Equivalence of color codes and surface codes

A Project Report

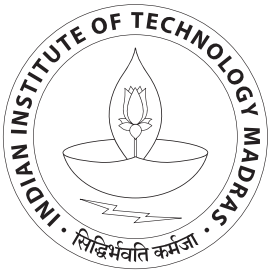
submitted by

ARJUN NITIN BHAGOJI

in partial fulfilment of the requirements
for the award of the degree of

MASTER OF TECHNOLOGY
and
BACHELOR OF TECHNOLOGY

under the guidance of
Dr. Pradeep Sarvepalli *and* ***Dr. Andrew Thangaraj***



DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.
June 2015

THESIS CERTIFICATE

This is to certify that the thesis entitled **Equivalence of color codes and surface codes**, submitted by **Arjun Nitin Bhagoji**, to the Indian Institute of Technology, Madras, for the award of the degree of **Master of Technology** and **Bachelor of Technology**, is a bona fide record of the research work carried out by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Pradeep Sarvepalli
Research Guide
Assistant Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Andrew Thangaraj
Research Co-Guide
Associate Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Place: Chennai
Date:22/06/2015

ACKNOWLEDGEMENTS

I would like to thank my guide, Dr. Pradeep Sarvepalli and my co-guide Dr. Andrew Thangaraj for his invlauable help and guidance over the course of this project. I am indebted to the many excellent professors who have taught me over the course of my five years here at IIT Madras. In particular, I have been inspired by and learnt the most from Dr. V. Balakrishnan, Dr. Krishna Jagannathan, Dr. Radha Krishna Ganti and Dr. L. Sriramkumar. I would also like to thank Dr. Prabha Mandayam for all her help and advice over the past year, both technical and non-technical. I benefited from multiple discussions with Arun Aloshious, a Ph.D student of Dr. Pradeep's.

ABSTRACT

In a recent work, Bombin, Duclos-Cianci, and Poulin showed that every local translationally invariant 2D topological stabilizer code is locally equivalent to a finite number of copies of Kitaev's toric code. In this thesis, we focus on color codes and relax the constraint on translation invariance. We show that any color code can be mapped to exactly two copies of a related surface code. The surface code in our map is induced by the color code and easily derived from the color code. Furthermore, our map does not require any ancilla qubits for the surface codes. We also indicate the various degrees of freedom in constructing the map and the consequent variations. We derive an algorithm to map an arbitrary color code to a pair of surface codes and demonstrate how this mapping can be used with various examples and simulations.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS i

ABSTRACT ii

LIST OF TABLES v

LIST OF FIGURES vii

1 Introduction 1

2 Preliminaries 3

2.1 Quantum error models . . . 3

2.1.1 Bit and phase flip channels . . . 3

2.1.2 Depolarizing channel . . . 4

2.2 Quantum Codes . . . 4

2.2.1 Syndromes and error correction . . . 4

2.2.2 Stabilizer formalism . . . 5

3 Topological Codes 7

3.1 Kitaev's toric code . . . 7

3.2 Surface codes . . . 11

3.3 Color codes . . . 13

4 Equivalence between color codes and surface codes 14

4.1 Deducing the map—A linear algebraic approach . . . 15

4.2 Comparison with other maps . . . 24

5 Examples 26

5.1 Square-octagonal lattice . . . 26

5.2 Hexagonal lattice 30

6 Error models and simulations **33**

6.1 Analytic error probabilities 33

6.1.1 Marginal errors: hexagonal lattice 34

6.1.2 Exact single qubit errors: hexagonal lattice 34

6.1.3 Marginal errors: square octagonal lattice 37

6.2 Decoding procedure 38

6.2.1 Simulations 38

LIST OF TABLES

4.1 Comparison with previous work 24

LIST OF FIGURES

3.1 X-type stabilizer on the vertices and Z-type stabilizer on the plaquettes . . 8

3.2 **Cycle on the left**: Homologically trivial cycle. The two syndromes shown could have arisen from either the top chain or bottom chain, hence the ambiguity in the syndrome. If the top chain was the original error and the bottom one our estimate, the encoded state is undamaged by this error. **Cycle on the right**: Homologically non-trivial cycle. The cycle winds around the torus and is clearly not the boundary of any chain of plaquettes. It commutes with all the stabilizers but is not a product of them. In fact, it is the product of an encoded or logical Z operation and a plaquette stablilizer. 9

3.3 The basis for the 4 logical operators that act on the encoded qubits. 10

4.1 Illustrating the contraction of a color code via τ_c and the resultant surface code. Only portions of the codes are shown. The c-colored faces are vertices in $\tau_c(\Gamma)$. The faces $f \notin \mathsf{F}_c(\Gamma)$ remain faces in $\tau_c(\Gamma)$ and are also labeled f in $\tau_c(\Gamma)$, while the c-colored edge $e = (u, v)$ in Γ is mapped to an edge in $\tau_c(\Gamma)$, so we retain the label e. Every vertex in Γ is incident on a unique c-colored edge, so we can also extend τ_c to vertices u, v and edges unambiguously by defining $\tau_c(u) = \tau_c(v) = \tau_c(u, v) = e$. 16

4.2 Circled hopping operator is the dependent operator. In addition to 17

4.3 Mapping the independent hopping operators $H^{\epsilon_r}_{u,v} = H^{\epsilon_r}_{f_1 \leftrightarrow f_2} = Z_u Z_v$ and $H^{\epsilon_b}_{u',v'} = H^{\epsilon_b}_{f \leftrightarrow f'} = Z_{u'} Z_{v'}$ on Γ onto two copies of $\tau(\Gamma)$ i.e. Γ_1 and Γ_2; $\pi(H^{\epsilon_r}_{f_1 \leftrightarrow f_2}) = [Z_{\tau(u)}]_1$ acts only on Γ_1 while $H^{\epsilon_b}_{f \leftrightarrow f'} = [X_{\tau(u')} X_{\tau(v')}]_2$ acts only on Γ_2. 19

5.1 Color-coded mapping of independent hopping operators from the color code to the two copies of the toric code. Red: $H^{\mu_{c'}}_{1,8}$ is mapped to $[X_{\tau(1)} X_{\tau(8)}]_1$; blue: $H^{\epsilon_c}_{5,6}$ is mapped to $[Z_{\tau(5)}]_1 = [Z_{\tau(6)}]_1$; green: $H^{\mu_c}_{3,4}$ is mapped to $[Z_{\tau(3)}]_2 = [Z_{\tau(4)}]_2$; pink: $H^{\epsilon_{c'}}_{16,9}$ is mapped to $[X_{\tau(9)} X_{\tau(16)}]_2$. 27

5.2 Mapping for X type errors on $G1$, starting with X_1, which is the single qubit X operator chosen according to Lemma 4. The dependent hopping operator is $H^{\mu_{c'}}_{2,3}$, which clearly maps to a Z stabilizer on $[\tau(G1)]_2$, as in Eq. 4.10. . . . 28

5.3 Mapping for Z type errors on $G1$, starting with Z_2, which is the single qubit Z operator chosen according to Lemma 4. The dependent hopping operator is $H^{\epsilon_{c'}}_{1,8}$, which clearly maps to a Z stabilizer on $[\tau(G1)]_1$, as in Eq. 4.9. 29

5.4 Color-coded mapping of independent hopping operators from the color code to the two copies of the surface code. Red: $H^{\mu_{c'}}_{2,3}$ is mapped to $[X_{\tau(2)}X_{\tau(3)}]_1$; blue: $H^{\epsilon_c}_{5,6}$ is mapped to $[Z_{\tau(5)}]_1 = [Z_{\tau(6)}]_1$; green: $H^{\mu_c}_{3,4}$ is mapped to $[Z_{\tau(3)}]_2 = [Z_{\tau(4)}]_2$; pink: $H^{\epsilon_{c'}}_{6,1}$ is mapped to $[X_{\tau(6)}X_{\tau(1)}]_2$. 30

5.5 Mapping for X type errors on $G1$, starting with X_2, which is the single qubit X operator chosen according to Lemma 4. The dependent hopping operator is $H^{\mu_{c'}}_{6,1}$, which clearly maps to a Z stabilizer on $[\tau(G1)]_2$, as in Eq. 4.10. . . . 31

5.6 Mapping for Z type errors on $G1$, starting with Z_1, which is the single qubit Z operator chosen according to Lemma 4. The dependent hopping operator is $H^{\epsilon_{c'}}_{2,3}$, which clearly maps to a Z stabilizer on $[\tau(G1)]_1$, as in Eq. 4.9. 32

6.1 One plaquette of the hexagonal color code mapped to two copies of the triangular surface code. Note the labeling of qubits used to derive the error probabilities. 34

6.2 One plaquette of the square-octagonal color code mapped to two copies of Kitaev's toric code. Note the labeling of qubits used to derive the error probabilities. 36

6.3 Phase decoding performance of standard matching decoding with uniform error probabilities for toric code lattices of size 2.2^m. Threshold obtained is approximately 11%. 38

CHAPTER 1

Introduction

From Richard Feynman's idea in 1982 of a computer based on quantum mechanical principles, to David Deutsch's notion of a universal quantum computer developed later in the 1980s and Peter Shor's remarkable 1994 proofs that both integer factorization and the 'discrete logarithm' could be solved in an efficient manner, quantum computing has had a short but eventful history [10]. Of fundamental importance in quantum computing are questions of both scope and implementation. We seek to understand which problems can be solved in less time and space using quantum computers and how to develop quantum algorithms for these problems. To have widely available quantum computers which can be used to run these algorithms on a day-to-day basis, we need to look at issues of implementation. Quantum computing implementations suffer from various crippling problems at the moment such as decoherence and the lack of scalability. Many groups around the world are working on these problems and some small-scale demonstrations have been made, with some critical breakthroughs made just this year (2015) using an implementation of quantum computers that is the central topic of this thesis.

Apart from computing, the other great technical advance of the modern age is that of communication. Starting with Claude E. Shannon's remarkable papers in 1948, great advances have been made in the field of communication. Many good error-correcting codes have been constructed to try and achieve the reliable transmission of information through a noisy channel. In 1995, Ben Schumacher proved the quantum analog of one of Shannon's theorems and since then, much work has gone into developing quantum error-correcting codes. First, the eponymous Calderbank-Shor-Steane codes and later stabilizer codes, discovered independently by Calderbank, Shor, Steane and Rains, and Gottesman, have provided us with evidence that good quantum codes exist to help protect the information stored in quantum bits or qubits. Quantum codes help in both the protection of qubits used for computation as well as for the transmission of information.

In 1997, Kitaev [7] came up with a new approach to quantum computing known as topological quantum computing. In a topological quantum computer, quasiparticles known as anyons are braided to form quantum gates. These quantum computers are much more stable than the ones made using the standard approach of trapped particles as the information for these computers is stored in their topological degrees of freedom which remain unaffected by local perturbations of the qubits. There are various types of topological codes, such as surface codes and color codes, and they have different properties which have an influence on their suitabil-

ity for use in a practical quantum computer. Recently, scientists at IBM [3] , implemented a surface code based quantum computer and performed error-correction on it.

In this thesis, we study the relation between two types of topological codes, namely, surface codes and color codes. As we shall see in subsequent chapters, they each have their advantages and disadvantages, and relating the two paves the way towards realizing quantum computers which have robust yet efficient error-correcting schemes.

CHAPTER 2

Preliminaries

In this chapter, we will take a brief look at some ideas and concepts essential to the rest of the thesis. We start by studying quantum codes and their connection to classical codes. Then, we move on to an important class of quantum codes known as stabilizer codes. The decoding of quantum codes is then explained in some detail, with various standard methods presented. Finally, a short introduction to quantum circuits and belief propagation is given.

2.1 Quantum error models

Noise is as omnipresent in quantum information processing systems as it is in classical ones. The effect of noise in a quantum system is to change the state of qubits, which are being used for the transmission of information or to perform computations, in undesirable ways.

2.1.1 Bit and phase flip channels

The most common quantum noise model is the *bit flip channel*, which is very similar to its classical counterpart, the *binary symmetric channel*. The binary symmetric channel flips the transmitted bit with a probability $p > 0$ and leaves it unchanged with probability $1 - p$. Similarly, the bit flip channel changes the state of a qubit from $|0\rangle$ to $|1\rangle$ and vice versa with probability p. The operators acting on a single qubit in the bit flip channel are:

$$E_{b0} = \sqrt{1-p}I \qquad \& \qquad E_{b1} = \sqrt{p}X. \tag{2.1}$$

A quantum error channel that has no classical analog is the *phase flip channel*. With probability p, it changes the phase of qubit from positive to negative or vice versa. It leaves the phase unchanged with probability $1 - p$. The operators that act on a single qubit in a phase flip channel are:

$$E_{p0} = \sqrt{1-p}I \qquad \& \qquad E_{p1} = \sqrt{p}Z. \tag{2.2}$$

2.1.2 Depolarizing channel

Consider a quantum system in a state ρ. In a *depolarizing channel*, this state ρ is replaced by the completely mixed state $I/2$ with probability p, which is the probability that it is depolarized. The state is left unchanged with probability $1-p$ and so the state of the system after passing through a depolarizing channel is

$$\mathcal{E}(\rho) = \frac{pI}{2} + (1-p)\rho. \tag{2.3}$$

Now, we observe that we can write $\frac{I}{2}$ as follows:

$$\frac{I}{2} = \frac{\rho + X\rho X + Y\rho Y + Z\rho Z}{4} \tag{2.4}$$

and substitute it back into Eq. 2.3 to get

$$\rho = (1 - \frac{3p}{4})\rho + \frac{p}{4}(X\rho X + Y\rho Y + Z\rho Z) \tag{2.5}$$

which can be parametrized using $p' = \frac{3p}{4}$ so we can interpret the depolarizing as acting on the state with each of X, Y and Z with probability $\frac{p'}{3}$ and leaving the state unchanged with probability $1-p'$:

$$\rho = (1-p')\rho + \frac{p'}{3}(X\rho X + Y\rho Y + Z\rho Z). \tag{2.6}$$

2.2 Quantum Codes

The ideas related to quantum error-correcting codes that are necessary and relevant for the remainder of the thesis are explained using the example of the 3-qubit bit flip code, which is quite similar to the 3-bit repetition code used in classical error correction.

2.2.1 Syndromes and error correction

Consider the bit flip channel with probability p of flipping a qubit (as defined earlier) acting on a qubit passing through it. Let the initial state be $|\psi\rangle = a|0\rangle + b|1\rangle$, where $a, b \in \mathbb{C}$. With probability p, the Pauli operator X acts on this state and flips $|0\rangle$ to $|1\rangle$ and vice versa. Without any encoding of this state, the error probability is just p. To decrease the probability of error, we can now encode $|0\rangle$ as $|000\rangle$ and $|1\rangle$ as $|111\rangle$. We denote $|000\rangle$ as $|0_L\rangle$, as a logical $|0\rangle$ and

$|111\rangle$ as $|1_L\rangle$, as a logical $|1\rangle$. Now, the state $|\psi\rangle$ is encoded as

$$|\psi\rangle \rightarrow a|000\rangle + b|111\rangle. \tag{2.7}$$

Now suppose the encoded state is passed through the bit flip channel, and the channel acts independently on each of the 3 qubits in the encoded state. Then, we can measure the syndrome of the state after it has passed through the channel using the following projection operators:

$$P_0 = |000\rangle\langle 000| + |111\rangle\langle 111|, \tag{2.8}$$
$$P_1 = |100\rangle\langle 100| + |011\rangle\langle 011|, \tag{2.9}$$
$$P_2 = |010\rangle\langle 010| + |101\rangle\langle 101|, \tag{2.10}$$
$$P_3 = |001\rangle\langle 001| + |110\rangle\langle 110|. \tag{2.11}$$

P_0 corresponds to no errors while each P_i corresponds to a bit flip error on qubit i. We get the syndrome by performing the operation $\langle\psi||P_i||\psi\rangle$, and depending on the syndrome that is obtained after measurement, the necessary error correction steps are performed. If there is no error, nothing is done to the outut. Otherwise, depending on which qubit there was an error on, an X operator is applied on that again in order to correct it. Note that as long as only single qubit bit-flip errors occur on the encoded state, this procedure to correct errors work. If there are two or more bit-flip errors, then we will not be able to recover the initial state perfectly. Thus, the probability of error we get using this code is $3p^2(1-p) + p^3 = 3p^2 - 2p^3$, which is an improvement over the error p obtained without any coding.

2.2.2 Stabilizer formalism

Consider again the 3-qubit bit flip code from the previous section. The logical qubits that make up the encoded state, $|000\rangle$ and $|111\rangle$ are left unaffected by Z_1Z_2 and Z_2Z_3. $|000\rangle$ and $|111\rangle$ are the unique states that are left unaffected these operators, i.e. they are *stabilized* by these operators. These generate the stabilizer group for this particular code and are known as the stabilizer generators. The three qubit bit flip code, as we have seen, can perfectly correct all single qubit bit flip errors, i.e. it can detect and correct errors in the set $E = \{I, X_1, X_2, X_3\}$. The stabilizer generators Z_1Z_2 and Z_2Z_3 anti-commute with every with every possible product of two operators from the set E.

Also, we see that we perform syndrome measurements using the stabilizers. For example, if the error X_1 occurred, it will anti-commute with the first stabilizer and commute with the second, giving the syndrome -1 and 1. Similarly, each of the other single qubit X errors will give a unique syndrome, and so we can detect and correct all these errors perfectly, mirroring

our earlier analysis without stabilizers.

The stabilizer formalism gives us a clean and elegant representation of quantum codes and can be used even to describe quantum gates. The analysis done above can be simply extended to other, more complex codes.

CHAPTER 3

Topological Codes

One of the essential requirements for any quantum computing system is fault tolerance. In quantum circuits, qubits interact with each other through various means and if qubit is in error, then the qubit it is interacting with is also likely to become erroneous. For example, in a CNOT gate if the control qubit is in error, it is likely that its error will spread to the target qubit. In this manner, there may be a catastrophic propagation of errors leading to a failure in the computation. To avoid this and have fault-tolerant quantum computation, we can replace each physical qubit with an encoded qubit and act on these encoded qubits with encoded gates, after each of which we can perform error correction. The encoded gates that are used should not allow errors to propagate to a large number of qubits, i.e. they should be 'local' in a sense. It has been shown that an universal set (the Hadamard, phase, CNOT and $\frac{\pi}{8}$ gates) of encoded gates can be implemented in a fault tolerant manner. An important proerty of gates which automatically makes them fault-tolerant is tranversality- that is, if the gate can be implemented in a bitwise manner, it is called a transversal gate. Looking for transversality while designing quantum gates gives us a general guiding principle by which to construct fault-tolerant circuits.

It is in this context that topological quantum codes, first introduced by Kitaev [7], are seen as a promising approach towards a practical quantum computer. Topological quantum codes, by the nature of their arrangement of qubits in a lattice, allow for local syndrome extraction and thus fault tolerant quantum computation. Topological codes are a subset of stabilizer codes, with the stabilizers typically located on lattice vertices and faces. We will see that in topological codes, the encoded information is stored in global degrees of freedom which are not affected by local errors. Gates can be designed to act on the information contained in these global degrees of freedom, which will make them fault tolerant towards local errors, which can be easily corrected without affecting the state of the quantum system. Topological codes can be constructed and studied in dimensions greater than 2, but we will focus on two-dimensional topological codes. It is instructive to study the code originally proposed by Kitaev in [7] before looking at other topological code constructions.

3.1 Kitaev's toric code

In Kitaev's toric code, the physical qubits can be understood to be located on the edges of a square lattice embedded on a torus, i.e. a square with its opposite sides identified. Without

loss of generality, the X-type stabilizers are taken to reside on the vertices of the lattice and the Z-type stabilizers on the faces or plaquettes of the lattice. Each X-type stabilizer is a tensor product of the 4 X-type operators at that vertex (and identity on all other qubits) and each Z-type stabilizer is a tensor product of the 4 Z-type operators surrounding the plaquette (and identity on all other qubits). Even though X and Z operators anti-commute, all the stabilizers commute with each other since they always cross one another an even number of times. If

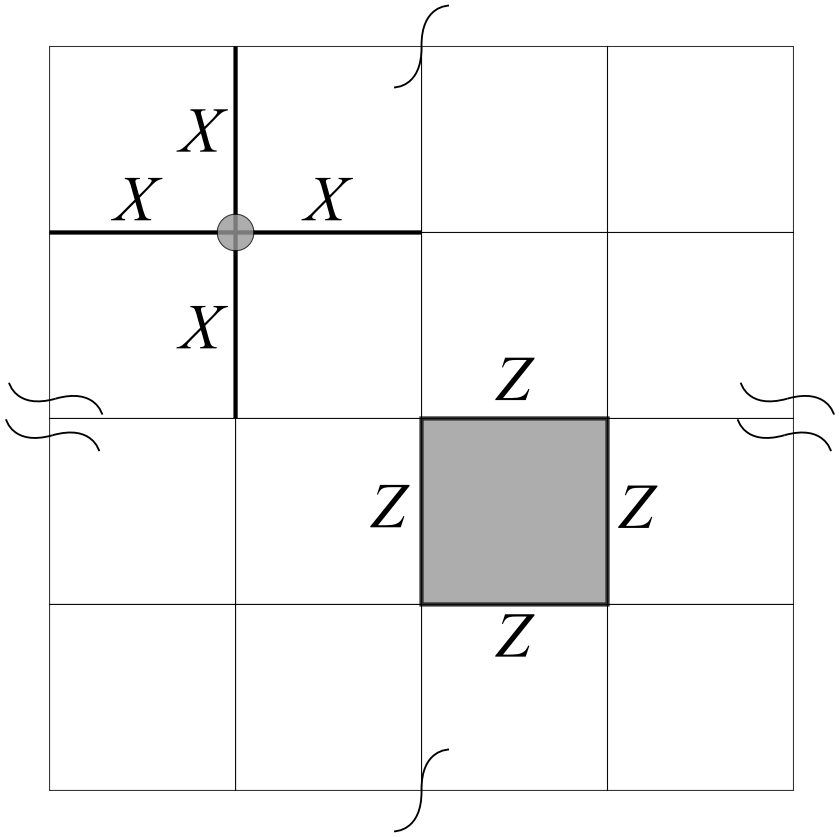


Figure 3.1: X-type stabilizer on the vertices and Z-type stabilizer on the plaquettes

the qubits are embedded on a $l \times l$ lattice, then there are $2l^2$ physical qubits. Also, due to the periodicity of the toric code lattice, there are $l^2 - 1$ independent stabilizers of each type, X and Z, so the number of encoded qubits is

$$n - g = 2l^2 - (2(l^2 - 1)) = 2. \tag{3.1}$$

Without loss of generality and for clarity of exposition, we will assume errors come only from the phase flip channel, i.e. Z-type errors. An identical analysis can be perfomed with X-type errors and combinations of X and Z type errors. Suppose there is a single Z-type error on the toric code. Then, it will affect the two X-type stabilizers residing at the vertices which the edge in error connects and those will take on the value -1. We can also think of a charge residing at those two points in the lattice, created due to the Z-type error. If there is a chain

(sequence of edges) of Z-type errors, then the charge will only reside at the vertices at the two ends of the chain, i.e. only those stabilizers will have a -1 value associated with them. In the charge picture, X and Z-type errors create different types of charges, which are localized on the plaquettes and vertices respectively.

The syndromes that we obtain from the toric code are highly ambiguous due to the fact that multiple chains can begin and end at the same vertices. If the cycle formed by the error and its estimate is a homologically trivial cycle, i.e. it is the boundary of a chain of plaquettes, then the encoded state is not affected by that error. The homologically trivial cycle that we obtain is in fact a product of stabilizers and thus does not affect the code. On the other hand, if the cycle formed is a homologically non-trivial cycle, i.e. it is not the boundary of any chain of plaquettes and it winds around the torus, then the encoded state is affected by that error. Although the non-trivial cycle commutes with all the stabilizers, it is not a product of them.

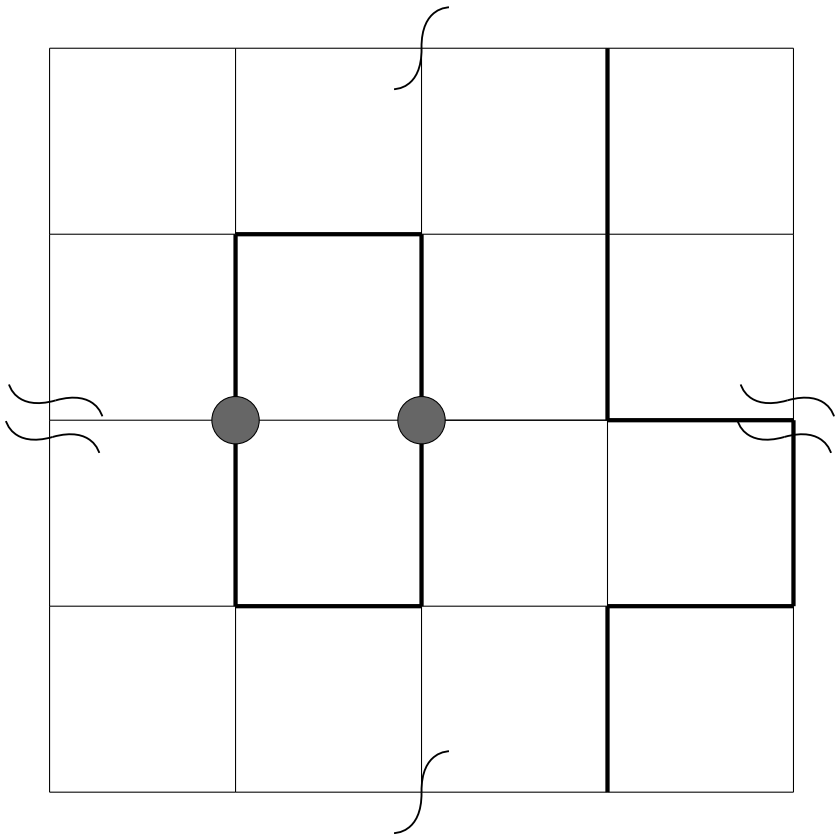


Figure 3.2: **Cycle on the left**: Homologically trivial cycle. The two syndromes shown could have arisen from either the top chain or bottom chain, hence the ambiguity in the syndrome. If the top chain was the original error and the bottom one our estimate, the encoded state is undamaged by this error.
Cycle on the right: Homologically non-trivial cycle. The cycle winds around the torus and is clearly not the boundary of any chain of plaquettes. It commutes with all the stabilizers but is not a product of them. In fact, it is the product of an encoded or logical Z operation and a plaquette stablilizer.

Thinking in terms of charges, in a homologically trivial cycle, a charge is created, moved in a cycle that does not wind around the torus and brought back to annihilate with the other charge comprising the initially formed pair. In this case, the quantum system remains undamaged. However, if the charge winds around the torus before annihilating its paired charge, then the system is damaged. From this, we can clearly see that an encoded or logical Z operator is formed by a non-trivial cycle of Z-type errors winding around any of the two cycles of the torus. Thus, there are two logical Z operators, $Z1'$ and $Z2'$ on the primary lattice and two logical X operators, $X1'$ and $X2'$ that come from non-trivial cycles on the dual lattice. The dual lattice is the lattice formed by interchanging the positions of vertices and plaquettes on the primary lattice. i.e. we place plaquettes with their centers at the vertices of the primary lattice and shrink plaquettes to get the dual lattice vertices.

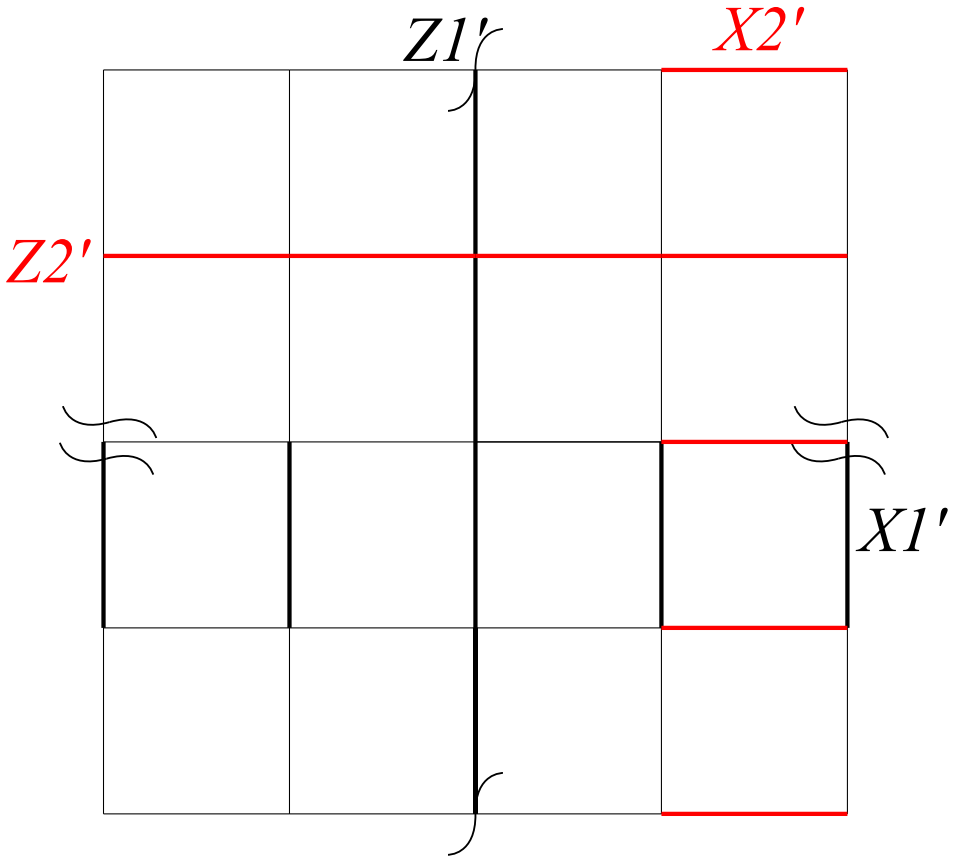


Figure 3.3: The basis for the 4 logical operators that act on the encoded qubits.

The weight of Pauli operator is defined as the number of qubits on which it acts with a non-trivial single qubit Pauli operator. Then, the distance of a stabilizer code is the weight of the Pauli operator with minimum weight that affects the encoded qubits but still commutes with all the stabilizers. Thus, for a toric code on a $l \times l$ lattice, the code distance is l. We see that we can increase the code distance by just increasing the size of the lattice, which in turn increases the number of physical qubits. However, increasing the size of the lattice does not affect the

number of encoded qubits.

The qubits of the square lattice can also be placed on planar topology with a boundary as opposed to a toriodal one without boundaries to give a topological code. In the case of the *planar code*, however, not all the stabilizers are the same. Those at the boundary behave differently from those in the interior of the lattice. Further generalizations of the toric code are discussed in subsequent sections.

3.2 Surface codes

Kitaev's toric code is an example of a more general class of codes known as *surface codes*. The main idea of surface codes, like the toric code, is to store information in homological or global degrees of freedom. Surface codes are not restricted to a lattice or surface of a particular type. In general, the surfaces that are chosen for embedding the lattice are closed, connected and orientable. We choose closed surfaces because typically we only have a finite number of qubits in the code and a closed surface suffices. Orientable (having an inside and outside) and connected (we can move from one point to another without jumps) surfaces are chosen for simplicity's sake, in order to avoid surfaces with odd, non-intuitive topologies.

We use the following notation: the Pauli group on n qubits is denoted $\mathcal{P}_n$. We denote the vertices of a graph Γ by $\mathsf{V}(\Gamma)$, and the edges by $\mathsf{E}(\Gamma)$. The set of edges incident on a vertex v is denoted as $\delta(v)$ and the edges in the boundary of a face by $\partial(f)$. Assuming that Γ is embedded on a suitable surface we use $\mathsf{F}(\Gamma)$ to denote the faces of the embedding and do not always make explicit reference to the surface.

A surface code on a graph Γ is a stabilizer code where the qubits are placed on the edges of Γ and whose stabilizer S is given by

$$S = \langle A_v, B_f \mid v \in \mathsf{V}(\Gamma), f \in \mathsf{F}(\Gamma) \rangle, \tag{3.2}$$

where $A_v = \prod_{e \in \delta(v)} X_e$ and $B_f = \prod_{e \in \partial(f)} Z_e$. The Pauli group on the qubits of a surface code is denoted as $\mathcal{P}_{\mathsf{E}(\Gamma)}$. Now, these stabilizer generator are subject to the following constraints:

$$\prod_{v \in \mathsf{V}} A_v = 1 \qquad \& \qquad \prod_{f \in \mathsf{F}} B_f = 1, \tag{3.3}$$

so not all of them are independent. Denoting the number of edges, vertices and faces by E, V and F respectively, from Eq. 3.3 we get that there are $V - F + 2$ independent generators. We know that the number of encoded qubits in a stabilizer code is just the number of physical qubits minus the number of independent generators, so for the surface code we have $k =$

$E-(V+F-2)$ encoded qubits. The quantity $V+F-E$ is known as the *Euler characteristic* of the surface and it is equal to $2(1-g)$ for closed orientable surfaces, where g is the genus of the surface. The genus of a surface can be understood as the number of handles it has, for example, a coffee mug has genus 1 while a sphere has genus 0. This gives us the number of encoded qubits for a surface code as

$$k = E - V - F - 2 = 2 - 2(1-g) = 2g. \tag{3.4}$$

There are four types of topological charges on a surface code: i) electric charge (denoted ϵ) localized on the vertices, ii) magnetic charge (denoted μ) living on the plaquettes, iii) the composite electric and magnetic charge denoted $\epsilon\mu$ which resides on both the plaquettes and vertices, and iv) the vacuum denoted ι. Of these, only two charges are independent. We shall take this pair to be the electric and magnetic charges. A charge composed with another charge of the same type gives the vacuum i.e. $c \times c = \iota$. The electric charges are created by Z-type errors and magnetic charges by X-type errors on the surface code.

A *hopping operator* is any element of the Pauli group that moves the charges. On a surface code, we can move the electric charges from one vertex to another by means of a Z-type Pauli operator. We denote by $H^{\epsilon}_{u\leftrightarrow v}$ the operator that moves ϵ from vertex u to v and vice versa. If we consider the magnetic charges then the movement can be accomplished by means of an X-type Pauli operator. The operator that moves a magnetic charge from face f to f' (or vice versa) is denoted by $H^{\mu}_{f\leftrightarrow f'}$.

Elementary hopping operators are those which move charges from one vertex to an adjacent vertex or from one plaquette to an adjacent plaquette. Let $e = (u, v)$ be the edge incident on the vertices u, v. We denote the elementary hopping operator along e as H^{ϵ}_{e}, where $H^{\epsilon}_{e} = Z_e$. It is a specific realization of $H^{\epsilon}_{u\leftrightarrow v}$. Similarly, the elementary operator that moves μ across e is denoted as H^{μ}_{e}. Let e be the edge shared by the faces f and f', then $H^{\mu}_{f\leftrightarrow f'}$ can be realized by H^{μ}_{e} where $H^{\mu}_{e} = X_e$. Observe that $H^{\epsilon}_{u\leftrightarrow v}$ and $H^{\mu}_{f\leftrightarrow f'}$ anti-commute when they act along the same edge, while operators for the same type of charges commute. In general, $H^{\epsilon}_{u\leftrightarrow v}$ and $H^{\mu}_{f\leftrightarrow f'}$ commute if and only if they cross an even number of times.

In spite of the many advantages that surface codes have, they are limited in terms of the number of transversal gates that can be implemented. To overcome this, we look at another class of topological codes that allow us to implement the entire Clifford group transversally.

3.3 Color codes

Color codes are a seprate class of topological codes that are similar to surface codes but not identical, as they have an additional property of 'color' associated with their faces. Unlike the surface codes which do not For 2D color codes, we consider lattices known as 2-colexes that can be colored using 3 colors. A 2-colex is a trivalent, 3-face-colorable complex. A stabilizer code is defined on a 2-colex by attaching qubits to every vertex code and defining the stabilizer S as

$$S = \langle B_f^X, B_f^Z \mid v \in \mathsf{F}(\Gamma) \rangle \text{ where } B_f^\sigma = \prod_{v \in f} \sigma_v. \tag{3.5}$$

We denote the Pauli group on these qubits as $\mathcal{P}_{\mathsf{V}(\Gamma)}$; the c-colored faces of Γ by $\mathsf{F}_c(\Gamma)$ and the c-colored edges of Γ by $\mathsf{E}_c(\Gamma)$. We restrict our attention to 2-colexes which do not have multiple edges (the surface codes could contain multiple edges though). This is not a severe restriction because a 2-colex with multiple edges can be modified to another 2-colex without them but encoding the same number of qubits and possessing the same error correcting capabilities (in terms of distance). We also assume that all embeddings are 2-cell embeddings so that all faces are homeomorphic to unit discs.

On a color code, the topological charges live on the faces. In addition to being electric and/or magnetic, they also carry a color depending on which face they are present. Let us denote the electric charge on a c-colored face as ϵ_c, the magnetic charge as μ_c and the composite charge as $\epsilon_c \mu_c$. The electric charges are not all independent [1]. Any pair (two out of three colors) of them can be taken as the independent set of electric charges. Similarly, only two magnetic charges are independent. As for surface codes, electric (magnetic) charges are created by Z (X) errors on the color code.

Similarly, we can define hopping operators for color codes. Let $f, f' \in \mathsf{F}_c(\Gamma)$ be two plaquettes connected by an edge (u, v) where $u \in f$ and $v \in f'$. Then $H^{\epsilon_c}_{f \leftrightarrow f'}$ and $H^{\mu_c}_{f \leftrightarrow f'}$ are the operators that move ϵ_c and μ_c from f to f'. A realization of these operators along (u, v) is $H^{\epsilon_c}_{u,v} = Z_u Z_v$ and $H^{\mu_c}_{u,v} = X_u X_v$. An element of the stabilizer can be viewed as a combination of hopping operators which move a charge around and bring it back to the original location. Since this movement cannot be detected, we can always adjoin an element of the stabilizer to the hopping operators.

CHAPTER 4

Equivalence between color codes and surface codes

Toric codes [7] proposed by Kitaev are one of the most studied classes of topological quantum codes and of fundamental importance in fault tolerant quantum computing. Although toric codes and their generalization–surface codes, have many attractive features (such as local stabilizer generators, low complexity decoders, efficient fault tolerant protocols, a high circuit threshold [5, 6, 11]), they have a limited set of transversal gates. On other hand, a class of topological color codes can implement the entire Clifford group transversally [1]. This might suggest that color codes are inequivalent to toric codes. However, in a very surprising development, Bombin et al. [2] showed that translationally invariant 2D color codes can be mapped to a finite number of copies of Kitaev's toric code.

Our goal is to find a map between a color code and some related surface codes. We shall denote this map by π for the rest of the paper. We shall first describe the construction of π in an informal fashion, emphasizing the principles underlying the map, and then rigorously justify all the steps. The key observation, due to [2], is that there are four types of charges on a surface code and sixteen types of charges on a color code. This is the starting point for relating the color code to surface codes. The two pairs of independent charges on the color code i.e. $\{\epsilon_c, \mu_{c'}\}$ and $\{\epsilon_{c'}, \mu_c\}$ suggest that we can decompose the color code into a pair of toric codes by mapping $\{\epsilon_c, \mu_{c'}\}$ charges onto one toric code and $\{\epsilon_{c'}, \mu_c\}$ onto another. However, charge "conservation" is not the only constraint. We would like a map that preserves in some sense the structure of the color code and allows us to go back and forth between the color code and the surface codes. We shall impose some conditions on this map keeping in mind that we would like to use it in the context of decoding color codes.

First, observe that the electric charges on the surface codes live on the vertices while the magnetic charges live on the plaquettes. But, if we consider the pair of charges $\{\epsilon_c, \mu_{c'}\}$, they both live on plaquettes—one on the c-colored plaquettes and another on c'-colored plaquettes. A natural way to make the association to a surface code is to contract all the c-colored plaquettes in the embedding of Γ. This will give rise to a new graph $\tau_c(\Gamma)$. We can now place the charges ϵ_c and $\mu_{c'}$ on the vertices and plaquettes of $\tau_c(\Gamma)$ respectively. Similarly, the charges $\{\mu_c, \epsilon_{c'}\}$ can live on the vertices and plaquettes of *another* instance of $\tau_c(\Gamma)$. We impose the following (desirable) constraints on the map π. It must be (i) linear, (ii) invertible, (iii) local, (iv) efficiently computable, (v) preserve the commutation relations between the (Pauli) error operators on $\mathsf{V}(\Gamma)$ i.e. $\mathcal{P}_{\mathsf{V}(\Gamma)}$, and (vi) consistent in the description of the movement of charges

on the color code and surface codes. These constraints are not necessarily independent and in no particular order. It is possible to relax some of the constraints above.

One immediate application of our results, as in [2, 4, 9], is an alternate decoding scheme for color codes via surface codes.

4.1 Deducing the map—A linear algebraic approach

The maps proposed in [2] are based on the following ideas: i) conservation of topological charges ii) identification of the hopping operators and iii) preserving the commutation relations between the hopping operators. These ideas are central to our work as well. However, we take a simpler linear algebraic approach to find the map.

Suppose we have a 2-colex Γ. Then, upon contracting all the c-colored faces including their boundary edges, we obtain another complex. We denote this operation as τ_c and the resulting complex as $\tau_c(\Gamma)$ (see Fig. 4.1). We suppress the subscript if the context makes it clear and just write τ. There is a one-to-one correspondence between the c-colored faces of Γ and the vertices of $\tau(\Gamma)$, so we can label the vertices of $\tau(\Gamma)$ by $f \in \mathsf{F}_c(\Gamma)$. We also label them by $\tau(f)$ to indicate that the vertex was obtained by contracting f. Similarly, the edges of $\tau(\Gamma)$ are in one-to-one correspondence with the c-colored edges of Γ, so an edge $\tau(\Gamma)$ is labeled the same as the parent edge $e = (u, v)$ in Γ. The faces which are not in $\mathsf{F}_c(\Gamma)$ are mapped to faces of $\tau(\Gamma)$. Therefore, we label the faces as f or more explicitly as $\tau(f)$, where $f \notin \mathsf{F}_c(\Gamma)$. Thus, the complex $\tau(\Gamma)$ has the vertex set $\mathsf{F}_c(\Gamma)$, edge set $\mathsf{E}_c(\Gamma)$ and faces $\mathsf{F}_{c'}(\Gamma) \cup \mathsf{F}_{c''}(\Gamma)$. Since every vertex v in Γ has a unique c-colored edge incident on it, we can associate to it an edge in $\tau(\Gamma)$ as $\tau(v)$.

Now, each c-colored face in Γ can host ϵ_c and μ_c. With respect to $\tau_c(\Gamma)$, they both reside on the vertices of Γ_c. So we shall place them on two different copies of $\tau_c(\Gamma)$ denoted Γ_1 and Γ_2. Then, the charges ϵ_c and μ_c will play the role of an electric charge on Γ_1 and Γ_2, respectively. So, we shall make the identification $\epsilon_c \equiv \epsilon_1$ and $\mu_c \equiv \epsilon_2$. The associated magnetic charges on Γ_i will have to reside on $\mathsf{F}(\Gamma_i)$. Possible candidates for these charges must come from $\epsilon_{c'}$, $\epsilon_{c''}$ and $\mu_{c'}$, $\mu_{c''}$. The following lemma addresses these choices.

Lemma 1 (Charge mapping)**.** *Let c, c', c'' be three distinct colors. Then, $\{\epsilon_c, \mu_{c'}\}$ and $\{\epsilon_{c'}, \mu_c\}$ are permissible pairings of the charges so that the color code on Γ can be mapped to a pair of surface codes on $\Gamma_i = \tau_c(\Gamma)$. In other words, $\epsilon_1 \equiv \epsilon_c$, $\mu_1 \equiv \mu_{c'}$, $\epsilon_2 \equiv \mu_c$ and $\mu_1 \equiv \epsilon_{c'}$, where ϵ_i and μ_i are the electric and magnetic charges of the surface code on Γ_i.*

Proof. First, observe that operators that move the electric charges ϵ_c and $\epsilon_{c'}$ are both Z-type, therefore they will always commute. This means that if ϵ_c is identified with the electric charge

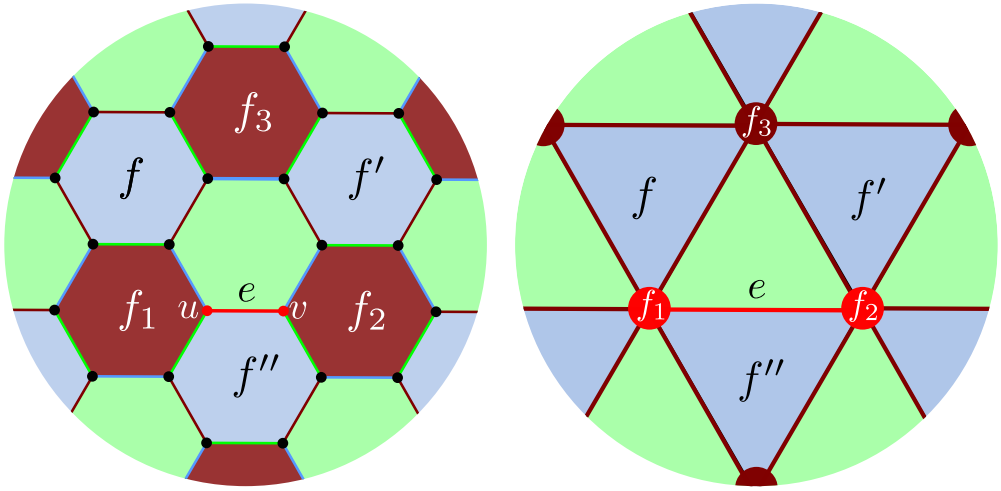


Figure 4.1: Illustrating the contraction of a color code via τ_c and the resultant surface code. Only portions of the codes are shown. The c-colored faces are vertices in $\tau_c(\Gamma)$. The faces $f \notin \mathsf{F}_c(\Gamma)$ remain faces in $\tau_c(\Gamma)$ and are also labeled f in $\tau_c(\Gamma)$, while the c-colored edge $e = (u, v)$ in Γ is mapped to an edge in $\tau_c(\Gamma)$, so we retain the label e. Every vertex in Γ is incident on a unique c-colored edge, so we can also extend τ_c to vertices u, v and edges unambiguously by defining $\tau_c(u) = \tau_c(v) = \tau_c(u, v) = e$.

on a surface code, $\epsilon_{c'}$ cannot be the associated magnetic charge. That leaves either μ_c and $\mu_{c'}$. Of these, observe that any operator that moves μ_c will always overlap with any operator that moves ϵ_c an even number of times. Therefore, this leaves only $\mu_{c'}$. The operators that move ϵ_c and $\mu_{c'}$ commute/anti-commute when they overlap an even/odd number of times just as the electric and magnetic charges of a surface code justifying the association $\epsilon_1 \equiv \epsilon_c$ and $\mu_1 \equiv \mu_{c'}$. A similar argument shows the validity of the equivalence $\epsilon_2 \equiv \mu_c$ and $\mu_2 \equiv \epsilon_{c'}$. □

Let Γ have n vertices and F_c vertices of color c. Then, Γ_i has F_c vertices, $n/2$ edges and $F_{c'} + F_{c''}$ faces. Together Γ_1 and Γ_2 have n qubits. We desire that π accurately reflect the movement of the independent charges on the color code and the surface codes. So, π must map the hopping operators of the charges of the color code on Γ to the hopping operators of the surface code on Γ_i. As mentioned earlier, $H^{\epsilon_c}_{u,v}$ moves electric charges on c-colored plaquettes and $H^{\epsilon_{c'}}_{u,v}$ electric charges on c'-colored plaquettes. But, although these charges may appear to be independent, due to the structure of the color code they are not. A c''-colored plaquette on the color code is bounded by edges whose color alternates between c and c'. The Z-type stabilizer associated to this plaquette, i.e. B^Z_f, can be viewed as being composed of $H^{\epsilon_c}_{u,v}$ hopping operators that move ϵ_c, in which case we would expect to map B^Z_f onto Γ_1. But, B^Z_f can also be viewed as being composed of $H^{\epsilon_{c'}}_{i,j}$. Thus, we see that there are two possible combinations of hopping operators that give the same plaquette stabilizer; one composed entirely of hopping operators of c-colored charges and the other of hopping operators of c'-colored charges. This suggests that there are dependencies among the hopping operators and some of them, while

ostensibly acting on only one kind of charge, could still be moving the other type of charges. However, the overall effect on the other charge must be trivial, i.e. it must move the charge back to where it started. A similar argument can be made for B_f^X which moves the magnetic charges. The next lemma makes precise these dependencies.

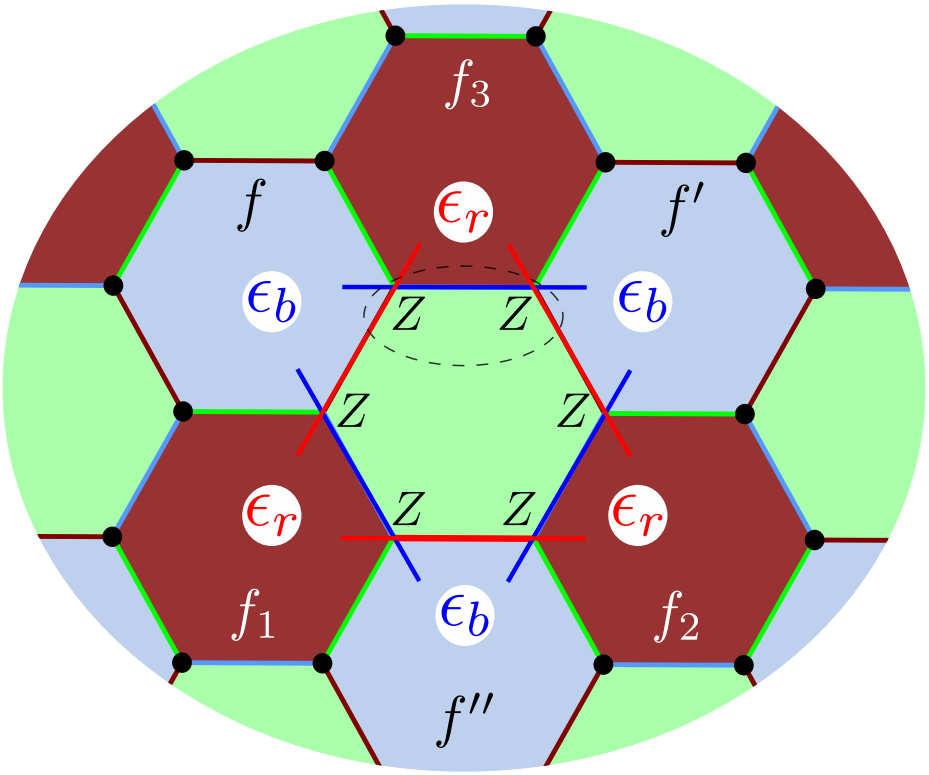


Figure 4.2: Circled hopping operator is the dependent operator. In addition to

Lemma 2 (Dependent hopping operators)**.** *Let $f \in \mathsf{F}_{c''}(\Gamma)$ and $1, \ldots, 2\ell_f$ be the vertices in its boundary so that $(2i-1, 2i) \in \mathsf{E}_c(\Gamma)$, $(2i, 2i+1) \in \mathsf{E}_{c'}(\Gamma)$ for $1 \leq i \leq \ell_f$ and $2\ell_f + 1 \equiv 1$. If π is invertible, then $\pi(B_f^\sigma) \neq I$ and there are $4\ell_f - 2$ independent elementary hopping operators along the edges of f.*

Proof. The stabilizer generator B_f^Z is given as

$$B_f^Z = \prod_{i=1}^{2\ell_f} Z_i = \prod_{i=1}^{\ell_f} Z_{2i-1}Z_{2i} = Z_1 Z_{2\ell_f} \prod_{i=1}^{\ell_f - 1} Z_{2i}Z_{2i+1} \tag{4.1}$$

$$= \prod_{i=1}^{\ell_f} H^{\epsilon_c}_{2i-1,2i} = H^{\epsilon_{c'}}_{1,2\ell_f} \prod_{i=1}^{\ell_f - 1} H^{\epsilon_{c'}}_{2i,2i+1}. \tag{4.2}$$

We see that B_f^Z can be expressed as the product of ℓ_f hopping operators of type $H^{\epsilon_c}_{u,v}$ or type

$H^{\epsilon_{c'}}_{u,v}$. Further, we have

$$\pi(B^Z_f) \;=\; \prod_{i=1}^{\ell_f} \pi(H^{\epsilon_c}_{2i-1,2i}) = \pi(H^{\epsilon_{c'}}_{1,2\ell_f}) \prod_{i=1}^{\ell_f-1} \pi(H^{\epsilon_{c'}}_{2i,2i+1})$$

If $\pi(B^Z_f) = I$, then $\ker(\pi) \neq I$ which means that π is not invertible and it would not be possible to preserve the information about the syndromes, as $\pi(B^Z_f)$ would commute with all the error operators. So, we require that $\pi(B^Z_f) \neq I$. This means that only one of these hopping operators is dependent and there are $2\ell_f - 1$ independent hopping operators. The linear independence of the remaining $2\ell_f - 1$ operators can be easily verified by considering their support. Similarly, B^X_f also implies that there are another $2\ell_f - 1$ independent hopping operators, giving us $4\ell_f - 2$ in total. □

We are now ready to define the action of π on elementary hopping operators. Without loss of generality we can assume if $f \in \mathsf{F}_{c''}(\Gamma)$ has $2\ell_f$ edges, then the dependent hopping operators of f are $H^{\epsilon_{c'}}_{1,2\ell_f}$ and $H^{\mu_{c'}}_{2m,2m+1}$ i.e. $Z_1 Z_{2\ell_f}$ and $X_{2m}X_{2m+1}$, where $1 \leq m \leq \ell_f$ and $2\ell_f + 1 \equiv 1$.

Lemma 3 (Elementary hopping operators)**.** *Let $f, f' \in \mathsf{F}_c(\Gamma)$ where the edge (u, v) is incident on f and f'. Then, the following choices reflect the charge movement on Γ onto the surface codes on Γ_i.*

$$\pi(H^{\epsilon_c}_{u,v}) \;=\; \left[Z_{\tau(u)}\right]_1 = \left[Z_{\tau(v)}\right]_1 \tag{4.3}$$

$$\pi(H^{\mu_c}_{u,v}) \;=\; \left[Z_{\tau(u)}\right]_2 = \left[Z_{\tau(v)}\right]_2, \tag{4.4}$$

where $[T]_i$ indicates the instance of the surface code on which T acts. Now if $f, f' \in \mathsf{F}_{c'}(\Gamma)$ and $(u, v) \in \mathsf{E}_{c'}(\Gamma)$ such that $u \in f$ and $v \in f'$ and $H^{\epsilon_{c'}}_{u,v}$ and $H^{\mu_{c'}}_{u,v}$ are chosen to be independent hopping operators of f, then

$$\pi(H^{\epsilon_{c'}}_{u,v}) = \left[X_{\tau(u)}X_{\tau(v)}\right]_2; \pi(H^{\mu_{c'}}_{u,v}) = \left[X_{\tau(u)}X_{\tau(v)}\right]_1. \tag{4.5}$$

Proof. We only prove for $H^{\epsilon_c}_{u,v}$ and $H^{\mu_{c'}}_{u,v}$. Similar reasoning can be employed for $H^{\mu_c}_{u,v}$ and $H^{\epsilon_{c'}}_{u,v}$. (i) $H^{\epsilon_c}_{u,v}$: This operator moves ϵ_c from f to f' in Γ. These faces are mapped to adjacent vertices in $\tau(\Gamma)$. By Lemma 1, ϵ_c is mapped to ϵ_1, so $\pi(H^{\epsilon}_{u,v})$ should move ϵ_1 from the vertex $\tau(f)$ to the vertex $\tau(f')$ on Γ_1. Many hopping operators can achieve this; choosing the elementary operator gives $\pi(Z_u Z_v) = [Z_{\tau(u,v)}]_1$. Since $\tau(u, v) = \tau(u) = \tau(v)$, Eq. (4.3) follows. (ii) $H^{\mu_{c'}}_{u,v}$: This operator moves $\mu_{c'}$ from f to f'. Since $\mu_{c'}$ is mapped to μ_1, $\pi(H^{\mu_{c'}}_{u,v})$ should move μ_1 from the plaquette $\tau(f)$ to $\tau(f')$ on Γ_1. The operator on the first surface code which achieves this is an X-type operator on qubits $\tau(u)$ and $\tau(v)$ in Γ_1, i.e. $[X_{\tau(u)}X_{\tau(v)}]_1$. In both cases we choose the hopping operators to be of minimum weight. □

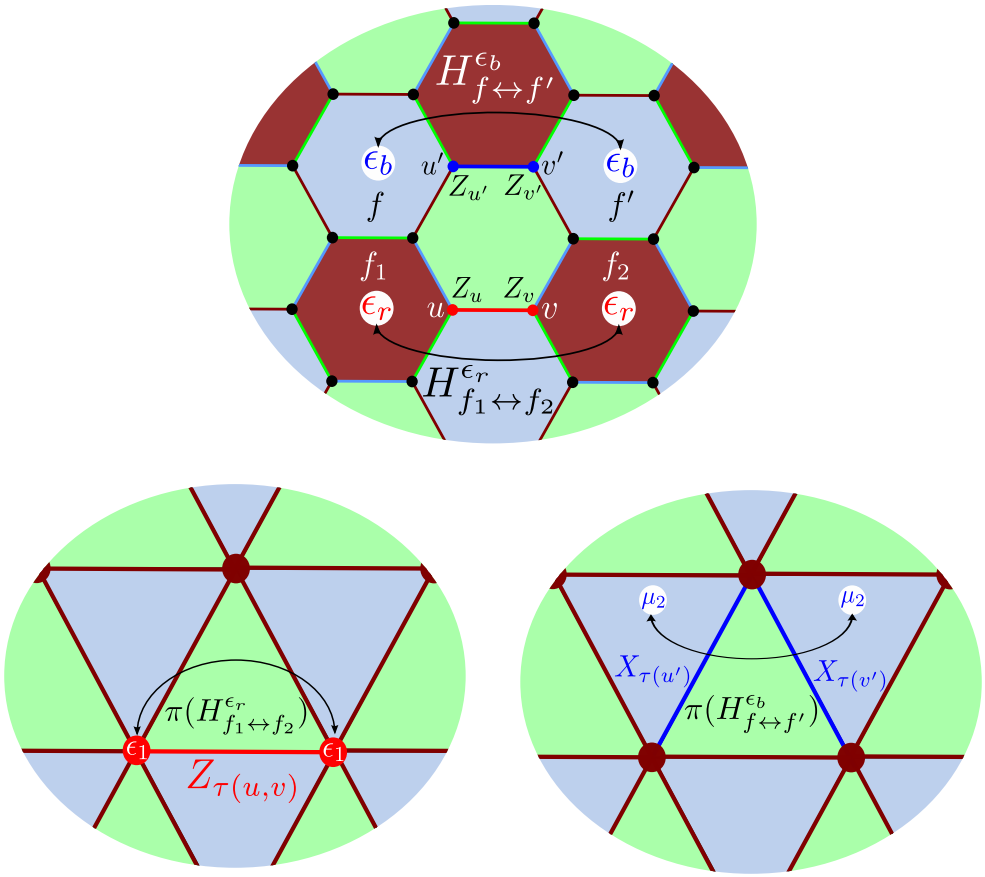


Figure 4.3: Mapping the independent hopping operators $H^{\epsilon_r}_{u,v} = H^{\epsilon_r}_{f_1\leftrightarrow f_2} = Z_u Z_v$ and $H^{\epsilon_b}_{u',v'} = H^{\epsilon_b}_{f\leftrightarrow f'} = Z_{u'} Z_{v'}$ on Γ onto two copies of $\tau(\Gamma)$ i.e. Γ_1 and Γ_2; $\pi(H^{\epsilon_r}_{f_1\leftrightarrow f_2}) = [Z_{\tau(u)}]_1$ acts only on Γ_1 while $H^{\epsilon_b}_{f\leftrightarrow f'} = [X_{\tau(u')} X_{\tau(v')}]_2$ acts only on Γ_2.

Lemma 3 does not specify the mapping for the dependent hopping operators but it can be obtained as a linear combination of the independent ones. Alternative choices to those given in Lemma 3 exist for π. These choices are essentially alternate hopping operators on the surface codes which accomplish the same charge movement. Such operators can be obtained by adding stabilizer elements to those given in Eqs. (4.3)–(4.5).

In this paper we explore the choice when the operators $H^{\epsilon_{c'}}_{1,2\ell_f}$ and $H^{\mu_{c'}}_{2m,2m+1}$ are dependent. The c''-faces form a covering of all the vertices of Γ and they are non-overlapping. The elementary hopping operators along the edges on such plaquette do not interact with the elementary hopping operators of other plaquettes in $\mathsf{F}_{c''}(\Gamma)$. So we can consider each $f \in \mathsf{F}_{c''}(\Gamma)$ independently. This also makes sense from our constraint to keep π local. Based on Lemmas 2 and 3, we can map the independent elementary hopping operators of f along c-colored edges. They map elementary hopping operators on Γ to elementary hopping operators on Γ_i.

$$\pi(Z_{2i-1}Z_{2i}) = [Z_{\tau(2i)}]_1 \text{ and } \pi(X_{2i-1}X_{2i}) = [Z_{\tau(2i)}]_2 \tag{4.6}$$

Next, we consider the hopping operators that involve the c'-colored edges. Without loss of generality we assume that the edge $Z_1Z_{2\ell_f}$ is the one which carries the dependent hopping operator and $X_{2m}X_{2m+1}$ carries the other dependent hopping operator. Then letting $2\ell_f+1 \equiv 1$ we have

$$\pi(Z_{2i}Z_{2i+1}) = [X_{\tau(2i)}X_{\tau(2i+1)}]_2 \; ; 1 \leq i < \ell_f \tag{4.7}$$

$$\pi(X_{2i}X_{2i+1}) = [X_{\tau(2i)}X_{\tau(2i+1)}]_1 \; ; 1 \leq i \neq m \leq \ell_f. \tag{4.8}$$

All these operators and their images under π are linearly independent as can be seen from their supports. From Lemma 3 we obtain the images for the dependent hopping operators:

$$\pi(H^{\epsilon_{c'}}_{1,2\ell_f}) = [X_{\tau(1)}X_{\tau(2\ell_f)}]_2 \prod_{i=1}^{\ell_f} [Z_{\tau(2i)}]_1 \tag{4.9}$$

$$\pi(H^{\mu_{c'}}_{2m,2m+1}) = [X_{\tau(2m)}X_{\tau(2m+1)}]_1 \prod_{i=1}^{\ell_f} [Z_{\tau(2i)}]_2 \tag{4.10}$$

To complete the map it remains to find the action of π for two more independent errors on the color code. One choice is any pair of single qubit operators X_i and Z_j, where $1 \leq i, j \leq 2\ell_f$. Or we can consider the images under π. We can see from Eqs. (4.6)–(4.8) that the images are also linearly independent and only single qubit X-type of errors remain to be generated. One choice is any $[X_{\tau(i)}]_1$ on Γ_1 and $[X_{\tau(j)}]_2$ on Γ_2, where $1 \leq i, j \leq 2\ell_f$. That is, we need to find E, E' such that $\pi(E) = [X_{\tau(i)}]_1$ and $\pi(E') = [X_{\tau(j)}]_2$ respect the commutation relations.

Lemma 4 addresses this choice.

Lemma 4 (Splitting)**.** *The following choices lead to an invertible π while respecting the commutation relations with hopping operators in Eqs.* (4.6)–(4.8)*.*

$$\pi(gX_1) = [X_{\tau(1)}]_1 \text{ where } g \in \{I, B_f^X, B_f^Y, B_f^Z\} \tag{4.11}$$

$$\pi(gZ_{2m}) = [X_{\tau(2m)}]_2 \text{ where } g \in \{I, B_f^X\} \tag{4.12}$$

Proof. Each face $f \in \mathsf{F}_{c''}(\Gamma)$ accounts for $2\ell_f$ qubits i.e. $4\ell_f$ independent operators. Now $[X_{\tau(1)}]_1$ and $[X_{\tau(2m)}]_2$ form a linearly independent set of size $4\ell_f$ along with the images of the independent elementary hopping operators on f. Thus, the elementary hopping operators and the preimages of $[X_{\tau(1)}]_1$ and $[X_{\tau(2m)}]_2$ account for all the $4\ell_f$ operators on qubits on f. Considering all faces in $\mathsf{F}_{c''}(\Gamma)$, we have $\sum_f 4\ell_f = 2n$ operators which generate $\mathcal{P}_{\mathsf{V}(\Gamma)}$. Since their images are independent and $\Gamma_1 \cup \Gamma_2$ has exactly as many qubits as Γ, π must be invertible.

Next, we prove these choices respect the commutation relations as stated. Consider $[X_{\tau(1)}]_1$: this error commutes with all the operators in Eq. (4.6)–(4.8) except $\pi(Z_1Z_2) = [Z_{\tau(1)}]_1$. There are $4\ell_f - 3$ such hopping operators on f with which $\pi^{-1}([X_{\tau(1)}]_1)$ must commute. As a consequence of the rank-nullity theorem there are $2^{4\ell_f-(4\ell_f-3)}$ such operators. It can be verified that $\langle X_1, B_f^X, B_f^Z\rangle$ account for these operators. But $\pi^{-1}([X_{\tau(1)}]_1)$ must also anti-commute with Z_1Z_2. This gives the choices in Eq. (4.11) since operators in $\langle B_f^X, B_f^Z\rangle$ commute with Z_1Z_2. Now let us determine $\pi^{-1}([X_{\tau(2m)}]_2)$. Once again with reference to Eq. (4.6)–(4.8) we see that it must commute with $4\ell_f - 3$ hopping operators on f. It also commutes with $\pi^{-1}([X_{\tau(1)}]_1)$ since $[X_{\tau(2m)}]_2$ commutes with $[X_{\tau(1)]_1}$. Again, due to a dimensionality argument there are $2^{4\ell_f-(4\ell_f-2)}$ choices for $\pi^{-1}([X_{\tau(2m)}]_2)$. Since $[X_{\tau(2m)}]_2$ anti-commutes with $[Z_{\tau(2m)}]_2$ its preimage must anti-commute with $\pi^{-1}([Z_{\tau(2m)}]_2) = X_{2m-1}X_{2m}$ giving two choices Z_{2m} and $Z_{2m}B_f^X$. We can check that Z_{2m} satisfies all the required commutation relations as does the choice $Z_{2m}B_f^X$. □

In Lemma 4 we first assigned $\pi^{-1}([X_\tau(1)]_1)$ followed by $\pi^{-1}([X_{\tau(2m)}]_2)$. Changing the order restricts g to $\{I, B_f^X\}$ in Eq. (4.7) while $g \in \{I, B_f^X, B_f^Y, B_f^Z\}$ in Eq. (4.8).

Lemma 5 (Preserving commutation relations)**.** *The map π preserves commutation relations of error operators in $\mathcal{P}_{\mathsf{V}(\Gamma)}$.*

Proof of Lemma 5. We only sketch the proof. It suffices to show that the commutation relations hold for a basis of $\mathcal{P}_{\mathsf{V}(\Gamma)}$. We consider the basis consisting of the hopping operators along c and c' edges in Eq. (4.6)–(4.8) and the single qubit operators given in Lemma 4. The proof of Lemma 4 shows that the commutation relations are satisfied for the single qubit operators. Consider a hopping operator along c'-colored edge. This anti-commutes with exactly two hopping

operators along c-colored edges on Γ. For instance consider $Z_{2i}Z_{2i+1}$. From Eq. (4.6)–(4.8) this anti-commutes with $X_{2i-1}X_{2i}$ and $X_{2i+1}X_{2i+2}$. Their images under π are $[X_{\tau(2i)}X_{\tau(2i+1)}]_2$, $[Z_{\tau(2i)}]_2$ and $[Z_{\tau(2i+1)}]_2$ for which it is clear that the commutation relations are satisfied.

The operators along the c-colored edges are given in Eq. (4.6). Suppose we consider $\pi(Z_{2i-1}Z_{2i})$; then it anti-commutes with $X_{2i-2}X_{2i-1}$ and $X_{2i}X_{2i+1}$. We only need to verify for those operators which are independent. Assume that they are both independent, then their images are $X_{\tau(2i-2)}X_{\tau(2i-1)}$ and $X_{\tau(2i)}X_{\tau(2i+1)}$ respectively. They anti-commute with $\pi(Z_{2i-1}Z_{2i}) = [Z_{\tau(2i-1)}]_1 = [Z_{\tau(2i)}]_1$. If only one of the operators is independent, then we need only verify for that operator. The preceding argument already establishes this result. We can argue in a similar fashion to show that commutation relations are preserved for the operators of the type $X_{2i-1}X_{2i}$ and $X_{2i}X_{2i+1}$. □

Lemma 6 (Preserving code capabilities)**.** *Under π, stabilizers of the color code on Γ are mapped to stabilizers on the surface codes on Γ_1 and Γ_2.*

Proof of Lemma 6. To prove this, it suffices to show that the stabilizers associated with plaquettes of all three colors are mapped to stabilizers on the surface codes. If $\Gamma_i = \tau_c(\Gamma)$, then we show that the stabilizers associated with $f \in \mathsf{F}_{c'}(\Gamma) \cup \mathsf{F}_{c''}(\Gamma)$ are mapped to the plaquette stabilizers on Γ_i. If $f \in \mathsf{F}_{c'}(\Gamma)$, then $B_f^Z = \prod_{i=1}^{\ell_f} H_{2i-1,2i}^{\epsilon_c}$. By Lemma 3 this is mapped to $\prod_i^{\ell_f}[Z_{\tau(2i)}]_1 = \prod_{e\in\partial(\tau(f))}[Z_e]_1$. Using a similar argument we can show that $B_f^Z \in \mathsf{F}_{c''}(\Gamma)$ is also a plaquette stabilizer on Γ_1. Since faces in $\mathsf{F}_{c'}(\Gamma)\cup\mathsf{F}_{c''}(\Gamma)$ are in one to one correspondence with the faces of $\tau(\Gamma)$, they account for all the face stabilizers on Γ_1. By considering B_f^X, we can similarly show that they map to the face stabilizers on Γ_2.

Now consider a face $f \in \mathsf{F}_c(\Gamma)$. Consider B_f^Z, this can be decomposed into hopping operators $H_{u,v}^{\epsilon_{c'}}$ along c'-edges. By Lemma 3, such an operator maps to $[X_{\tau(u)}X_{\tau(v)}]_2$ and an additional stabilizer on one of the faces of Γ_1 if $H_{u,v}^{\epsilon_{c'}}$ is a dependent hopping operator. Thus B_f^Z maps to a vertex operator on $\tau(f)$ in Γ_2 and possibly a combination of plaquette stabilizers. Since every vertex in Γ_2 is from a face in Γ, we can account for all the vertex operators on Γ_2. Similarly, by considering the stabilizer B_f^X we can account for all the vertex operators on Γ_1. □

Theorem 7. *Any 2D color code (on a 2-colex Γ without parallel edges) is equivalent to a pair of surface codes $\tau(\Gamma)$ under the map π defined as in Algorithm 1.*

Proof Sketch. By charge conservation we require two copies of $\tau(\Gamma)$ to represent the color code using surface codes. Lines 2–3 follow from Lemma 1. Since c''-colored faces in $\mathsf{F}_{c''}(\Gamma)$ cover all the qubits of the color code, we account for all the single qubit operators on the color code by the for-loop in lines 4–10. The closed form expressions for single qubit errors in lines 6–9 are a

Algorithm 1 Mapping a 2D color code to surface codes

Input: A 2-colex Γ without parallel edges; Γ is assumed to have a 2-cell embedding.

Output: $\pi : \mathcal{P}_{\mathsf{V}(\Gamma)} \to \mathcal{P}_{\mathsf{E}(\Gamma_1)} \otimes \mathcal{P}_{\mathsf{E}(\Gamma_2)}$, where $\Gamma_i = \tau_c(\Gamma)$.

1: Pick a color $c \in \{r, g, b\}$ and contract all edges of Γ that are colored $\{r, g, b\} \setminus c$ to obtain $\tau(\Gamma)$. Denote two instances of $\tau(\Gamma)$ as Γ_1 and Γ_2.

2: Choose charges ϵ_c, μ_c, $\epsilon_{c'}$ and $\mu_{c'}$ on Γ, where $c' \neq c$.

3: Set up correspondence between charges on Γ and Γ_i as follows: $\epsilon_1 \equiv \epsilon_c$, $\mu_1 \equiv \mu_{c'}$, $\epsilon_2 \equiv \mu_c$ and $\mu_2 \equiv \epsilon_{c'}$.

4: **for** each c''-colored face f in $\mathsf{F}(\Gamma)$ **do**

5: Let the boundary of f be $v_1, \ldots v_{2\ell_f}$.

6: Choose a pair of c'-colored edges in $\partial(f)$, say $(v_{2\ell_f}, v_1)$ and (v_{2m}, v_{2m+1}). Let $[T]_i$ denote that T acts on Γ_i.

$$\pi(Z_{v_1}) = \left[X_{\tau(v_1)}\right]_2 \prod_{i=1}^{m} \left[Z_{\tau(v_{2i})}\right]_1 \tag{4.13}$$

7: For $1 \leq j \leq \ell_f$ compute the mapping (recursively) as

$$\pi(Z_{v_{2j}}) = \pi(Z_{v_{2j-1}}) \left[Z_{\tau(v_{2j})}\right]_1 \tag{4.14}$$
$$\pi(Z_{v_{2j-1}}) = \pi(Z_{v_{2j-2}}) \left[X_{\tau(v_{2j-2})} X_{\tau(v_{2j-1})}\right]_2 \tag{4.15}$$

8: For $1 \leq j \leq m$ compute the mapping as

$$\pi(X_{v_1}) = \left[X_{\tau(v_1)}\right]_1 \tag{4.16}$$
$$\pi(X_{v_{2j}}) = \pi(X_{v_{2j-1}}) \left[Z_{\tau(v_{2j})}\right]_2 \tag{4.17}$$
$$\pi(X_{v_{2j-1}}) = \pi(X_{v_{2j-2}}) \left[X_{\tau(v_{2j-2})} X_{\tau(v_{2j-1})}\right]_1 \tag{4.18}$$

9: For $m + 1 \leq j \leq \ell_f$ compute the mapping as

$$\pi(X_{v_{2\ell_f}}) = \left[X_{\tau(v_{2\ell_f})}\right]_1 \tag{4.19}$$
$$\pi(X_{v_{2j-1}}) = \pi(X_{v_{2j}}) \left[Z_{\tau(v_{2j})}\right]_2 \tag{4.20}$$
$$\pi(X_{v_{2j}}) = \pi(X_{v_{2j+1}}) \left[X_{\tau(v_{2j})} X_{\tau(v_{2j+1})}\right]_1 \tag{4.21}$$

10: **end for**

direct consequence of Lemmas 3, 4 and the choices given in Eqs. (4.6)–(4.8) and Eqs. (4.11)–(4.12). By considering the images of the stabilizers of the color code, we can show that they are mapped to the stabilizers of the surface codes on Γ_i (see Lemma 6). From Lemma 5, the commutation relations among the hopping operators on the color code in Eq. (4.6)–(4.8) and the single qubit operators in Eq. (4.11)–(4.12) are preserved. Hence, the errors corrected by the color code are the same as those corrected by the surface codes on Γ_i. Thus the color code is equivalent to two copies of $\tau(\Gamma)$. □

4.2 Comparison with other maps

Authors (Approach)	Translation invariance	No. of copies	Ancilla qubits	Other comments
Bombin et. al.	Yes	Finite, could be many	Sometimes	Applicable for all translation invariant 2D stabilizer codes
Delfosse (Algebraic topology and hypergraphs)	No	3	Yes	Not bijective
Present work (Linear algebraic)	No	2	No	Bijective

Table 4.1: Comparison with previous work

The result in [2], as well the subsequent papers [14**?**] which explore the equivalence between stabilizer codes and toric codes in great detail, have one important qualifier, namely translational invariance. For 2D color codes, Delfosse [4] relaxed the constraint on translation invariance and mapped a 2D color code to three surface codes. In this paper, we propose an alternate map based on linear algebra. We map arbitrary color codes, including those that are not translationally invariant, onto two copies of a surface code.

The main differences between the proposed map (and its variations) and that of [2] are: first, the map therein requires local translation symmetry. Second, in the map in [2], the number of toric codes onto which the color code is mapped need not always be two. On the other hand, our map always gives exactly two surface codes. These surface codes need not be copies of the toric code on the square lattice. Third, the proposed map does not require any ancilla qubits, as may be the case for some codes under the map in [2]. Even for arbitrary color codes, our map is efficiently computable locally and we compute the images for all the single qubit errors on the color code in closed form. On the other hand, the results in [2] go beyond color codes and include all local translationally invariant 2D stabilizer codes and certain subsystem codes.

Our work differs from that of [4] in the following aspects. The map in [4] projects onto three copies of surface codes. Therefore, our map leads to a lower decoding complexity compared to [4]. Furthermore, unlike our map which is a bijective map onto the surface codes, the map in [4] is not bijective, although it is injective. This has a bearing in the context of decoding. In some cases, a decoder using the map in [4] may not be able to lift an error (estimate) from surface codes to (the parent) color code. This will not occur with a decoder using our map.

Following the submission of our paper, we became aware of the work by Kubica, Yoshida, and Pastawski [9] who showed equivalence between color codes and toric codes for all dimensions $D \geq 2$. In 2D, for color codes without boundaries, their result is similar to ours but there are substantial differences. First, they map the color code onto two *different* surface codes, we map onto two copies of the *same* surface code. Second, we use linear algebra to study these equivalences, which is simpler than the approach taken in [9] (or [2, 4]).

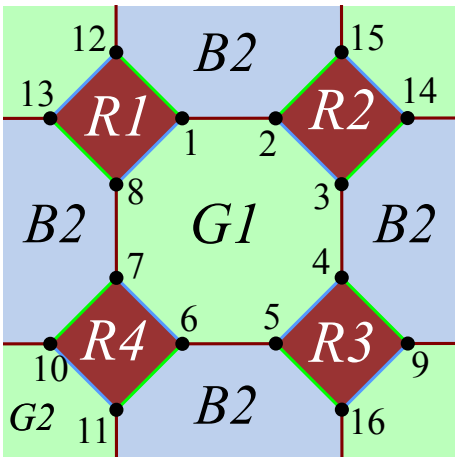
CHAPTER 5

Examples

In this chapter, we will look at the mapping from color codes to surface codes for three types of lattices, two with translational symmetry and one without. The lattices with translational symmetry are the hexagonal lattice and the square-octagonal lattice. These are two of the three 2D regular lattices that are 3-colorable, the third one being the truncated trihexagonal lattice [12]. The hexagonal lattice has already been used in the previous chapter to illustrate various lemmas. As in (preliminaries section), F_r, F_g and F_b represent the sets of red, green and clue faces respectively.

5.1 Square-octagonal lattice

Consider a 16 qubit color code on a square octagonal lattice embedded on a torus.



For this code, the number of independent stabilizer generators is

$$\begin{aligned} g &= 2(|F_r| + |F_g| + |F_b|) - 4, \\ &= 2(4 + 2 + 2) - 4 = 12, \end{aligned}$$

and so the number of encoded qubits is

$$\begin{aligned} k &= n - g, \\ &= 16 - 12, \\ &= 4. \end{aligned}$$

Thus, the code parameters are $[16, 4, 7]_2$. The code is mapped to two copies of Kitaev's toric code, each of which has 8 qubits. The mapping of the hopping operators is illustrated below:

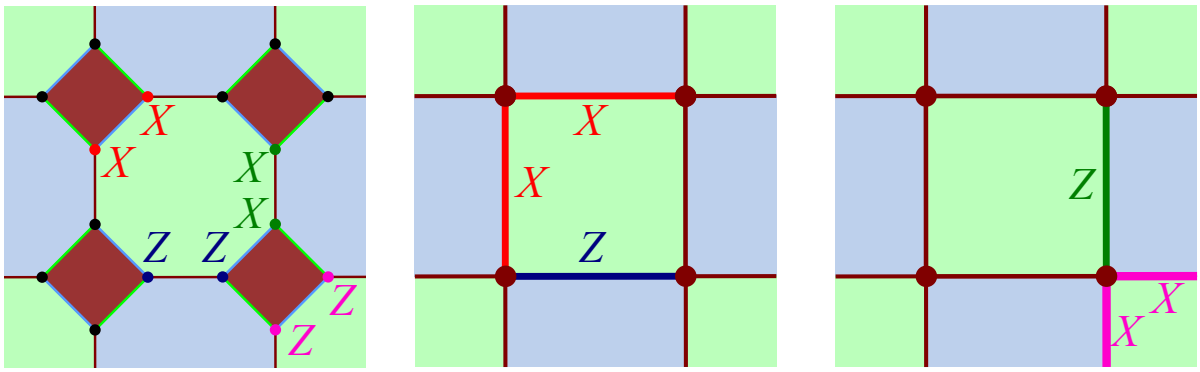


Figure 5.1: Color-coded mapping of independent hopping operators from the color code to the two copies of the toric code. Red: $H_{1,8}^{\mu_{c'}}$ is mapped to $[X_{\tau(1)}X_{\tau(8)}]_1$; blue: $H_{5,6}^{\epsilon_c}$ is mapped to $[Z_{\tau(5)}]_1 = [Z_{\tau(6)}]_1$; green: $H_{3,4}^{\mu_c}$ is mapped to $[Z_{\tau(3)}]_2 = [Z_{\tau(4)}]_2$; pink: $H_{16,9}^{\epsilon_{c'}}$ is mapped to $[X_{\tau(9)}X_{\tau(16)}]_2$.

Without loss of generality, we choose the dependent Z operator to be $H_{1,8}^{\epsilon_{c'}}$ on $G1$ and the dependent X operator on $G1$ to be $H_{2,3}^{\mu_{c'}}$. From Lemma 4, we choose the single qubit X operator to be X_1 and the single qubit Z operator to be Z_2, i.e., we take $m = 1$ in 4.11. These choices lead to the following valid map for X and Z type errors on $G1$ from the color code to the two copies of Kitaev's toric code. It can be easily verified that the map satisfies all commutation relations and that stabilizers on the color code map to stabilizers on the toric code. The map for errors on plaquette $G2$ can be derived in an identical fashion, by making choices consistent with Lemmas 2 and 4. The color code is shown on the left and the surface codes on the right. Each of the single qubit errors and their images are shown (in bold red).

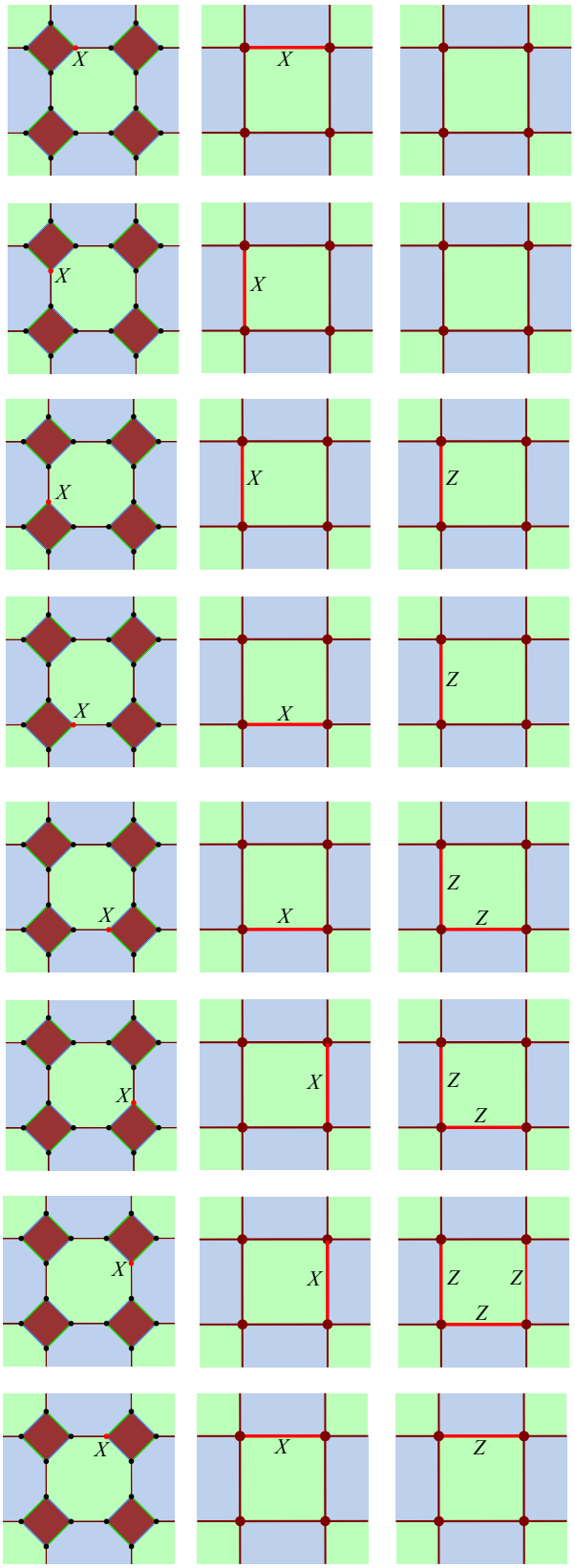


Figure 5.2: Mapping for X type errors on $G1$, starting with X_1, which is the single qubit X operator chosen according to Lemma 4. The dependent hopping operator is $H^{\mu_{c'}}_{2,3}$, which clearly maps to a Z stabilizer on $[\tau(G1)]_2$, as in Eq. 4.10.

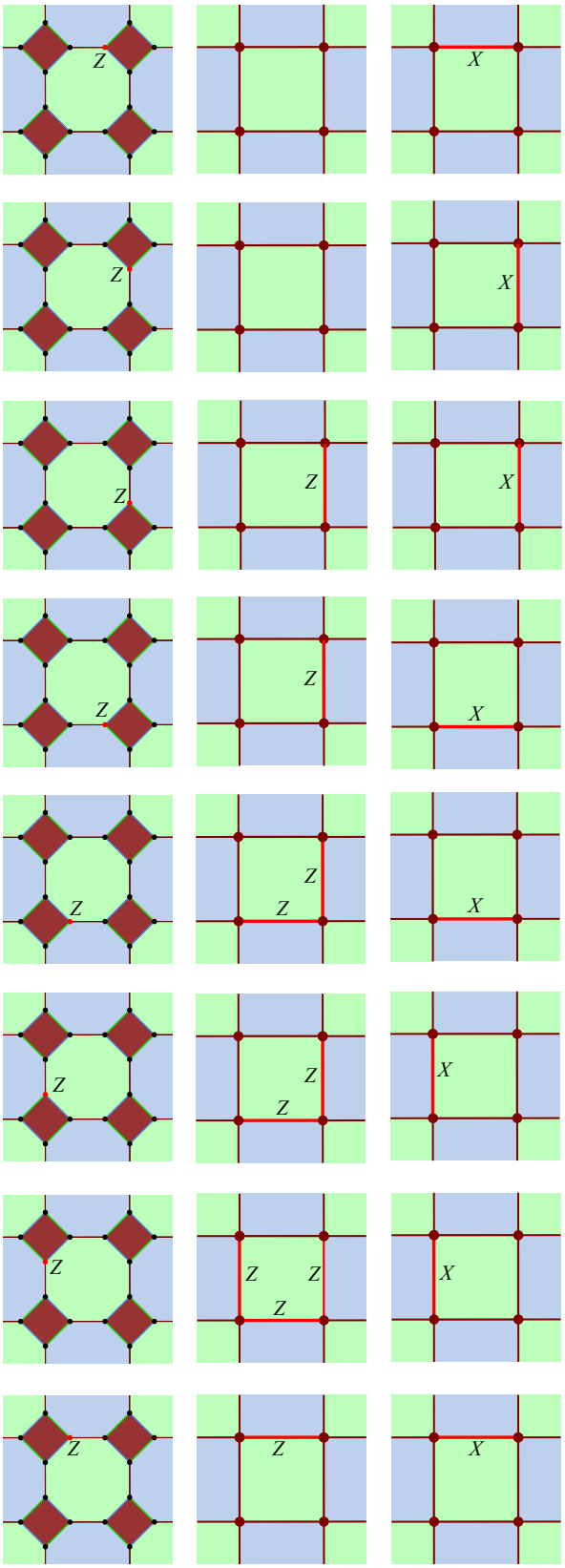
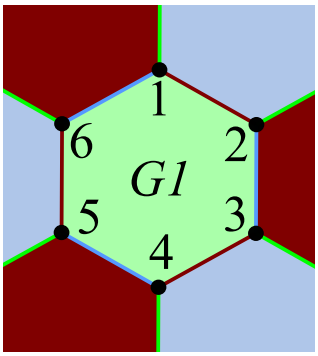


Figure 5.3: Mapping for Z type errors on $G1$, starting with Z_2, which is the single qubit Z operator chosen according to Lemma 4. The dependent hopping operator is $H^{\epsilon_{c'}}_{1,8}$, which clearly maps to a Z stabilizer on $[\tau(G1)]_1$, as in Eq. 4.9.

5.2 Hexagonal lattice

Consider a single green plaquette of a hexagonal code containing 6 qubits and its surrounding plaquettes.



The hexagonal color code is mapped to two copies of a surface code on a triangular lattice. The mapping of the independent hopping operators is illustrated below:

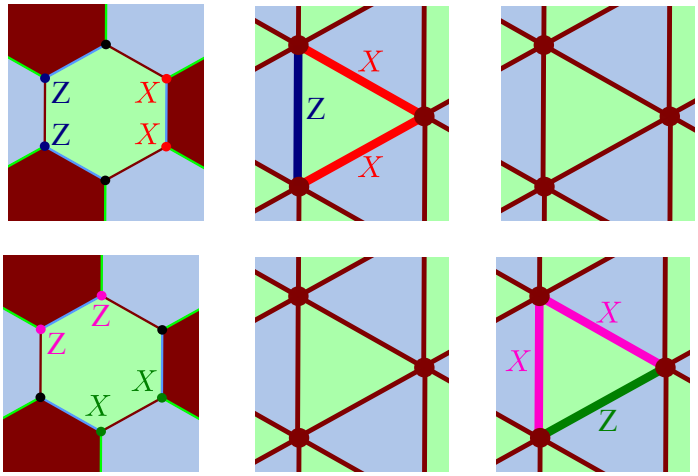


Figure 5.4: Color-coded mapping of independent hopping operators from the color code to the two copies of the surface code. Red: $H_{2,3}^{\mu_{c'}}$ is mapped to $[X_{\tau(2)}X_{\tau(3)}]_1$; blue: $H_{5,6}^{\epsilon_c}$ is mapped to $[Z_{\tau(5)}]_1 = [Z_{\tau(6)}]_1$; green: $H_{3,4}^{\mu_c}$ is mapped to $[Z_{\tau(3)}]_2 = [Z_{\tau(4)}]_2$; pink: $H_{6,1}^{\epsilon_{c'}}$ is mapped to $[X_{\tau(6)}X_{\tau(1)}]_2$.

Without loss of generality, we choose the dependent Z operator on $G1$ to be $H_{2,3}^{\epsilon_{c'}}$ and the dependent X operator on $G1$ to be $H_{6,1}^{\mu_{c'}}$. From Lemma 4, we choose the single qubit X operator to be X_2 and the single qubit Z operator to be Z_1. These choices lead to the following valid map for X and Z type errors on $G1$ from the color code to the two copies of the triangular surface code. It can be easily verified that the map satisfies all commutation relations and that

stabilizers on the color code map to stabilizers on the surface code. The map for errors on other plaquettes can be derived in an identical fashion, by making choices consistent with Lemmas 2 and 4.

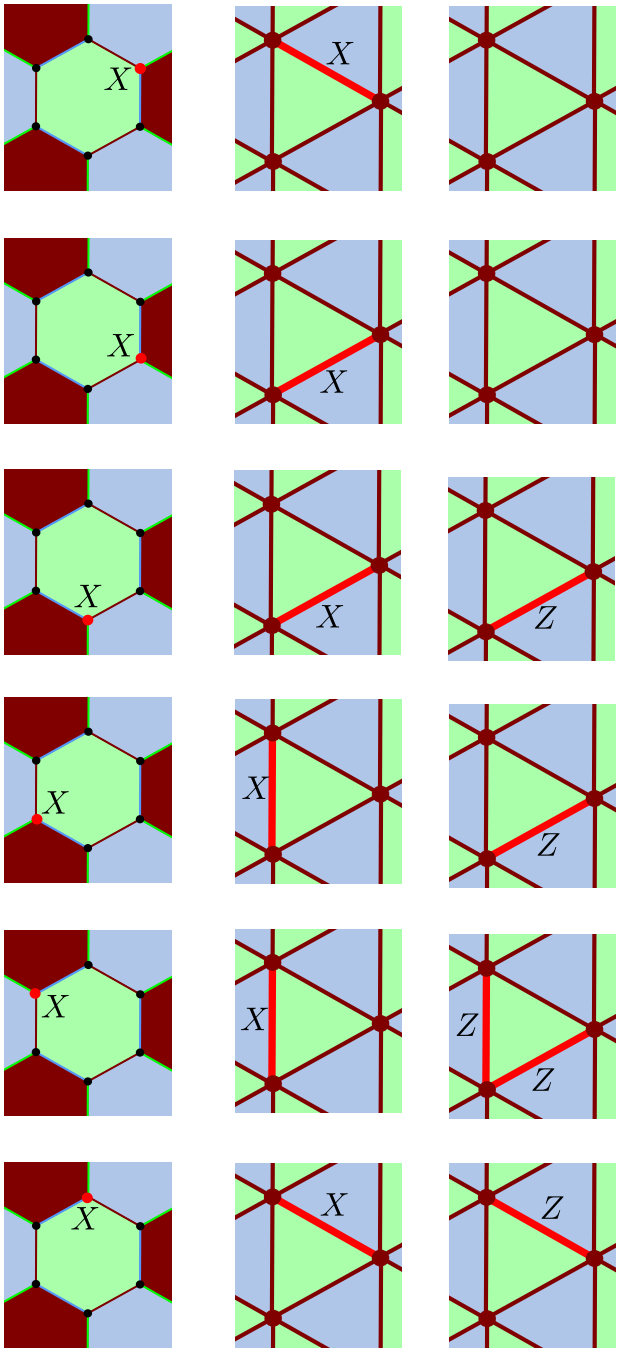


Figure 5.5: Mapping for X type errors on $G1$, starting with X_2, which is the single qubit X operator chosen according to Lemma 4. The dependent hopping operator is $H^{\mu_{c'}}_{6,1}$, which clearly maps to a Z stabilizer on $[\tau(G1)]_2$, as in Eq. 4.10.

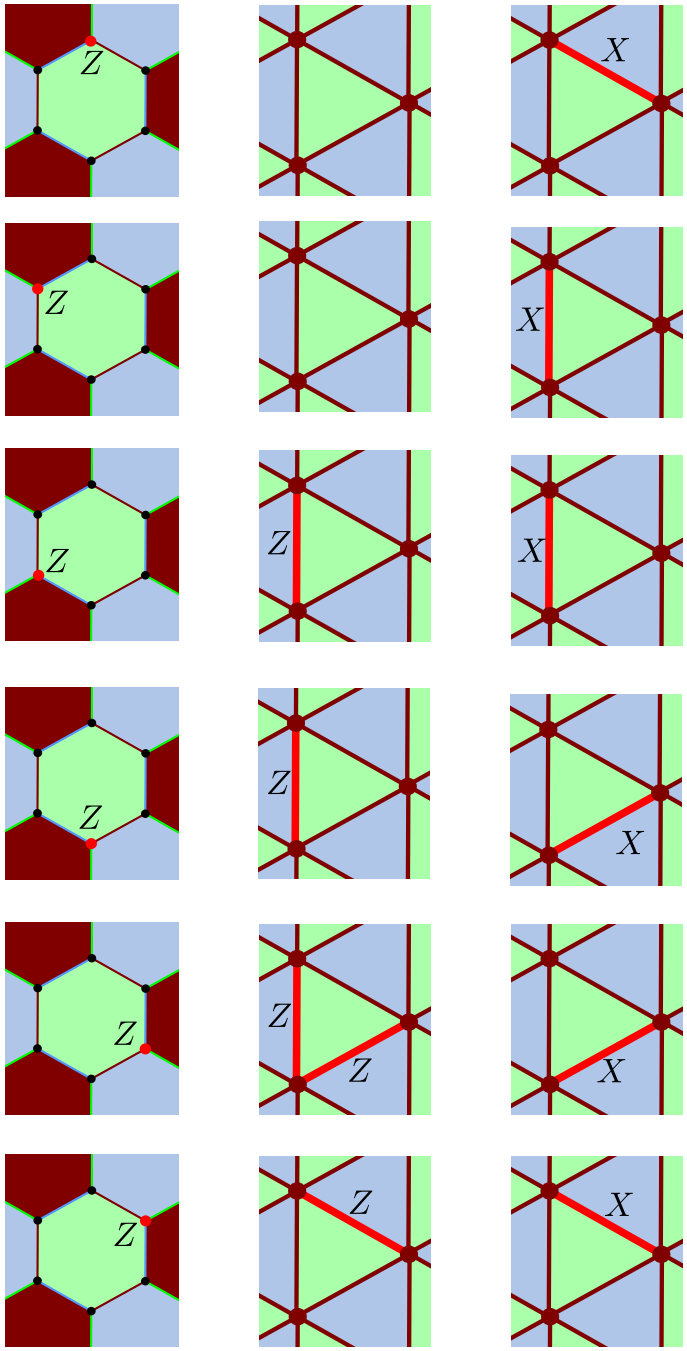


Figure 5.6: Mapping for Z type errors on $G1$, starting with Z_1, which is the single qubit Z operator chosen according to Lemma 4. The dependent hopping operator is $H_{2,3}^{\epsilon_{c'}}$, which clearly maps to a Z stabilizer on $[\tau(G1)]_1$, as in Eq. 4.9.

CHAPTER 6

Error models and simulations

One of the main objectives of finding a map between color codes and surface codes is to allow for the efficient decoding of color codes. In that regard, we will derive some analytic expressions for the error probabilities on the surface code given error probabilities on the color code. This will enable us to assign the correct weights for qubits on the surface code when the matching decoder is used for decoding. An overview of the decoding techniques used is given as pseudo-code and finally the simulation results are presented.

At the moment, for our analysis, we have only used the bit flip and phase flip channels. In subsequent work, we hope to calculate error probabilities and perform simulations for the depolarizing channel as well.

6.1 Analytic error probabilities

For the rest of this section, $p_{i,\sigma}$ denotes the probability of an error $\sigma \in I, X, Y, Z$ occuring on qubit i. For notational convenience, and in departure from the notation in the rest of the thesis, qubits on the color code are labeled using letters while those on the surface codes using digits. We assume X and Z-type errors occur independently of one another on each qubit in the color code and calculate the probabilities of error on each qubit of the surface codes based on the map in (refer to section on Examples). For each qubit, we assume the errors that occur on it are independent of the other qubits and calculate both the marginal error probability and the exact single qubit error probability. In our calculations, we assume that the probabilities of error on the qubits of the color code are not equal, allowing for flexibility in the overall color code error model we consider. The expressions are derived only for a son

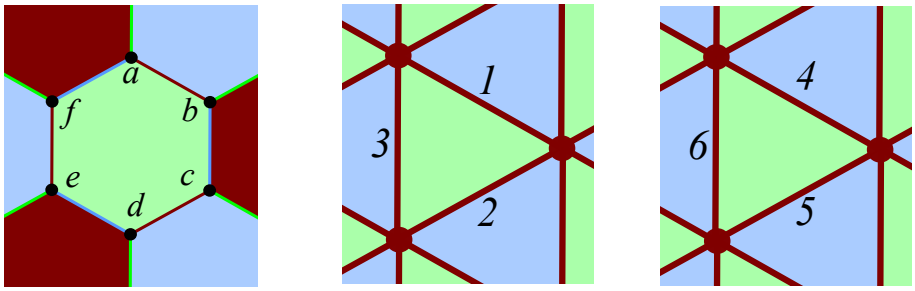


Figure 6.1: One plaquette of the hexagonal color code mapped to two copies of the triangular surface code. Note the labeling of qubits used to derive the error probabilities.

6.1.1 Marginal errors: hexagonal lattice

The error probabilities for each qubit on the two copies of the triangular surface codes are given below:

$$p_{1,Z} = p_{b,Z}, \tag{6.1}$$

$$p_{2,Z} = p_{c,Z}, \tag{6.2}$$

$$p_{3,Z} = p_{c,Z}(1-p_{d,Z})(1-p_{e,Z}) + p_{d,Z}(1-p_{c,Z})(1-p_{e,Z}) + p_{e,Z}(1-p_{c,Z})(1-p_{d,Z}) \tag{6.3}$$

$$+ p_{c,Z}p_{d,Z}p_{e,Z}, \tag{6.4}$$

$$p_{4,Z} = p_{a,X}, \tag{6.5}$$

$$p_{5,Z} = p_{d,X}(1-p_{e,X})(1-p_{f,X}) + p_{e,X}(1-p_{d,X})(1-p_{f,X}) + p_{f,X}(1-p_{d,X})(1-p_{e,X}) \quad + p_{d,X}p_{e,X}p_{f,X}, \tag{6.6}$$

$$p_{6,Z} = p_{f,X}, \tag{6.7}$$

$$p_{1,X} = p_{a,X}(1-p_{b,X}) + p_{b,X}(1-p_{a,X}), \tag{6.8}$$

$$p_{2,X} = p_{c,X}(1-p_{d,X}) + p_{d,X}(1-p_{c,X}), \tag{6.9}$$

$$p_{3,X} = p_{e,X}(1-p_{f,X}) + p_{f,X}(1-p_{e,X}), \tag{6.10}$$

$$p_{4,X} = p_{a,Z}(1-p_{b,Z}) + p_{b,Z}(1-p_{a,Z}), \tag{6.11}$$

$$p_{5,X} = p_{c,Z}(1-p_{d,Z}) + p_{d,Z}(1-p_{c,Z}), \tag{6.12}$$

$$p_{6,X} = p_{e,Z}(1-p_{f,Z}) + p_{f,Z}(1-p_{e,Z}). \tag{6.13}$$

$$\tag{6.14}$$

6.1.2 Exact single qubit errors: hexagonal lattice

Exact single qubit error probabilities for each qubit on the two copies of the triangular surface codes are given below. These give the probability that a particular X or Z error occurs on only

the qubit under consideration and all other qubits are error free:

$$p_{1,Z} = p_{a,Z}p_{b,Z}(1-p_{c,Z}-p_{c,X})(1-p_{d,Z}-p_{d,X})(1-p_{e,Z}-p_{e,X})(1-p_{f,Z}-p_{f,X}), \tag{6.15}$$

$$p_{2,Z} = p_{c,Z}p_{d,Z}(1-p_{a,Z}-p_{a,X})(1-p_{b,Z}-p_{b,X})(1-p_{e,Z}-p_{e,X})(1-p_{f,Z}-p_{f,X}), \tag{6.16}$$

$$p_{3,Z} = p_{e,Z}p_{f,Z}(1-p_{a,Z}-p_{a,X})(1-p_{b,Z}-p_{b,X})(1-p_{c,Z}-p_{c,X})(1-p_{d,Z}-p_{d,X}), \tag{6.17}$$

$$p_{4,Z} = p_{a,X}p_{b,X}(1-p_{c,Z}-p_{c,X})(1-p_{d,Z}-p_{d,X})(1-p_{e,Z}-p_{e,X})(1-p_{f,Z}-p_{f,X}), \tag{6.18}$$

$$p_{5,Z} = p_{c,X}p_{d,X}(1-p_{a,Z}-p_{a,X})(1-p_{b,Z}-p_{b,X})(1-p_{e,Z}-p_{e,X})(1-p_{f,Z}-p_{f,X}), \tag{6.19}$$

$$p_{6,Z} = p_{e,X}p_{f,X}(1-p_{a,Z}-p_{a,X})(1-p_{b,Z}-p_{b,X})(1-p_{c,Z}-p_{c,X})(1-p_{d,Z}-p_{d,X}), \tag{6.20}$$

$$p_{1,X} = p_{b,X}(1-p_{a,Z}-p_{a,X})(1-p_{c,Z}-p_{c,X})(1-p_{d,Z}-p_{d,X})(1-p_{e,Z}-p_{e,X})(1-p_{f,Z}-p_{f,X}), \tag{6.21}$$

$$p_{2,X} = p_{c,X}(1-p_{a,Z}-p_{a,X})(1-p_{b,Z}-p_{b,X})(1-p_{d,Z}-p_{d,X})(1-p_{e,Z}-p_{e,X})(1-p_{f,Z}-p_{f,X}), \tag{6.22}$$

$$p_{3,X} = p_{c,X}p_{d,X}p_{e,X}(1-p_{a,Z}-p_{a,X})(1-p_{b,Z}-p_{b,X})(1-p_{f,Z}-p_{f,X}), \tag{6.23}$$

$$p_{4,X} = p_{a,Z}(1-p_{b,Z}-p_{b,X})(1-p_{c,Z}-p_{c,X})(1-p_{d,Z}-p_{d,X})(1-p_{e,Z}-p_{e,X})(1-p_{f,Z}-p_{f,X}), \tag{6.24}$$

$$p_{5,X} = p_{d,X}p_{e,X}p_{f,X}(1-p_{a,Z}-p_{a,X})(1-p_{b,Z}-p_{b,X})(1-p_{c,Z}-p_{c,X}), \tag{6.25}$$

$$p_{6,X} = p_{f,Z}(1-p_{f,Z}-p_{f,X})(1-p_{b,Z}-p_{b,X})(1-p_{c,Z}-p_{c,X})(1-p_{d,Z}-p_{d,X})(1-p_{e,Z}-p_{e,X}). \tag{6.26}$$

(6.27)

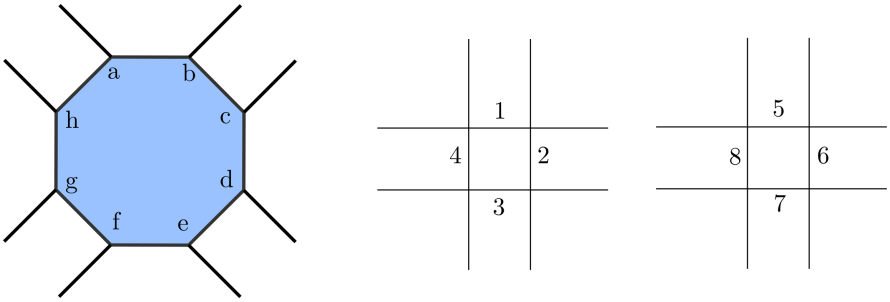


Figure 6.2: One plaquette of the square-octagonal color code mapped to two copies of Kitaev's toric code. Note the labeling of qubits used to derive the error probabilities.

6.1.3 Marginal errors: square octagonal lattice

The error probabilities for each qubit on the two copies of Kitaev's toric code are given below:

$$p_{1,Z} = p_{a,Z}, \tag{6.28}$$

$$p_{2,Z} = p_{d,Z}(1-p_{e,Z})(1-p_{f,Z})(1-p_{g,Z})(1-p_{h,Z}) + p_{e,Z}(1-p_{d,Z})(1-p_{f,Z})(1-p_{g,Z})(1-p_{h,Z}) \tag{6.29}$$

$$+ p_{f,Z}(1-p_{d,Z})(1-p_{e,Z})(1-p_{g,Z})(1-p_{h,Z}) + p_{g,Z}(1-p_{d,Z})(1-p_{e,Z})(1-p_{f,Z})(1-p_{h,Z}) \tag{6.30}$$

$$+ p_{h,Z}(1-p_{d,Z})(1-p_{e,Z})(1-p_{f,Z})(1-p_{g,Z}) \tag{6.31}$$

$$p_{3,Z} = p_{f,Z}(1-p_{g,Z})(1-p_{h,Z}) + p_{g,Z}(1-p_{f,Z})(1-p_{h,Z}) \tag{6.32}$$

$$+ p_{h,Z}(1-p_{f,Z})(1-p_{g,Z}), \tag{6.33}$$

$$p_{4,Z} = p_{h,Z}, \tag{6.34}$$

$$p_{5,Z} = p_{b,X}, \tag{6.35}$$

$$p_{6,Z} = p_{c,X}, \tag{6.36}$$

$$p_{7,Z} = p_{c,X}(1-p_{d,X})(1-p_{e,X}) + p_{d,X}(1-p_{e,X})(1-p_{c,X}) + p_{e,X}(1-p_{c,X})(1-p_{d,X}) + p_{c,X}p_{d,X}p_{e,X} \tag{6.37}$$

$$p_{8,Z} = p_{d,X}(1-p_{e,X})(1-p_{f,X})(1-p_{g,X})(1-p_{c,X}) + p_{e,X}(1-p_{d,X})(1-p_{f,X})(1-p_{g,X})(1-p_{c,X}) \tag{6.38}$$

$$+ p_{f,X}(1-p_{d,X})(1-p_{e,X})(1-p_{g,X})(1-p_{c,X}) + p_{g,X}(1-p_{d,X})(1-p_{e,X})(1-p_{f,X})(1-p_{c,X}) \tag{6.39}$$

$$+ p_{c,X}(1-p_{d,X})(1-p_{e,X})(1-p_{f,X})(1-p_{g,X}), \tag{6.40}$$

$$p_{1,X} = p_{a,X}(1-p_{b,X}) + p_{b,X}(1-p_{a,X}), \tag{6.41}$$

$$p_{2,X} = p_{c,X}(1-p_{d,X}) + p_{d,X}(1-p_{c,X}), \tag{6.42}$$

$$p_{3,X} = p_{e,X}(1-p_{f,X}) + p_{f,X}(1-p_{e,X}), \tag{6.43}$$

$$p_{4,X} = p_{g,X}(1-p_{h,X}) + p_{h,X}(1-p_{g,X}), \tag{6.44}$$

$$p_{5,X} = p_{a,Z}(1-p_{b,Z}) + p_{b,Z}(1-p_{a,Z}), \tag{6.45}$$

$$p_{6,X} = p_{c,Z}(1-p_{d,Z}) + p_{d,Z}(1-p_{c,Z}), \tag{6.46}$$

$$p_{7,X} = p_{e,Z}(1-p_{f,Z}) + p_{f,Z}(1-p_{e,Z}), \tag{6.47}$$

$$p_{8,X} = p_{g,Z}(1-p_{h,Z}) + p_{h,Z}(1-p_{g,Z}). \tag{6.48}$$

$$\tag{6.49}$$

The individual qubit errors can be calculated using a similar procedure as that for the hexagonal code.

6.2 Decoding procedure

We use our map from color codes to surface codes coupled with the perfect matching decoding algorithm [5] in order to decode errors on the color code. The perfect matching algorithm returns the minimum weight error corresponding to a syndrome s_{surf}, which is the collection of endpoints of the error. The algorithm given below assumes that errors on the color code are all of Z type, i.e. they come from a phase flip channel. An identical procedure can be used if the errors are all of X type, i.e. they come from a bit flip channel.

6.2.1 Simulations

A C++ program based on the procedure outlined in Algorithm 2 was used to carry out simulations to determine the error threshold. We used the Boost C++ libraries [13] and the Blossom V [8] matching decoding algorithm for our simulations. As a trial, we first obtained the threshold for Kitaev's toric code with phase flip errors. Our simulations returned a threshold of 11%.

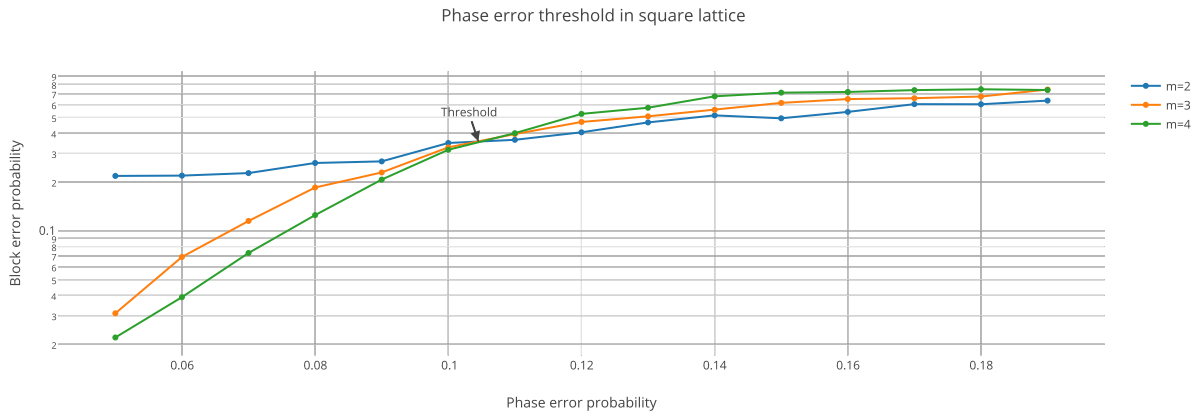


Figure 6.3: Phase decoding performance of standard matching decoding with uniform error probabilities for toric code lattices of size 2.2^m. Threshold obtained is approximately 11%.

Algorithm 2 Decoding a 2D color code

Input: A 2-colex Γ without parallel edges; Γ is assumed to have a 2-cell embedding; error probability p_i of phase flip channel for qubit i on color code where $i \in (1 \cdots, 16)$

1: Create a model of the color code with qubits as vertices and faces as stabilizers.
2: For each qubit i, generate an Z error with probability p_i independent of the other qubits to obtain the overall error E_Z.
3: If a qubit has an error, set its associated vertex's value to 1. Else, set it to 0.
4: **for** $f \in F_G$ **do**
5: Do a $GF(2)$ sum over all the vertices contained in f.
6: **if** sum=0 **then**
7: The face has a trivial syndrome which we set to 0 in the syndrome vector s_col
8: **else if** sum=1 **then**
9: The face has a non-trivial syndrome which we set to 1 in the syndrome vector s_col
10: **end if**
11: **end for**
12: Create a two models of the surface code with the qubits as edges and X type stabilizers as vertices.
13: Map the syndrome vector on Γ to the two models of a surface code, Γ_1 and Γ_2 using the syndrome map derived from Algorithm 1.
14: **for** $\Gamma_i, i \in \{1, 2\}$ **do**
15: Assign appropriate edge weights to each edge e based on the expressions in Section 6.1.
16: Collect all vertices which have a non-trivial syndrome in a set V_s.
17: Use Djikstra's algorithm to find the shortest distance between each pair of vertices and store the edges comprising the shortest path for each pair.
18: Create a complete graph G_d from all the vertices $v \in V_s$ with edge weights corresponding to the distance obtained in the previous step.
19: Run the matching algorithm on the distance graph G_d to get a matching M.
20: Look up the stored shortest paths from Step on the lattice corresponding to the edges of M.
21: Find the edges associated with the symmetric difference of all the shortest path from Step 14. This is the required minimum weight error pattern E_{min}.
22: Use the inverse map to determine the error $E_{est}^{\Gamma_i}$ on the color code Γ corresponding to E_{min}.
23: **end for**
24: Combination of the two inverted error patterns $E_{est}^{\Gamma_1}$ and $E_{est}^{\Gamma_2}$ will give the corrected errorE_{est}on Γ.
25: **if** E_{est} is a homologically trivial cycle **then**
26: Decoding is successful.
27: **else if** E_{est} is a homologically non-trivial cycle **then**
28: Decoding has failed.
29: **end if**

REFERENCES

[1] **Bombin, B.** and **M. A. Martin-Delgado** (2006). Topological quantum distillation. *Phys. Rev. Lett.*, **97**(180501).

[2] **Bombin, H.**, **G. Duclos-Cianci**, and **D. Poulin** (2012). Universal topological phase of two-dimensional stabilizer codes. *New J. Phys. 14, 073048 (2012). of Computation*, **14**, 073048.

[3] **Christine, V.** (2015). Ibm scientists achieve critical steps to building first practical quantum computer. http://www-03.ibm.com/press/us/en/pressrelease/46725.wss#resource. Accessed: 2015-06-21.

[4] **Delfosse, N.** (2014). Decoding color codes by projection onto surface codes. *Phys. Rev. A*, **89**, 012317.

[5] **Dennis, E.**, **A. Kitaev**, **A. Landahl**, and **J. Preskill** (2002). Topological quantum memory. *J. Math. Phys.*, **43**, 4452–4505.

[6] **Duclos-Cianci, G.** and **D. Poulin** (2010). Fast decoders for topological codes. *Phys. Rev. Lett.*, **104**(050504).

[7] **Kitaev, A.** (2003). Fault-tolerant quantum computation by anyons. *Annals. of Physics*, **303**, 2–30.

[8] **Kolmogorov, V.** (2009). Blossom v: A new implementation of a minimum cost perfect matching algorithm. *Mathematical Programming Computation (MPC)*, 43–67.

[9] **Kubica, A.**, **B. Yoshida**, and **F. Pastawski** (2015). Unfolding color codes. ArXiv:1503.02065.

[10] **Nielsen, M.** and **L. Chuang**, *Quantum Computation and Quantum Information*. 2010.

[11] **Raussendorf, R.** and **J. Harrington** (2001). Fault-tolerant quantum computation with high threshold in two dimensions. *Phys. Rev. Lett.*, **98**(150504).

[12] **Sarvepalli, P.** and **R. Raussendorf** (2012). Efficient decoding of topological color codes. *PHYSICAL REVIEW A*, **85**, 022317.

[13] **Siek, J.**, *The Boost Graph Library*. 2002.

[14] **Yoshida, B.** (2011). Classification of quantum phases and topology of logical operators in an exactly solved model of quantum codes. *Annals of Physics*, **326**, 15–95.