as SNR rises, the system becomes more resilient against noise, leading to a significant drop in BER. Beyond this transition, in the high SNR region,

further SNR enhancements result in marginal BER reductions. Engineers strive to operate near the optimal point to strike a balance between SNR and BER, considering practical constraints such as cost and power limitations that may restrict operating at very high SNR levels.

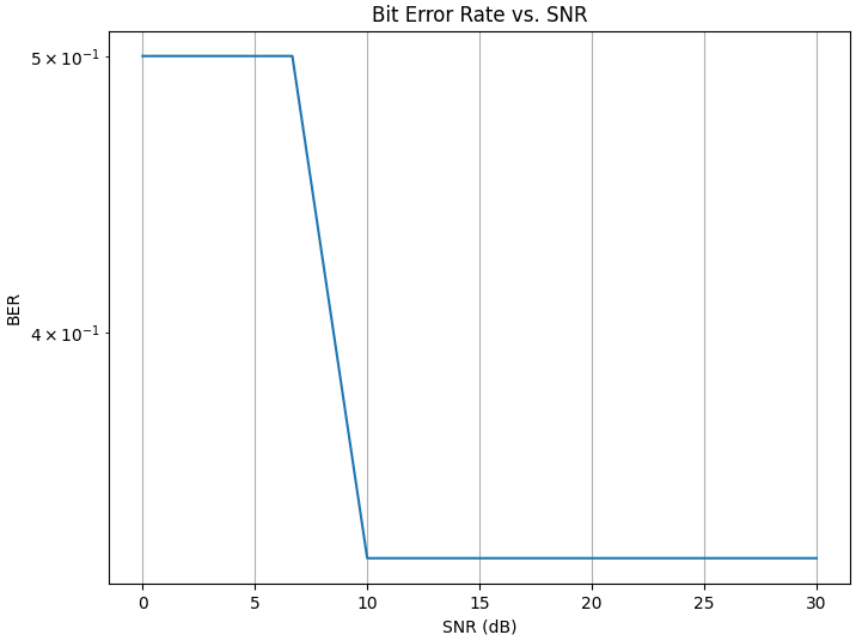

Figure 4.17: The variation of BER in function of SNR when using Message Passing decoding

4.2.3 Comparison between Hard Decision Decoding and Message Passing Decoding

According to the data gathered from the previous section, a comparison of hard decision decoding versus message passing decoding of LDPC codes indicates considerable variations in error correcting capabilities and dependability. Hard decision decoding, as shown in the results, has a higher Bit Error Rate (BER) than message forwarding decoding. Hard choice decoding requires making binary decisions on received bits, which increases error rates, especially in noisy situations with low signal-to-noise ratios (SNR). In contrast, Message Passing decoding, which is based on belief propagation and uses soft information such as probabilities or log-likelihood ratios, provides higher decoding performance with lower error rates, particularly in noisy environments. Message passing decoding, achieved through methods like the Sum-Product Algorithm (SPA), shows improved BER performance and reliability in data transfer.

4.3 Analyzing the performance of Steane codes

In this section, we explore the implementation and simulation of the Steane code using Python. The Steane code is a type of quantum error-correcting code that is particularly well-suited for correcting single-qubit errors. By simulating the Steane code, we can gain insights into its error-correcting capabilities and performance in quantum computation.

The Steane code is a [[7,1,3]] code, meaning it encodes one logical qubit into seven physical qubits and can correct arbitrary single-qubit errors. A popular quantum error-correcting code that belongs to the Calderbank-Shor-Steane (CSS) family of codes which there exists a subfamily of CSS codes that are also QLDPC codes. The Steane Code It is constructed using classical binary codes that are derived from the classical [7,4,3] Hamming code.

4.3.1 Simulation Part 1:

In this section, the simulation is performed using Python, leveraging libraries such as Qiskit for quantum computing and NumPy for numerical computations.

The implementation of the Steane code simulation is broken down into several key components: the definition of stabilizer generators, encoding circuits, error application, and error correction.

4.3.1.1 Stabilizer Formalism:

The Steane code utilizes the stabilizer formalism, where the code space is defined as a set of commuting stabilizer generators. These generators are typically expressed in terms of Pauli matrices: S0, S1, and $S2$ represent the stabilizer generators of a quantum error-correcting code. Each stabilizer generator S_i is associated with a set of qubits on which it acts.

S0 = [3, 4, 5, 6]: This means that the first stabilizer generator S0 acts on qubits 3, 4, 5, and 6.

S1 = [1, 2, 5, 6]: This means that the second stabilizer generator S1 acts on qubits 1, 2, 5, and 6.

S2 = [0, 2, 4, 6]: This means that the third stabilizer generator S2 acts on qubits 0, 2, 4, and 6.

In practice, the stabilizer generators are represented by specific Pauli operators (X, Y, Z, or combinations thereof) acting on the qubits listed. For example, if S0 were a generator of the form $Z_3Z_4Z_5Z_6$, it would mean that S0 applies a Z (Pauli-Z) operation on qubits 3, 4, 5, and 6.

4.3.1.2 Encoding Process:

Encoding a qubit using the Steane code involves applying a series of quantum gates to transform the state of the logical qubit into the state of seven physical qubits. The encoding process ensures that any single-qubit error can be detected and corrected. We construct the encoding circuit for the Steane code by applying the necessary quantum gates to transform the logical qubit into the encoded state, in this case using the Hadamard gate alongside with the CNOT gate.

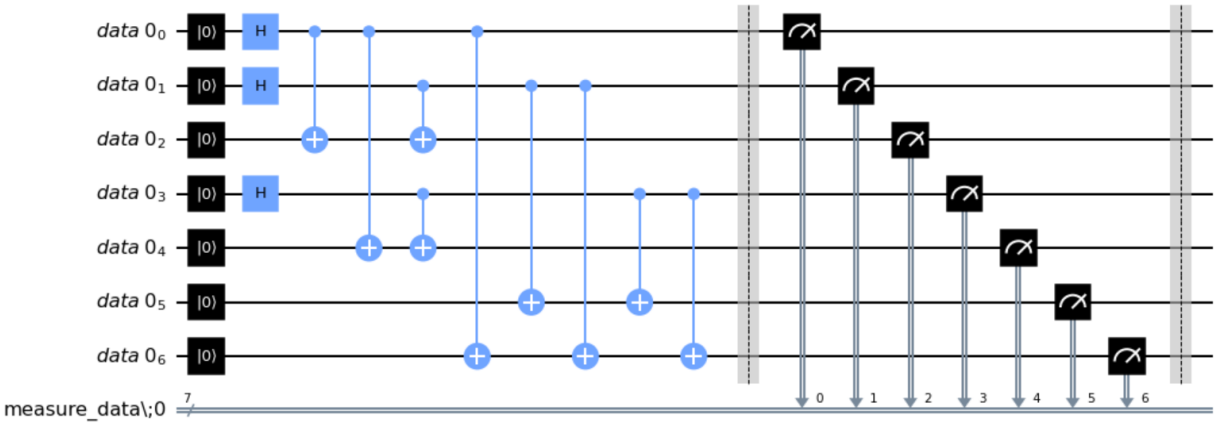

Figure 4.18: Showcasing the steane code encoding process.

A) Error Application

To simulate the error correction capabilities of the Steane code, we apply random single-qubit errors to the encoded state. In this case to qubit 7.

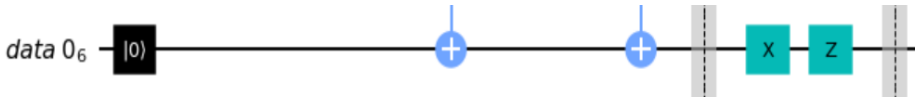

Figure 4.19: Applying random single-qubit errors to the encoded state

B) Error Correction

The error correction procedure involves measuring the syndromes and applying the appropriate corrections based on the measurement outcomes, by adding three X (measuring the bit flip errors) ancillas and three Z ancillas (measuring the phase flip errors).

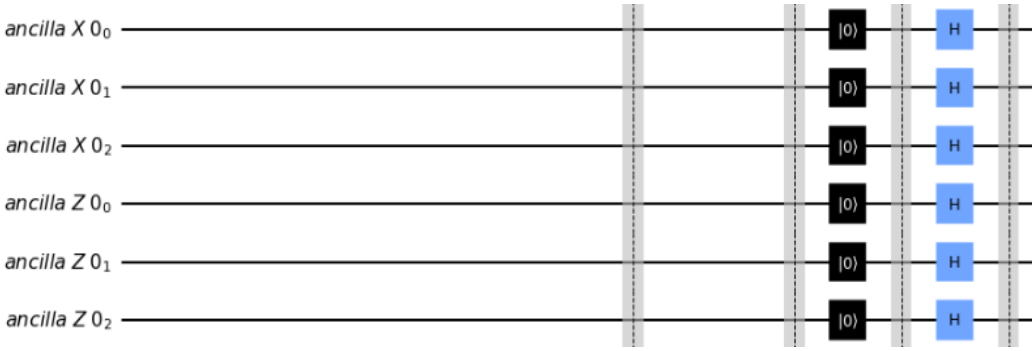

Figure 4.20: Adding six ancillas for the syndromes measurement and error correction

4.3.1.3 Simulation Results

We evaluate the performance of the Steane code by running the simulation for various error rates and measuring the fidelity of the recovered logical qubit.

A) Error Rate Analysis

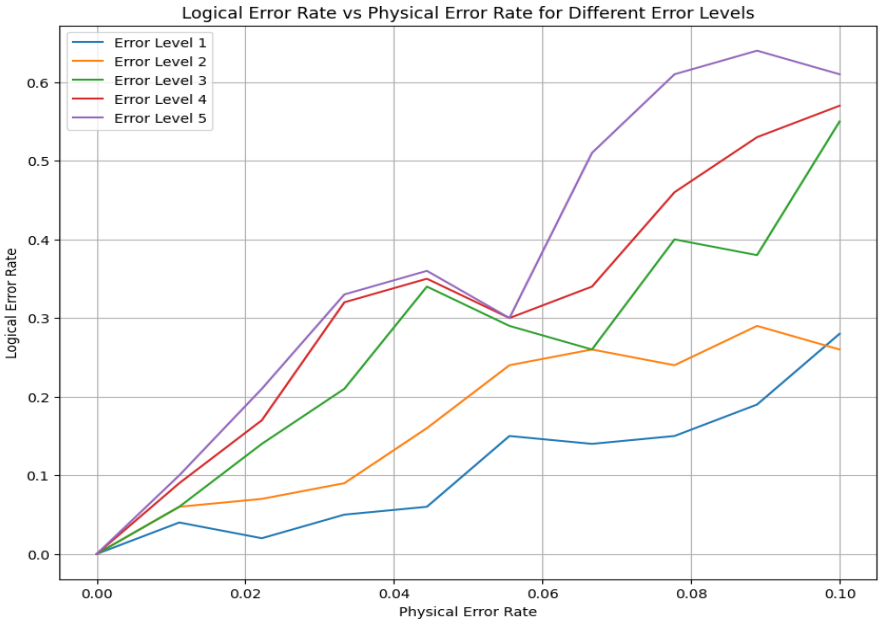

Figure 4.21: Physical error vs Logical error rate for different error levels for the Steane code

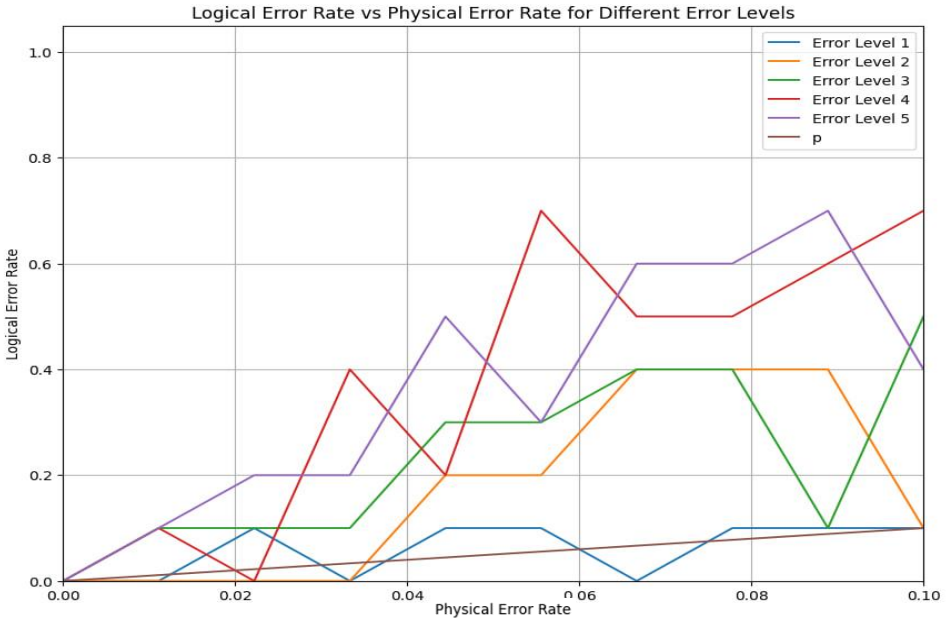

Figure 4.22: Physical error vs Logical error rate for different error levels for the Steane code using 'p' as reference

1. **Analysis of the Graph:**

- **Axes:**

- X-axis (p): The physical error probability, which is the probability that a physical qubit undergoes an error.
- Y-axis (Logical p): The logical error probability, which is the probability that the logical qubit (after error correction) undergoes an error.

- **Lines and Legend:**

- Level 1, Level 2, Level 3, Level 4, Level 5: These lines represent the error probabilities for different levels of error correction.
- p: This line (brown) shows the physical error probability as a reference where logical error probability equals the physical error probability without any error correction.

2. **Key Observations**

- **Performance of Error Correction Levels:**

- Level 1 shows the lowest logical error rate, indicating the most effective error correction at lower physical error probabilities.
- Level 5 shows the highest logical error rate, indicating the least effective error correction at higher physical error probabilities.

- For lower physical error probabilities, all levels perform better than the reference line p, but as physical error probability increases, higher levels (Level 5) show an increasing logical error rate more sharply compared to lower levels (Level 1).

- **Threshold Behavior:**

- At lower physical error probabilities, all levels are effective in reducing logical error rates.
- At higher physical error probabilities, the logical error rate for higher levels increases faster than for lower levels. This indicates that higher levels are less robust to increasing physical error rates.

This demonstrates that simpler error correction schemes (Level 1) might be more robust against higher physical error rates compared to more complex schemes (Level 5).

4.3.2 Simulation Part 2:

In this section we test another Steane Code by changing the placement of the H (Hadamard gate) and the CNOT gate in order to see the impact of the placement of quantum gates on the encoding process and therefor the error correction process and keeping everything else the same as in stabilizer generators, error application, and error correction.

4.3.2.1 Encoding Process:

We construct the encoding circuit for this second Steane code by applying the necessary changes to the placement of quantum gates (H and CNOT) to transform the logical qubit into the encoded state. As following:

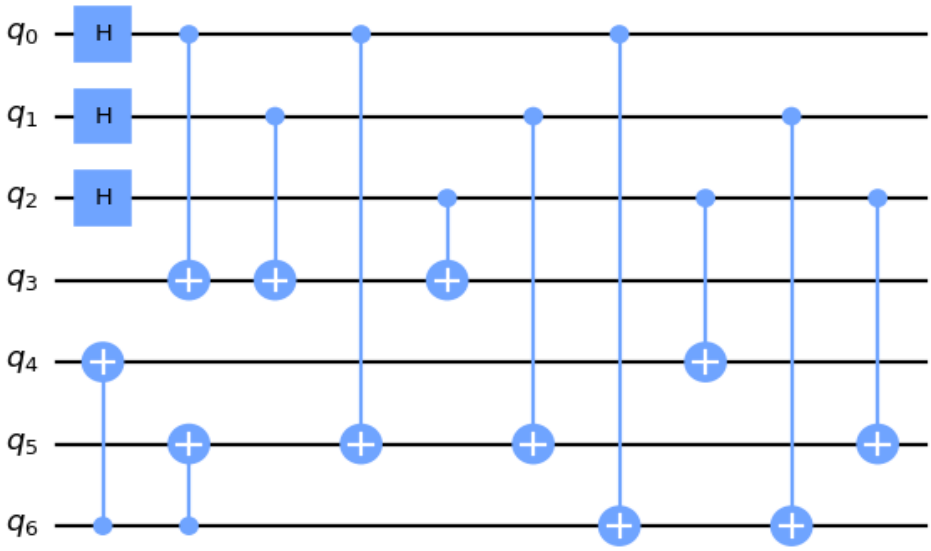

Figure 4.23: The encoding circuit using the second Steane Code

4.3.2.2 Simulation Results

As seen before we evaluate the performance of the code by running the simulation for various error rates of the recovered logical qubit.

A) Error Rate Analysis

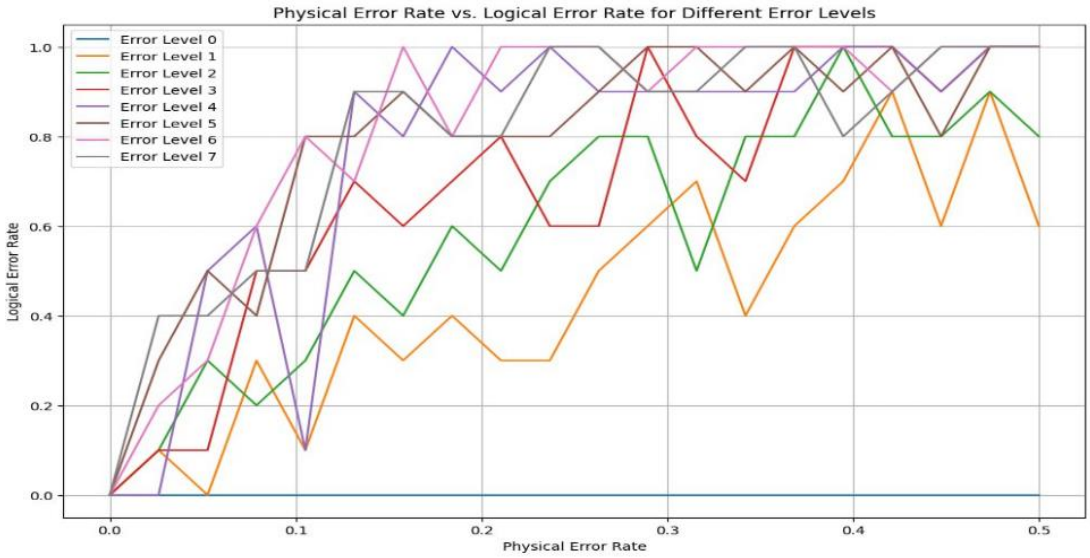

Figure 4.24: Physical error vs Logical error rate for different error levels for the second Strane code

1. **Analysis of the Graph:**

This graph represents the relationship between the physical error rate and the logical error rate for different error correction levels in a quantum error correction (QEC) code. Here's a detailed breakdown of what the graph means:

- **Axes:**
- X-axis (Physical Error Rate): This is the rate of errors occurring at the physical qubit level. It ranges from 0 to 0.5, representing the probability of an error occurring on a physical qubit during computation.
- Y-axis (Logical Error Rate): This is the rate of errors occurring at the logical qubit level after error correction has been applied. It also ranges from 0 to 1, representing the probability of an error occurring on the logical qubit.
- **Lines:**
- Each line represents the logical error rate for a specific error correction level (labeled Error Level 0 through Error Level 7) as the physical error rate varies.
- Error Level 0 (Blue Line): Represents the baseline logical error rate without any error correction applied. As expected, this line shows a linear relationship, indicating a one-to-one correspondence between physical and logical errors.
- Higher Error Levels (Other Colors): Represent the performance of different levels of error correction.
- **Interpretation:**
- Error Correction Performance: Generally, in an effective QEC scheme, higher levels should show better error correction (i.e., lower logical error rates for a given physical error rate).
- Erratic Behavior: The erratic or fluctuating lines for the higher error levels indicate instability in error correction performance.

2. **Key Observations**

- **Performance of Error Correction Levels:**
- Low Physical Error Rates: For very low physical error rates (near 0), all levels should ideally show a low logical error rate. However, fluctuations indicate some instability even at low error rates.
- High Physical Error Rates: As the physical error rate increases, the logical error rates for higher levels increase more rapidly and erratically, indicating that the QEC is becoming less effective or even counterproductive.

4.3.3 Steane Code and Quantum Circuits:

The encoding process involves a specific sequence and placement of quantum gates, including Hadamard (H) and CNOT gates. Explaining the radical shift of logical and physical error rate between figure (4.7) and figure (4.9) to explore more this section we also compare between the corrected state that the correct encoding should produce and the measured ones.

4.3.3.1 For the first Steane code figure (4.18):

Correct: | Measured:

0000000 | 0000000 === True

0111100 | 0111100 === True

1011010 | 1011010 === True

1100110 | 1100110 === True

1101001 | 1101001 === True

1010101 | 1010101 === True

0110011 | 0110011 === True

0001111 | 0001111 === True

Measured State == Logical |0> State? : True

In this case, the "Measured" states match the "Correct" states exactly. This indicates that the encoding circuit correctly prepared the logical |0⟩ state using the Steane code. The logical |0⟩ state in the Steane code is a specific superposition of the 7-qubit states, and the measurements should reflect this superposition.

4.3.3.2 For the second Steane code figure (4.23):

Correct: | Measured:

0000000 | 0000000 === True

0111100 | 1111000 === False

1011010 | 1100110 === False

1100110 | 0011110 === False

1101001 | 1010101 === False

1010101 | 0101101 === False

0110011 | 0110011 === True

0001111 | 1001011 === False

Measured State == Logical |0> State?: [True, False, False, False, False, False, True, False]

Here, the "Measured" states does not match every "Correct" state. This indicates that the circuit did not correctly prepare the logical |0⟩ state.
Which lead us to conclude that changing the placement of H and CNOT gates, the structure of the encoded state changes. The Steane code relies on specific gate sequences to create the necessary entanglement and superposition. If you place the H and CNOT gates incorrectly, the resulting state may not represent the correct logical |0⟩ state. The "Correct" states list all the states that the correct encoding should produce, which are derived from the logical |0⟩ state. If the circuit is incorrect, it may produce a different set of states, or it may not produce the expected superposition and entanglement.

4.4 Conclusion:

In this chapter, we conducted simulations of both Low-Density Parity-Check (LDPC) codes and Steane codes using Python, with a focus on evaluating their efficiency and error rate performance. LDPC codes, known for their sparse parity-check matrices, exhibited notable efficiency in resource utilization and decoding processes, particularly in scenarios involving larger block sizes and complex error patterns. On the other hand, Steane codes, characterized by their structured [[7,1,3]] framework, demonstrated effective error correction capabilities, especially against single-qubit errors. Despite their differences, both LDPC and Steane codes showcased promising error rate reduction capabilities under varying noise levels, highlighting their significance in quantum error correction research and potential applications in quantum computing.

General Conclusion

In conclusion, this master thesis has shed light on the potential of quantum communication and error correction techniques in addressing the contemporary challenges of secure and efficient data transmission. By exploring the feasibility and performance implications of deploying quantum technologies, this research has demonstrated the promise of quantum communication in meeting the growing demands of digital connectivity. Furthermore, the analysis of quantum error correction codes has underscored their crucial role in preserving the integrity of transmitted quantum information, despite the challenges posed by noise and decoherence. It is evident that both classical and quantum error correction techniques have their own strengths and limitations, and their effectiveness varies depending on the specific application and environmental factors. While classical error correction may excel in certain scenarios, quantum error correction offers unique advantages in others. Therefore, a nuanced understanding of both classical and quantum error correction is essential for designing robust and high-performance communication systems capable of meeting the diverse needs of modern society. This research sets the stage for further exploration and development in the field of quantum communication, paving the way for innovative solutions to address the evolving communication challenges of the future.

Abstract

The evolution of communication systems has been a transformative journey, with error correction playing a crucial role in ensuring reliable data transmission and storage. Classical error correction techniques have been developed to mitigate the effects of noise and errors in digital communication systems. As quantum computing advances, error correction becomes even more critical, with quantum computers being susceptible to higher error rates compared to classical computers. Quantum errors can manifest as bit flips, phase flips, or a combination of both, making them more challenging to correct. Quantum error correction (QEC) is essential for performing error-free quantum algorithms and scaling quantum computers beyond the current limitations. Despite the differences between classical and quantum error correction, it is still possible to develop quantum error-correcting codes based on classical error-correcting structures. As communication systems continue to evolve, incorporating both classical and quantum error correction techniques will be crucial for ensuring reliable data transmission and storage in the face of increasingly complex noise and error patterns.

ملخص

كان تطور أنظمة الاتصالات رحلة تحويلية، حيث لعب تصحيح الأخطاء دورًا حاسمًا في ضمان نقل البيانات وتخزينها بشكل موثوق. تم تطوير تقنيات تصحيح الأخطاء الكلاسيكية للتخفيف من آثار الضوضاء والأخطاء في أنظمة الاتصالات الرقمية. مع تقدم الحوسبة الكمومية، يصبح تصحيح الأخطاء أكثر أهمية، حيث تكون أجهزة الكمبيوتر الكمومية عرضة لمعدلات أخطاء أعلى مقارنة بأجهزة الكمبيوتر الكلاسيكية. يمكن أن تظهر الأخطاء الكمومية مع انقلاب البت أو انقلاب الطور أو مزيج من كليهما، مما يجعلها أكثر صعوبة في التصحيح. يعد تصحيح الخطأ الكمي (QEC) ضروريًا لإجراء خوارزميات كمومية خالية من الأخطاء وتوسيع نطاق أجهزة الكمبيوتر الكمومية بما يتجاوز القيود الحالية. على الرغم من الاختلافات بين تصحيح الخطأ الكلاسيكي والكمي، لا يزال من الممكن تطوير رموز تصحيح الأخطاء الكمومية بناءً على الهياكل الكلاسيكية لتصحيح الأخطاء. مع استمرار تطور أنظمة الاتصالات، سيكون دمج تقنيات تصحيح الأخطاء الكمية والكلاسيكية أمرًا بالغ الأهمية لضمان نقل البيانات وتخزينها بشكل موثوق في مواجهة أنماط الضوضاء والخطأ المعقدة بشكل متزايد.

Résumé

L’évolution des systèmes de communication a été un parcours transformateur, la correction des erreurs jouant un rôle crucial pour assurer la transmission et le stockage fiables des données. Des techniques classiques de correction des erreurs ont été développées pour atténuer les effets du bruit et des erreurs dans les systèmes de communication numériques. À mesure que l’informatique quantique progresse, la correction des erreurs devient encore plus critique, les ordinateurs quantiques étant sensibles à des taux d’erreur plus élevés que les ordinateurs

classiques. Les erreurs quantiques peuvent se manifester sous forme de retournements de bits, de phases ou d'une combinaison des deux, ce qui les rend plus difficiles à corriger. La correction d'erreur quantique (QEC) est essentielle pour exécuter des algorithmes quantiques sans erreur et faire évoluer les ordinateurs quantiques au-delà des limites actuelles. Malgré les différences entre la correction d'erreur classique et quantique, il est encore possible de développer des codes de correction d'erreur quantique basés sur des structures de correction d'erreur classiques. À mesure que les systèmes de communication continuent d'évoluer, l'intégration de techniques classiques et de correction d'erreurs quantiques sera cruciale pour assurer la transmission et le stockage fiables des données face à des modèles de bruit et d'erreurs de plus en plus complexes.

Keywords

Quantum Communication, Qubits, Entanglement, Superposition, Quantum Mechanics, Classical error correction codes, Quantum error correction codes, Low density parity check code, steane code

Mots-clés

Communication quantique, Qubits, Enchevêtrement, Superposition, Mécanique quantique, Codes de correction d'erreur classiques, Codes de correction d'erreur quantique, Code de contrôle de parité de faible densité, code stéarine

الكلمات الرئيسية

الاتصال الكمي، الكيوبتات، التشابك، التراكب، ميكانيكا الكم، رموز تصحيح الخطأ الكلاسيكية، رموز تصحيح الخطأ الكمي، رمز فحص التكافؤ المنخفض الكثافة، رمز ستاين

References

[1] «Classical Communications channels and their limitations,» 04 04 2024. [En ligne].

[2] «The limitations of Classical Sensing,» 11 04 2024. [En ligne].

[3] D. Bacco, «Towards a full fledged quantum and classical communication in optical fiber,» 11 11 2019. [En ligne].

[4] Y. Yamamato, Fundamentals of Noise processes.

[5] Robert Koenig et S. Smith, «Limits on classical communication from quantum entropy power inequalities,» May 2012. [En ligne].

[6] M.Ould-Khaoua, . L. Mackenzie et R. Sutherland, Journal of Systems Architecture, 1998.

[7] W. S. Wong et R. Brockett, «Systems with finite communication bandwidth constraints.II.Stabilization with limited information feedback,» *IEEE,* 1999.

[8] W. S. Wong et R. Brockett, «Systems with finite communication bandwidth constraints.I.State estimation problems,» *IEEE,* 1997.

[9] David.A.Weston, Electromagnetic Interference and compatibility, 2016.

[10] J. G. Proakis et M. Salehi, Digital Communication Fifth Edition, 2008.

[11] S. Haykin, Communication System Fourth Edition, 2000.

[12] T. S. Rappaport, Wireless Communications: Principles and practise, 2024.

[13] J. G. Proakis et M. Salehi, Communication Systems Engineering Second Edition, 2002.

[14] W. Stallings, Wireless Communications and networks, 2002.

[15] «An Overview of security challenges in communication networks,» 2016.

[16] A. Security, «Quantum Secure Communication through Entanglement-based networks,» 15 02 2023. [En ligne].

[17] R. B. Larson et V. Bromm, *The first stars in the Universe,* 2004.

[18] «Chronology of the Universe,» Wikipedia, 2015. [En ligne].

[19] J. Wang, G. Guo et Z. Shan, «SoK: Benchmarking the performance of a quantum computer,» *ARxiv,* 2022.

[20] J. C. Bardin, D. H. Slichter et D. J. Reilly, «Microwaves in Quantum Computing,» *IEEE,* 2021.

[21] M. Vesely, Application of Quantum Computers in Foreign Exchange Reserves Management, 2022.

[22] D. Perri, O. Gervasi, M. Simonetti et S. Tasso, High Performance Computing and Computational Intelligence Applications with Multichaos Perspective, 2022.

[23] L. Viola, E. Knill et R. Laflamme, Constructing Qubits in Physical Systems, 2001.

[24] A. Lopatnikova, M.-N. Tran et S. A. Sisson, «An Introduction to Quantum Computing for Statisticians and Data Scientists,» 2021.

[25] Z. Hu, X. Zheng, D. An, C. Zhou, Y. Yang et R. Li, «New analytic buckling solutions of side-cracked rectangular thin plates by the sympletic superposition method,» *Internation Journal of Mechanical Science,* 2021.

[26] F. Giacomini et C. Brukner, «Einstein's Equivalence principle for superpositions of gravitational fields and quantum reference frames,» *Arxiv,* 2020.

[27] K. Andrei, «Roots of quantum computing supremacy: superposition, entanglement or complementarity?,» *The European Physical Journal Special Topics,* 2021.

[28] F. Giacomini et C. Brukner, «Quantum superposition of spacetimes obeys Einstein's equivalence principle,» *AVS Quantum Sciecne,* 2022.

[29] M. Zukowski et C. Brukner, «Bell's Theorem for general N-qubit states,» *Phys.Rev.Leh,* 2002.

[30] Quera, «Bell Inequality,» [En ligne].

[31] n. Authors, «Bell's Inequality,» May 2024. [En ligne].

[32] M. Parry, «Generating entangled photons with nonlinear metasurfaces,» *Adv.Photon,* 2021.

[33] A. R. Echarri, J. D. Cox et F. G. De Abajo, «Direct generation of entangled photons with nonlinear optical waveguides,» *Nanophotonics,* 2021.

[34] A. Villar, A. Lohrmann et A. Ling , «Experimental entangled photon pair generation using crystals with parallel optical axes,» 2018.

[35] D. Tordera et S. Dutta, «How to apply CNOT on polarization qubits,» 2018. [En ligne].

[36] A. Crespi, R. Ramponi, R. Osellame, L. Sansoni, I. Bongioanni, F. Sciarrino, G. Vallone et P. Mataloni, «Integrated photonic quantum gates for polarization qubits,» *Nature Communications,* 2011.

[37] M. A. Nielsen et I. L. Chuang, Quantum Computation and Quantum Information, 2010.

[38] Wikipedia, «List of quantum logic gates,» 19 04 2024. [En ligne].

[39] I. Q. Documentation, «Quantum circuit model,» [En ligne].

[40] FasterCapital, «Quantum Gate: Opening the Quantum Gates: A Journey into Quantum Computing,» 07 04 2024. [En ligne].

[41] S. Shettigar, QUANTUM LOGIC GATES - A PRESENTATION, 2023.

[42] C. Portmann et R. Renner, «Security in quantum cryptography,» *Rev. Mod. Phys.,* 2022.

[43] T. Fernàndez-Caramés et P. Fraga-Lamas, «Towards Post-Quantum Blockchain: A Review on Blockchain Cryptography Resistant to Quantum Computing Attacks,» *IEEE Access,* 2020.

[44] J. Yin, Y.-H. Li, S.-K. Liao, M. Yang, Y. Cao, L. Zhang, J.-G. Ren, W.-Q. Cai, W.-Y. Liu, S.-L. Li, R. Shu, Y.-M. Huang, L. Deng, L. Li, Q. Zhang, N.-L. Liu, Y.-A. Chen, C.-Y. Lu, X.-B. Wang, F. Xu, J.-Y. Wang, C.-Z. Peng, A. K. Ekert et J.-W. Pan, «Entanglement-based secure quantum cryptography over 1,120 kilometres,» *Nature,* 2020.

[45] A.-B. Al-Ghamdi, A. Al-Sulami et A. O. Aljahdali, «On the security and confidentiality of quantum key distribution,» *SECURITY AND PRIVACY,* 2020.

[46] F. Basso Basset, M. Valeri, E. Roccia, V. Muredda, D. Poderini, J. Neuwirth, N. Spagnolo, M. B. Rota, G. Carvacho, F. Sciarrino et R. Trotta, «Quantum key distribution with entangled photons generated on demand by a quantum dot,» *Science Advances,* 2024.

[47] O. Amer, V. Garg et W. O. Krawec, «An Introduction to practical quantum key distribution,» 2020.

[48] L.-J. Wang, K.-Y. Zhang, J.-Y. Wang, J. Cheng, Y.-H. Yang, S.-B. Tang, D. Yan, Y.-L. Tang, Z. Liu, Y. Yu, Q. Zhang et J.-W. Pan, «Experimental authentication of quantum key distribution with post-quantum cryptography,» *npj Quantum Information,* 2021.

[49] M. Mehic, M. Niemiec, S. Rass, J. Ma, M. Peev, A. Aguado, V. Martin, S. Schauer, A. Poppe, C. Pacher et M. Voznak, «Quantum Key Distribution: A Networking Perspective,» *ACM Comput. Surv,* 2020.

[50] A. Nellis, «The quantum internet, explained,» [En ligne].

[51] M. Mastriani, «Is Instantaneous quantum internet possible?,» *hal.Science,* 2019.

[52] S. Pattanayak, Quantum Machine learning with python, 2021.

[53] N. Vidhya, V. Seethalakshmi et S. Suganyadevi, Quantum Computing: A shift from bits to qubits, 2023.

[54] Y. Lu, A. Sigov, L. Ratkin, L. A. Ivanov et M. Zuo, «Quantum computing and industrial information integration: A review,» *Journal of Industrial Information Integration,* 2023.

[55] Y. Zhou, E. M. Stoudenmire et X. Waintal, «What Limits the Simulation of Quantum Computers?,» *Phys. Rev. X,* 2020.

[56] F. Tacchino, A. Chiesa, S. Carretta et D. Gerace, «Quantum Computers as Universal Quantum Simulators: State-of-the-Art and Perspectives,» *Advanced Quantum Technologies,* 2020.

[57] S. T. Marella, H. S. Kumar Parisa et Y. Zhao, Quantum Computing and Communications, 2020.

[58] C. E. Shannon, A mathematical Theory of Communication, 1948.

[59] M. K. Saini, «Block coding in digital electronics,» 17 07 2023. [En ligne].

[60] B. Vucetic et J. Yuan, Turbo Codes: Principles and applications, 2000.

[61] Wikipedia, «Error Correction Codes,» 21 04 2024. [En ligne].

[62] N. H. I. Ibraheem, Error-Detecting and Error-Correcting using Hamming and Cyclic codes, 2009.

[63] O. Khadir, A simple coding-decoding algorithm for the Hamming Code, 2022.

[64] I. M. Farhan, «Block code: communication engineering III for electrical,» 2023.

[65] J. V. Lint, Introduction to Coding Theory Third Edition, 1999.

[66] W. C. Huffman et V. Pless, Fundamentals of error correcting codes, 2003.

[67] E. Pasalic, Coding Theory and Applications: Cyclic Codes, 2013.

[68] R. G. Gallager, Low Density Parity Check Codes, 2003.

[69] M. Rovini, Low Density Parity Check Codes: A tutorial, 2004.

[70] P. Radosavljevic, A. F. De Baynast et J. R. Cavallaro, Optimized Message Passing Schedules for LDPC Decoding, 2005.

[71] R. Johannesson et K. S. Zigangirov, Fundamentals of Convolutional Coding, 1999.

[72] J. Lieb, R. Pinto et J. Rosenthal, «Convolutional Codes,» *ArXiv,* 2020.

[73] K. T. Phelps et C. A. Rodger, Encyclopedia of Physical Science and Technology, 2003.

[74] P. Panteleev et G. Kalachev, «Quantum LDPC codes with almost linear Minimum Distance,» *ArXiv,* 2020.

[75] N. P. Breuckmann et J. N. Eberhardt, «Balanced Product Quantum Codes,» *IEEE Transactions on Information Theory,* 2021.

[76] N. P. Breuckmann et J. N. Eberhardt, «Quantum Low-Density Parity-Check Codes,» *American Physical Society,* 2021.

[77] N. Raveendran et B. Vasic, «Trapping Sets of quantum LDPC codes,» *Quantum,* 2021.

[78] I. B. Djordjevic, Quantum information processing, quantum computing, and quantum error correction: an engineering approach, 2021.

[79] S. S. Garani, P. J. Nadkarni et A. Raina, «Theory Behind Quantum Error Correcting Codes: An Overview,» *Journal of the Indian Institute of Science,* 2023.

[80] A. Grospellier, L. Grouès, A. Krishna et A. Leverrier, «Combining hard and soft decoders for hypergraph product codes,» *Quantum,* 2021.

[81] I. Dumer et N. Gharavi, «Codes approaching the Shannon limit with polynomial complexity per information bit,» *ArXiv,* 2021.

[82] I. Bocharova, B. Kudryashov, E. Ovsyannikov, V. Skachek et T. Uustalu, «Design and Analysis of NB QC-LDPC Codes over Small Alphabets,» *IEEE Transactions on Communications,* 2022.

[83] P. Charanarur, V. Gad et R. Gad, «Sensor's data transmission with BPSK using LDPC (Min-Sum) error corrections over MIMO channel: Analysis over RMSE and BER,» *Materials Today: Proceedings,* 2020.

[84] Z. Chen, K. J. Satzinger, J. Atalaya, A. N. Korotkov, A. Dunsworth, D. Sank, C. Quintana, M. McEwen, R. Barends, P. V. Klimov, S. Hong, C. Jones, A. Petukhov, D. Kafri, Demura, Sean, Burkett, Brian, Gidney, Craig, Fowler, Austin G., Paler, Alexandru, Putterman, Harald, Aleiner, Igor, Arute, Frank, Arya, Kunal, Babbush, Ryan, Bardin, Joseph C., Bengtsson, Andreas, Bourassa, Alexandre, Broughton, Michael, Buckley, Bob B., Buell, David A., Bushnell, Nicholas, Chiaro, Benjamin, Collins, Roberto, Courtney, William, Derk, Alan R., Eppens, Daniel, Erickson, Catherine, Farhi, Edward, Foxen, Brooks, Giustina, Marissa, Greene, Ami, Gross, Jonathan A., Harrigan, Matthew P., Harrington, Sean D., Hilton, Jeremy, Ho, Alan, Huang, Trent, Huggins, William J., Ioffe, L. B., Isakov, Sergei V., Jeffrey, Evan, Jiang, Zhang, Kechedzhi, Kostyantyn, Kim, Seon, Kitaev, Alexei, Kostritsa, Fedor et Landhuis, David, «Exponential suppression of bit or phase errors with cyclic error correction,» *Nature,* 2021.

[85] A. Al-Shemmery, G. George et M. G. K. G. Paracha, «Software Implementation of Quantum Error-Correction,» *Undergraduate Research Scholars Program,* 2022.

[86] A. Noiri, T. Nakajima, T. Kobayashi, S. Tarucha et K. Takeda, «Quantum error correction with silicon spin qubits,» *Nature,* 2022.

[87] S. Krinner, Lacroix, Nathan, Remm, Ants, Di Paolo, Agustin, Genois, Elie, Leroux, Catherine, Hellings, Christoph, Lazar, Stefania, Swiadek, Francois, Herrmann, Johannes, Norris, Graham J., Andersen, Christian Kraglund, Muller, Markus, Blais, Alexandre, Eichler, Christopher et Wallraff, Andreas, «Realizing repeated quantum error correction in a distance-three surface code,» *Nature,* 2022.

[88] Á. Márton et János K. Asbóth, «Coherent errors and readout errors in the surface code,» *Quantum,* 2023.

[89] T. R. Scruby, Logical gates by code deformation in Topological Quantum Codes, 2021.

[90] T. Tansuwannont et Leung, Debbie, «Fault-tolerant quantum error correction using error weight parities,» *American Physical Society,* 2021.

[91] B. W. Reichardt, «Fault-tolerant quantum error correction for Steane's seven-qubit color code with few or no extra qubits,» *Quantum,* 2020.

[92] J. R. McClean, Jiang, Zhang, Rubin, Nicholas C., Babbush, Ryan et Neven, Hartmut, «Decoding quantum errors with subspace expansions,» *Nature Communications,* 2020.

[93] N. P. Breuckmann et Higgott, Oscar, «Subsystem Codes with High Thresholds by Gauge Fixing and Reduced Qubit Overhead,» *Phys. Rev. X,* 2021.

[94] M. Taylor et Charles Woodward, Holography, cellulations and error correcting codes, 2021.

[95] W. Cai, Ma, Yuwei, Wang, Weiting, Zou, Chang-Ling et Sun, Luyan, «Bosonic quantum error correction codes in superconducting quantum circuits,» *Fundamental Research,* 2021.

[96] P. R. S. M. W. L. F. L. J. L. e. a. Reinhold, «Error-corrected gates on an encoded qubit,» *Nature Physics,* p. arxiv.org, 2020.

[97] V. B. M. M. M. S. A. H. e. a. J. F., «Logical-qubit operations in an error-detecting surface code,» *Nature Physics,* 2022.

[98] D. S. S. B. J. M. G. e. a. C. Piveteau, «Error mitigation for universal gates on encoded qubits,» *Physical Review Letters,* 2021.

[99] I. H. K. S. D. B. a. B. J. B. L. Z. Cohen, «Low-overhead fault-tolerant quantum computing using long-range connectivity,» *Science Advances,* 2022.

[100] C. A. P. S. M. D. C. e. a. P. Das, «A scalable decoder micro-architecture for fault-tolerant quantum computing,» *arXiv preprint,* 2020.

[101] M. S. K. e. a. R. Sweke, «Reinforcement learning decoders for fault-tolerant quantum computation,» *Machine Learning,* 2020.

[102] M. Shirvanimoghaddam, «Primitive rateless codes,» *IEEE Transactions,* 2021.

[103] V. J. U. P. e. a. M. J. Salariseddigh, «Deterministic K-Identification for MC Poisson Channel With Inter-Symbol Interference,» *IEEE Open Journal of the Communications Society,vol,* 2024.

[104] V. J. U. P. e. a. M. J. Salariseddigh, «Deterministic identification for MC ISI-Poisson channel,» *in ICC 2023-IEEE International Conference on Communications,* 2023.

[105] J. E. M. B. Pereira, Quantum error-correcting codes, 2022.

[106] D. C. S. X. N. a. L. H. R. Cane, «Mitigation of decoherence-induced quantum-bit errors and quantum-gate errors using Steane's code,» *IEEE Access,* 2020.