The Thesis Committee for Josey Hanish
certifies that this is the approved version of the following thesis:

Decoding Pauli-Z Errors on the 3-Dimensional Tetrahedral Color Code with Boundaries

APPROVED BY

SUPERVISING COMMITTEE:

Brian La Cour, Supervisor

Greg Sitz, Honors Advisor in Physics

Decoding Pauli-Z Errors on the 3-Dimensional Tetrahedral Color Code with Boundaries

Department of Physics

Josey Hanish, Author

Date

Brian La Cour, Supervisor

Date

Decoding Pauli-Z Errors on the 3-Dimensional Tetrahedral Color Code with Boundaries

by

Josey Hanish

Thesis

Presented to the Faculty of the College of Natural Sciences of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Bachelor of Science

Dean's Scholars Honors Degree in Physics

The University of Texas at Austin

May 2020

Acknowledgements

This thesis is about my contribution to a project undertaken by a research group at Applied Research Laboratories: University of Texas at Austin from Fall 2018 to Spring 2020.

I would like to thank my coworkers at ARL:UT; this project would not have been possible without them. Skylar Turner guided the direction of this project, did much of the background research, and wrote code to generate physical errors, implement the improved lift procedure, and check for logical errors. Skylar also suggested running the MWPM algorithm on the restricted lattice instead of the full lattice and adapted the Z-error decoder to do so. Eion Blanchard wrote code to generate the lattice geometries and to implement the X-error decoder. Noah Davis and Brian La Cour provided valuable insights. Brian also secured funding for this project and provided appreciated feedback on drafts of this thesis.

My contribution was to write code to implement the Z-error decoder, to find the rank of a binary matrix, and to revise the code for parallel computing. I also submitted such parallel jobs to the supercomputer queue and retrieved results.

This work was supported by the Air Force Research Laboratory under Grant No. FA8750-18-1-0042. Large-scale numerical calculations were possible thanks to an allocation from the Texas Advanced Computing Center.

Josey Hanish

The University of Texas at Austin
May 2020

Abstract

Decoding Pauli-Z Errors on the 3-Dimensional Tetrahedral Color Code with Boundaries

Josey Hanish

The University of Texas at Austin, 2020

Supervisor: Brian La Cour

In quantum computers, each logical qubit must be encoded in several physical qubits to protect against noise in physical qubits. The tetrahedral color code is such an encoding. In the primal lattice, the three-dimensional tetrahedral color code is a bitruncated octahedral lattice, and in the dual lattice, the tetrahedral color code is a four-colorable body-centered cubic lattice. This color code can be utilized for measurement-based quantum computing, for which all entanglement is present in a cluster state at the beginning of the computation and gate operations are performed by local measurements and classical operations. Moreover, this color code admits a gate set that is both transversal, i.e., realized by qubit-wise operations, and universal

when supplemented by measurement and classical computing. During cluster state preparation and computation, errors may occur on the physical qubits. Therefore, it is necessary to have a decoder that, using the syndrome of these errors, finds a set of qubits to correct and performs the operations needed to correct those qubits. The decoder is considered successful if and only if the correction does not cause a logical error on the logical qubit. In this thesis, I present a decoder for Pauli Z, or phase-flip, errors on the three-dimensional tetrahedral color code with nonperiodic boundaries. This decoder for Z-errors includes as a subroutine a decoder for Pauli X errors. The decoder uses a restriction procedure to map the tetrahedral color code to a toric code. The toric code is another quantum error correcting code. Then, an existing toric code decoder interprets the error syndrome. This research on the bounded color code builds upon previous work on the unbounded color code and the two-dimensional color code. Under independently identically distributed noise, evidence indicates an error probability threshold for Z-errors between 0.01% and 0.02% and for X-errors between 2.5% and 3.3%. I also present example errors and illustrate how the decoder attempts to correct those errors.

Table of Contents

Chapter 1

Introduction

Qubits are to a quantum computer as classical bits are to a classical computer. Whereas classical bits can attain only two states, 0 or 1, a qubit can attain any linear combination of these states, $a\,|0\rangle + b\,|1\rangle$, where $a, b \in \mathbb{C}$ and $|a|^2 + |b|^2 = 1$. In fact, in the formalism of quantum computing, $|0\rangle$ and $|1\rangle$ are orthonormal basis vectors. If $a, b \in \mathbb{R}$, a qubit can be visualized as a vector pointing from the origin to a point on the unit circle, where a is the x-coordinate and b is the y-coordinate.

By a change of basis, this plane can be reparameterized to the second canonical basis, $\left\{|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}, |-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}\right\}$. Often, for neatness, the normalization factor of $\sqrt{2}$ is omitted when writing the states. The normalization can easily be found by taking the inner product of the state with itself. I will make that omission throughout this thesis.

Measuring a qubit in some orthogonal basis makes the qubit "snap to" one of the orthogonal basis vectors. For instance, measuring a qubit in the $\{\ |+\rangle,\ |-\rangle\ \}$ basis will not only give a result of either $|+\rangle$ or $|-\rangle$, the qubit will have actually moved from its original state to $|+\rangle$ or $|-\rangle$, respectively. Note that this will destroy all information about how far the qubit was from $|0\rangle$ and $|1\rangle$.

Another advantage of qubits over classical bits is that qubits can be entangled with each other. The state of two classical bits can be one of 00, 01, 10, or 11. But two qubits can be in any state described by $a\,|00\rangle + b\,|01\rangle + c\,|10\rangle + d\,|11\rangle$, where $|a|^2 + |b|^2 + |c|^2 + |d|^2 = 1$. For instance, two qubits can be in the state $|00\rangle + |11\rangle$. Note that this state is not separable - it cannot be written as a tensor product[1] of

[1]Tensor product: $\otimes : \mathbb{C}^2 \times \mathbb{C}^2 \longrightarrow \mathbb{C}^4, [a, b]^T \otimes [c, d]^T = [ac, ad, bc, bd]^T$

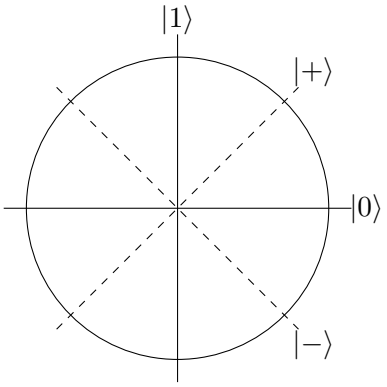


Figure 1.1: The two canonical orthogonal bases in quantum computing: $\{\ |0\rangle,\ |1\rangle\ \}$ and $\{\ |+\rangle,\ |-\rangle\ \}$.

two qubits. This state is therefore entangled. In fact, this state is one of the Bell states, the maximally entangled states of two qubits.

When the entangled state $|00\rangle+|11\rangle$ is measured (in the $\{\ |00\rangle,\ |01\rangle,\ |10\rangle,\ |11\rangle\ \}$ basis), the state will "snap to" either $|00\rangle$ or $|11\rangle$. The entanglement is lost, because the state is now separable. For example, if the state "snaps to" $|00\rangle$, then the state can be written as $\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]\otimes\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$ So, if one wants to correct errors in some entangled state, one cannot simply measure the state - that will destroy the entanglement.

1.1 Quantum Error Correction

Every physical system that can be used to encode qubits is vulnerable to noise. Solid state qubits are subject to thermal noise, and photonic qubits are subject to photon loss or noisy communication channels. Since reaching absolute zero or manufacturing perfectly flawless fiber-optic cables is impossible, qubits must be encoded in a way that makes the information resistant to this noise. Such an encoding is called a quantum error-correcting code. In a quantum error-correcting code, a logical qubit is encoded in $n > 1$ code (physical) qubits. A logical qubit is denoted $|\psi\rangle_L$.

A simple example of a quantum error-correcting code is the three-qubit repeating code[2] [1, 2]. Let $|0\rangle_L = |000\rangle$ and $|1\rangle_L = |111\rangle$ such that a general state is $a\,|000\rangle + b\,|111\rangle$. Alice transmits this state through a noisy channel that may

[2]The example in this paragraph is adapted from [1].

perform a bit-flip operation on a code qubit. The three one-qubit error states are $a\,|100\rangle + b\,|011\rangle$, $a\,|010\rangle + b\,|101\rangle$, and $a\,|001\rangle + b\,|110\rangle$. Bob cannot measure the qubits directly, because that would destroy the superposition. Instead, Bob must use two ancillary qubits to find a syndrome of the error. A syndrome gives information about the relationships between the qubits without measuring the values of the qubits themselves, so the state is preserved. Here, Bob initializes his ancillas to $|0\rangle$. He applies a Controlled-NOT gate with the first code qubit as the control and the first ancilla as the target. He applies another Controlled-NOT gate with the second code qubit as the control and the first ancilla again as the target. Now, if the values of the first two code qubits are the same, the ancilla is $|0\rangle$. If the values are different, the ancilla is $|1\rangle$. Bob repeats this procedure with the second and third code qubits and the second ancilla. With these two bits of ancillary information, Bob can figure out which one code qubit, if any, experienced a bit-flip error. Given at least five code qubits, this scheme can be extended to protect against phase-flip errors as well [2].

There are infinitely many errors that could happen to a qubit, because there are infinitely many unitary operations - the space of unitary operations is continuous. Fortunately, there exists a result that shows that if an error-correction scheme can correct X and Z errors, it can correct any single-qubit error [2].

1.1.1 The Stabilizer Formalism

Often in quantum error correction, quantum states are described with the stabilizer formalism instead of with the wavefunction. The wavefunction representation describes the state as a function ψ of space and time (or momentum and time) such that the Schroedinger equation, $\hat{H}\,|\psi(\vec{x},t)\rangle = i\hbar\frac{\partial}{\partial t}\,|\psi(\vec{x},t)\rangle$, is satisfied. The stabilizer formalism describes the same quantum state with a set S of stabilizer operators. The quantum state is the state that, when operated on by operators from S, stays the same.

For example, recall the state $|00\rangle + |11\rangle$. Let X_i be the X, or NOT, gate operating on the i-th qubit. $X_1X_2(|00\rangle + |11\rangle) = |00\rangle + |11\rangle$, so the state is stabilized by X_1X_2 [3]. This state is also stabilized by Z_1Z_2, and in fact, the stabilizer set $\{\, X_1X_2, Z_1Z_2 \,\}$ is sufficient to identify this state [3].

In Section 1.3.1, the tetrahedral color code lattice is described by its stabi-

lizers. Thus, the tetrahedral color code is a "stabilizer code."

1.2 Topological Codes

The tetrahedral color code is also a topological code. In a topological code, the quantum information of the logical qubit is nonlocally distributed among the code qubits; the information is stored in global degrees of freedom [4]. Topological systems protect quantum information naturally because of the energy gap between the ground state and the excited states [5].

Furthermore, the three-dimensional tetrahedral color code allows topological quantum computation without quasiparticles (anyons) or "braiding," the arrangement of quasiparticles in space to perform logic gates [5]. (In fact, "Topological Computation without Braiding" was the title of the original paper on this code.) Since anyons are less well-understood and more difficult to produce in the lab than solid-state or photonic qubits, topological quantum computation without quasiparticles is desirable [6].

Finally, the three-dimensional tetrahedral color code allows universal[3] computation without magic state distillation, a computationally costly operation that allows universal computation in two-dimensional systems [7].

1.3 Characteristics of the Tetrahedral Color Code Lattice

1.3.1 Geometry

The primal lattice $\mathcal{L}$ is a tetrahedral slice of the bitruncated octahedral lattice. The primal lattice represents the physical configuration of the qubits. On the primal lattice, qubits are vertices and edges are connections between qubits. All qubits are 4-valent (thus tetrahedral), connected to 4 other qubits, except for the qubits at the corners of the primal bounding tetrahedron, which are 3-valent. Z-stabilizers are on the faces of the primal lattice and X-stabilizers are on the cells. One color is associated with each edge of the primal lattice, two colors are associated with each face, and one color is associated with each cell and each facet. A facet is one of

[3]Defined in Section 1.3.3

the four triangular faces of the primal bounding tetrahedron. The color associated with a cell is the color that is *absent* from its bounding edges; the facets are labeled likewise; i.e. cells and facets are four-colorable. A face is labeled by the two colors that are *not* present in the two *cells* it connects, which is equivalent to labeling a face by the color of its edges.

The dual lattice $\mathcal{L}^*$ allows for a more intuitive interpretation of the decoder algorithms. The dual lattice is a slice of the body-centered cubic lattice, along with four boundary vertices that are connected to the bulk vertices. The four boundary vertices correspond to the four primal facets. On the dual lattice, qubits are tetrahedra. This includes the tetrahedra for which 1, 2, or 3 vertices are a boundary vertex, but not the tetrahedron where all four vertices are the boundary vertices. Z-stabilizers are on edges, and X-stabilizers are on vertices. One color is associated with each vertex, i.e., the vertices are four-colorable. Edges are labeled by two colors each, the colors that are not present on the endpoints of the edge. The fact that the vertices are four-colorable is essential to our implementation of the Restriction Decoder [8].

Primal Lattice $\mathcal{L}$			Dual Lattice $\mathcal{L}^*$		
Qubits	Vertices	0D	Qubits	Tetrahedra	3D
Connections	Edges	1D	Connections	Faces	2D
Z-stabilizers	Faces	2D	Z-stabilizers	Edges	1D
X-stabilizers	Cells	3D	X-stabilizers	Vertices	0D
Boundary	Facets	2D	Boundary	Vertices	0D

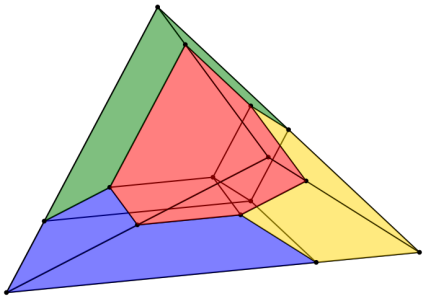
Table 1.1: Corresponding objects in the primal and dual lattices.

In the simplest possible lattice, shown in Figure 1.2, each face is adjacent to four qubits and each cell is adjacent to eight qubits. In larger lattices, a face can be adjacent to up to six qubits and a cell can be adjacent to up to 24 qubits.

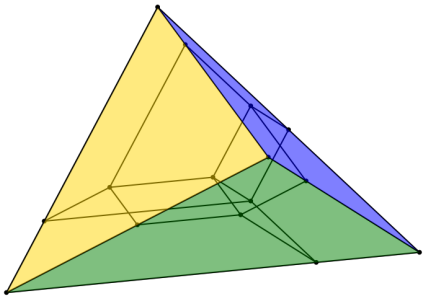
1.3.2 Error Syndromes

A X-error is also known as a bit-flip. The action of an X-error on a physical qubit in the state $|\psi\rangle = a|0\rangle + b|1\rangle$ is $X|\psi\rangle = \left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]\left[\begin{smallmatrix} a \\ b \end{smallmatrix}\right] = \left[\begin{smallmatrix} b \\ a \end{smallmatrix}\right]$. The action of a Z-error, also known as a phase-flip, is $Z|\psi\rangle = \left[\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right]\left[\begin{smallmatrix} a \\ b \end{smallmatrix}\right] = \left[\begin{smallmatrix} a \\ -b \end{smallmatrix}\right]$. X and Z are the Pauli-X and Pauli-Z matrices σ_X, σ_Z.

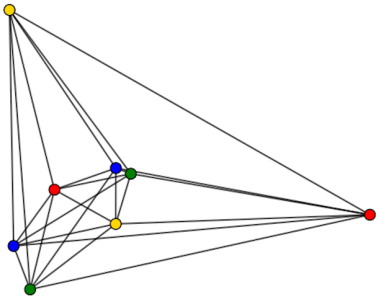
A Z-stabilizer will show a syndrome if an odd number of the physical qubits



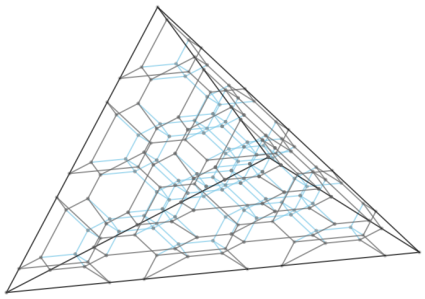
(a) The simplest possible primal lattice (distance 3) with cell colors shown.



(b) The simplest possible primal lattice with facet colors shown. This lattice is shown with the same spatial orientation as Figure 1.2a.



(c) The dual to the distance-3 lattice with vertex colors shown. The four interior vertices correspond to primal cells, and the four exterior (boundary) vertices correspond to primal facets.



(d) The distance-7 primal lattice. Here, the bitruncated octahedral structure is more apparent. Black lines are the edges of the bounding tetrahedron, gray lines are edges on a facet, and blue lines are interior edges.

Figure 1.2

adjacent to that face have experienced an X-error. In the dual, Z-stabilizers with a syndrome can be interpreted as edge-like excitations. Likewise, an X-stabilizer will show a syndrome if an odd number of the physical qubits adjacent to that cell has experienced a Z-error. This is a point-like excitation in the dual.

When the quantum state is in the code space, the measurement of a stabilizer operator is positive one, $\hat{S}\,|\psi\rangle = +\,|\psi\rangle$. When an error brings the state out of the code space, the measurement will return some other value. This is the physical interpretation of a syndrome on a stabilizer.

1.3.3 Gate Set

Gates are operations that transform a quantum state into another quantum state. For example, the X, or NOT, gate transforms the state $|0\rangle$ into $|1\rangle$ and vice versa. The three-dimensional tetrahedral color code admits the gates:

- $T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
- $CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
- $CP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
- $P = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$

Details of how these gates are applied can be found in [9].

Uniquely, this gate set is both transversal and universal when supplemented by measurement and classical computing. Universal means that any logical operation can be approximated to an arbitrary precision by applying only gates from the set. Transversal means that the gate is applied qubitwise - the gate can be applied to the logical qubit by applying a gate to each code qubit individually; the gate

on the logical qubit is a tensor product of gates on the physical qubits. (See [9]). The Eastin-Knill theorem states that a quantum error correcting code cannot have a gate set that is both universal and transversal [10]. But in our case, the allowed operations are not restricted to the gate set; measurement and classical computing are also permitted. Thus, the Eastin-Knill theorem is circumvented.

1.4 A Use of the Tetrahedral Color Code: Measurement-Based Quantum Computing

In measurement-based quantum computing (MBQC), proposed by Robert Raussendorf in 2001, all entanglement exists as a resource at the beginning of the computation. This entangled state is called a cluster state. By measuring (logical) qubits in the cluster state, the state is "sculpted" into the final result. The measurement-based model is equivalent to the circuit model, although it is not always trivial to translate between the two. The 3D tetrahedral color code is well-suited to MBQC because the operations required for MBQC are quantum-local in this code [7].

Another challenge is that the basis in which some logical qubit is measured may be affected by the result of previous measurements. The proper basis must be calculated by a classical computer while the quantum state waits. This leaves time for errors to accumulate on the quantum state, which is the source of the iid errors investigated in this thesis and in [11][4].

1.5 Previous Work

Robert Raussendorf introduced measurement-based quantum computing in 2001, suggesting implementation on systems that realize Ising-type interactions [12].

The original topological code was Kitaev's surface/toric code, introduced in 1997 [13, 4]. The only gate that is transversal for the surface code is CNOT[5] [5].

Hèctor Bombìn introduced the tetrahedral color code in 2007 [5]. In the same paper, he showed that the code admits a universal gate set. In 2018, he published another transversal, universal gate set for this code and wrote about error propagation under those operations [9]. He also proposed using the three-dimensional

[4] [11] is the paper written on this topic by the author's group in parallel with this thesis.

[5] The Controlled-NOT gate, which entangles two qubits.

tetrahedral color code to implement measurement-based quantum computing on a photonic architecture [7]. [7] also describes steps for preparing the 3D tetrahedral color code lattice with ancilla qubits.

In this work, two decoders are used, both published by Aleksander Kubica. In 2019, Kubica introduced a method to map the color code onto the toric code, called the Restriction Decoder. The Restriction Decoder builds upon Kubica's Sweep Decoder (2018), a cellular automaton decoder for the toric code. A more accessible description of the Restriction Decoder can be found in [14].

Chapter 2

Z-Error Decoding Algorithm

Recall that, on the dual lattice, Z-errors are indicated by X-syndromes, which correspond to vertices, while qubits correspond to tetrahedra. The goal of the decoder is to find a set of tetrahedra for which errors on those tetrahedra would correspond to the vertex-like syndrome observed. Then, those qubits (tetrahedra) can be corrected by applying another phase-flip operation.[1] The same argument applies to X-errors, which can be corrected with the bit-flip operation.

In fact, the set of qubits that the decoder identifies to correct does not have to be identical to the set of qubits that originally experienced an error, as long as the set of original error qubits and the set of corrected qubits differ (XOR) only by a stabilizer. If the difference is not a stabilizer, the logical qubit is flipped - a logical error. Logical errors are considered decoder failures.

The general procedure this decoder uses is the following: (1) Twice-restrict the lattice (defined below). (2) Connect 0D vertices into 1D edges. (3) Connect 1D edges into 2D faces. (4) "Lift" 2D faces into 3D tetrahedra (qubits). (5) Return the union of sets of error qubits found from all possible restrictions.

2.1 Restriction

Recall that the vertices of the dual lattice are four-colorable such that no two vertices of the same color are connected. To restrict the lattice on a color $c \in \{ r, y, b, g \}$,

[1] $Z \cdot Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I_2$. All Pauli operators are involutions, $\sigma_i^2 = I$.

means to remove all vertices of that color and all edges connected to those vertices [8]. For X-errors, the lattice only needs to be restricted once. For Z-errors, the lattice must be restricted twice - first remove c_1, then remove c_2, such that $c_1, c_2 \in \{ r, y, b, g \}$ without replacement. The restriction procedure reduces the color code to the toric code [8, 14].

2.2 Minimum-Weight Perfect Matching

A minimum-weight perfect matching (MWPM) is a pairing of vertices on a weighted graph such that the edges in the pairing have the minimum possible weight [15]. The goal of this step is to pair syndrome vertices such that the edges of the error qubits (tetrahedra) can be reconstructed.

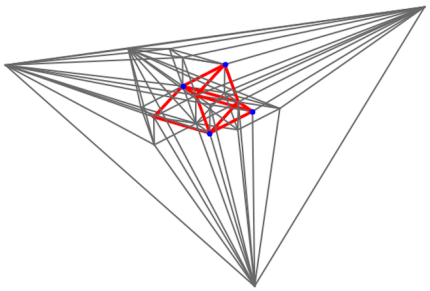
First, we restrict the syndrome to only those vertices that are present on the twice-restricted lattice.

Then, we convert the restricted lattice to a weighted graph for MWPM. The vertices of the MWPM graph are the syndrome vertices on the restricted lattice. We find the shortest path between each of the vertices in the restricted syndrome. The number of edges along that path becomes the weight of the edge between those vertices on the MWPM graph.

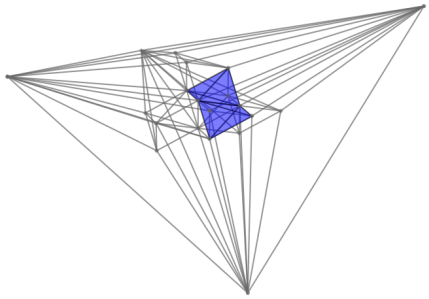
Recall that boundary vertices do not show syndromes, even though an error qubit could be adjacent to a boundary vertex. Essentially, syndrome information is missing on the boundary vertices. But if an error qubit is adjacent to the boundary vertex, the MWPM algorithm must be allowed to match to the boundary vertices in order to recover those edges of the tetrahedron where one endpoint is a boundary vertex. Therefore, the closest boundary vertex is submitted to the matching algorithm, even though it cannot show a syndrome. Moreover, since the lattice is twice-restricted, only two boundary vertices will remain.

Unsuccessful Approaches

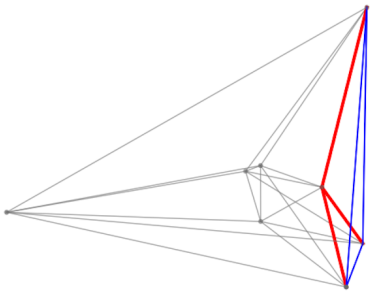
The first version of the code for this step matched the syndrome on the non-restricted lattice. This was unsuccessful because there are multiple shortest paths between vertices on the unrestricted lattice. The paths on the edges that would have been removed by restriction ended up as winglike triangles that should not have been part of the edge-like syndrome (Figure 2.1a). These "wings" were removed by identifying all vertices that were adjacent to only two edges in the edge-



(a) An example of a "wing" (left, red) and the point-like syndrome (blue).



(b) The original error tetrahedra that caused that syndrome. Note that the "wing" doesn't correspond with the edges of the error tetrahedra.



(c) A case where an edge-like syndrome ending at a boundary vertex is a legitimate syndrome.

Figure 2.1

like syndrome, then removing those two edges that were adjacent to that vertex. However, if such an inappropriate edge was adjacent to a boundary vertex, the "wing" removal would not find it. (Recall that syndromes do not appear on edges connecting boundary vertices.) This issue could not be fixed, because there are sometimes legitimate reasons for an edge to go to the boundary vertex and then stop - see the syndrome in Figure 2.1c.

That version of the code also sometimes matched bulk vertices to boundary vertices, when there was a more appropriate bulk vertex that should have been the match. This was imperfectly remedied by enforcing a weight-2 penalty for matchings between a bulk vertex and a boudnary vertex.

The Z-error decoder calls the X-error decoder, described in Sections 2.3 and 2.4, as a subroutine.[2] If the edge-like syndrome is not a valid syndrome of X-errors, the X-error decoder fails to find a set of qubits to correct, and therefore the Z-error decoder fails.

2.3 Sweep Decoder

Now we return to the full lattice in order to convert the edge-like syndrome to a mesh of triangular faces. This step and the lift prodecure in Section 2.4 are the same as in the X-error decoder; in fact, our implementation of the Z-error decoder calls the X-error decoder as a subroutine. First, the full lattice is once-restricted, so three colors remain. This procedure maps the color code to a toric code, so the toric sweep decoder can be used.

The sweep decoder is a cellular automaton that converts an edge-like syndrome to a mesh of (in this case triangular) faces [16]. The sweep operation operates in some direction that may not be parallel to any edge of the lattice. For each vertex in the restricted lattice that is adjacent to a syndrome edge, we select the smallest set of faces whose boundary matches the syndrome edges adjacent to that vertex.

For each vertex in the restricted lattice, we identify which syndrome edges are adjacent to that vertex. If any syndrome edges are adjacent to that vertex, we select the smallest set of faces in the sweep direction whose edges adjacent to that vertex match those adjacent syndrome edges. We flip the edges of those faces and recalculate the syndrome; i.e., edges that had a syndrome are now unexcited, and edges that did not have a syndrome are now excited. We repeat this procedure until the entire lattice has been swept through.

2.4 Lift Procedure

Next, we must "lift" the set of triangular faces to a set of tetrahedra [8]. One arbitrary color from $\{ r, g, b, y \}$ is chosen as the lift color. For each vertex of this color, tetrahedra are identified such that the faces of the triangular mesh adjacent to that vertex are included in the 2D boundary of those tetrahedra. Our lift procedure projects the triangular mesh to the facet, in the case of lifting on a boundary vertex,

[2] This is permissible for the reasons described in [14]

or to a topological sphere, in the case of lifting on a bulk vertex [11]. The projection is then decoded using the Peel Algorithm [17].

Unsuccessful Approaches

In the primal lattice, the number of qubits on a facet is proportional to the square of the code distance. Thus, in the dual, the number of qubits (tetrahedra) adjacent to a boundary vertex is also proportional to the square of the code distance.

The original Restriction Decoder was intended for the tetrahedral color code with periodic boundaries [8], which does not exhibit such quadratic scaling. Specifically, on the code with periodic boundaries, each vertex in the dual is adjacent to 24 or fewer qubits. Therefore, one can implement a naïve lift procedure by combinatorically scanning all possible combinations of qubits (tetrahedra) until a set of tetrahedra whose boundary locally matches the triangular mesh is found.

Because of the quadratic scaling on the lattice with non-periodic boundaries, this naïve approach was computationally unrealistic when lifting on the boundary vertices. Our solution to this problem, described above, was intended to fix the computational complexity of lifting on boundary vertices, but it turned out to work well on both boundary and bulk vertices.

2.5 Concluding the Algorithm and Remarks

We repeat the restrict-sweep-lift procedure of Sections 2.3 and 2.4 for three of the four colors. The remaining color is the "pivot" or "lift" color. We chose the pivot color to be red, but since the lattice is symmetric, any color would work equally. Finally, we take the symmetric difference of all three correction sets to find the final set of qubits to be corrected. This decoder is deterministic - given some specific syndrome, it will always return the same correction.

The sweep decoder, the lift procedure, and the symmetric difference together form the decoder for X-errors, whose syndromes are edge-like. As another part of this project, the group wrote an X-error decoder [11]. In our implementation, we call that decoder as a subroutine of the Z-error decoder.

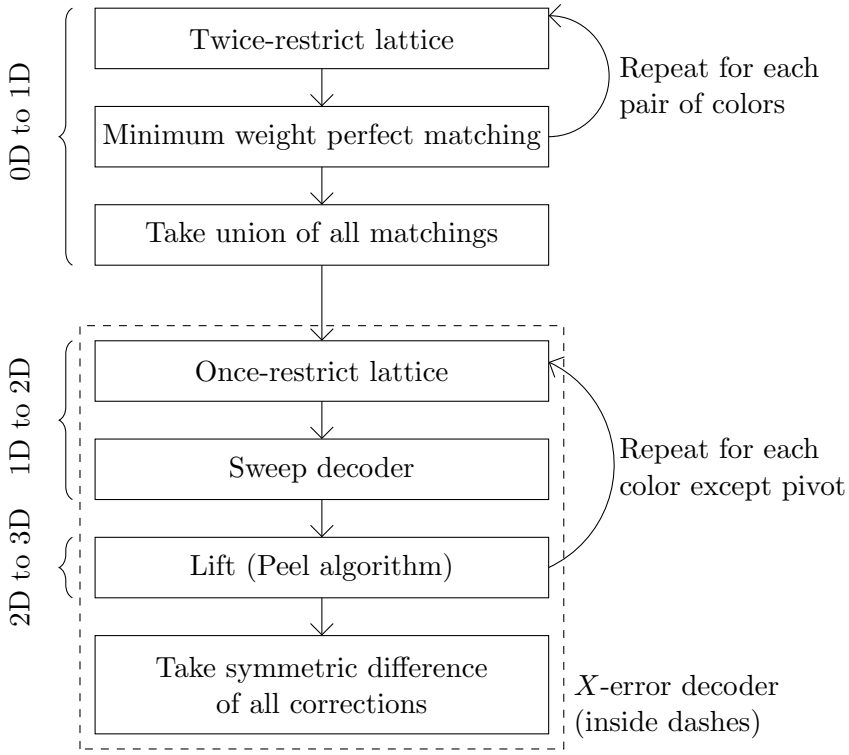


Figure 2.2: The Z-error decoding procedure

2.6 Check for Logical Errors

To determine whether the decoder caused a logical error, we compare the set of qubits corrected by the decoder to the original set of error qubits. In the following, Q denotes the set that is the symmetric difference (XOR) of the set of original error qubits and the set of corrected qubits. This is the set of qubits that have experienced a phase-flip after the correction is applied. Recall that applying a phase-flip to an original error qubit brings it back into its usual state. The phase-flips on the remaining qubits, Q, comprise an operator that may or may not be a logical Z operator.

If applying Z-gates to a set of qubits does not cause a logical error, that set of qubits must be a linear combination of Z-stabilizers.[3] We can represent the stabilizers as a a binary matrix of size $m \times n$, where m is the number of code qubits and n is the number of stabilizers. Each row in the matrix represents one stabilizer. The value in the i-th position is 1 if that stabilizer is adjacent to the i-th qubit and 0 otherwise. Recall from above that all faces (Z-stabilizers) are adjacent to either 4 or 6 qubits, so each row will have 4 or 6 ones. The rank of this matrix is the number of independent stabilizers. Q can be represented as a vector $\vec{Q}$ of length m, where the i-th value[4] is 1 if the i-th qubit experienced a phase-flip and 0 otherwise. Then, $\vec{Q}$ can be appended to the stabilizer matrix. If $\vec{Q}$ is a linear combination of stabilizers, this will not affect the rank of the matrix; otherwise, the rank of the matrix will increase by 1.

The function built into the Python module numpy to calculate the rank of a matrix uses regular addition instead of addition mod 2. This is inappropriate for our purpose because our matrices are binary. So, we wrote our own function to calculate the rank of a binary matrix using row operations mod 2 and Gaussian elimination.

Interestingly, in testing, all our logical errors occurred when Q had an odd number of elements. This observation suggests an easier way to check whether an operator is a logical operator. Namely, we do not need to check the rank of the matrix with $\vec{Q}$ appended if there are an even number of elements in Q, or equivalently, if

[3]Not X-stabilizers. This is easy to get mixed up, because Z-errors show their syndrome on X-stabilizers.

[4]The order of the qubits has no physical meaning. The order must be consistent with the ordering of qubits used to build the stabilizer matrix.

the sum of elements of $\vec{Q}$ is odd. It also suggests that all logical Z operators on this lattice act on an odd number of qubits.

Chapter 3

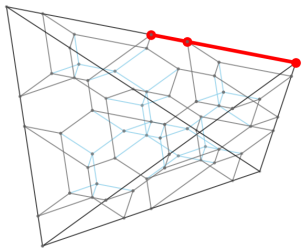
Results

3.1 Example Errors and Corrections

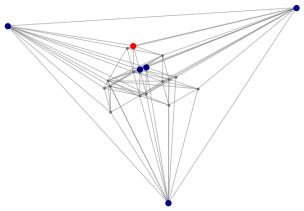
In this section, I present some examples of errors on the primal lattice, their syndromes, how the decoder attempts to correct that syndrome, and the final result of the correction. This section should not be taken as a characterization of all possible error types that do or do not cause logical errors, although such a characterization would be an interesting future project. The examples in this section are on lattices of size $d = 5$ and $d = 7$.

Examples That Cause Logical Errors

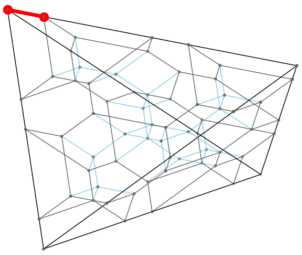
The syndrome of an error along the primal edge of size greater than $\frac{d}{2}$, where d is the total number of qubits along that edge, is identical to the syndrome of the remaining qubits on that edge. (See Figure 3.1b.) The decoder connects the error to the opposite corner. (See Figure 3.1.)



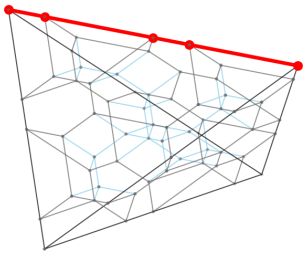
(a) The original error qubits, a chain on the primal edge whose length is greater than half the total length of the edge.



(b) The syndrome of that error. Red indicates syndrome vertices. Blue indicates vertices that are adjacent to error tetrahedra but have no syndrome because an even number of error tetrahedra touch that vertex.



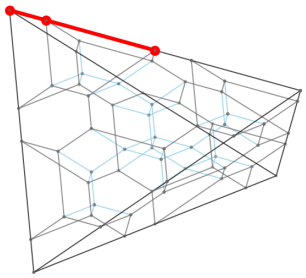
(c) The qubits the decoder recommends for correction. The syndrome, shown in Figure 3.1b, is the same as the syndrome of this error, and the decoder assumes the smaller error.



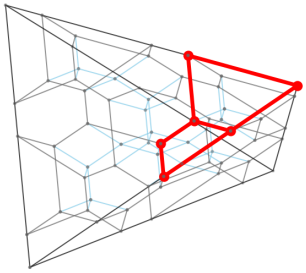
(d) The final result is that all qubits on the edge are flipped. This causes a logical error because the edge cannot be expressed as a linear combination of Z-stabilizers.

Figure 3.1

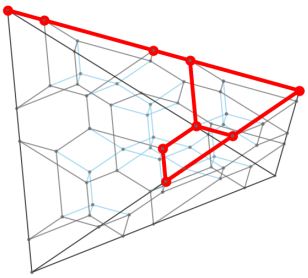
In some cases, for an error identical to the above, but on a different edge, the decoder both connects the error to the opposite corner and unnecessarily corrects some loops on an adjacent facet. (See Figure 3.2.) Although the lattice itself is symmetrical, the decoder is asymmetrical because of the sweep direction and the pivot color. This is why the same error on different edges may have different corrections.



(a) Another chain of errors on the primal edge whose length is greater than half the length of the edge.



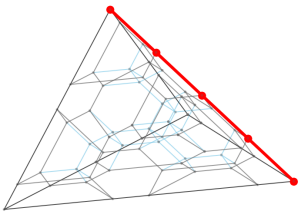
(b) The qubits the decoder recommends for correction. The decoder completes the edge to the opposite corner. The decoder also flips four qubits on the facet, which has no effect on the final result because the loop surrounds a Z-stabilizer. (See Figure 3.5.



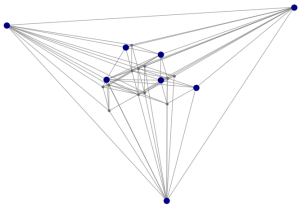
(c) The final result is a logical error.

Figure 3.2

If the error is along the entire edge, as shown in Figure 3.3, it will have no syndrome, so the decoder cannot correct it. This causes a logical error. Errors that are loops around a face in the bulk or on a facet also have no syndrome, but since those errors commute with Z-stabilizers, such loops do not cause logical errors.



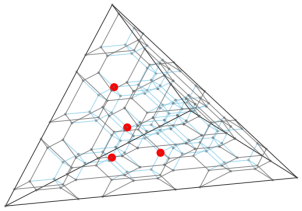
(a) An error along the entire primal edge.



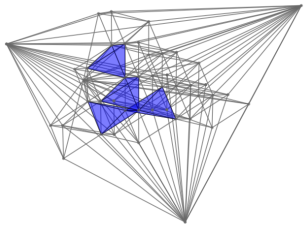
(b) This error has no syndrome. Blue indicates vertices that are adjacent to error tetrahedra but have no syndrome because an even number of error tetrahedra touch that vertex.

Figure 3.3

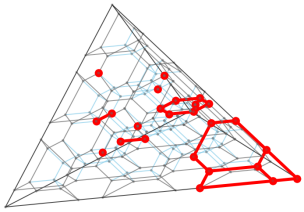
Errors on the edge are not the only errors that can cause logical errors. Figure 3.4 shows an error in the bulk that results in a logical error. In this case, some of the tetrahedra in the dual touch each other, so there is not syndrome information on the vertices where two tetrahedra touch.



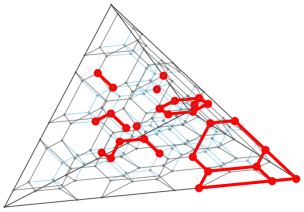
(a) Some qubits in the bulk that appear to be unconnected.



(b) In the dual, those qubits (tetrahedra) are adjacent on vertices. There is no syndrome on those vertices where two tetrahedra touch.



(c) The correction recommended by the decoder.

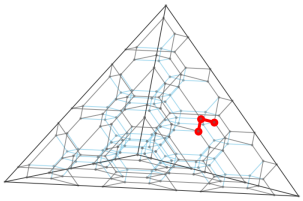


(d) The final set of qubits that are flipped. This is a logical Z-error because it is not a linear combination of Z-stabilizers.

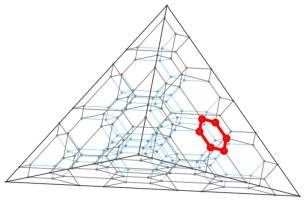
Figure 3.4

Examples That Do Not Cause Logical Errors

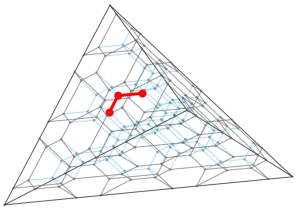
A half-hexagon error on a facet or in the bulk, as shown in Figure 3.5, will either correct the original error or complete the hexagon. Completing the hexagon is not a logical error because Z-stabilizers are on the faces of the primal lattice. A loop-like error is trivially a linear combination of Z-stabilizers; it corresponds to the stabilizer on the face enclosed by the loop. It is therefore not a logical error.



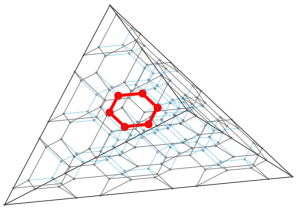
(a) A half-hexagon error on a facet.



(b) The decoder completes the hexagon. The result is that all six qubits are flipped. This corresponds with the Z-stabilizer on that face, so it is not a logical error.



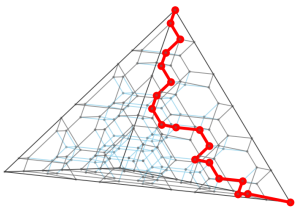
(c) A half-hexagon error in the bulk.



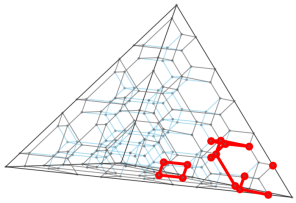
(d) Again, the decoder completes the hexagon, so the final result is not a logical error.

Figure 3.5

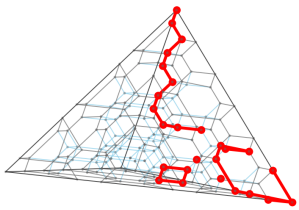
Although several of the logical error examples above resulted from a correction that connected one primal corner to another, an error that connects primal corners does not necessarily cause a logical error. Figure 3.6 shows such an error that the decoder corrected successfully. This correction results in a more complicated linear combination of Z-stabilizers than in Figure 3.5.



(a) The original error qubits.



(b) The correction returned by the decoder.



(c) The final set of flipped qubits. This is not a logical error because the set of flipped qubits is a linear combination of Z-stabilizers, although this is not trivially apparent from looking at the figure.

Figure 3.6

3.2 Threshold Probability for IID Errors

We numerically test the decoder using Monte Carlo methods. Each physical qubit in the lattice experiences an error with an independently and identically distributed (iid) probability p. Then, the decoder is given a syndrome of those errors and attempts to correct the errors. Finally, to assess the efficacy of the decoder, we check whether the correction caused a logical error. A logical error is considered a failure.

At high error probabilities, the decoder failure probability is high because as the lattice size increases, the number of errors (which is equal to the number of qubits $\times p$, on average) increases more rapidly than the lattice's capacity to correct errors. But at small error probabilities, larger lattices may be more likely to successfully correct an error than smaller lattices. The probability below which larger lattices are more successful is called the threshold probability.

We note the following:

- When building a quantum computer, working in the below-threshold regime is desirable. The lattice can be made arbitrarily large to push the failure probability arbitrarily close to 0.
- Although the behavior of the decoder, given some error syndrome, is determin-

istic, decoder failure is expressed as a probability because the error is randomly selected.

- The X- and Z-error decoders were tested separately because corrections of X- and Z-errors are independent.

We chose error probabilities (p) ad hoc to locate the threshold regime for this decoder. Broadly, we began with threshold regimes indicated by the original Restriction Decoder [8], decreased p until the threshold regime for this decoder was reached, then ran simulations for finer-grained increments of p in that region. These data points are illustrated in Figures 3.7 and 3.8.

For each data point, the number of Monte Carlo simulations run was a power of 10 between 10^4 and 10^7. By one "simulation," I mean that the simulation generated a random iid error, found the syndrome, ran the decoder for that syndrome, and checked whether the result was a logical error. In reality, since these tests were run on the Lonestar 5 supercomputer, 24 simulations were run in parallel. Smaller lattices and smaller probabilities required a larger number of simulations in order to generate a significant number of logical errors. We made an effort to run a number of simulations large enough that the error bars on vertically aligned points would not overlap. Error bars represent the standard deviation of the mean, given by $\sqrt{p_f(1-p_f)/N}$, where p_f is the decoding failure probability and N is the number of Monte Carlos simulations. In Figures 3.7 and 3.8, some error bars are not visible because the points themselves, as rendered, are larger than the bars. The number of simulations was upper-bounded by the runtime limit on the Lonestar 5 supercomputer, which was 48 wall-clock hours.

We find evidence for a threshold for the X-error decoder between 2.5% and 3.3% and for a threshold for the Z-error decoder between 0.01% and 0.02%. (See Figures 3.7 and 3.8.[1]) In comparison, the original paper on the restriction decoder found a threshold of 10.2% for both X- and Z-errors on the 2D color code with periodic boundaries; note that code has both a different dimensionality and a different boundary condition than the code studied in this thesis [8]. A recent paper on decoding with connected components found the threshold on the 2D triangluar color code with boundaries to be 0.2% [18]. (The decoder described in this thesis did

[1]Figures 3.7 and 3.8 were also published in [11].

not use connected components.) In addition, an investigation of the 3D gauge (not tetrahedral) color code with boundaries found a threshold of 0.46% for X-errors; Z-errors should admit the same threshold due to the symmetry of the gauge color code [19]. To the best of my knowledge, there are no previous results for thresholds on the 3D tetrahedral color code with boundaries, so I cannot compare our result directly with those of other groups. Furthermore, there may exist better decoders for the three-dimensional tetrahedral color code that admit higher thresholds.

It makes sense for the X-threshold to be higher than the Z-threshold, because decoding Z-errors is more difficult for two reasons. First, there is more degeneracy among syndromes of Z-errors than of X-errors because there are fewer X-stabilizers than Z-stabilizers. Second, the syndrome of Z-errors is missing information on the boundary vertices. Furthermore, the Z-error decoder uses the X-error decoder as a subroutine, so the performance of the former will always be bounded by that of the latter.

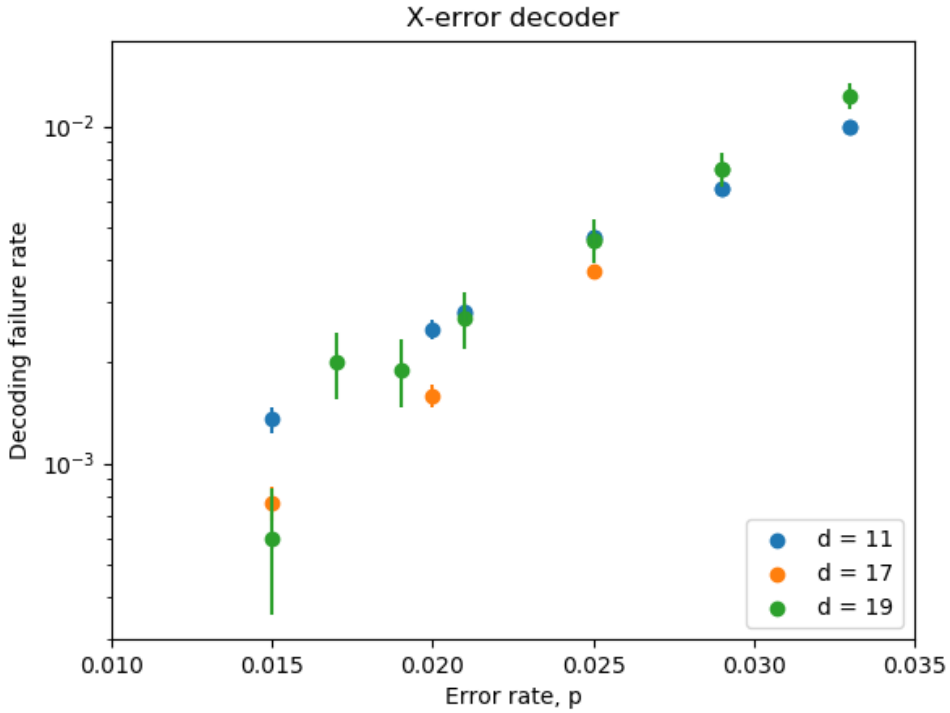


Figure 3.7: X-error decoder performance under iid local noise.

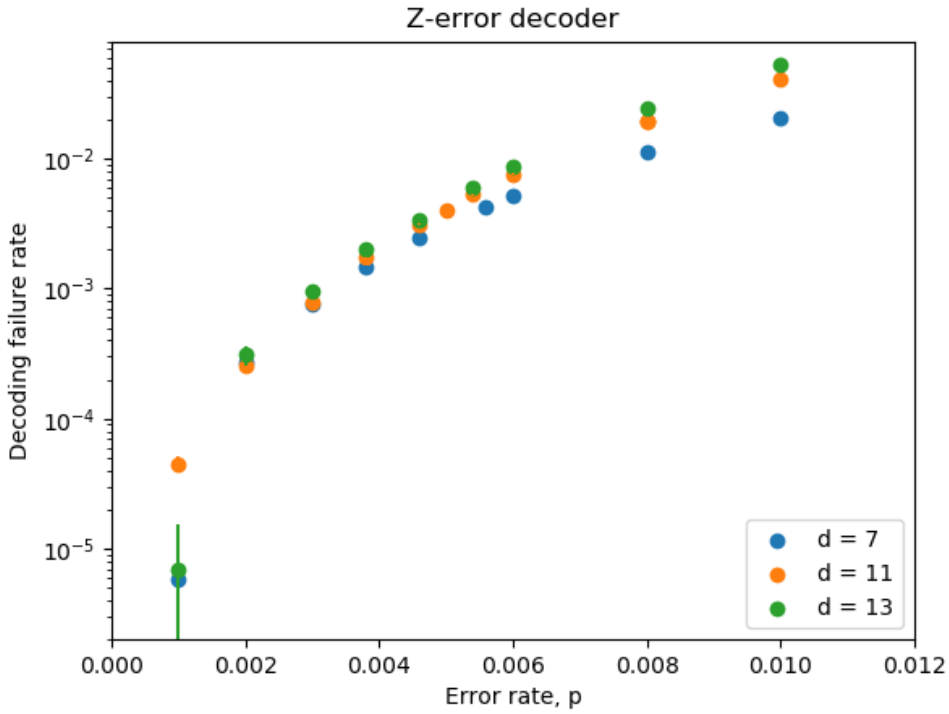


Figure 3.8: Z-error decoder performance under iid local noise.

Chapter 4

Conclusions

In this thesis, I have described the three-dimensional tetrahedral color code with boundaries. The 3D tetrahedral color code is a topological quantum error code, which encodes quantum information in global, as opposed to local, degrees of freedom. The code encodes one logical qubit in at least fifteen physical qubits. Furthermore, this code admits a logical gate set that is both transversal and universal. The code can be described by stabilizers - namely, Z-stabilizers that are on the faces and X-stabilizers that are on the cells of the primal lattice. Measuring the stabilizers results in an error or non-error syndrome.

A decoder is an algorithm that, given the error syndrome, finds a set of qubits that could have caused that error and applies operations to correct the error. I described a decoder for Z-errors, which have a point-like syndrome on the dual lattice. This decoder uses a restriction procedure to map the tetrahedral color code to the toric code. The decoder uses minimum-weight perfect matching to connect the point-like syndrome to an edge-like syndrome, uses the Sweep Decoder to connect the edge-like syndrome to a triangular mesh, and uses the Peel Algorithm to lift the triangular mesh to a tetrahedral set of qubits. This set of qubits is then the correction recommended by the decoder. Since the sweep and lift steps required for the Z-error decoder are the same procedures required for an X-error decoder, the Z-error decoder can call an X-error decoder as a subroutine. (The research group in which I work also wrote an X-error decoder as part of this project; see [11].)

The metric that measures decoder performance evaluates whether the symmetric difference between the set of original error qubits and the set of qubits cor-

rected by the decoder causes a logical error. I presented some examples of errors that do and do not cause logical errors in Section 3.1. Then, I presented a threshold error probability for iid X and Z errors. In the regime below the threshold error probability, making the lattice arbitrarily large will push the probability of successful correction arbitrarily close to 1. Evidence was found for a threshold for the X-error decoder between 2.5% and 3.3% and for a threshold for the Z-error decoder between 0.01% and 0.02%. Although this threshold is lower than thresholds found on other codes ([8, 18, 19]), the three-dimensional tetrahedral color code merits further investigation for the reasons described in Section 4.1. Furthermore, there may exist better decoders for the three-dimensional tetrahedral color code that admit higher thresholds.

4.1 Future Work

A possible extension to this project would be to test a just-in-time decoding scheme, as proposed by Bombìn in [7]. A just-in-time scheme would allow the three-dimensional color code to be implemented on a two-dimensional array of physical qubits by allowing the third dimension to be time. We chose the Sweep Decoder as our toric code decoder because the Sweep Decoder is a cellular automaton that lends itself to such a chronological ordering. In the just-in-time scheme, the decoder has less information than in the regular 3D scheme, because stabilizers on future areas of the lattice cannot be measured yet. Thus, the thresholds presented in this paper set an upper bound for the thresholds of the just-in-time scheme. A just-in-time application would be a potential reason to choose a three-dimensional color code instead of a two-dimensional code with a higher threshold, so this application should be explored further.

An extension of Chamberland's work on connected components to the 3D tetrahedral color code might result in higher thresholds than those presented in this thesis [18].

This work used iid errors for the threshold simulations. A more realistic noise model would take into account the noise propagation across gates described in [9]. This could lead towards a fault-tolerance threshold for measurement-based quantum computing on the color code.

Finally, a characterization of error configurations that cause or do not cause

logical errors could be used to predict, given some configuration of physical errors, whether the logical qubit will experience a logical error. In threshold calculations, this would circumvent the computational demand of running the decoder and checking whether the result is a logical error. This could also be used to determine which physical qubits are most likely to cause a logical error, so experimentalists constructing quantum computers could prioritize insulating the most troublesome qubits from noise.

References

[1] N. David Mermin. *Quantum Computer Science*. Cambridge University Press, New York, NY, 2007. Pages 100–109.

[2] Scott Aaronson. [Notes for] Lecture 27, Thurs April 27: Quantum error correction. `https://www.scottaaronson.com/qclec/27.pdf`, 2017.

[3] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, U.K., 2000.

[4] Hèctor Bombìn. Topological codes. In Daniel A. Lidar and Todd A. Brun, editors, *Quantum Error Correction*. Cambridge University Press, New York, NY, 2013.

[5] Hèctor Bombìn and Miguel A. Martin-Delgado. Topological computation without braiding. *Phys. Rev. Lett.*, 98:160502, 2007. doi: 10.1103/PhysRevLett.98.160502.

[6] Scott Aaronson. [Notes for] Lecture 29, Thurs May 4: Experimental realizations of QC. `https://www.scottaaronson.com/qclec/29.pdf`, 2017.

[7] Hèctor Bombìn. 2D quantum computation with 3D topological codes. arXiv:1810.09571, 2018.

[8] Aleksander Kubica and Nicolas Delfosse. Efficient color code decoders in $d \geq 2$ dimensions from toric code decoders. arXiv:1905.07393, 2019.

[9] Hèctor Bombìn. Transversal gates and error propagation in 3D topological codes. arXiv:1810.09575, 2018.

[10] Bryan Eastin and Emanuel Knill. Restrictions on transversal encoded quantum gate sets. *Phys. Rev. Lett.*, 102:110502, 2009. doi: 10.1103/physrevlett.102.110502.

[11] Skylar Turner, Josey Hanish, Eion Blanchard, Noah Davis, and Brian La Cour. A decoder for the color code with boundaries. arXiv:2003.11602, 2020.

[12] Robert Raussendorf and Hans Briegel. A one-way quantum computer. *Phys. Rev. Lett.*, 86:5188, 2001. doi: 10.1103/PhysRevLett.86.5188.

[13] A. Yu. Kitaev. Fault-tolerant quantum computation by anyons. *Ann. Phys. (N Y)*, 303:2, 2003. doi: 10.1016/S0003-4916(02)00018-0.

[14] Arun B. Aloshious and Pradeep Kiran Sarvepalli. Projecting three-dimensional color codes onto three-dimensional toric codes. *Phys. Rev. A*, 98:012302, 2018. doi: 10.1103/PhysRevA.98.012302.

[15] Michel X. Goemans. Lecture notes on bipartite matching. `https://math.mit.edu/ goemans/18433S09/matching-notes.pdf`, 2009.

[16] Aleksander Kubica and John Preskill. Cellular-automaton decoders with provable thresholds for topological codes. *Phys. Rev. Lett.*, 123:020501, 2019. doi: 10.1103/physrevlett.123.020501.

[17] Arun B. Aloshious and Pradeep Kiran Sarvepalli. Decoding toric codes on three dimensional simplical complexes. arXiv:1911.06056, 2019.

[18] Christopher Chamberland, Aleksander Kubica, Theodore J Yoder, and Guanyu Zhu. Triangular color codes on trivalent graphs with flag qubits. *New J. Phys.*, 22:023019, 2020. doi: 10.1088/1367-2630/ab68fd.

[19] Benjamin J. Brown, Naomi H. Nickerson, and Dan E. Browne. Fault-tolerant error correction with the gauge color code. *Nat. Commun.*, 7:12302, 2016. doi: 10.1038/ncomms12302.