Quantum LDPC Codes and Belief Propagation (BP) Decoders

Submitted by

Debashis Saikia
debashis23@iisertvm.ac.in

BS-MS Student
Department of Physics
Indian Institute of Science Education and Research, Thiruvananthapuram

Under the guidance of

Dr. Arun B Aloshious
aloshious@iitg.ac.in

Assitant Professor
Department of Electronics and Electrical Engineering
Indian Institute of Technology, Guwahati

From 12th May to 10th July

Acknowledgement

I sincerely thank my supervisor for his constant guidance and support throughout this internship. I am also grateful to the Indian Institute of Technology, Guwahati, for providing a wonderful research environment and the opportunity to explore quantum error correction. My thanks also extend to everyone who encouraged and supported me.

Abstract

Quantum low-density parity-check (qLDPC) codes have emerged as promising candidates for scalable fault-tolerant quantum computation due to their potential to achieve constant rate and linear distance with sparse connectivity. During this internship, I conducted an in-depth study of qLDPC code constructions, focusing particularly on hypergraph product and lifted product codes. I explored the theoretical foundations of these codes and reviewed key papers that contribute to understanding their structure and performance. Additionally, I implemented Python-based code generators that can construct instances of these codes programmatically, facilitating further analysis and experimentation.

Contents

1 Introduction

Quantum computers have the potential to revolutionize computation by solving specific problems exponentially faster than their classical counterparts. Notable examples include factoring large integers, simulating quantum systems, and optimizing complex functions [1, 2, 3]. However, the practical realization of large-scale quantum computation faces a significant problem: qubits are highly fragile. Qubits, the fundamental units of quantum information, are susceptible to decoherence and operational errors due to their inevitable interaction with the surrounding environment. Even small amounts of noise can corrupt the quantum state and destroy the computation if not mitigated properly.

To mitigate this, quantum error correction (QEC) is essential. The core idea of QEC is to encode quantum information into a larger Hilbert space so that errors can be detected and corrected without collapsing the encoded state. Among the most prominent and well-understood frameworks for QEC are stabilizer codes [1, 4, 3]. These codes extend classical linear codes into the quantum realm by using the mathematical structure of the Pauli group. A stabilizer code is defined by an abelian subgroup of the n-qubit Pauli group, known as the stabilizer group, whose joint $+1$ eigenspace constitutes the code space. Errors are detected by measuring the stabilizer generators, and their syndromes guide the correction process.

Within this framework, logical qubits are manipulated via logical operators, which are elements of the normalizer of the stabilizer group. These operators act on the encoded information without disturbing the stabilizer constraints. Importantly, logical operators come in pairs—analogous to X and Z in the single-qubit case—and must obey the same anti-commutation relations to preserve quantum behavior [1, 4]. The interplay between the stabilizer group and its normalizer underpins the structure of quantum codes and defines how information is stored and processed fault-tolerantly.

Despite the elegance of stabilizer codes, their practical use in fault-tolerant architectures faces challenges, especially regarding scalability. Conventional codes, such as the surface code, offer high error thresholds but require large overheads due to local, low-weight constraints. This is where quantum low-density parity-check (qLDPC) codes have emerged as a promising frontier. Inspired by classical LDPC codes, qLDPC codes aim to balance efficient error correction with minimal resource overhead by enforcing that each qubit participates in only a small number of stabilizer checks, and each stabilizer acts on only a small number of qubits [5].

Recent advances have demonstrated that qLDPC codes can simultaneously achieve constant rate, high distance, and sparse connectivity, making them suitable candidates for scalable quantum computation. They offer an architectural advantage in hardware feasibility and decoding efficiency, especially in regimes with desirable massive parallelism and locality. Understanding such codes' theoretical underpinnings and practical performance is central to pushing the boundaries of fault-tolerant quantum computation.

2 Background

The most exciting aspect of quantum error-correcting codes combines research from various elements, like classical coding theory, topology, homology, and combinatorics. For background on quantum error correcting codes, we refer to the review by Terhal [6] and Breukmann [7], and for some specific topics, Gottesman's thesis [4] and Nielsen and Chuang [1].

2.1 Stabilizer Formalism

The stabilizer framework [1, 4] provides an elegant way to describe a large class of quantum error-correcting codes in terms of group theory. We begin with the n-qubit Pauli group.

$$\mathcal{P}_n = \left\{ i^l \, P_1 \otimes P_2 \otimes \cdots \otimes P_n \;\middle|\; P_j \in \{I, X, Y, Z\},\ 0 \le l \le 3 \right\}, \tag{1}$$

where I is the 2×2 identity matrix, X, Y, Z are the Pauli operators, and $\otimes$ denotes the tensor product. The factor i^l captures the possible global phases.

A *stabilizer group* $\mathcal{S}$ is an Abelian subgroup of $\mathcal{P}_n$ that does not contain $-I$. Suppose $\mathcal{S}$ has m independent generators $S_1, \ldots, S_m$. Then the associated stabilizer code is defined as the simultaneous $+1$ eigenspace of all generators:

$$C = \{\, |\psi\rangle \in (\mathbb{C}^2)^{\otimes n} \;\mid\; S_i|\psi\rangle = |\psi\rangle \ \ \forall i \,\}. \tag{2}$$

Since $\mathcal{S}$ has 2^m elements, the code space has dimension

$$\dim(C) = 2^{n-m} = 2^k, \tag{3}$$

which defines an (n, k) stabilizer code.

Binary Representation. One of the most powerful features of the stabilizer formalism is that it admits a binary description. Each single-qubit Pauli is mapped to a pair of bits:

$$I \mapsto (0,0), \quad X \mapsto (1,0), \quad Z \mapsto (0,1), \quad Y \mapsto (1,1). \tag{4}$$

Thus, an n-qubit Pauli operator corresponds to a binary vector of length $2n$,

$$P \;\longleftrightarrow\; (\mathbf{a} \,|\, \mathbf{b}) \in \mathbb{F}_2^{2n} \tag{5}$$

where $\mathbf{a}, \mathbf{b} \in \mathbb{F}_2^n$ record the X and Z components, respectively.

If we apply this mapping to the m stabilizer generators, we obtain the *parity-check matrix*

$$H = \left[H_X \mid H_Z \right], \tag{6}$$

with $H_X, H_Z \in \mathbb{F}_2^{m \times n}$. In other words, H is an $m \times 2n$ binary matrix whose rows encode the action of each generator.

Commutation Condition. Two Pauli operators commute if and only if their binary vectors satisfy a simple symplectic condition. For rows $(\mathbf{a}|\mathbf{b})$ and $(\mathbf{a}'|\mathbf{b}')$ of H, the commutation relation is

$$\mathbf{a} \cdot \mathbf{b}' + \mathbf{a}' \cdot \mathbf{b} \equiv 0 \pmod 2. \tag{7}$$

This is equivalently expressed in matrix form as

$$H \Lambda H^T = 0, \tag{8}$$

where

$$\Lambda = \begin{bmatrix} 0 & I_n \\ I_n & 0 \end{bmatrix}. \tag{9}$$

Thus, the requirement that the stabilizer group be Abelian translates neatly into a quadratic condition over $\mathbb{F}_2$.

2.2 CSS codes

Let us consider two classical codes $C_X, C_Z \in \mathbb{F}_2^n$. Then a CSS (*Calderbank-Shor-Steane*) code is defined if C_X and C_Z which are orthogonal,i.e., they satisfy thee following relation[8, 7]:

$$C_X \subset C_Z^{\perp} \tag{10}$$

There is another way to define the CSS code, which is more popular and easier to verify. Let us consider H_X and H_Z, where H_X and H_Z are the corresponding parity check matrices of C_X and C_Z. Here, H_X represents the X-checks and H_Z represents the Z-checks. If the defined code is CSS, i.e., $C_X \subset C_Z^{\perp}$, it is equivalent to saying;

$$H_X H_Z^T = 0 \tag{11}$$

And the final parity check matrix becomes:

$$H = \left[\begin{array}{c|c} H_X & 0 \\ 0 & H_Z \end{array} \right] \tag{12}$$

with the commutativity relation 11.

2.3 Homological Perspective on CSS codes

Chain Complexes. In mathematics, a chain complex is defined by a sequence of abelian groups (or modules) which are connected to the adjacent abelian groups (or modules) via homomorphisms (called boundary operators or differentials), such that the composition of consecutive homomorphisms is zero [9]. The equation 14 shows an example of a chain complex.

$$C = \left(C_n \xrightarrow{\partial_n} \cdots \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0 \right) \tag{13}$$

So, in this homological perspective, a CSS code can be written as a chain complex of length three:

$$C = \left(C_2 \xrightarrow{\partial_2 = H_Z^T} C_1 \xrightarrow{\partial_1 = H_X} C_0 \right) \tag{14}$$

Here, the condition of consecutive homomorphisms; $\partial_n \partial_{n+1} = 0$ is also satisfied due to the CSS condition $H_X H_Z^T = 0$

2.4 Tanner Graphs

Tanner graphs can be considered as the visual representations of the codes. Each code has a parity check matrix and the non-zero elements that represent the operations of checks on the bits, for example: we have a parity check matrix E, then $E_{i,j}$ tells you how the i^{th} check (or stabilizer) acts on the j^{th} bit. This is similar to the concept of an incidence matrix in graph theory; considering this aspect will end up with two sets of vertices: a set of checks and a set of bits, which are connected according to the parity check matrix.

The Tanner graph of [7, 4, 3] Hamming code is shown in the figure 1.

In case of the Tanner graph of classical code, the graph is bipartite, i.e., the vertices can be separated into two disjoint sets; bits and X-checks (*bit flip*), but in case of quantum error-correcting codes, it is a tripartite graph,i.e., the vertices can be separated into three different sets which are mutually disjoint; qubits, X-checks (*bit flip*), and Z-checks (*phase flip*). An example of a Tanner graph of a quantum code is shown in the figure 2, which is the hypergraph product code constructed by taking both H_1 and H_2 as the parity check matrix of a 3-bit repetition code

3 Quantum LDPC Codes

In classical coding theory, it is much easier to form low-density parity-check (LDPC)codes with a constant encoding rate ($\frac{k}{n}$) and linear distance ($d \propto n$) using any random codes [10, 11]. In comparison, forming a quantum low-density parity-check(qLDPC) code is much harder, and even it is still a major open problem whether such qLDPC codes exist that can actually show similar properties to their classical counterparts. For example, surface codes, one of the most studied quantum error-correcting codes, have $d \propto \sqrt{n}$ [7]. However, in recent times, much research has been done in this field, and many other codes have also been suggested that have better performance than the surface codes.

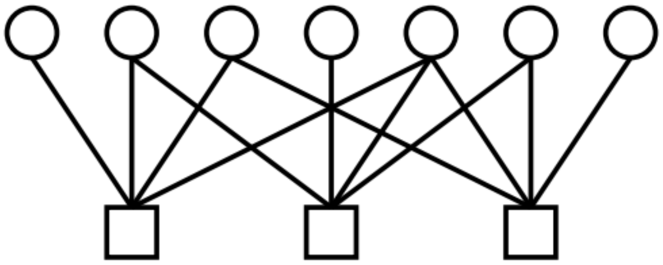

Figure 1: The Tanner graph of the [7, 4, 3] Hamming code. Circles represent the bits and square represent the checks (*constraints)*.

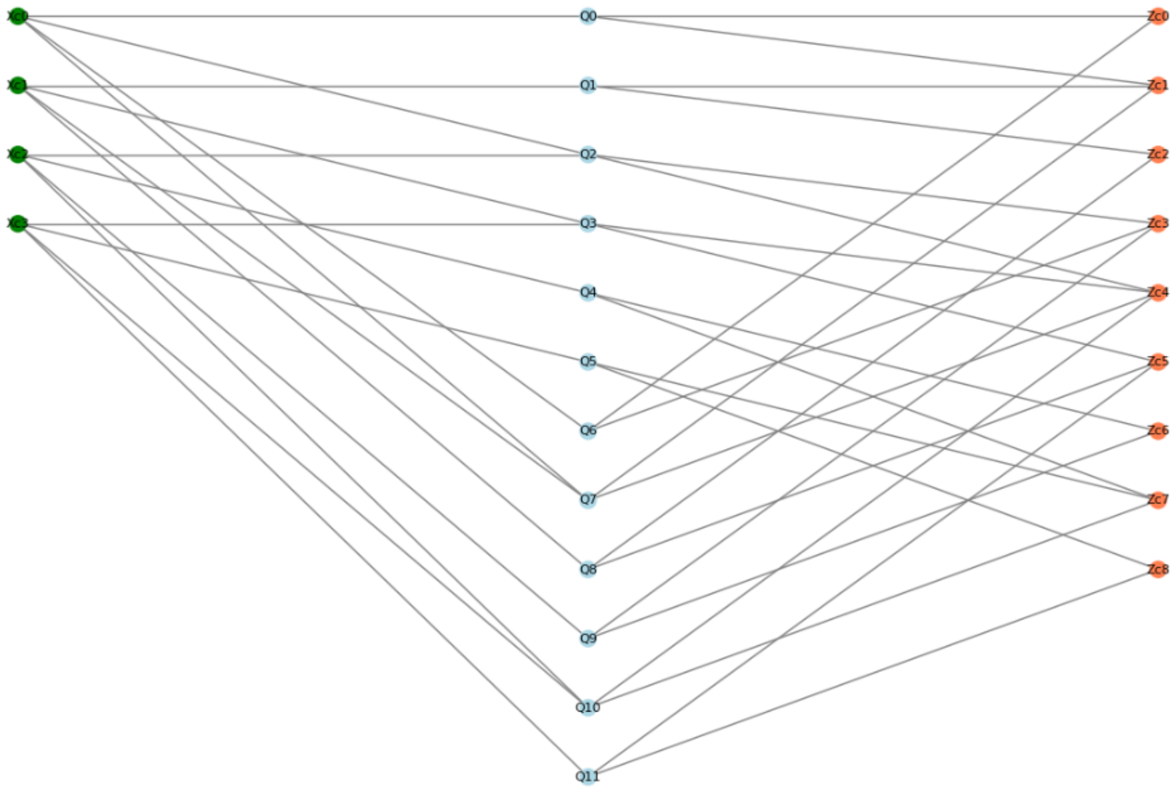

Figure 2: The Tanner graph of a hypergraph product code generated using a 3-bit repetition code. On the left side, green dots represent the X-checks, on the right side, orange dots represent the Z-checks, and at the center, the blue dots represent the qubits.

Name	k	Distance
Hypergraph (Tensor) products	$\Theta(n)$	$\Theta(\sqrt{n})$
Tensor products (Ramanujan complexes)	$\Theta(\sqrt{n})$	$\Omega(\sqrt{n}\,\text{polylog}\, n)$
Fibre bundle codes	$\Theta(n^{3/5}/\text{polylog}\, n)$	$\Omega(n^{3/5}/\text{polylog}\, n)$
Lifted product codes	$\Theta(n^{\alpha}\log n)$	$\Omega(n^{1-\alpha}/\log n)$
Balanced product codes	$\Theta(n^{4/5})$	$\Omega(n^{3/5})$

Table 1: Parameters of various quantum code families which have better performance than Surface codes

3.1 Hypergraph Product Codes

It is the generalization of graphs. In standard graphs, an edge connects two vertices, i.e., edges are pairs of nodes; however, in a hypergraph, an edge can join more than two vertices, i.e., edges are arbitrary sets of nodes.

A *hypergraph product code* (HGP) is a CSS-type quantum LDPC code constructed from two classical binary codes $C_1 = [n_1, k_1, d_1]$ and $C_2 = [n_2, k_2, d_2]$ via the chain-complex tensor (or "product") construction. One forms boundary maps

$$\partial_2 = (\partial_{C_1} \otimes I,\ I \otimes \partial_{C_2}), \quad \partial_1 = (I \otimes \partial_{C_2} + \partial_{C_1} \otimes I) \quad (15)$$

on $C_2 \otimes C_2 \to C_1 \otimes C_2 \oplus C_2 \otimes C_1 \to C_1 \otimes C_1$, ensuring $\partial_1 \circ \partial_2 = 0$, so that this defines a valid CSS code [12, 13]. The resulting quantum code has $n = n_1 n_2 + n_2 n_1$ physical qubits and encodes

$$k = k_1 k_2 + k_1^{\perp} k_2^{\perp} \quad (16)$$

logical qubits, where $k^{\perp} = n - k$ refers to the classical dual code [13, 14, 15]. Crucially, its minimum distance scales as

$$d \geq \min\{d_1, d_2, d_1^{\perp}, d_2^{\perp}\}, \quad (17)$$

Often giving square-root scaling of distance in code size—an improvement over toric codes while still retaining constant rate [14]. HGP codes are promising because they achieve constant encoding rate, sparse checks, and nontrivial distance scaling, making them attractive for fault-tolerant quantum computing.

Illustrative Example 1.

Consider two matrices

$$H_1 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \qquad H_2 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

Here, H_1 has dimensions 2×2 ($m_1 = 2, n_1 = 2$), while H_2 has dimensions 2×3 ($m_2 = 2, n_2 = 3$). Thus, the number of physical qubits is

$$n = n_1 n_2 + m_1 m_2 = 2 \cdot 3 + 2 \cdot 2 = 10.$$

The X-stabilizer matrix is constructed as

$$H_X = \begin{bmatrix} I_{m_1} \otimes H_2 & H_1^T \otimes I_{m_2} \end{bmatrix},$$

while the Z-stabilizer matrix is

$$H_Z = \begin{bmatrix} H_1 \otimes I_{n_2} & I_{n_1} \otimes H_2^T \end{bmatrix}.$$

These become

$$H_X = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix},$$

and

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

Both H_X and H_Z are binary matrices acting on the $n = 10$ qubits. By construction, one verifies

$$H_X H_Z^T = 0,$$

ensuring commutativity of the stabilizers.

The explicit forms of H_X and H_Z shown above were generated using the code available in the

Graph Theoretic Perspective: From a graph-theoretic perspective, consider a bipartite graph $G = (V \cup C, E)$ with variable nodes V, check nodes C, and edges E. The hypergraph product construction can be described by forming the Cartesian product of G with itself, denoted

$$G_Q = G \times G,$$

which serves as the factor graph of the resulting quantum code Q.

The vertex set of G_Q naturally separates into three categories:

1. *Qubit nodes:* pairs of variable nodes $V \times V$ together with pairs of check nodes $C \times C$,
2. *X-type stabilizers:* nodes in $V \times C$,
3. *Z-type stabilizers:* nodes in $C \times V$.

This structure shows that the construction introduces two distinct kinds of qubits: those associated with pairs of variable nodes ("VV" type) and those associated with pairs of check nodes ("CC" type).

Edges in G_Q are placed according to the Cartesian product rule. Specifically, two nodes (a_1, a_2) and (b_1, b_2) in G_Q are connected if either

$$(a_1, b_1) \in E \quad \text{with } a_2 = b_2,$$

or

$$(a_2, b_2) \in E \quad \text{with } a_1 = b_1.$$

This reformulation highlights the combinatorial essence of the hypergraph product: The code's qubits and stabilizers emerge directly from the tensor-product structure of the underlying bipartite graph.

3.2 Lifted Product Codes

Lifted Product (LP) codes, also known as *Generalized Hypergraph Product codes*[15], introduced by Panteleev and Kalachev [14], form an important family of *quantum LDPC (qLDPC) codes* that achieve near-optimal asymptotic parameters. They are obtained by applying a *lift* to a Tanner graph associated with two classical binary linear codes, thereby producing a large structured quantum code from small base matrices. This approach generalizes the hypergraph product construction by incorporating permutation matrices, which allows one to improve distance scaling while maintaining sparsity.

Let $H_1 \in \mathbb{F}_2^{m_1 \times n_1}$ and $H_2 \in \mathbb{F}_2^{m_2 \times n_2}$ be two binary parity-check matrices defining classical codes. The Tanner graph of H_1 (resp. H_2) consists of n_1 (resp. n_2) variable nodes and m_1 (resp. m_2) check nodes, with edges indicating nonzero entries.

In the lifted construction, instead of associating each nonzero entry with the identity matrix I_L as in the hypergraph product, one replaces it with an $L \times L$ permutation matrix P_{ij} drawn from a set of carefully chosen permutations. Formally, define a *lifted adjacency matrix* $\widetilde{H}_1$ by replacing each 1 in H_1 with some $P_{ij} \in \mathcal{P}_L$, and each 0 with the $L \times L$ zero matrix. Similarly, construct $\widetilde{H}_2$.

The lifted product then defines the stabilizer check matrices

$$H_X = \left[I_{m_1} \otimes \widetilde{H}_2 \quad \widetilde{H}_1^T \otimes I_{m_2} \right], \tag{18}$$

$$H_Z = \left[\widetilde{H}_1 \otimes I_{n_2} \quad I_{n_1} \otimes \widetilde{H}_2^T \right]. \tag{19}$$

The number of physical qubits is

$$n = n_1 n_2 L + m_1 m_2 L,$$

while the number of stabilizer checks is given by the row dimensions of H_X and H_Z. By construction, the commutativity condition

$$H_X H_Z^T = 0$$

holds identically, since the Kronecker structure and permutation lifts preserve orthogonality.

The main advantage of the lifted product construction lies in its flexibility: By carefully choosing the lifting degree L and the associated permutations, one can achieve improved minimum distance compared to hypergraph product codes. Recent results show that lifted product codes yield families of asymptotically good *qLDPC codes* with linear distance scaling, marking a breakthrough in quantum coding theory.

4 Belief Propagation Decoder

Belief propagation (BP), introduced by Gallager in his seminal work on LDPC codes [11] and later generalized by Pearl in the context of graphical models, is the fundamental message-passing algorithm for probabilistic inference on factor graphs [16]. Consider a codeword $\mathbf{x} = (x_1, \ldots, x_n)$ transmitted over a memoryless channel, with received vector $\mathbf{y} = (y_1, \ldots, y_n)$. The decoding problem can be posed as finding the marginal a posteriori probability

$$P(x_i \mid \mathbf{y}) = \frac{1}{Z} \sum_{\mathbf{x} \setminus x_i} P(\mathbf{y} \mid \mathbf{x}) P(\mathbf{x}), \tag{20}$$

where Z is a normalizing constant and the sum runs over all codewords consistent with $\mathbf{x}$ except x_i. Direct computation is intractable, since the sum spans an exponentially large set of codewords.

BP makes this computation feasible by exploiting the factorization of the joint distribution induced by the Tanner graph. If $\mathcal{V}$ denotes the set of variable nodes and $\mathcal{C}$ the set of check nodes, the joint posterior can be expressed as

$$P(\mathbf{x} \mid \mathbf{y}) \propto \prod_{i \in \mathcal{V}} P(y_i \mid x_i) \prod_{j \in \mathcal{C}} \mathbb{I}\left(\bigoplus_{i \in \mathcal{N}(j)} x_i = 0 \right), \tag{21}$$

where $\mathbb{I}(\cdot)$ is the indicator of parity-check satisfaction and $\oplus$ denotes modulo-2 addition. BP proceeds by passing messages along edges of the Tanner graph: from variables to checks and vice versa. At iteration ℓ, the variable-to-check message is

$$m_{i \to j}^{(\ell)}(x_i) = \kappa \, P(y_i \mid x_i) \prod_{j' \in \mathcal{N}(i) \setminus j} m_{j' \to i}^{(\ell - 1)}(x_i), \tag{22}$$

where κ is a normalization constant. The check-to-variable update is given by

$$m_{j \to i}^{(\ell)}(x_i) = \sum_{\{x_k\}_{k \in \mathcal{N}(j) \setminus i}} \mathbb{I}\left(\bigoplus_{k \in \mathcal{N}(j)} x_k = 0 \right) \prod_{k \in \mathcal{N}(j) \setminus i} m_{k \to j}^{(\ell)}(x_k). \tag{23}$$

The belief at variable node i is then computed as

$$b_i^{(\ell)}(x_i) = \kappa \, P(y_i \mid x_i) \prod_{j \in \mathcal{N}(i)} m_{j \to i}^{(\ell)}(x_i), \tag{24}$$

with the hard decision taken as $\hat{x}_i = \arg\max_{x_i \in \{0,1\}} b_i^{(\ell)}(x_i)$.

These equations highlight the generality of BP: the algorithm computes exact marginals when the Tanner graph is cycle-free, and provides highly accurate approximations in the presence of cycles. Johnson [10] emphasizes that this original formulation underpins all practical variants, including the log-domain sum–product and the min–sum algorithms used in LDPC decoders.

4.1 Sum Product

The sum–product algorithm (SPA) is the standard form of belief propagation used in decoding low-density parity-check (LDPC) codes. First introduced by Gallager [11] and later formalized using factor graphs by Kschischang et al. [16], SPA iteratively exchanges probabilistic messages between variable and check nodes in the Tanner graph. In practice, these messages are expressed as log-likelihood ratios (LLRs), with updates involving sums and hyperbolic tangent functions. While exact only on cycle-free graphs, SPA achieves near-capacity performance on practical loopy graphs [10].

Consider a binary LDPC code with parity-check matrix H. The decoding problem can be posed as estimating the transmitted codeword $\mathbf{x}$ from the received vector $\mathbf{y}$ over a noisy channel. For the binary-input additive white Gaussian noise (AWGN) channel, the channel log-likelihood ratio (LLR) for bit i is given by

$$r_i = \ln \frac{P(y_i \mid x_i = 0)}{P(y_i \mid x_i = 1)}. \tag{25}$$

In the Tanner graph representation, messages are exchanged between variable nodes (corresponding to bits) and check nodes (corresponding to parity constraints). Initially, the variable-to-check message is set as

$$M_{j,i}^{(0)} = r_i, \qquad \forall (i,j) \in E, \tag{26}$$

where E denotes the set of edges in the Tanner graph.

At iteration ℓ, the check-to-variable message is computed by combining incoming messages under the parity constraint. In LLR form, this update is expressed as [10]

$$E_{j,i}^{(\ell)} = \ln \left(\frac{1 + \prod_{i' \in \mathcal{N}(j) \setminus i} \tanh\left(\frac{M_{j,i'}^{(\ell-1)}}{2}\right)}{1 - \prod_{i' \in \mathcal{N}(j) \setminus i} \tanh\left(\frac{M_{j,i'}^{(\ell-1)}}{2}\right)} \right), \tag{27}$$

where $\mathcal{N}(j)$ is the neighborhood of check node j. This expression can be derived by rewriting the parity-check constraint in terms of conditional probabilities and using the identity

$$\ln \frac{1+x}{1-x} = 2\tanh^{-1}(x). \tag{28}$$

The variable-to-check update then aggregates the channel information with all extrinsic incoming check messages:

$$M_{j,i}^{(\ell)} = r_i + \sum_{j' \in \mathcal{N}(i) \setminus j} E_{j',i}^{(\ell)}, \tag{29}$$

where $\mathcal{N}(i)$ denotes the set of check nodes connected to variable i.

The belief, or posterior LLR, for bit i after ℓ iterations is

$$L_i^{(\ell)} = r_i + \sum_{j \in \mathcal{N}(i)} E_{j,i}^{(\ell)}. \tag{30}$$

A hard decision is obtained as

$$\hat{x}_i^{(\ell)} = \begin{cases} 0, & L_i^{(\ell)} > 0, \\ 1, & L_i^{(\ell)} < 0. \end{cases} \tag{31}$$

The algorithm terminates if all parity-check equations are satisfied, i.e.,

$$H \cdot \hat{\mathbf{x}}^T = \mathbf{0}. \tag{32}$$

Although BP is exact only on cycle-free graphs, Johnson [10] shows that its iterative nature allows it to perform remarkably well on loopy Tanner graphs encountered in LDPC codes. This balance between computational feasibility and near-optimal error correction explains why BP decoding remains the cornerstone of modern error-correcting code implementations.

Algorithm 1 Belief Propagation (Sum-Product) Decoding for LDPC Codes

```
0: Input: Channel log-likelihood ratios (LLRs) $L_i$ for each variable node $v_i$
0: Initialization: For each edge $(v_i, c_j)$, set $m^{(0)}_{v_i \to c_j} = L_i$
0: for iteration $t = 1$ to $T_{\max}$ do
0:   for each check node $c_j$ do
0:     for each variable node $v_i \in \mathcal{N}(c_j)$ do
0:       Compute message:
           $$m^{(t)}_{c_j \to v_i} = 2\tanh^{-1}\left(\prod_{v_k \in \mathcal{N}(c_j)\setminus v_i} \tanh\left(\frac{m^{(t-1)}_{v_k \to c_j}}{2}\right)\right)$$
0:     end for
0:   end for
0:   for each variable node $v_i$ do
0:     for each check node $c_j \in \mathcal{N}(v_i)$ do
0:       Update message:
           $$m^{(t)}_{v_i \to c_j} = L_i + \sum_{c_k \in \mathcal{N}(v_i)\setminus c_j} m^{(t)}_{c_k \to v_i}$$
0:     end for
0:   end for
0:   Decision: For each $v_i$, compute
           $$\hat{x}^{(t)}_i = \mathrm{sign}\left(L_i + \sum_{c_j \in \mathcal{N}(v_i)} m^{(t)}_{c_j \to v_i}\right)$$
0:   if all parity-checks satisfied then
0:     Output: Estimated codeword $\hat{x}$ and stop
0:   end if
0: end for
0: Output: Final $\hat{x}$ after $T_{\max}$ iterations =0
```

4.2 Min–Sum

The min–sum algorithm arises as a practical approximation of the sum–product algorithm. Specifically, the check-node update in SPA, which involves products of hyperbolic tangents, can be simplified by noting that

$$2\tanh^{-1}\left(\prod_k \tanh\left(\frac{M_k}{2}\right)\right) \approx \left(\prod_k \mathrm{sign}(M_k)\right) \cdot \min_k |M_k|. \tag{33}$$

Thus, instead of evaluating costly tanh and $\tanh^{-1}$ functions, the min–sum rule uses the sign product and the minimum magnitude among incoming messages. This approximation reduces computational complexity significantly, with only a moderate performance loss compared to full SPA decoding.

4.3 Ordered Statistics Decoding (OSD)

Ordered Statistics Decoding (OSD)[17] is a powerful reliability-based decoding technique to enhance the performance of linear block codes beyond standard hard-decision decoding. The main idea is to exploit the relative reliability of received bits to identify the most likely error patterns. Let a received vector over a binary-input AWGN channel be $\mathbf{y} = (y_1, y_2, \ldots, y_n)$, and let the corresponding log-likelihood ratios (LLRs) be

$$r_i = \ln \frac{P(y_i \mid x_i = 0)}{P(y_i \mid x_i = 1)}, \quad i = 1, \ldots, n. \tag{34}$$

In OSD, the bits are first permuted according to the magnitude of their reliabilities:

$$|r_{i_1}| \geq |r_{i_2}| \geq \cdots \geq |r_{i_n}|, \tag{35}$$

so that the most reliable positions are at the front. Next, the generator matrix G of the code is transformed via Gaussian elimination into systematic form aligned with the reliability order:

$$G' = [I_k \mid P], \tag{36}$$

where I_k is the $k \times k$ identity corresponding to the most reliable positions, and P is the parity portion.

The core decoding step is to consider all error patterns of weight up to t in the k most reliable positions. For each candidate error vector $\mathbf{e}_k$, the estimated codeword is computed as

$$\hat{\mathbf{c}} = \mathbf{r}_s \oplus \mathbf{e}_k \cdot G', \tag{37}$$

where $\mathbf{r}_s$ is the hard-decision version of the permuted received vector. The likelihood of each candidate codeword is evaluated using the original LLRs:

$$\Lambda(\hat{\mathbf{c}}) = \sum_{i=1}^{n} (-1)^{\hat{c}_i} r_i, \tag{38}$$

and the decoder selects the codeword with the *maximum likelihood* (minimum Λ). By exploiting the ordering of reliabilities, OSD achieves near-ML performance with dramatically reduced complexity compared to exhaustive search.

5 Conclusion

This project provided a comprehensive exploration of quantum error correction, beginning with the stabilizer formalism and extending into advanced families of quantum LDPC codes such as the Hypergraph Product Code and the Lifted Product Code. By studying their structure and simulating their performance, I gained deeper insights into how these codes balance rate, distance, and efficiency — three crucial aspects for practical fault-tolerant quantum computing.

On the decoding side, experimenting with belief propagation–based algorithms (Sum-Product, Min-Sum, and Ordered Statistics Decoding) highlighted both their strengths and limitations in handling the degeneracy and correlations inherent to quantum codes. Implementing and simulating these decoders not only reinforced theoretical understanding but also provided hands-on experience with algorithmic design and performance benchmarking.

Overall, the internship has strengthened my foundations in quantum error correction, expanded my technical skills in simulation and coding, and offered a forward-looking perspective on the role of qLDPC codes in building scalable, fault-tolerant quantum computers. The experience leaves me well-prepared to pursue deeper research in this domain and contribute to the next generation of quantum technologies.

All the codes made during the internship are available in the GitHub repository [18].

6 Future Work

This project has laid the groundwork for a deeper exploration of decoding strategies for quantum LDPC codes. While current belief propagation–based decoders such as Sum-Product, Min-Sum, and Ordered Statistics Decoding offer promising performance, challenges remain in achieving efficient and scalable decoding that fully leverages the structure of qLDPC codes.

As a continuation of this work, I am pursuing a long-term research project in collaboration with my professor, focusing on developing and refining more effective decoders for qLDPC codes. The goal is to design decoding strategies that improve error-correction thresholds, reduce complexity, and enhance practical applicability for near-term and future quantum architectures.

References

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[2] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[3] J. Preskill, "Lecture notes for physics 229: Quantum information and computation." `https://www.preskill.caltech.edu/ph229/`, 1998. California Institute of Technology.

[4] D. Gottesman, *Stabilizer codes and quantum error correction*. California Institute of Technology, 1997.

[5] D. Gottesman, "Fault-tolerant quantum computation with constant overhead," *arXiv preprint arXiv:1310.2984*, 2013.

[6] B. M. Terhal, "Quantum error correction for quantum memories," *Reviews of Modern Physics*, vol. 87, no. 2, pp. 307–346, 2015.

[7] N. P. Breuckmann and J. N. Eberhardt, "Quantum low-density parity-check codes," *PRX quantum*, vol. 2, no. 4, p. 040101, 2021.

[8] A. R. Calderbank, E. M. Rains, P. M. Shor, and N. J. Sloane, "Quantum error correction via codes over gf (4)," *IEEE Transactions on Information Theory*, vol. 44, no. 4, pp. 1369–1387, 1998.

[9] Wikipedia contributors, "Chain complex."

[10] S. J. Johnson, *Iterative error correction: Turbo, low-density parity-check and repeat-accumulate codes*. Cambridge university press, 2010.

[11] R. Gallager, "Low-density parity-check codes," *IRE Transactions on information theory*, vol. 8, no. 1, pp. 21–28, 2003.

[12] J.-P. Tillich and G. Zemor, "Quantum ldpc codes with positive rate and minimum distance proportional to n 1/2,"

[13] B. Audoux and A. Couvreur, "On tensor products of css codes," *Annales de l'Institut Henri Poincaré D*, vol. 6, no. 2, pp. 239–287, 2019.

[14] P. Panteleev and G. Kalachev, "Quantum ldpc codes with almost linear minimum distance," *IEEE Transactions on Information Theory*, vol. 68, no. 1, pp. 213–229, 2021.

[15] P. Panteleev and G. Kalachev, "Degenerate quantum ldpc codes with good finite length performance," *Quantum*, vol. 5, p. 585, 2021.

[16] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001.

[17] J. Roffe, D. R. White, S. Burton, and E. Campbell, "Decoding across the quantum low-density parity-check code landscape," *Phys. Rev. Res.*, vol. 2, p. 043423, Dec 2020.

[18] D. Saikia, "Quantum-error-correcting-codes: Simulation implementations of qldpc and decoder algorithms." GitHub repository, Indian Institute of Technology Guwahati, 2025. `https://github.com/debashis-saikia/Quantum-Error-Correcting-codes`.