Identifying and Analyzing Topological Properties of Quantum Error-Correcting Codes

Honours Thesis (PHYS 598 – Fall 2024/Winter 2025)

Hung Nguyen

Department of Physics and Astronomy, University of Calgary

Supervisor: Dr. David Feder

Department of Physics and Astronomy, University of Calgary

Institute for Quantum Science and Technology, University of Calgary

(Dated: April 11, 2025)

Abstract

Quantum Error-Correcting Codes (QECCs) are crucial for protecting qubits from noise in quantum computing. Among the most powerful QECCs is the two-dimensional Toric Code, whose topological properties help safeguard quantum information across multiple qubits simultaneously. Such topological QECCs exhibit global features that remain invariant under local perturbations—a desirable trait for error correction. Additionally, all stabilizer QECCs (including the Toric Code) can be represented via graphs, which reflect the design and classification of quantum codes while revealing key properties. However, recognizing topological structure in graph form is not always straightforward. This project aims to uncover how the Toric Code's topological features manifest within graph states. Graph representations were constructed by converting Stabilizer Generators into adjacency matrices using the symplectic formalism, and local Clifford equivalents were systematically explored through local complementations and permutation symmetries. We analytically derived a graph representation of the 2×2 Toric Code, identifying a unique 'windmill' structure that is exactly equivalent to the original code under local Clifford operations. Inspired by this structure, we then examined candidate graphs for the 3×3 Toric Code, revealing configurations that replicate its essential structural and coding properties. This work provides deeper insight into how topological features emerge in graph-based QECCs, and it offers tools for constructing new codes aimed at future fault-tolerant quantum computing applications.

I. INTRODUCTION

The concept of quantum computing originated more than 40 years ago from Benioff (1980) and Feynman (1982), independently [1, 2]. In 1985, the theoretical realization of a universal quantum computer was published by Deutsch [3]. But not until 1994 when Peter Shor invented one of the first meaningful quantum algorithms – discrete logarithms and factoring – had quantum computing entered the centre of research [4]. Quantum computing promises various revolutionizing applications in cryptography, optimization, and most importantly, simulation of quantum systems [2, 5–7].

The physical realization of quantum computers, however, has since been a great engineering challenge. Ambient noise, no matter how little, significantly affects the performance of current quantum computers [8]. Naturally, a great portion of current research in quantum computing focuses on Fault-Tolerant Quantum Computing – the study of how to minimize environmental noise using algorithmic approaches or hardware designs. The algorithmic approaches majorly focus on the development of quantum error-correcting codes (QECCs), which is the main focus of this report.

In 1995, Shor demonstrated the first quantum error-correcting code, which is known as the Shor code [9]. Shortly after, multiple error-correcting codes were developed. Among these, particularly notable are the Calderbank-Shor-Steane (CSS) codes, which are a class of code constructed from classical linear codes [10, 11]. In 1996, Gottesman introduced the *stabilizer formalism*, providing the language to describe quantum error-correcting codes [12]. Using this formalism, Kitaev developed the beautiful idea of using topology to protect quantum information, resulting in the *Toric Code* [13]. Due to the high error resistance of the Toric Code, it has become the most popular QECC to be implemented in quantum computers in the form of the *surface code* – a special case of the Toric Code [14].

On another note, the stabilizer formalism also provides a way to represent QECCs using graphs. The graph representation is a powerful tool to study the properties of QECCs because it offers an intuitive way to visualize the connectivity of qubits and the relationships between Stabilizer Generators. Furthermore, it allows for using well-established graph-theoretic approaches to analyze and optimize QECCs. QECCs can be constructed from graphs as how they can be constructed from Stabilizer Generators. Thus, it is of great interest to study the prominent error-correcting features of the Toric Code – its topological properties – in the graph representation. However, the topological order of the Toric Code is not immediately apparent in its graph representation. This project seeks to identify and analyze the topological properties of the Toric Code in its graph representation, and to study how these properties can be used to construct new QECCs.

II. BACKGROUND

This section provides the necessary but not sufficient background to understand the methods and results presented in this report. For a more comprehensive introduction to quantum computation and quantum error correction, please refer to the most popular textbook in the field, *Quantum Computation and Quantum Information* by Nielsen and Chuang [15]. For an in-depth introduction to the homological description of the Toric Code, please refer to *Quantum Error Correction* by Lidar and Brun [16].

A. Qubits and Quantum Gates

A *qubit* is a two-level quantum system, which can be represented as a linear combination of the *computational basis states* $|0\rangle$ and $|1\rangle$:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{1}$$

where α and β are complex numbers satisfying the normalization condition

$$|\alpha|^2 + |\beta|^2 = 1.$$

The $|+\rangle$ and $|-\rangle$ states are defined as:

$$|+\rangle = \frac{(|0\rangle + |1\rangle)}{\sqrt{2}}, \quad |-\rangle = \frac{(|0\rangle - |1\rangle)}{\sqrt{2}}. \tag{2}$$

A *Quantum Gate* U is a unitary operator of dimension $2^n \times 2^n$ that acts on n qubits, whereas $U\ U^\dagger = U^\dagger\ U = I$.

The next Subsection describes Pauli matrices, which are essential for the content of this report. For more details on quantum gates mentioned in this report, please refer to Appendix A.

1. *Pauli Matrices*

If you have taken a quantum mechanics course, you should be familiar with the Pauli spin matrices σ_x, σ_y, and σ_z. In this report, we will instead use the following notations:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{3}$$

Pauli matrices are Hermitian operators, which means that they satisfy the condition $P^\dagger = P$. Hermitian Quantum Gates are important because they correspond to *observables* in quantum mechanics. As a matter of fact, the computational basis states are the eigenstates of the Z operator. Similarly, the $|+\rangle$ and $|-\rangle$ states are the eigenstates of the X operator, with eigenvalues $+1$ and -1, respectively.

B. Qubit Errors

Any single-qubit Quantum Gate U can be expressed as a linear combination of the Pauli matrices

$$U = a_0 I + a_1 X + a_2 Z + a_3 XZ, \tag{4}$$

where a_0, a_1, a_2, and a_3 are complex numbers satisfying the normalization condition $|a_0|^2 + |a_1|^2 + |a_2|^2 + |a_3|^2 = 1$ [15].

A single-qubit error can be modelled as a Quantum Gate E acting on a qubit, which, in turn, can be expanded using the expansion above. For a qubit in the state $|\psi\rangle$, the effect of the error E thus results in the state

$$E|\psi\rangle = (a_0 I + a_1 X + a_2 Z + a_3 XZ)|\psi\rangle. \tag{5}$$

Taking *syndrome measurements* collapses the state of the qubit to one of the four possible states $\psi\rangle$, $X|\psi\rangle$, $Z|\psi\rangle$, or $XZ|\psi\rangle$. The syndrome measurement is a projective measurement that can be performed on the qubit to determine the type of Pauli error that has occurred.

The result of the syndrome measurement is a classical bit string that indicates which error has occurred.

To recover the original state $|\psi\rangle$, one can apply the corresponding Pauli Operator to neutralize the error. This shows how the ability to correct a discrete set of errors $E_P = \{I, X, Z, XZ\}$ enables the recovery of single-qubit states from an *arbitrary* error E. In general, *correcting a set of basis errors corrects all of their linear combinations* [15]. This is why one only needs to implement a scheme that corrects a discrete set of errors to be able to conveniently correct a *continuum* of errors. One of the schemes that address this problem are called *quantum error-correcting codes*.

C. Quantum Error-Correcting Codes

Quantum information is prone to decoherence and quantum noise [15]. Therefore, error correction schemes are required to ensure the meaningful execution of quantum algorithms. In classical information theory, *bits* – the most basic unit of information, can have value of 0 or 1 – can be copied and thus information can be retained by counting on the majority of the bits. In quantum information theory, qubits – however – cannot be copied due to the No Cloning Theorem [17]. To circumvent this, the quantum information is *encoded* into a *subspace* called a *code space*. To be more specific, the quantum information is encoded into a larger Hilbert space in such a way that errors (like Pauli errors) would move the system into an orthogonal space called the *error space*. This orthogonality allows the errors to be detected by measurements without collapsing the quantum state [16]. Since we know which error affected our system, we can simply apply the inverse of that error to neutralize it. This is the principle of QECCs.

When a qubit in the state

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

is encoded into a *Logical Qubit*, the Logical Qubit would then be in the state

$$|\psi_L\rangle = \alpha|0_L\rangle + \beta|1_L\rangle,$$

where the $|0_L\rangle$ and $|1_L\rangle$ states are specific linear combinations of the basis states of the code space.

A QECC is often denoted by

$$[n,\ k,\ d]$$

where n is the number of encoding qubits, k is the number of Logical Qubits, and d is the code distance. The code distance is the minimum number of qubit errors needed to cause a logical error. We say an $[n,\ k,\ d]$ QECC uses n qubits in total to encode k Logical Qubits, and can protect information perfectly under d simultaneous errors.

D. Stabilizer Codes

1. Group theory

A *group* $\mathcal{G}$ is a non-empty set equipped with a binary operation $*$ that satisfies the following [18]:

1. Closure: $\forall\ a,\ b \in \mathcal{G},\ a * b \in \mathcal{G}$
2. Associativity: $\forall\ a,\ b,\ c \in \mathcal{G},\ a * (b * c) = (a * b) * c$
3. Identity: $\exists\ e \in \mathcal{G} :\ \forall\ a \in \mathcal{G},\ a * e = e * a = a$
4. Inverse: $\forall\ a \in \mathcal{G},\ \exists\ b \in \mathcal{G} :\ a * b = b * a = e$
5. Commutativity (optional): $\forall\ a,\ b \in \mathcal{G},\ a * b = b * a$.

If a group satisfies the fifth condition, it is called a *commutative group* or an *Abelian group.*

A *normalizer* of $\mathcal{S}$ in the group $\mathcal{G}$ is a subset $\mathcal{N}$ defined as

$$\mathcal{N} = \{g \in \mathcal{G} \mid g\mathcal{S}g^{-1} = \mathcal{S}\}.$$

2. Stabilizer group

The *Pauli group* $\mathcal{P}_n$ is a non-Abelian group defined as

$$\mathcal{P}_n = \{i^k P_1 \otimes P_2 \otimes \cdots \otimes P_n \mid P_j \in \{I,\ X,\ Y,\ Z\},\ k \in \{1,\ 2,\ 3,\ 4\}\}. \tag{6}$$

A *stabilizer group* $\mathcal{S}$ is an Abelian *subgroup* of the Pauli group that does *not* contain the element $-I$ [16]. The elements of a stabilizer group are *involutory*, which means they are their own inverses and they have eigenvalues $+1$ and -1. If not all elements of $\mathcal{S}$ are independent of one another, the stabilizer group can be represented by m independent elements S_i (called the *Stabilizer Generators*)

$$\mathcal{S} = \langle S_1,\ S_2, \ldots,\ S_m \rangle.$$

3. Stabilizer subspace

We define the subspace acting as our code space as all the states $|\psi\rangle$ that satisfy

$$S_i|\psi\rangle = |\psi\rangle, \quad \forall\ S_i \in \mathcal{S}.$$

The QECCs whose code space can be described indirectly by stabilizer groups are called *stabilizer codes*. The motivation for this is that describing $\mathcal{S}$ is much simpler than specifically describing the subspace (containing all states $|\psi\rangle$) that it stabilises.

4. Logical Operators

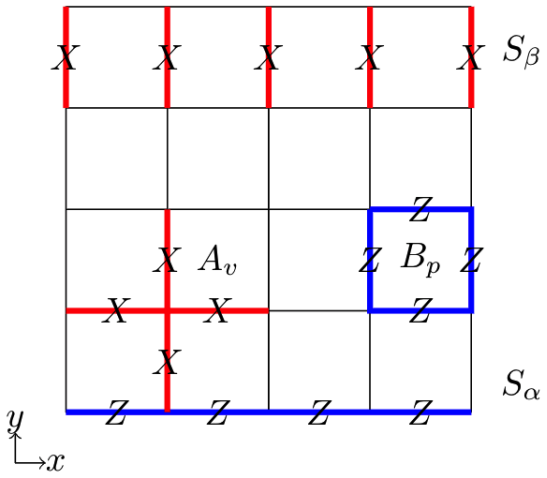
The *Logical Operators* are operators that takes the system from a code space basis to another code space basis. Essentially, the Logical Operators are operators in the set $\mathcal{N} - \mathcal{S}$, where $\mathcal{N}$ is the normalizer of $\mathcal{S}$ in the Pauli group $\mathcal{P}_n$ [16]. This notion describes the Logical

Operators $\bar{X}$ and $\bar{Z}$ where

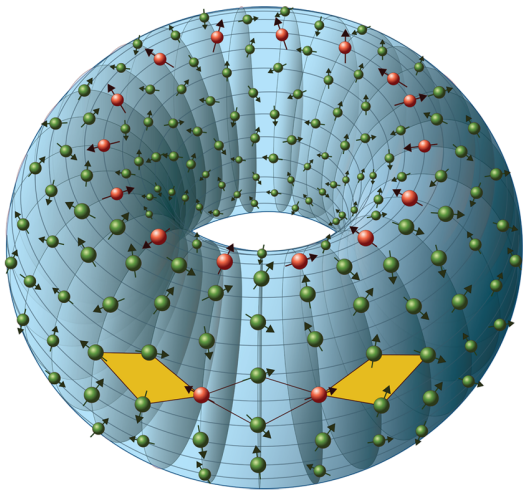
$$\bar{X}|0_L\rangle = |1_L\rangle, \quad \bar{X}|1_L\rangle = |0_L\rangle, \quad \bar{Z}|0_L\rangle = |0_L\rangle, \quad \bar{Z}|1_L\rangle = -|1_L\rangle. \tag{7}$$

As can be seen, the action of the Logical Operators on Logical Qubits are analogous to the action of Pauli Operators on qubits.

E. Toric Code



(a) The toric lattice. The qubits are laid out on the edges of the lattice. The Stabilizer Generators are star operators A_v and plaquette operators B_p. The Logical Operators are $\bar{Z} \equiv S_\beta$ and $\bar{X} \equiv S_\alpha$. Source: [19].



(b) The periodic boundary conditions of the lattice reflect the topology of a torus. Source: [20].

Figure 1: Visualizations of the Toric Code. (a) Toric lattice. (b) Torus layout.

The Toric Code is a stabilizer QECC defined on an $N \times N$ square lattice, where qubits are placed along the edges of the lattice (see Figure 1a). Its name, *toric* code, comes from the fact that the lattice is effectively wrapped onto the surface of a torus, with periodic boundary conditions that reflect the torus's topology (see Figure 1b).

The Stabilizer Generators of the Toric Code are: the star operators A_v, containing X operators acting on qubits surrounding certain lattice vertices; the plaquette operators B_p,

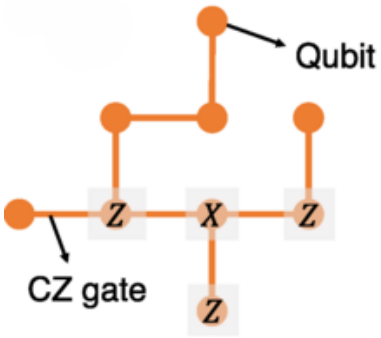


Figure 2: A quantum graph states: vertices are qubits in $|+\rangle$ states; edges represent CZ gates. A stabilizer is shown to be obtained from the graph. Source: [21].

containing Z operators acting on qubits surrounding certain lattice plaquettes.

The Logical Operators of the Toric Code include the $\bar{Z}$ operators, which consist of X operators acting along any double lattice paths that loop around the lattice, and the $\bar{X}$ operators, which consist of Z operators acting along any single lattice paths that loop around the lattice. This is any path that loops around the lattice is topologically equivalent to any other path that loops around the lattice. For a proof, please refer to Chapter 19 of [16]. There are two Logical Operators of each type, looping along different dimensions of the lattice (top-bottom and left-right), associated with two Logical Qubits. The Logical Operators shown in Figure 1a represent the simplest paths that loop around the lattice, and are associated with different Logical Qubits.

F. Graph representation of Stabilizer Codes

1. Graph states

Graphs are mathematical structures with vertices and edges used to visualize connectivity. A quantum state that can be represented by a graph is called a *graph state*. In the graph state, vertices represent qubits in the states $|+\rangle$, and edges represent CZ gates between qubits (see Figure 2).

Each graph state is exactly equivalent to a set of stabilizers [22]. The stabilizers $S^{(a)}$

associated with vertex a of a graph $G = (V,\ E)$ has the form

$$S^{(a)} = X_a \prod_{b \in neighbor(a)} Z_b \tag{8}$$

where $neighbor(a)$ is the set of neighbors of a, X_a is the Pauli X acting on a, and Z_b is the Pauli Z acting on b [22].

2. Local Clifford equivalence – local complementation

Not only is a graph directly equivalent to a stabilizer code, stabilizers can be *equivalent* to graph states as well [23–25]. Any stabilizer state can be transformed into a graph state under *local Clifford* (LC) operations (see Appendix A 0 c). The graphs obtained from these operations are not unique and are all *LC-equivalent* to one another.

One can obtain a graph state from the stabilizers by following these steps:

1. Represent all stabilizers as binary vectors of the form $(X|Z)$ (this is called *binary representation* or *symplectic representation*). [26]

2. Use row operations and column permutations to obtain a symplectic matrix of the form $(I|\Gamma)$ [23].

3. Read off Γ as an adjacency matrix for the graph state [24].

The LC-equivalence appears in the graph picture as *local complementation*. Two graph states are LC-equivalent if and only if the associated graphs are related by a sequence of local complementations [23]. The action of local complementation on a vertex is taking the *complement graph* of all neighbors of that vertex (see Figure 3). A complement graph is obtained by replacing all edges between the neighbors of the vertex with non-edges and vice versa.

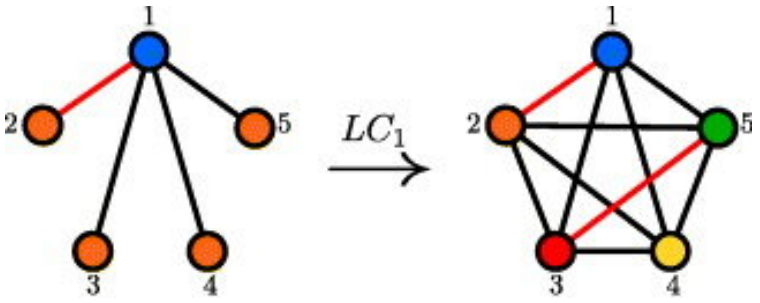


Figure 3: The effect of local complementation on vertex 1 of a graph. Source: [27].

III. METHODS

A. Finding the suitable graph

What is of interest is observing the topological properties of the Toric Code in its graph representation. This topological order is reflected by the periodicity of the qubits on the lattice. Specifically, the toric graph should remain invariant when the vertices are permuted in a periodic manner that corresponds to the periodicity of qubits on the lattice. For example, the 3×3 toric graph should be *automorphic* (exactly equal) to the graph obtained by permuting the vertices using the mappings $\{1 \to 7,\ 7 \to 13,\ 13 \to 1\}$ or $\{17 \to 15,\ 15 \to 13,\ 13 \to 17\}$.

To achieve this, we first constructed the graph for the Toric Code, as further demonstrated in Section III B. Then, we found the local complements of this graph that satisfied the provided vertex permutation conditions. To do this, a program was written in C to explore the orbit of local complementations for a given graph, as specified in Section III C.

B. Generating the 2×2 Toric graph

Table I: Stabilizer Generators of the 2×2 Toric Code. The indices correspond to the qubits in the lattice presented in Figure 4a.

Type	**Generators**
B_1	$Z_1 Z_2 Z_4 Z_5$
B_2	$Z_2 Z_3 Z_4 Z_7$
B_3	$Z_1 Z_5 Z_6 Z_8$
B_4	$Z_3 Z_6 Z_7 Z_8$
A_1	$X_1 X_2 X_3 X_6$
A_2	$X_1 X_3 X_4 X_8$
A_3	$X_2 X_5 X_6 X_7$
A_4	$X_4 X_5 X_7 X_8$

The Stabilizer Generators of the 2×2 Toric Code are given in Table I, where the indices correspond to the qubits in the lattice presented in Figure 4a. The symplectic representation of these Stabilizer Generators is given by the following matrix:

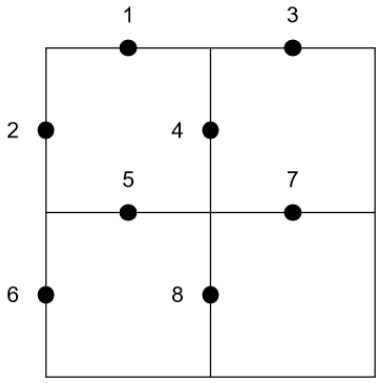
$$S = \begin{pmatrix} B_1 \\ B_2 \\ B_3 \\ A_1 \\ A_2 \\ A_3 \\ \bar{Z}_1 \\ \bar{Z}_2 \end{pmatrix} = \left(\begin{array}{cccccccc|cccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right). \quad (9)$$

By performing Gaussian elimination and column swaps (of qubits 4, 5, and 7), we can

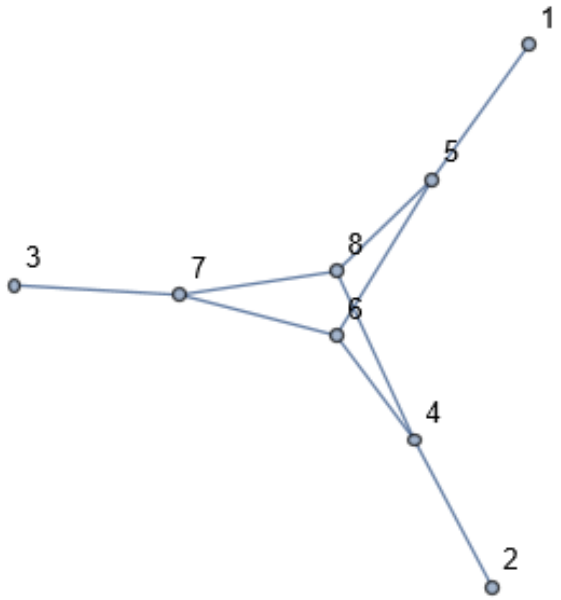
obtain the graph-state standard form

$$S' = (I|\Gamma) = \left(\begin{array}{cccccccc|cccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{array}\right). \quad (10)$$

The adjacency matrix Γ results in the graph showed in Figure 4b.



(a) The 2×2 toric lattice. The qubits are represented by the vertices of the graph, and the edges represent the Stabilizer Generators. The qubits are labelled from 1 to 8, and the Stabilizer Generators are labelled as A_i and B_i.



(b) The graph representation of the 2×2 Toric Code, obtained from the adjacency matrix Γ from Eq. 10.

Figure 4: Visualizations of the 2×2 Toric Code. (a) Toric lattice. (b) Graph representation.

C. Finding the local complements

The next task was to find the local complements of the graph we just found. Thus, I designed a program in C to do this [28]. The choice of the programming language was decided based on its fast runtime and reliable capability to allocate memory and debug errors. Another reason to choose C is because there is a well-developed package to work on graphs written by McKay and Piperno [29].

1. Queries and Functionality

The program asks for the number of vertices and edges of the user-given graph. Please note that *the vertex indices in the program starts from 0 instead of 1* so all the vertex labels obtained from the lattice must be offset by 1. It then asks if the user wants to check if they want to compare this graph to another graph. If the user chooses to do so, they enter the new graph and the program later confirms whether or not the two graphs are LC-equivalent. The program then asks the number of vertex permutations the user wants to check, and the specific vertex permutations. Finally, the program inquires the user for their preferred depth of the local complementation path.

After the queries, the program outputs the local complement that satisfies all vertex permutations and the vertex path that leads to it, if any was found. If the user has chosen the option to compare graphs, the program also gives the result of this. Finally, the program outputs the number of unique graphs that satisfy all the vertex permutations, up to isomorphism. If the number is greater than one, all unique graphs will be displayed.

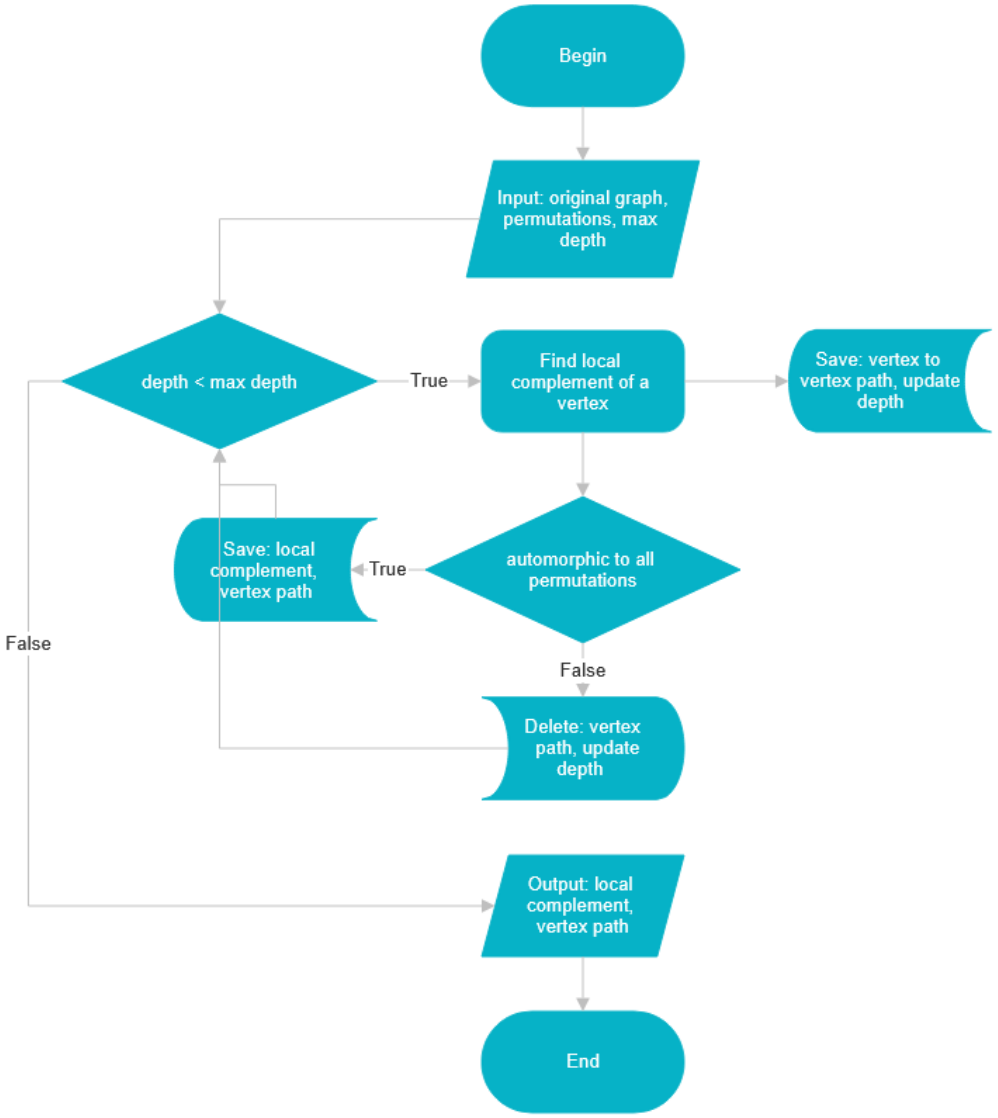
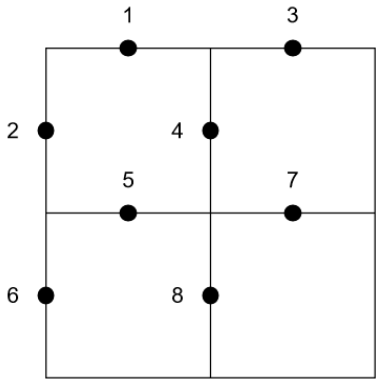


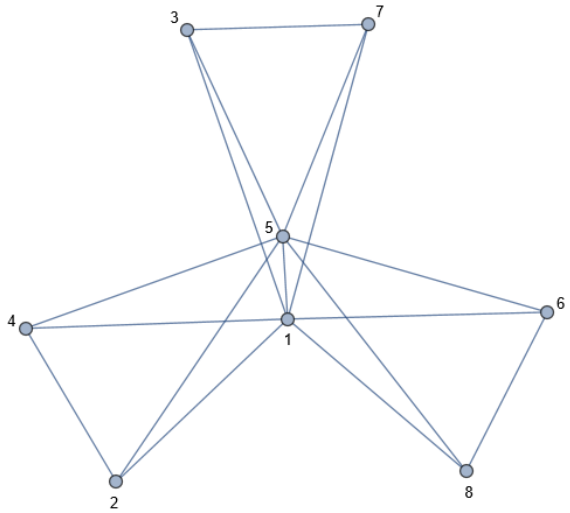
Figure 5: The flowchart showing the main logical flow of the program, not including the function to compare graphs and saving unique graphs for the sake of demonstration.

IV. RESULTS

A. The 2×2 Toric graph



(a) The 2×2 toric lattice, presented for the sake of demonstration.



(b) The graph representation of the 2×2 Toric Code, LC-equivalent to the graph in Figure 4b.

Figure 6: Visualizations of the 2×2 Toric Code. (a) Toric lattice. (b) Graph representation.

We chose to check two vertex permutations,

$$\{1 \leftrightarrow 5,\ 3 \leftrightarrow 7\} \tag{11}$$

and

$$\{2 \leftrightarrow 4,\ 6 \leftrightarrow 8\}. \tag{12}$$

This resulted in the graph in Figure 6b, which is unique for this choice of permutations, up to isomorphism. As can be seen, the graph is invariant to the qubit rearrangements that are related to the continuous transformation of the torus surface. This continuous transformation should not result in the change in quantum information that is encoded by the Toric Code, and it is a good sign that it does not change the graph state we generated. This indicates that the characteristic topological order of the Toric Code was successfully

captured by graph representation.

1. *Implications*

A natural question is *how do we extend this to a bigger lattice?* However, the program has computational complexity $O(n^d)$, where n is the number of vertices and d is the maximum depth of vertex traversal. Thus, even for a 3×2 lattice of 12 qubits/vertices, it would take at least 12^{12} operations to traverse all the possible local complements. Therefore, we decided that it is infeasible to find local complements this way for bigger lattices.

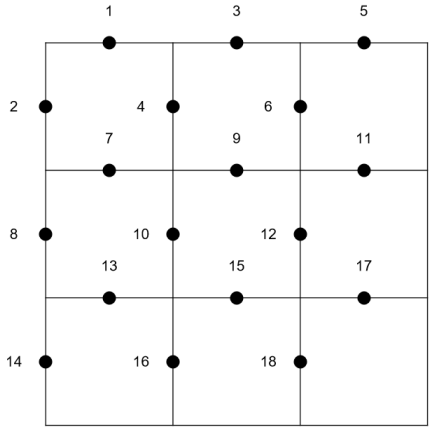
What we obtained was a proof of concept and an idea of how the graph of bigger lattices would look like. And we observed that the graph in Figure 6b has a 'windmill' structure, with the 'blades' containing qubits as logical $\bar{Z}$ operators, all connected to another Logical Operator $\{1,\ 5\}$ acting as the 'hub'. All the vertices in the same 'blades' and same 'hub' form a complete graph within themselves, and all the vertices in each 'blade' form a complete graph with all in the 'hub'. For the graphs that are isomorphic to this graph, they have a different 'hub' operator. This makes sense because all choice of Logical Operator are topologically equivalent to one another, as long as they form an incontractible loop around the lattice.

B. The 3×3 Toric graph

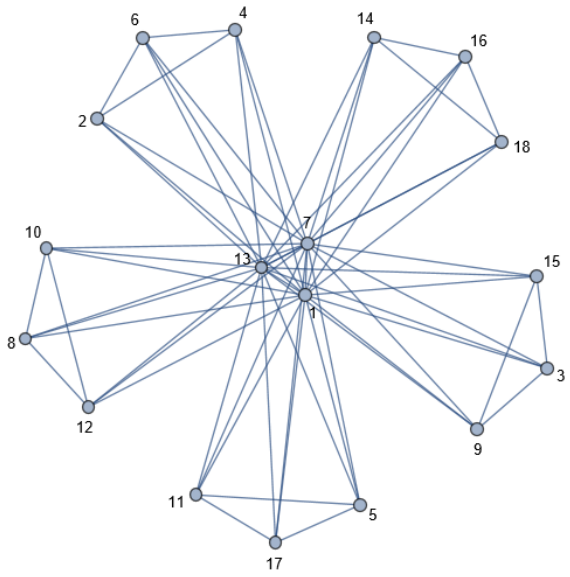
Using the insights from the 2×2 toric graph, we constructed the 3×3 graph, as displayed in Figure 7b. All the logical $\bar{Z}$ operators are identified and then connected to an arbitrary logical $\bar{Z}$, which in this case is $\{1,\ 7,\ 13\}$.

Using the programs provided by Mohsen Mehrani – my collaborator – we were able to deduce some error-correcting characteristic of this newly obtained graph. The code produced from this graph possesses 4 ground states, has code distance 3, and can be generated from 16 Stabilizer Generators. This aligns with the characteristic of the 3×3 Toric Code: 4 ground states correspond to 2 Logical Qubits; code distance 3; 16 Stabilizer Generators correspond

to 7 star operators, 7 plaquette operators, and 2 logical $\bar{Z}$ operators. Therefore, we believe this graph is equivalent to the Toric Code, despite not being able to exactly find the local complementation path between the two.



(a) The 3×3 toric lattice.



(b) The graph representation of the 3×3 Toric Code, with the 'windmill' structure.

Figure 7: Visualizations of the 3×3 Toric Code. (a) Toric lattice. (b) Graph representation.

V. CONCLUSION

In this research, we investigated the topological properties inherent to quantum error-correcting codes, particularly by analyzing the graph-state representation of the Toric Code. By converting Stabilizer Generators into adjacency matrices and systematically exploring local Clifford operations, we identified specific graph structures that effectively describe the essential topological features of the codes.

Through our detailed analysis of the 2×2 Toric Code, we discovered a unique 'windmill' graph configuration. This graph was shown to be invariant under certain periodic vertex permutations, clearly reflecting the Toric Code's underlying topological robustness. Building upon these insights, we successfully extended our methodology to construct a candidate

graph for the larger 3×3 lattice. This new graph closely mirrors the structural and coding properties characteristic of the original Toric Code, as verified through an error-correcting analysis.

Our results highlight the potential of using graph-theoretic techniques to visualize and deeply understand topological quantum codes. This approach not only enhances conceptual clarity but also provides practical tools for constructing scalable quantum error-correcting codes crucial for fault-tolerant quantum computing.

Future research opportunities include expanding this analysis to even larger lattice sizes, pursuing optimizations in graph-based QECC designs, and extending investigations to other homological codes (see Ref. [30]) that share similar topological orders.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisor, Dr. David Feder, for his guidance, support, and insightful discussions throughout my research. His expertise, patience, and encouragement have profoundly influenced my understanding and the quality of my work.

I am also deeply thankful to my collaborator, Mohsen Mehrani, whose contributions, engaging discussions, and generous sharing of knowledge were crucial to the success of this project. His idea of selecting permutations to analyze the 2×2 graph and the provision of essential tools for verifying the properties of graph states greatly enhanced our research outcomes.

Finally, I extend appreciation to my peers, family, and friends for their continuous encouragement and support for me during this project.

---

[1] P. Benioff, The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines, Journal of Statistical Physics **22**, 563 (1980).

[2] R. P. Feynman, Simulating physics with computers, International Journal of Theoretical Physics **21**, 467 (1982).

[3] D. Deutsch, Quantum theory, the church–turing principle and the universal quantum computer, Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences **400**, 97 (1985).

[4] P. W. Shor, Algorithms for quantum computation: discrete logarithms and factoring, in *Proceedings 35th annual symposium on foundations of computer science* (Ieee, 1994) pp. 124–134.

[5] L. K. Grover, A fast quantum mechanical algorithm for database search, in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing* (1996) pp. 212–219.

[6] C. H. Bennett and G. Brassard, Quantum cryptography: Public key distribution and coin tossing, in *Proceedings of the IEEE International Conference on Computers, Systems, and Signal Processing* (IEEE, Bangalore, India, 1984) pp. 175–179.

[7] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, Quantum computation by adiabatic evolution, arXiv preprint quant-ph/0001106 (2000).

[8] A. Barenco, Quantum physics and computers, Contemporary physics **37**, 375 (1996).

[9] P. W. Shor, Scheme for reducing decoherence in quantum computer memory, Physical review A **52**, R2493 (1995).

[10] A. R. Calderbank and P. W. Shor, Good quantum error-correcting codes exist, Physical Review A **54**, 1098 (1996).

[11] A. Steane, Multiple-particle interference and quantum error correction, Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences **452**, 2551 (1996).

[12] D. Gottesman, Class of quantum error-correcting codes saturating the quantum hamming bound, Physical Review A **54**, 1862 (1996).

[13] A. Kitaev, Fault-tolerant quantum computation by anyons, Annals of Physics **303**, 2 (2003).

[14] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Surface codes: Towards practical large-scale quantum computation, Physical Review A—Atomic, Molecular, and Optical Physics **86**, 032324 (2012).

[15] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information* (Cambridge university press, 2010).

[16] D. A. Lidar and T. A. Brun, *Quantum error correction* (Cambridge university press, 2013).

[17] W. K. Wootters and W. H. Zurek, A single quantum cannot be cloned, Nature **299**, 802 (1982).

[18] D. Dummit and R. Foote, *Abstract Algebra* (Wiley, 2003).

[19] P. Liao and D. L. Feder, Graph-state representation of the toric code, Physical Review A **104**, 10.1103/physreva.104.012432 (2021).

[20] D. Gottesman, Keeping one step ahead of errors, Physics **5**, 50 (2012).

[21] S.-H. Lee and H. Jeong, Universal hardware-efficient topological measurement-based quantum computation via color-code-based cluster states, Physical Review Research **4**, 10.1103/physrevresearch.4.013010 (2022).

[22] S. Anders and H. J. Briegel, Fast simulation of stabilizer circuits using a graph-state representation, Physical Review A—Atomic, Molecular, and Optical Physics **73**, 022334 (2006).

[23] M. Van den Nest, J. Dehaene, and B. De Moor, Graphical description of the action of local clifford transformations on graph states, Physical Review A **69**, 10.1103/physreva.69.022316 (2004).

[24] M. Grassl, A. Klappenecker, and M. Rotteler, Proceedings of the ieee international symposium on information theory (2002).

[25] D. Schlingemann, Stabilizer codes can be realized as graph codes, arXiv preprint quant-ph/0111080 (2001).

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

[27] M. Hajdusek and M. Murao, Direct evaluation of pure graph state entanglement, New Journal of Physics **15** (2013).

[28] H. Nguyen, Github: Phys598-honoursthesis-code (2025), accessed: 2025-04-09.

[29] B. D. McKay and A. Piperno, Practical graph isomorphism, ii, Journal of symbolic computation **60**, 94 (2014).

[30] N. Delfosse, P. Iyer, and D. Poulin, Generalized surface codes and packing of logical qubits (2016), arXiv:1606.07116 [quant-ph].

APPENDICES

Appendix A: Some notable quantum gates

a. Hadamard Gate

The Hadamard gate is a single-qubit gate that is defined as:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{A1}$$

The Hadamard gate takes the computational basis states to the $|+\rangle$ and $|-\rangle$ states and vice versa.

b. Controlled-Z gate

The Controlled-Z (CZ) gate is a two-qubit gate that applies a Z operation to the target qubit if the control qubit is in the state $|1\rangle$. Its matrix representation is given by:

$$CZ = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}.$$

The CZ gate is symmetric and commutative, meaning the roles of the control and target qubits are equal. It is a crucial component in constructing graph states, as it introduces entanglement between qubits connected by an edge in the graph.

c. Clifford gates

The Clifford group $\mathcal{C}_n$ is a normalizer of the Pauli group $\mathcal{P}_n$. The Clifford gates are elements of the Clifford group and are generated by the phase gate (H), the Hadamard gate (H), and the CX (CNOT) gate or CZ gate.

The Phase gate (S) is defined as:

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}.$$

The Controlled-NOT (CNOT) gate is a two-qubit gate that flips the target qubit if the control qubit is in the state $|1\rangle$. Its matrix representation is:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$