Towards Self-Correcting Quantum Memories

Kamil Michnicki

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2015

Reading Committee:

Marcel Den Nijs, Chair

Aram Harrow

Chris Laumann

Program Authorized to Offer Degree:
UW Departhment of Physics

University of Washington

Abstract

Towards Self-Correcting Quantum Memories

Kamil Michnicki

Chair of the Supervisory Committee:
Professor Marcel Den Nijs
Physics

This thesis presents a model of self-correcting quantum memories where quantum states are encoded using topological stabilizer codes and error correction is done using local measurements and local dynamics. Quantum noise poses a practical barrier to developing quantum memories. This thesis explores two types of models for suppressing noise. One model suppresses thermalizing noise energetically by engineering a Hamiltonian with a high energy barrier between code states. Thermalizing dynamics are modeled phenomenologically as a Markovian quantum master equation with only local generators. The second model suppresses stochastic noise with a cellular automaton that performs error correction using syndrome measurements and a local update rule.

Several ways of visualizing and thinking about stabilizer codes are presented in order to design ones that have a high energy barrier: the non-local Ising model, the quasi-particle graph and the theory of welded stabilizer codes. I develop the theory of welded stabilizer codes and use it to construct a code with the highest known energy barrier in 3-d for spin Hamiltonians: the welded solid code. Although the welded solid code is not fully self correcting, it has some self correcting properties. It has an increased memory lifetime for an increased system size up to a temperature dependent maximum.

One strategy for increasing the energy barrier is by mediating an interaction with an external system. I prove a no-go theorem for a class of Hamiltonians where the interaction terms are local, of bounded strength and commute with the stabilizer group. Under these conditions the energy barrier can only be increased by a multiplicative constant.

I develop cellular automaton to do error correction on a state encoded using the toric code. The numerical evidence indicates that while there is no threshold, the model can extend the memory lifetime significantly. While of less theoretical importance, this could be practical for real implementations of quantum memories. Numerical evidence also suggests that the cellular automaton could function as a decoder with a soft threshold.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

My sincere thank you to Aram Harrow for his constant encouragement and guidance through the academic jungle. Thank you to Dave Bacon for his early guidance, to Steve Flammia for his encouragement and feedback and to Arthur Fine for his early mentoring. Thank you to my fellow graduate students and group members, Elizabeth Crosson, David Rosenbaum, Lukas Svec, Paul Pham, Kevin Zatloukal and Jijiang Yan for all of the camaraderie, feedback and and time spent in journal club. I am particularly appreciative of the University of Washington Department of Physics department for training and supporting me, the University of Washington Department of Computer Science for the support, office and coffee room and the MIT Center for Theoretical Physics for their hospitality during my time visiting.

DEDICATION

to all of my friends and family.

Chapter 1

INTRODUCTION

This thesis is about how to build a scalable quantum memory that is stabilized by passive dynamics.

A quantum memory is a stepping stone to a fully scalable quantum computer. Quantum computers have numerous applications. One of the most exciting is the ability of a quantum computer to directly simulate quantum mechanics. The space and time resources that a classical computer needs to simulate a quantum system often scale very poorly. Just to keep track of a superposition of n spins requires 2^n coefficients whereas a quantum computer could store the state directly using n qubits. There are many quantum algorithms that are provably more efficient than their classical counterparts. These include such algorithms as Grover's search [39] and quantum factoring [72]. Quantum factoring uses a polynomial number of primitive operations to factor numbers reliably. Despite prime numbers being a popular topic among mathematicians since Euclid, the best known algorithms for factoring run in a time exponential in the root of the number of digits in the number to be factored [68]. The quantum algorithm takes exponentially fewer gates to perform.

A useful abstraction for modeling quantum noise is the channel formalism. Here a quantum state is encoded in an error correcting code. Although there is noise during transmission, if the noise is weak enough, it can be reversed by measuring error syndromes and performing error correction. I focus exclusively on a foundational class of quantum error correcting codes called stabilizer codes. I cover quantum communication and stabilizer codes in chapter 2.

Models for quantum memories are more involved than quantum communication. Models

for quantum communication assume perfect encoding and decoding and have noise only while the quantum state is being transmitted. Quantum memories, on the other hand, are always noisy. They have noise at every stage: encoding, transmitting to the future and decoding. I talk about strategies for encoding and decoding in the absence of reliable operations. This falls under the theory of fault tolerant quantum circuits and I cover it in chapter 2.

The memories that I consider in this thesis are based on stabilizer codes that have syndrome measurements that are local. Such stabilizer codes are called topological stabilizer codes. They are called this because operations that transform the encoded quantum state often resemble geometric objects such as strings. Different strings have the same effect so long as they are related by a series of local deformations. The topological properties of an operation determine its action on the encoded quantum state. I cover topological stabilizer codes and different ways of visualizing them in chapter 3.

Self-correcting quantum memories are quantum systems whose internal dynamics perform error correction. Memories that use local dissipation of heat to do error-correction are particularly interesting because all that they require to function is a cold enough environment. This can be contrasted with storing a state in a fully fault tolerant quantum computer where measurements have to be made actively [1, 8, 52, 53, 69, 74]; something that may be more challenging to implement. I cover self-correcting quantum memories in chapter 4.

We do not know exactly how thermalization works on the microscopic level. Instead I use a phenomenological model to model thermalization. It uses local stochastic dynamics that thermalize the system to a Gibbs state.

The Ising model, a model for ferromagnetism, is a great example of a self-correcting classical memory. Here there is an energy penalty between neighboring spins whose orientations do not agree. In two dimensions this leads to domains of spins that all agree with each other. At a low enough temperature, local thermalization tends to decrease the energy by shrinking

these domains. This self-correcting behavior keeps the net magnetization stable for a long time. The challenge is to find a quantum memory that is protected in the same way.

There are certain criteria for when a topological stabilizer code will have a long lifetime against thermalizing dynamics. Surprisingly, these conditions do not involve the dynamics at all. Instead they involve the equilibrium distribution, the Gibbs distribution. I cover this and generalizations of the criteria in chapter 4.

Proving that a system has a long lifetime is often hard but there is a good indicator for whether a topological stabilizer code would make a good self-correcting memory: the energy barrier. For a domain to grow to the point of causing a decoding error, it has to go over some minimum energy called the energy barrier. I discuss it in chapter 4.

My main contribution is to improve the current best known energy barrier for topological stabilizer codes in 3-d. To do this, I invented a technique for combining codes which I call welding. I use this technique to combine several blocks of code with a constant energy barrier into a code with an energy barrier that scales as a power-law L^a for a system of qubits of width L. I cover the theory of welding in chapter 5 and develop the welded solid code, the code that achieves a high energy barrier.

Errors in a self-correcting memory manifest themselves as quasi-particles under a particular interpretation. Coupling them to an external system can prevent them from diffusing and corrupting the memory. In chapter 6, I put limits on which circumstances such an external coupling is beneficial.

Finally, I discuss other models of self-correcting quantum memories that straddle the divide between active and passive correction. These are the cellular automata decoders. I discuss this in chapter 7.

Chapter 2

QUANTUM ERROR CORRECTION

Classical error correction arose out of a need to reliably send information between two parties over a noisy channel. The theory was originally developed by Shannon [71]. Quantum error correcting codes are motivated by the same need only now we are transmitting a quantum state instead of a classical one. Shor [73] showed that this could be done and developed the first quantum error correcting code to do it. It was further developed by Shor, Steane and Calderbank [24, 77, 78]. See [69] for an early history. Both models involve encoding information into a redundant number of subsystems so that if a subset of these get damaged, the original message or quantum state can still be recovered, or decoded, with an error correction procedure. In both models encoding and decoding are assumed to be perfect.

The simplest classical code is the repetition code, wherein one simply resends a bit repeatedly. This linear error correcting code already provides an amazing improvement: the decoding error rate falls down exponentially with the number of bits used provided the error rate is below a threshold.

Quantum error correction rests on a crucial fact: you can measure whether an error occurred without measuring the stored superposition. This allows you to correct the error and regain the original quantum state.

A large class of quantum error correcting codes, called stabilizer codes [36], are based on a generalization of classical linear codes. I discuss a stabilizer code that encodes one qubit into seven physical qubits and corrects any error operation on any one of the physical qubits.

We can then encode another qubit using seven encoded qubits, with a total of 49 qubits. This concatenated code is even more resilient to errors than the original provided the error rate is not too big. Generally, we can keep concatenating like this and drive the decoding error rate down exponentially in the number of physical qubits. This constitutes an example of a threshold theorem for quantum error correction: below a certain error rate called the threshold error rate, a quantum state can be sent and recovered with arbitrary precision.

The threshold theorem for noisy communication relies on perfect encoding and decoding, something that may not be realistic experimentally. A more realistic model is where we encode and decode a quantum state using a faulty quantum computer, i.e. a quantum computer with faulty gates. There are many threshold results for fault tolerant quantum computing [1, 8, 52, 53, 69, 74]. I cover some of the basic concepts in the section on fault tolerance. A quantum memory is a fault tolerant quantum computer where all that we require is to swap information in and out, and preserve it for a long time.

2.1 Communication over a noisy channel

2.1.1 Quantum error correcting condition.

Imagine that there are two people, Alice and Bob and they are trying to communicate a quantum message, i.e. a quantum state, composed of many entangled qubits. Alice sends one qubit at a time to Bob. There is only noise during transmission and not before and after. Furthermore, the noise on each qubit is assumed to be stochastic and independent.

How do they communicate reliably? Alice encodes the message into a redundant number of subsystems using a quantum error correcting code. The code is designed so that an error on a small enough number of subsystems does not change the stored quantum information. I denote the encoding map with ϕ_{in} where $\phi_{\text{in}}(\rho) = \bar{\rho}$. The density matrix with a bar $\bar{\rho}$ denotes the encoded state of ρ. After encoding, the quantum state is sent over a noisy

channel which acts by transforming the state to $E(\bar{\rho})$. Once the information is received, an error correcting operation is applied that transforms the state to $C(E(\bar{\rho})) = \bar{\rho}_{\text{final}}$. The transmission is successful if $\bar{\rho}_{\text{final}} = \bar{\rho}$.

Decoding ambiguous in quantum communication. Decoding and error correction are synonymous for classical codes since measurement does not change a classical bit-string. However, quantum states change when they are measured. We have to decide whether decoding means error correction or measurement: I use"decoding" synonymously to "error correction". I say "readout" or "measurement" when I mean that the final encoded state is being measured.

Which noise model we should consider is ultimately an experimental question. I use a noise model where errors on different subsystems are not too correlated. Faulty subsystems are chosen independently and identically with a probability p, called the error rate. We then choose an operation that acts on the chosen qubits that an adversary would choose in order to maximize the probability of an error on the encoded state after the state is error corrected.

We are interested in bounding the probability that the quantum state changes during transmission. This is ambiguous since the only way we have of telling is by doing measurements, which may be done immediately or delayed after entangling the system with another. We'll consider a simplification. The noise map that I use in this chapter can be split into two parts, one correctable part E_1 is applied with a probability p_1 and an uncorrectable part E_2 is applied with a probability p_2. The error map is $E = p_1 E_1 + p_2 E_2$ where $E(\bar{\rho}) = p_1 E_1(\bar{\rho}) + p_2 E_2(\bar{\rho})$. In general a map can be expressed as $E_1(\rho) = \sum_i A_i \rho A_i^\dagger$.

Definition The quantum error correcting condition for a set of errors $\{A_i\}$ is that there exists a error correcting map C such that $C(A_i \bar{\rho} A_i^\dagger) = \bar{\rho} tr(A_i \bar{\rho} A_i^\dagger)$ for all A_i. That is, the error A_i is a correctable according to the encoding operation ϕ and the error correcting map C.

Our task will be to bound the probability of an uncorrectable error.

2.1.2 Classical repetition code

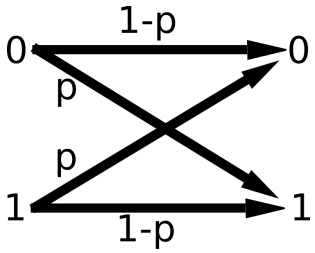

Figure 2.1: The binary symmetric channel. Each bit is transmitted with a probability $1-p$ and flipped with a probability p.

I first describe reliable classical communication through the binary symmetric channel. Here Alice wants to send a bit string to Bob by repeatedly sending a single bit. Each bit is flipped with a probability p and left unchanged with a probability $1-p$. Each use of the channel is independent. By encoding a bit-string redundantly using an error correcting code, Alice can reduce the probability of a decoding error arbitrarily provided the error rate is below a threshold. For more depth on classical error correction see [57].

The simplest error correcting code is the 3-bit repetition code. Suppose Alice wants to send a bit with a value b. To protect the encoded bit, Alice sends the same value three times. The bit-string bbb is the encoded message. Some of the bits might flip. When Bob observes the encoded bit-string some time later, he does error correction by taking the majority vote of the bit-string. For instance, if Bob sees the bit-string 011, he assumes the original bit-string was 111 and if he sees the bit-string 010, he assumes the original bit-string was 000.

How often does transmission fail for the 3-bit repetition code? The majority vote protocol increases the probability of transmission whenever the probability of a bit flip $p < \frac{1}{2}$. Alice decodes the bit string correctly if zero or one bits are flipped and incorrectly otherwise. This

happens with a probability $p_{\text{correct}} = (1-p)^3 + 3p(1-p)^2$. So the probability of interpreting incorrectly will be

$$p_{\text{error}} = 1 - p_{\text{correct}} = 3p^2 - 2p^3 \,. \tag{2.1}$$

When is the probability of transmitting the message correctly increased? This happens when $p > 3p^2 - 2p^3$. Rearranging we get the following inequality

$$0 > -2p^3 + 3p^2 - p = -p(1-p)(1-2p) \,. \tag{2.2}$$

Thus the survival probability of our bit is increased whenever $p < \frac{1}{2}$.

However, what if we wanted to drive the error rate down even further? What are some strategies that we could use? One simple strategy is to take the majority vote of more bits. We encode a logical bit 0 with the bit string 0^n and the logical bit 1 with 1^n. Here b^n just means the bit b repeated n times. Now what is the probability of there being a logical error using the majority vote protocol over n bits? That is, what is the probability of there being more than $n/2$ errors? Call this probability $P(X > \frac{n}{2})$. We prove in the box below that this probability goes down faster than exponentially with the number of bits whenever the probability is less than $1/2$. This is an example of a threshold theorem. When the error rate is below a threshold, we can drive down the probability of a decoding error exponentially with the number of encoding bits.

We calculate the probability that more than $n/2$ bits will be flipped for the n bit repetition code and the binary symmetric channel with an error rate p. Let X_i be a random variable that counts whether a bit flip error has occurred on bit i. Let X be the total number of bit flip errors. $X = \sum_i X_i$. We now bound $P(X \geq \frac{n}{2})$.

$$P(X \geq \frac{n}{2}) = P(e^{\alpha X} \geq e^{\alpha \frac{n}{2}}) \leq \frac{\mathbb{E}(e^{\alpha X})}{e^{\alpha \frac{n}{2}}} \,. \tag{2.3}$$

The above equation is true for any positive real number α. We assumed that errors on each bit occur independently. Hence $\mathbb{E}(e^{\alpha X})$ factorizes as $\mathbb{E}(e^{\alpha X}) = \mathbb{E}(e^{\alpha X_1})^n = (1 - p + pe^{\alpha})^n$. Hence,

$$P(X \geq \frac{n}{2}) \leq \left(\frac{(1 - p + pe^{\alpha})}{e^{\frac{\alpha}{2}}} \right)^n . \tag{2.4}$$

The quantity in the parentheses is less than 1 whenever $p < \frac{e^{\frac{\alpha}{2}}-1}{e^{\alpha}-1}$. For any $p < \frac{1}{2}$ we can find an $\alpha > 0$ such that the inequality is satisfied. Hence for any $p < \frac{1}{2}$, the n bit repetition code has an error rate that is exponentially suppressed in the number of encoding bits. In the limit that n goes to infinity, there will be zero probability of decoding incorrectly.

2.1.3 Linear codes

A 3-bit repetition code is a particular example of a linear code. We can do error correction by knowing the parity of pairs of bits without knowing the values of the individual bits. For instance, if we knew$\oplus$ means addition modulo 2. $b_1 \oplus b_i$[1]hen we can calculate whether the ith bit agrees or disagrees with the first bit. We flip the smaller of the two groups. Linear codes are error correction codes that use parties of groups of bits, perhaps more than two, as syndromes of errors.

We can encode these parity checks in a parity check matrix. Let B_i be a set of bits and $\mathcal{B} = \{B_a : \forall a \in \{1, .., m\}\}$ be a set of sets of bits. $\mathcal{B}$ defines the parity checks of a linear code. A bit-string b is a codeword whenever $\oplus_{i \in B_a} b_i = 0$ for all $B_a \in \mathcal{B}$. We can write this condition as a matrix equation. We define the matrix

$$M_{ij} = \begin{pmatrix} 1 & \text{if } j \in B_i \\ 0 & \text{otherwise} \end{pmatrix} .$$

The condition that each parity check is satisfied can be expressed as the matrix equation

[1] t

$\sum_j M_{ij} b_j \mod 2 = 0$ for all i. Hence the space of codewords is exactly the null space of the parity check matrix M. We can find a basis for the null-space of M, call it $\{V_i\}$. V_i is a vector and i indexes which vector we are talking about. The components of V_i are indexed with j as follows: V_{ij}. We can express every codeword as $b = \oplus_i m_i V_i$ where m_i is 0 or 1. This gives us a way to encode information. Let m be the message that we want to send. The jth bit of the encoded message is given by $\oplus_i m_i V_{ij}$. V is called the generator matrix of the code. For each vector V_i, there is a collection of vectors $\{D_i\}$ such that $\sum_j V_{ij} D_{i'j} = \delta_{ii'}$. This matrix D is how we decode the encoded message after error correction has been done. $Db = m$. $Vm = b$. So encoding and decoding are both done via matrix multiplication. As an example, here is the parity check matrix, generator matrix and decoding matrix for the 3-bit repetition code.

$$M = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

$$V = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

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

How many errors can a linear code correct? This requires specifying a decoder. One decoder is to flip the smallest number of bits of the corrupted string to achieve a codeword. We define the distance of the code to be the minimum Hamming distance between two words; that is, the minimum number of bits we have to flip to transform one codeword into another. If we see a bit string $e \oplus b$, where e is an error, then we can define an error correcting protocol

by just assuming the word $e \oplus b$ came from the codeword with the smallest Hamming distance. Thus a linear code with distance d can correct $\lfloor (d-1)/2 \rfloor$ errors. This may not be the most reliable way to decode but it often leads to a threshold. The most reliable way to decode is to look at the error model and determine what bits to flip to correct the message with the highest probability. However, both problems are NP-hard [12, 83].

Let's look at a code that is a bit more interesting than the repetition code: the Hamming code. It encodes 4 bits into 7 bits and has a distance of 3 or for short, it is a [7,4,3] code. We'll use a similar notation for quantum error correcting codes. We'll use the Hamming code later as a stepping stone to a fully quantum code. Here is the parity check matrix, the encoding matrix, followed by the decoding matrix.

$$M = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

$$V = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

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

The error correcting protocol is simple for the Hamming code. The parity check matrix has the property if e is a single bit error on bit i then the numerical value of the bit-string Me will be i. Like the repetition code, the Hamming code decreases the probability for a decoding error when ever the single channel error rate is below a threshold.

2.1.4 Concatenation

Concatenation is a way of combining two codes, an inner and outer code, to produce a third code such that the combined code is more reliable. It leads to a threshold theorem for classical communication; that below a certain error rate, errors can be driven down exponentially in the number of bits used in the encoding. Concatenation is how the first threshold theorems for quantum communication were proven [77].

A concatenated code is arranged into blocks of inner codes. The outer code uses the encoded bit from each inner code in order to encode bits using a second linear code, the outer code. The error correcting protocol is to first correct the inner code blocks by calculating their parity checks and flipping the appropriate bits. The outer code is then corrected by calculating the parity checks of the encoded bits of each block and then flipping the appropriate encoded bits.

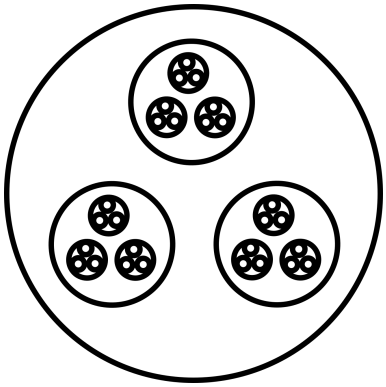

Figure 2.2: Each circle represents a majority vote of the circles that it contains. The smallest circles represent bits. This is the structure of the concatenated repetition code.

For instance, suppose we concatenate using the 3-bit repetition codes. We have 3 blocks of 3 bits. Each block is encoded with the repetition code and the 3 blocks are also encoded with the repetition code. Alice sends the bit string 000 000 000 to Bob one bit at a time. Suppose Bob receives the string 000 000 110. Bob will decode the last block in correctly when doing error correction on the first level and arrive at the string 000 000 111. However, when he compares the 3 blocks in second level of error correction, he can take the majority vote of the three blocks and arrive at the correct bit string 000 000 000.

What is the failure rate after n levels of concatenation? We assume i.i.d.(independent and identically distributed random variables) noise and a bit flip rate of p_0 at the zeroth level of concatenation. After the first level of concatenation, the failure rate will be $p_1 = 3p_0^2 - 2p_0^3$. Thus $p_1 < 3p_0^2$ which is an improvement when $p_0 < \frac{1}{3}$. After repeating concatenation n times, the failure rate is bounded by

$$p_n < \frac{(Cp_0)^{N^{\frac{\log 2}{\log 3}}}}{C}, \tag{2.5}$$

where $N = 3^n$ is the total number of bits in the encoding and $C = 3$. Hence the probability to transmit a bit incorrectly after n levels of concatenation goes down exponentially in the

number of bits used in the encoding.

I already showed an exponential suppression of error when doing a global majority vote on an N bit repetition code. What is interesting is that such an exponential suppression is generic for concatenated codes. For instance, if we had concatenated the Hamming code instead of the repetition code, again we would have an inequality of the same form

$$p_n < \frac{(Cp_0)^{N^{\frac{\log 2}{\log 7}}}}{C}, \tag{2.6}$$

where N is the total number of bits in the encoding. But now $C = 7$ instead of 3. This corresponds to seven ways of performing a single bit flip on a block of seven bits. In general, if a linear code corrects more than one error, it will have such a threshold theorem associated with it.

Decoding one block at a time often leads to a threshold but it is not necessarily the best threshold. For instance, concatenating the repetition code 2 times is the same as the 9 bit repetition code. However 4 errors is enough to cause a decoding error; two error in two blocks. However, a global majority vote could easily correct 4 errors.

2.1.5 From classical to quantum codes

Stabilizer codes are the quantum generalization of classical linear codes. I first re-frame linear codes in terms of the language of stabilizer codes and then point out the generalization.

As a start, let's see what happens when we use a quantum state to encode a block of linear code. We replace a bit b with a quantum state $|b\rangle$. Suppose we want to encode the quantum state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ using the 3-bit repetition code. The encoded $|b\rangle$ state is

$|bbb\rangle$ and so the encoded state $|\psi\rangle$ is

$$|\bar{\psi}\rangle = \alpha|000\rangle + \beta|111\rangle\,. \tag{2.7}$$

In order to correct bit flip errors we'll have to have a way of measuring the parity of a group of qubits. We can do this by measuring a product of Pauli-Z operators. For instance if we wanted to measure the parity between the first and second qubits, we would just measure Z_1Z_2. A $+1$ result indicates a parity of 0 and a -1 result indicates a parity of 1. In general we can measure the parity of a set of qubits Q by measuring the operator $\prod_{q\in Q} Z_q$. These are called syndrome measurements because they indicate a syndrome of an error.

We can recast the majority vote protocol in terms of quantum measurements and operations. Define the two parity checks $S_{13} = Z_1Z_3$ and $S_{23} = Z_2Z_3$. The majority vote protocol is

$$\begin{aligned}
&\text{if } S_{23} = 0 \text{ and } S_{13} = 0 \quad \rightarrow \quad \text{do nothing} \\
&\text{if } S_{23} = 0 \text{ and } S_{13} = 1 \quad \rightarrow \quad \text{apply } X_1 \\
&\text{if } S_{23} = 1 \text{ and } S_{13} = 0 \quad \rightarrow \quad \text{apply } X_2 \\
&\text{if } S_{23} = 1 \text{ and } S_{13} = 1 \quad \rightarrow \quad \text{apply } X_3
\end{aligned} \;.$$

This is guaranteed to work because when we measure the operators Z_1Z_2 and Z_2Z_3 then we are guaranteed to be in one of the subspaces

$$\begin{aligned}
&\{|000\rangle, |111\rangle\} \\
&\{|001\rangle, |110\rangle\} \\
&\{|010\rangle, |101\rangle\} \\
&\{|100\rangle, |011\rangle\}
\end{aligned} \;.$$

The protocol gives a rule for which qubit to flip in order to get into the code space $\{|000\rangle, |111\rangle\}$.

Suppose Alice sends the encoded quantum state $|\bar{\psi}\rangle$ to Bob one qubit at a time. A single qubit flips with a probability p by a Pauli X operator. Bob will then have the state

$$C(E(|\bar{\psi}\rangle\langle\bar{\psi}|)) = (1-(3p^2-2p^3))|\bar{\psi}\rangle\langle\bar{\psi}| + (3p^2-2p^3)\bar{X}|\bar{\psi}\rangle\langle\bar{\psi}|\bar{X} \tag{2.8}$$

where $\bar{X} = X_1X_2X_3$ denotes the encoded logical X operator. So Bob gets back the state $|\bar{\psi}\rangle$ with a probability $(1-(3p^2-2p^3))$ and the incorrect state $\bar{X}|\bar{\psi}\rangle$ with a probability $(3p^2-2p^3)$.

The repetition code does not fix all one qubit errors. For instance, a single qubit Pauli operation on the first qubit, e.g. the Z_1 operator, would change the relative phase of the code words even though the subspace is preserved. We will fix this problem by measuring parity checks in the rotated basis, the X basis, and not just the Z basis. This is essentially what stabilizer codes are.

2.1.6 Stabilizer codes

I now give the definition of a stabilizer code and review a code that corrects an arbitrary error on a single qubit: the Steane code. For classical linear codes, all of the parity measurements are made in the Z basis. Quantum mechanically, we can also do parity checks in the X and Y basis as well as a linear combination of X, Y and Z. As it turns out, doing parity checks in just the X, Y and Z basis is sufficient to provide a large class of good quantum codes.

Let S_C be an Abelian subgroup of the Pauli group G with real coefficients such that $-1 \notin S_C$. G is defined as

$$G = \{(-1)^k P_1 \otimes ... \otimes P_n : k \in \{0,1\} P_i \in \{I, \sigma_x, \sigma_y, \sigma_z\}\}. \tag{2.9}$$

S_C is called a stabilizer group of the code C and can be thought of as the group of all possible

generalized parity checks of the code C. The condition that $-1 \notin S_C$ is just a consistency condition; it makes it so that each operator can simultaneously have a $+1$ eigenvalue. A quantum state $|\psi\rangle$ is in the code space of the quantum error correcting code C if $h|\psi\rangle = |\psi\rangle$ for all $h \in S_C$.

We can locate errors by measuring stabilizers. These will be our syndromes. For instance, suppose that $h \in S_C$ does not commute with Z_1. We have that $hZ_1|\psi\rangle = -Z_1 h|\psi\rangle = -Z_1|\psi\rangle$. Hence, measuring h and getting a -1 eigenvalue indicates an error on an odd number of qubits of h. By measuring enough stabilizers we can locate where the error is likely to have originated from. Not only that, but we can construct codes that find both X and Z errors. We need only measure a generating set of operators R and then multiply the results of those measurements to find the eigenvalue of any stabilizer. For clarity, R is a generating set of S: $\langle R \rangle = S$.

There is an equivalence between a minimal generating set of a stabilizer group $\{h_1, ...h_{n-k}\}$and the set of single qubit Z operators $\{Z_1, ..., Z_{n-k}\}$. There is a Clifford group operator U such that $UZ_iU^\dagger = h_i$. See [63] for the proof. U is in the Clifford group if and only if for each $h \in G$, $UhU^\dagger \in G$. That is, Clifford group elements map Pauli operators to Pauli operators. If there are n qubits and a minimum of $n-k$ generators, then there must be k encoded qubits. The encoded logical operators must be $UZ_{n-k+j}U^\dagger = \bar{Z}_j$ and $UX_{n-k+j}U^\dagger = \bar{X}_j$. Furthermore, these encoded logical operators are also Pauli-operators since U is in the Clifford group. Encoded operations are only unique up to multiplication by an element of the stabilizer group. This is because if $h \in S_C$ and $\bar{L}$ is a logical operator then $\bar{L}h|\psi\rangle = \bar{L}|\psi\rangle$. We can use this encoding unitary to define an encoding map. Suppose we want to encode the state $\rho = \sum_i a_i L_i$ where L_i is a logical operator in the Pauli group. We append $n-k$

qubits, all in the $|0\rangle$ state so that the density matrix is

$$\frac{1+Z_1}{2} \otimes ... \otimes \frac{1+Z_{n-k}}{2} \otimes \rho \,. \tag{2.10}$$

Conjugating by U gives us

$$U \left(\frac{1+Z_1}{2} \otimes ... \otimes \frac{1+Z_{n-k}}{2} \otimes \rho \right) U^\dagger = \frac{1+h_1}{2} ... \frac{1+h_{n-k}}{2} \sum_i p_i \bar{L}_i \,. \tag{2.11}$$

We can define an encoding map $\phi_{\text{in}}(\sum_i a_i L_i) = \prod_{i=1}^{n-k} \frac{1+h_i}{2} \sum_k a_k \bar{L_k}$. Just as for linear codes the distance of a code will be the Hamming weight of the shortest logical operator.

All of the codes that I describe in this thesis have a generating set with terms that are composed of tensor products of just X operators or products of just Z operators. Such codes are called CSS codes, named after their discoverers Calderbank, Shor and Steane [23, 77]. For instance, $X_1X_3X_{10}$ and $Z_2Z_3Z_{10}Z_{12}$ could be generators of this generating set but Y_1Z_{12} could not be.

One way to look at a CSS code is that we first find a classical code that encodes many bits. We then use a subset of the logical operators of the classical code as stabilizers.

All generating sets of CSS codes in this thesis are in standard CSS form. A generating set R for a CSS stabilizer group is in standard CSS form if each element of R is either a tensor product of only single-qubit Pauli-X operators, or a tensor product of single-qubit Pauli-Z operators, e.g. $X \otimes I \otimes X$... but not $X \otimes Z \otimes I$....

The Steane code is an example of a quantum code that can correct any error on a single qubit. It has 7 physical qubits and encodes one logical qubit. The generating set for the

Steane code is given by:

$$S = \begin{matrix} \{I & I & I & Z & Z & Z & Z, \\ I & Z & Z & I & I & Z & Z, \\ Z & I & Z & I & Z & I & Z, \\ I & I & I & X & X & X & X, \\ I & X & X & I & I & X & X, \\ X & I & X & I & X & I & X\} \end{matrix} .$$

The logical operators of the Steane code are $\bar{X} = XXXXXXX$ and $\bar{Z} = ZZZZZZZ$ and $\bar{Y} = -i\bar{X}\bar{Z}$. This means that an encoded unitary operation on a single encoded qubit is $I\cos\theta + i\hat{n}\cdot\vec{\bar{\sigma}}\sin\theta$ where $\vec{\bar{\sigma}} = (\bar{X}, \bar{Y}, \bar{Z})$ and $\hat{n}$ is a unit vector.

The Z-stabilizers measure the parity checks for the Hamming code [45]. The syndromes of the Hamming code identify any bit flip on a single bit. In the quantum case this is equivalent to identifying a single qubit Pauli-X error. Similarly the X-stabilizers identify single qubit Z-error. We can see why as follows. Let

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

It is called the Hadamard matrix. It has the property that $HXH = Z$ and $HZH = X$. If we take a Z stabilizer h and transform it by conjugating it with $H^{\otimes 7}$ then $H^{\otimes 7}hH^{\otimes 7}$ is an X stabilizer! Similarly, a single qubit X-error transforms to a single qubit Z-error and hence the X stabilizers identify single qubit Z errors. Together, the X and Z stabilizers correct any single qubit error. A single qubit error can be an arbitrary unitary on a single qubit of the form $U = \cos I\theta + i\hat{n}\cdot\vec{\sigma}\sin\theta$ where $\hat{n}$ is a unit vector in 3-d and $\vec{\sigma} = (X, Y, Z)$. If we measure X and Z stabilizers, we project the state onto a discrete set of errors. One of four

things happens: 1. nothing happens and the operator I is applied 2. a bit flip happens; a single X gets applied 3. a phase flip happens; a single Z gets applied 4. both happen; a single $Y = -iZX$ gets applied.

How well does the Steane code perform against i.i.d. adversarial noise? An adversary chooses a set of faulty qubits. Each qubit is chosen with a probability p. What is the probability that the adversary chooses zero or one qubit and hence performs a correctable operation? The probability that the error is correctable is $(1-p)^6+7p(1-p)^6$. One can check numerically that whenever $p < \frac{1}{7}$ the probability that the state is transmitted is better than p. $\frac{1}{7}$ is a bound on the threshold but the actual threshold, where $p = (1-p)^6 + 7p(1-p)^6$, is slightly higher than $\frac{1}{7}$.

We now concatenate 7 encoded qubits to make a code with 49 physical qubits. Each group of 7 qubits corrects a single qubit error. However, if two errors happen in a block, that block can be corrected with the other six blocks. The way we do this is to replace X with a $\bar{X}$ operator and replace Z with a $\bar{Z}$ operator whenever they occur in the definition the Steane code. This gives us a Steane code on encoded blocks of code.

Let p_n be the failure probability for the n^{th} level of concatenation. When $p_0 < \frac{1}{7}$ then $p_{n+1} \leq Cp_n^2$. Thus the failure rate goes down doubly exponentially with the number of levels of concatenation[2] that we use while the number of qubits rises exponentially. It is quite remarkable that the error rate for decoding goes down so quickly. This is a threshold theorem for the Steane code for communication over a noisy channel.

2.2 Fault tolerant quantum computation and quantum memories

A fault-tolerant quantum computer simulates an error-free quantum computer using faulty components. A quantum memory is a simple quantum computer that can to swap quantum

[2]The argument is the same as for concatenated classical codes in section 2.1.4

information in and out and can store the state for a very long time. We can make a fully fault-tolerant quantum computer by combining a quantum memory with a quantum processor; a device that does a universal set of quantum operations on at least a pair of subsystems.

Self-correcting quantum memories are quantum memories that use the entropy of the environment to prevent errors from happening. If an error increases the energy of the memory, then that energy must be borrowed from the environment. The multiplicity of the environment typically goes up as you increase the energy. Hence a transition that reduces the energy of the system is more likely than a transition that increases it. For instance, consider an environment composed of m harmonic oscillators. The multiplicity of m that together store E units of energy is $O(E^{m-1})$. Borrowing δE units of energy reduces the multiplicity by $(E-\delta E)^{m-1} \sim e^{-\beta \Delta E}$ where the temperature $T = \frac{1}{\beta}$ is just the average energy per oscillator $T = \frac{E}{m}$ in the thermodynamic limit. In some systems, the multiplicity can actually go down as the energy goes up. This happens in spin systems with on-site magnetic fields with high occupation numbers.

At present, we have theoretical models of fault-tolerant circuits that simulate perfect quantum gates. There is a connection between the circuit model of fault-tolerance and self-correcting quantum memories. If we can find a theoretical model for a self-correcting quantum memory, then we can find a constant depth quantum circuit that performs error correction. At present we only know how to do error correction in 3-d with circuits whose depth grows unboundedly with the number of qubits used in the encoding. For instance, this is true for circuits that use concatenated codes [69] and for algorithms such as minimum weight perfect matching, which runs in a polynomial time [28,33], used to correct topological codes such as the toric code [52].

Interpreting whether a state is correct in a fault-tolerant implementation is more involved than with quantum communication over a noisy channel. For quantum communication we

assumed that we could decode perfectly. However, for faulty implementations, it is virtually impossible to correct a state completely. The reason is that by the time you have performed error correction, another round of errors has already happened. This presents two related challenges, how do we do fault-tolerant readout and how do we interpret if an intermediate state in our a circuit is correct?

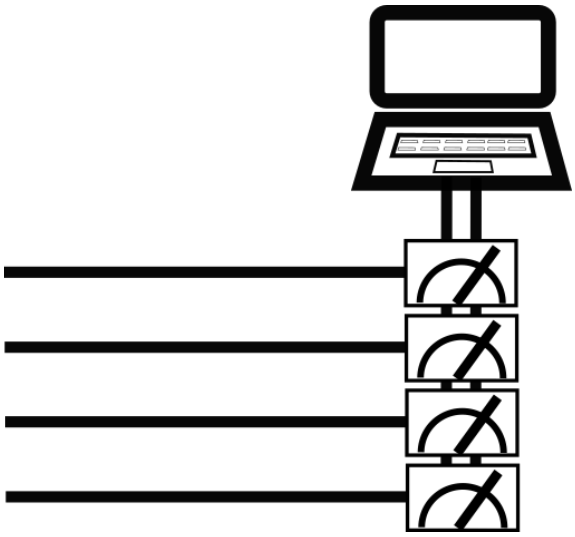

Figure 2.3: Each qubit is measured in the Z basis and error correction is done on the resulting bit-string by a classical computer.

One way to do a fault-tolerant quantum measurement is by a destructive measurement. The way we do this is to measure each qubit in the Z basis and then do classical error correction on the measurement data. Suppose we wanted to measure the logical operator $\bar{Z}$. Note that we need not fix Z errors since they commute with the operator $\bar{Z}$. Hence, we need not measure X-stabilizers either. Ideally, we would measure syndromes, perform an error correcting operation and then measure the logical operator $\bar{Z}$. Instead we measure single qubit Z operators for each qubit and store it in a string s. The Z stabilizers commute with these measurements and hence we can calculate the value that we would have gotten

by measuring these stabilizers by multiplying measurement the eigenvalues from the single qubit measurement data. We can then calculate which qubits we would have had to flip had we not measured each qubit destructively. Call this operator U_s. The operator U_s either commutes or anti-commutes with each single qubit Z operator. Hence, we flip each single qubit measurement value that overlaps with the corresponding qubit of the operator U_s. We can then calculate what the value of the logical operator $\bar{Z}$ would be if we had applied U_s and then measured $\bar{Z}$ by multiplying each measurement result that overlaps with the corresponding qubit of the operator U_s. The statistics of this calculated $\bar{Z}$ measurement are the same as if we had measured stabilizers perfectly, corrected errors perfectly and then measured the logical operator perfectly.

We can see from the example of a destructive measurement that it is important whether perfect error correction would lead to the correct state or not even though we cannot do perfect error correction. If it does, then our destructive measurement will have correct statistics. We use this idea to define whether an intermediate state in a quantum circuit is correct or not. We say it is correct if perfect error correction would result in the intended encoded quantum state. Otherwise the state is incorrect. We call the state that we would have if we were to do perfect error correction, even though we can't, the error corrected quantum state . Let P_s denote the projection onto the subspace with the syndrome string s. We define the error correction state formally as

$$C(\bar{\rho}) = \sum_s U_s P_s \rho P_s U_s^\dagger \,. \tag{2.12}$$

Similarly, we can define the error corrected logical operator. It is the operator that we would measure on the faulty state in order to get the same statistics as a measurement on the error corrected state. For a logical operator L, the error-corrected logical operator is

defined as

$$L_{\rm ec} = \sum_s P_s U_s^\dagger L U_s P_s \,. \tag{2.13}$$

We can verify that $tr(C(\bar{\rho})L) = tr(\rho \bar{L}_{\rm ec})$ by the cyclical rule of the trace operation.

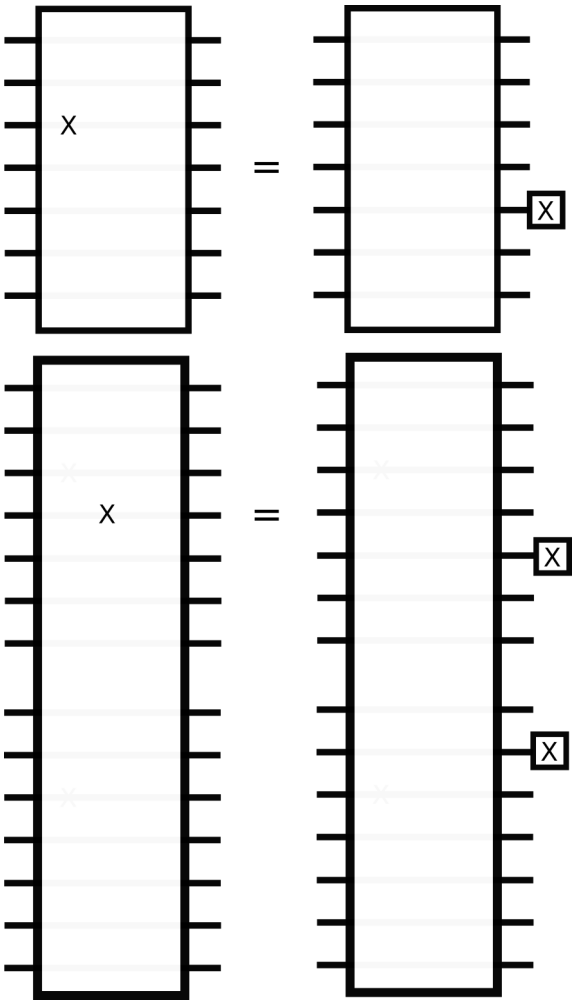

Figure 2.4: Sketch of fault tolerant circuits on one and two blocks. A fault, denoted with an X is equivalent to a circuit with at most one fault in each output block.

Next, I sketch how to do fault-tolerant implementations with faulty gates using concatenated codes. A faulty circuit is modeled as a perfect circuit with faulty locations where a

Pauli X, Y or Z error is applied. Each location is chosen independently and identically with a probability p and the Pauli error that is applied is chosen adversarially so as to maximize the probability of a decoding error according to the chosen decoding algorithm. Faults can be distinguished from errors. Errors refer to Pauli operators on a quantum state. A fault may actually reduce the number of errors on a state. For CSS codes, the error correction circuits can be done with gates that are in the Clifford group. This means that a set of faults in the middle of a circuit is equivalent to a perfect circuit followed by a different set of faults at the end of the circuit. We call this propagation of faults.

The basic strategy for designing fault tolerant circuits is to design the circuits so that faults do not propagate very far if the faults are sparse enough. Some basic rules for this are summarized in [9]. There the authors design two types of circuits. One for doing error correction and another for doing encoded Clifford group operations. Higher levels of error correction are done by replacing each qubit with an encoded qubit and each gate of a circuit with an encoded operation followed by error correction on each block. In this way the circuits are self-similar. We repeat this procedure until the probability of a decoding error is as low as we need it to be. This works if the fault rate is low enough; below a certain threshold. The basic rules for constructing circuits for error correction and encoded gates are:

1 If the error correction circuit has no faults, it does error correction on the input state.

2 If there is an error free state and a single fault during a error correction circuit, then it is equivalent to a perfect error correction circuit followed by a zero or one faults.

3 If there is a single fault before or during the execution of a circuit for an encoded gate, then it is equivalent to a perfect circuit followed by a single fault in each output block.

The third rule for constructing encoded gates can be achieved with transversal gates.

The circuits have gates that operate on at most a single qubit in each block. And each qubit is involved in at most one multi-qubit gate.

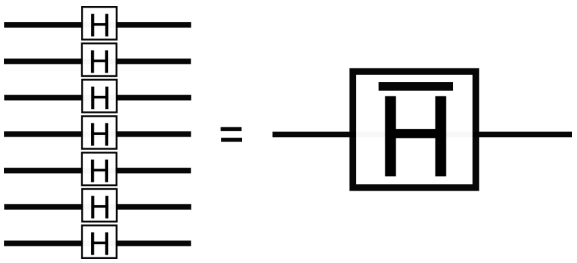

Figure 2.5: A fault-tolerant Hadamard.

For instance, a fault-tolerant implementation of a Hadamard operation, H, for the Steane code consists of just applying a Hadamard H on each qubit of a block. $\bar{H} = H^{\otimes 7}$. An error on any one of these qubits may become transformed by the Hadamard but it stays contained at that qubit. See [63, 69] for how to do fault-tolerant circuits for a universal set of gates.

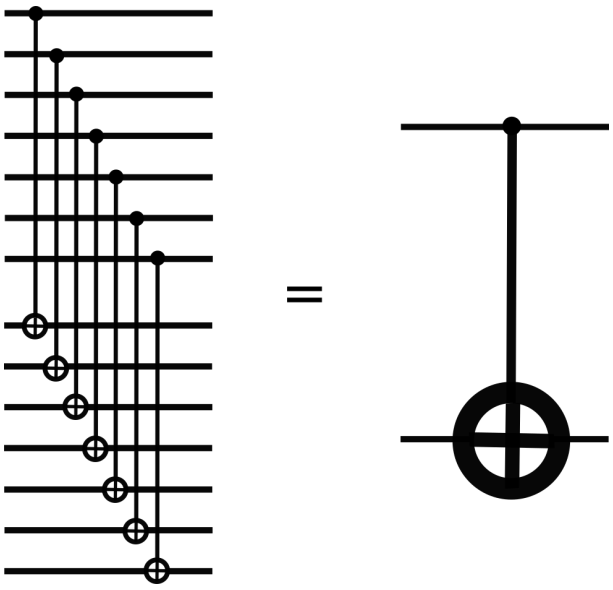

Figure 2.6: A transversal Cnot gate.

For a quantum memory, we need not be able to do a universal set of quantum gates. At a minimum we must be able to do a fault tolerant swap gate so that we can get quantum information in and out of the memory. We can construct a swap gate with three controlled not(Cnot) gates. A Cnot gate has a control qubit, i and a target qubit j. If the control qubit is in the 0 state, nothing happens but if the control qubit is in the 1 state, the second qubit is flipped. We can summarize this in the equation

$$C_{12}|a,b\rangle = |a, b \oplus a\rangle \,. \tag{2.14}$$

The Cnot gate is uniquely defined by its conjugations rules: $C_{ij}X_iC_{ij}^\dagger = X_iX_j$, $C_{ij}X_jC_{ij}^\dagger = X_j$, $C_{ij}Z_iC_{ij}^\dagger = Z_i$ and $C_{ij}Z_jC_{ij}^\dagger = Z_iZ_j$. We will now construct the encoded Cnot gates and verify it by verifying that it transforms the encoded logical operators in the same way. For two blocks that use identical CSS codes, we can simply do a Cnot by choosing the control qubits in one block and the target qubits in the other block. We pair up every qubit in this way so that the pairing is one-to-one and every qubit is paired to another qubit. We denote this operation as $\bar{C}_{ij}$ where i and j now denote blocks of qubits. $\bar{C}_{ij}$ has the correct conjugation rules for the logical operators $\bar{Z}$ and $\bar{X}$ because all of these encoded operations are just products of the unencoded operations.

The swap gate is defined by the property that $SWAP|a,b\rangle = |b,a\rangle$. We can perform this gate between qubits i and j by doing three Cnot gates: $SWAP_{ij} = C_{ij}C_{ji}C_{ij}$. We can show this by using equation 2.14 repeatedly. $C_{12}C_{21}C_{12}|a,b\rangle = C_{12}C_{21}|a, b\oplus a\rangle = C_{12}|b, b\oplus a\rangle = |b,a\rangle$. Similarly, to perform the encoded swap gate between blocks i and j we simply do three encoded Cnot gates. $\bar{SWAP} = \bar{C}_{ij}\bar{C}_{ji}\bar{C}_{ij}$.

Finally, I want to mention the equivalence between self-correcting memories and certain constant depth circuits for error-correction. The circuit goes as follows. First we measure

each stabilizer in a generating set and store it into a memory. Then for each qubit, we look at the Z-stabilizers(X-stabilizers) that act on that qubit. If more than half of these stabilizers are violated, then we apply a bit flip(phase flip). When the stabilizers in the generating set are local in some dimension, then this circuit will be local and of constant depth. Stabilizer codes with local generating sets are called topological stabilizer codes. I cover these in the next chapter.

Chapter 3

TOPOLOGICAL STABILIZER CODES

Topological stabilizer codes are stabilizer codes that have a local generating set, i.e. all the terms have support on a local neighborhood of qubits, in some finite dimension d. Topological stabilizer codes have codewords that exhibit topological order. Topological ordered states are states whose properties cannot fully be described by expectations of local operators. The study of topological order has a long history. One example is with lattice gauge theory founded by Wegner [85]. Another example is the fractional quantum hall effect [82]. Many people realized that the fractional quantum hall effect could not be described by the theory of Landau symmetry breaking and needed a new type of order, topological order, that takes zero temperature quantum effects into consideration [86]. The connection between quantum error correction and topological order was pointed out by Kitaev [51, 52]. This model is deeply related to the Dual of the 3-d Ising model considered by Wegner [85].

The Ising model is an example of a classical code that has local terms. I discuss that first and then move onto the toric code. I then re-frame the toric code and all CSS codes in terms of a generalized Ising model with non-local spin operators.

3.1 Ising code

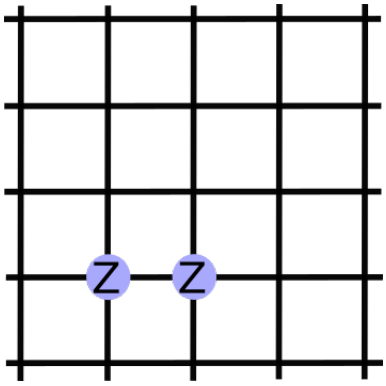

Figure 3.1: The 2-d Ising model. The edges represent Z-type terms in the Hamiltonian.

The stabilizer group of the repetition code is generated by $\{Z_iZ_j : \forall i,j \in V\}$ where V is the set of qubits. The code space of the Ising code is just $\{|0...0\rangle, |1...1\rangle\}$. The logical X operator is $\bar{X} = XX....X$ and the logical Z operator is $\bar{Z} = Z_i$ for any i. Because a single Z_i can change a superposition, the repetition code has a quantum distance of 1. Its classical distance is just the total number of qubits since we have to flip everyone to do an encoded bit flip.

There are many local generating sets of the repetition code. One of them is the set of terms in the 2-d Ising model. The terms in the generating set can be labeled by edges of a graph. Let E be the set of edges. The generating set is $\{Z_iZ_j : (i,j) \in E\}$. By multiplying a line of edges between qubit a and qubit b, we can generate Z_aZ_b for any pair of qubits a and b. Indeed any fully connected graph produces a generating set for the repetition code, even the 1-d Ising model. The 2-d Ising model is special because it has some self-correcting properties when local dynamics are allowed. For instance, it has a local error-correction rule which is resilient against noise from an i.i.d. binary symmetric channel.

Indeed it is just the classical repetition code with local syndromes in 2-d. Let i and j

denote qubits and vertices. The vertices are on a 2-d lattice as in figure 3.1. The parity check operators are denoted by the edges of the graph. Specifically they are $\{Z_i Z_j : (i,j) \in E\}$ where E is the set of edges. I call these operators edge operators. By multiplying a line of edges we can generate any operator of the form $Z_a Z_b$ for any qubits a and b. Hence the edge operators generate the parity checks of the repetition code.

3.2 Terms of the toric code

Next I discuss the toric code which is a local quantum code. I define the terms and discuss the logical operators. The logical operators have a macroscopic weight that scales with a power of the number of qubits in the code. Furthermore, there are decoders that preserve the original stored quantum state with a very high probability. That is, the toric code has a threshold against a single round of adversarial i.i.d. noise.

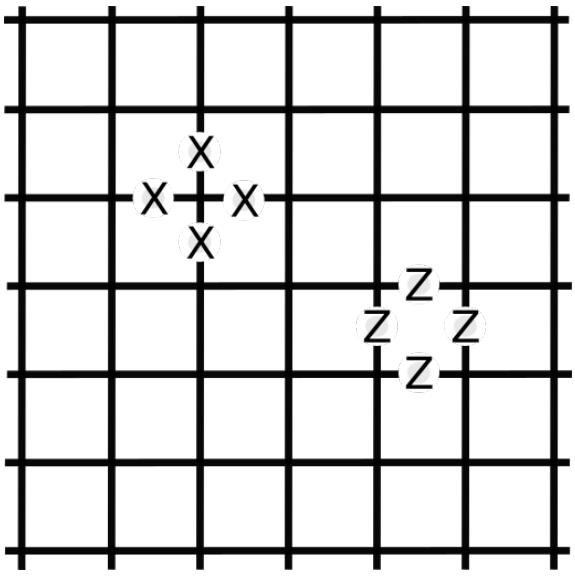

Figure 3.2: The toric code. Qubits are labeled by edges. The X and Z generators are indicated by the star of Xs and the plaquette of Zs. The boundaries are periodic.

We define the generators of the toric code with respect to a graph, shown in figure 3.2

(qubits are labeled by edges). The graph is a square lattice with $d \times d$ primitive cells. The boundary conditions are cyclical so that the graph has the topology of a torus. The stabilizers in the generating set are labeled by vertices of the graph and faces of the primitive cells. We refer to the faces as plaquettes. The vertices are

$$V = \{\boldsymbol{v} = (v_1, v_2) : v_i \in \{1, ..., N\}\}\,. \tag{3.1}$$

Let $\Gamma(\boldsymbol{v})$ be the set of edges that neighbor a vertex $\boldsymbol{v}$. For each $\boldsymbol{v}$ define the term $h^X_{\boldsymbol{v}} = \prod_{e\in\Gamma(\boldsymbol{v})} X_e$. These are called star operators. Let ∂f be the set of edges on the boundary of a plaquette f. For each plaquette f define a term $h^Z_f = \prod_{e\in\partial f} Z_e$. These are called plaquette operators. The generating set is a sum over elements of the set of vertices V and the set of faces F:

$$H = \cup_{\boldsymbol{v}\in V} h^X_{\boldsymbol{v}} \cup_{f\in F} h^Z_f\,. \tag{3.2}$$

The logical operator $\bar{X}$ of the toric code resembles a string that wraps around the torus. Here is how to understand this. Multiplying vertex operators generates closed loops of qubits and pairs of loops that go around the torus called non-trivial loops. These pairs can be made to be far apart so that each overlaps with a disjoint set of plaquettes. Hence each non-trivial loops commutes with each term in the generating set, yet is not generated by them. We conclude that the logical operator $\bar{X}$ is a tensor product of single qubit X operators on a single loop.

The logical operator $\bar{Z}$ also resembles a string that wraps around the torus. Plaquette operators are related to the star operators by a Hadamard and a transaltion by one half a lattice spacing up and to the right. Hence it has the same properties as the logical X operator.

There are no other logical operators as the only X and Z-type operators that commute

with the stabilizers are closed loops. Closed loops that do not wrap around the torus are generated by the stabilizer group as are pairs of loops that wrap around the torus. The only types of loops that are not generated are single loops around the handle or hole of the torus or both. These are the logical operators.

The set of logical operators completely determines the ground state degeneracy. There are two ways for each string to wrap around the torus, around the handle and around the center hole. Each way of wrapping corresponds to an encoded qubit. Hence the ground state degeneracy is $2 \times 2 = 4$.

I have described the toric code but rather in abstract terms. It's not very intuitive. I would like to introduce a more intuitive way of thinking about topological stabilizer codes in a way that generalizes to many other

3.3 Non-local Ising model

I want to give an alternative formulation of stabilizer codes that makes logical operators the focus rather than the terms in the Stabilizer group. I do this because this language will be particularly convenient to describe topological stabilizer codes. The key observation is that the product of any two equivalent logical operators[1] is a stabilizer and that furthermore any stabilizer can be expressed as the product of two logical operators.

We can show this with the equivalent logical operators L_1 and L_2 and the code state $|\psi\rangle$ by noting that if $L_1|\psi\rangle = L_2|\psi\rangle$ then $|\psi\rangle = L_1 L_1|\psi\rangle = L_1 L_2|\psi\rangle$. Hence the operator $L_1 L_2$ stabilizes the code space. Conversely, every stabilizer can be expressed as the product of two equivalent logical operators. If S is a stabilizer and L_1 is a logical operator, then we can define $L_2 = SL_1$. By definition then $S = L_2 L_1$.

Alternative descriptions of things often lead to new insights. One description of a sta-

[1]Two logical operators are equivalent when they have the same action on the ground state subspace. That is, L_1 and L_2 are equivalent if $L_1|\psi\rangle = L_2|\psi\rangle$ whenever $|\psi\rangle$ is a code state.

bilizer group then is to define a set of equivalent logical X operators, denoted by $\mathcal{L}_X$ and a set of equivalent logical Z operators, denoted by $\mathcal{L}_Z$. The following equation describes a stabilizer group with logical operator $\mathcal{L}_X$ and $\mathcal{L}_Z$.

$$\langle \{\bar{X}'\bar{X}'' : \forall \bar{X}', \bar{X}'' \in \mathcal{L}_X\} \cup \{\bar{Z}'\bar{Z}'' : \forall \bar{Z}', \bar{Z}'' \in \mathcal{L}_Z\} \,. \tag{3.3}$$

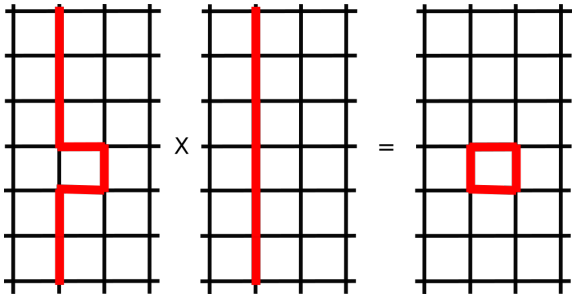

Figure 3.3: The product of two non-local spin operators of the Z type results in a plaquette operator.

We can describe the toric code more simply with this language. For instance, we can describe a set of equivalent logical Z operator of the toric code as the set of strings that either go around the handle or center of the torus. One string may be different from another by a small deformation. The product of the two operators gives a plaquette. See figure 3.3.

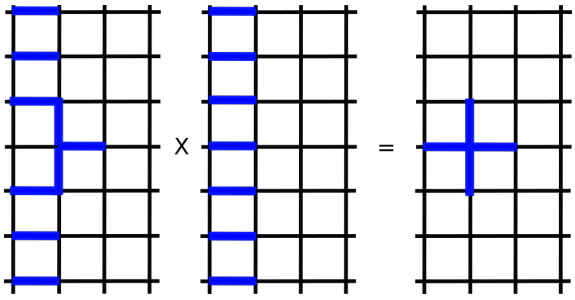

Figure 3.4: The product of two non-local spin operators of the X type results in a star operator.

Similarly, we can describe a set of non-trivial logical X-operators of the toric code as strings that cross edges of the graph. These strings again either go around the handle or the center of the torus. The product of two of these strings, one of which is deformed slightly, gives a plaquette operator.

This formulation of stabilizer codes very much looks like a generalization of the Ising code. The logical Z operators of the Ising code is a single Pauli Z operator and the stabilizers take the form of $Z_i Z_j$ for all combinations of logical Z operators. The difference is that for a stabilizer code, the logical Z operator is a non-local spin operator; it is a product of very many on-site Z operators. Furthermore, while the Ising model only has one logical X operator, the product of all of the on-sight X operators, stabilizer codes generically have very many equivalent logical X operators and so must have both products $\bar{Z}'\bar{Z}''$ and products $\bar{X}'\bar{X}''$ in the stabilizer group.

The link to topology is that for a topological stabilizer group, each $\bar{X}$ operator is related to every other $\bar{X}$ operator by a series of allowed local deformations and every $\bar{Z}$ operator is related to every other $\bar{Z}$ operator by another series of allowed local deformations. These deformations are exactly the local terms in the generating set of the code. Multiplying an X type non-local spin operator by an X type term leads to either another $\bar{X}$ operator or a product of multiple such operators. This ensures that all of the non-local spins of a particular type, X or Z, agree with each other in the ground state.

We can use this formulation as inspiration for a way of inventing new topological codes. First we specify the set of non-local spin operators. If we've chosen them to be local deformations of each other, then we can find a stabilizer group which has precisely those logical operators and stabilizers generated by the local deformations.

More specifically, we pick two sets of sets of qubits $M = \{A, B, ...\}$ and $N = \{I, J, ...\}$ with the requirement that the number of elements in the intersection of a set in M and a

set in N is odd. That is, if $S \in M$ and $T \in N$ then $|S \cap T|$ is an odd number so that the non-local $\bar{X}$ and $\bar{Z}$ operators anti-commute. Then we construct a a generating set of the form $R = \{\bar{X}_A \bar{X}_B : A, B : A \oplus B \text{ is local}\} \cup \{\bar{Z}_I \bar{Z}_J : I, J : I \oplus J \textit{ is local}\}$.

3.4 Decoding topological codes

There are several decoders for the toric code. There is minimum weight perfect matching [28, 33], two renormalization group decoders [32, 42], a cellular automata decoder [47] and many more. They all roughly do the same thing. They try to take violated stabilizers that are near each other and annihilate them together by putting an error path between them. This path creates the same violation and hence gets rid of them. What if the error path that we chose isn't the exact one that the environment chose? When the error rate is low, violated stabilizers are usually the result of only a few errors as opposed to very many. Thus annihilating them with a short string between them is likely to produce a local loop when we multiply the operation done by the environment and the operation we do in order to do error correction. This is because each error path with a length of L happens with a probability proportional to Fp^L where p is the error rate. This biases shorter strings.

The minimum weight perfect matching decoder is a great example. It pairs up violated stabilizers in a way that minimizes the total path length between them. Below a threshold error rate of about $\sim 15\%$ [84], the probability for a decoding error can be made arbitrarily small as a function of the system size. Other decoders have other ways of matching nearby pairs of violated stabilizers.

The minimum weight perfect matching decoder does not have the highest threshold. This is because while an error path that is longer may be less probable, there are many more long error paths than short ones. The best decoder takes this into account and groups violated stabilizers together in a way that maximizes the probability of decoding correctly according

to whatever error model we are using. This is called maximum likelihood decoding.

The various decoders differ in runtime and locality. Minimum weight perfect matching takes a polynomial time in the system size while the renormalization group decoders take a logarithmic time. The cellular automata decoders take a constant time to implement but must be repeated several times to fully correct all the errors. The time that error correction takes doesn't matter if it is only used for post-processing a destructive measurement. However, if our aim is to keep the contents of the memory safe by continually correcting, then we need an error correction scheme that takes a constant time to implement. Otherwise errors would pile up past the point where they can be corrected without a decoding error. Self-correcting memories and cellular automata both take a constant time to do error correction and are thus good candidates for quantum memories.

All decoders for topological stabilizer codes use the same principle to decode. They group clusters of violated stabilizers together in a way that keeps the weight of the recovery operation small.

I would like to add to this list the annealing decoder, which I introduce in the next chapter and my own version of a cellular automata decoder which I get to in chapter 7.

Chapter 4

SELF-CORRECTING MEMORIES

The idea behind self-correcting memories is that the physics of the material can be used to automatically correct data that is stored in it. A classical example is the ferromagnetic hard-disk drive where the net magnetization is stable below a critical temperature as a result of the tendency for the local magnetization to align across neighboring regions. In this sense, a ferromagnetic memory is self-correcting. The extension to a quantum memory is similar, the local dynamics correct any physical errors that would destroy the readout statistics of a stored quantum state. At present I do not know of any physical examples of quantum memories. In this chapter I discuss a particular theoretical model of how one could engineer a material so as to suppress the dynamics of the environment. In order for the energy of a material to go up, it has to borrow that energy from the environment. However the entropy of the environment goes down when it loses energy and so there are more configurations of the environment with high energy than low energy. If the material is in a particular energy eigenstate, we expect that local changes in our quantum material that decrease the energy of the material are more likely than local changes that increase the energy.

The challenge is to design a system Hamiltonian that keeps the would-be error-corrected state very close to the original. We work with Hamiltonians which are local. We make this assumption so that the model is more physically realistic. Let ϕ be the encoding map and ρ the state we would like to encode so that $\phi(\rho) = \bar{\rho}$. The encoded state changes over time according to the map E_t such that $\bar{\rho}(t) = E_t(\bar{\rho})$. E_t will be the the map of the thermalizing dynamics. Roughly, the Hamiltonian is good if $C(\bar{\rho}(t)) \sim \bar{\rho(0)}$ for a very long time.

What is a very long time? We want the system to have the property that if we increase the number of subsystems, the memory time increases. How fast? Models for classical self-correcting memories, such as the 2-d Ising model, stay correctable for a time that scales exponentially as $\sim e^{\alpha L}$ where α is a positive constant and L^2 is the number of spins in the system. Our aim is to find out if a quantum model exists in 3-d with the same scaling. The 4-d toric code [4, 30] and the 6-d color code [13, 15] both work as a self-correcting quantum memory but at present it is unclear how to get a memory lifetime that scales better than a constant as a function of the system size for a Hamiltonian with local interactions of a bounded strength in 3-d.

Kitaev [51] was the first to describe the idea of a self-correcting quantum memory with his toric code model. The toric code exhibits some self-correcting properties when presented with a weak perturbation [51]. It can also protect against dynamical errors with Anderson localization [76]. However, there is a lot of evidence that the toric code thermalizes in a finite time when subjected to thermalizing noise [2, 5, 22, 50]. The reason is a lack of a macroscopic energy barrier for performing logical operations with a series of local operations. A single bit flip X can create a pair of violated plaquettes and move them around until a nontrivial string of X operators is formed constituting a logical error. There are theorems that say only a constant energy barrier is possible in 2-d for stabilizer code Hamiltonians [18] and frustration free systems [56]. The no-go result for stabilizer Hamiltonians was predicted earlier in [50].

The 4-d toric code toric code [30] was postulated to be self-correcting due to a large energy barrier for all of its logical operators. This was theoretically verified in [4] where an exponential lifetime was shown for local stochastic thermalizing dynamics. Furthermore, there is numerical evidence that the 4-d toric code has an exponential lifetime below a critical error rate with respect to local engineered dissipation [65].

However, the real challenge is to develop a self-correcting quantum memory in three

spatial dimensions or less. At present, this is an outstanding open question with huge implications for quantum computing and basic physics.

There have been many attempts at creating theoretical models that exhibit self-correction in 3-d. Bacon [10] suggested that a 3-d subsystem code, the Bacon-Shor code, might be self-correcting due to a macroscopic energy barrier. However, it is still open whether it is stable in a thermal environment.

There have been several proposals where stabilizers are coupled to an external system. One proposal [44] couples terms in the toric code to bosons in a cavity. They get a memory lifetime that scales polynomially with the number of qubits in the toric code but the strength of the interaction must increase with the system size, going to infinity in the thermodynamic limit, in order to achieve this result. There is another model [67] where terms in the toric code are coupled to a lattice of bosons. They claim a fully self-correcting memory based on the existence of a macroscopic energy barrier. However, the coupling terms are of unbounded strength.

There have also been several proposals based on non-local interactions. [27] and [48] propose a repulsive interaction between quasi-particles. These non-local models achieve a polynomial lifetime as a function of the system size. However, such models may be physically unreasonable as interaction are bounded by the speed of light and thus such non-local Hamiltonians may only manifest as good approximations when the system size is small.

The only models based on stabilizer codes that are local and exhibit some self-correction are the cubic code [17, 20, 40] and the welded solid code [62]. The welded solid code is my contribution. Both models have an energy barrier. The cubic code came first with a logarithmic energy barrier of $O(\log L)$ for a code with $O(L^3)$ physical qubits. The welded solid code reaches a power-law energy barrier of $O(L^{\frac{2}{3}})$, an exponential improvement. Both models are likely only partially self-correcting, where an increase in system size leads to an

increase in lifetime up to a maximum system size. The memory lifetime of the cubic code was lower bounded by $L^{\alpha\beta}$ for a constant α, an inverse temperature β and a maximum systems size of $L^3 \sim e^\beta$. Using the same theoretical results of this paper, the welded code achieves a memory lifetime of $e^{\beta L^{\frac{2}{3}}}$ for $L^3 \sim e^\beta$. This exponential improvement in memory lifetime is completely due to the exponentially better energy barrier.

Both models evade a no-go theorem by Yoshida [88] where he proves that only a constant energy barrier is possible for translation invariant Hamiltonians with a bounded number of logical qubits. The cubic code has an unbounded number of logical qubits and the welded solid code is periodic over a length that changes with the system size. Indeed for periodic codes, the cubic code saturates the maximum possible energy barrier [41]. At present there is no known bound for the energy barrier of non-periodic topological codes such as the welded solid code.

I would like to point out that stabilizer code Hamiltonians are a very restrictive class. General spin Hamiltonians are frustrated and have terms that do not necessarily commute. These no-go theorems do not put any limitations on such systems.

4.1 Thermalizing dynamics.

The model that I use for thermalizing dynamics is a local stochastic quantum master equation. The system evolves according to Hamiltonian evolution for an infinitesimal time and then randomly emits or absorbs quanta of energy. These transitions correspond to measurements by the environment. Emission and absorption is modeled as a set local jumps, $\{A_i\}$, between energy eigenstates corresponding to a change of energy ω_i. This means that the jump operators satisfy the commutation relationship

$$[H, A_i] = \omega_i A_i \,. \tag{4.1}$$

By local I mean that for each i, A_i acts non-trivially on only a local neighborhood of spins, i.e. each spin is within a constant distance of any other spin. Each jump operator is applied with a probability $r_i \delta t$ in an infinitesimal time δt. We assume that if a particular transition up in energy exists, then the same transition which lowers the energy must exist too and vice-versa. This means that if A_i is in the set of jump operators, then $A_i^\dagger$ is also in this set. Furthermore we assume that the rate to increase energy is related to the rate to decrease by an exponential factor $e^{-\beta\omega_i}$ so that the jumps that increase energy are less likely than jumps that decrease energy. Furthermore, if the system is ergodic, each energy eigenstate is eventually visited from any initial energy eigenstate, then the thermal probability distribution $\rho_\beta \sim e^{-\beta H}$ is the equilibrium distribution and detailed balance is satisfied. A system satisfies detailed balance if every process is balanced by its reverse process. In this case $e^{-\beta\omega_i} r_i tr A_i \rho_\beta A_i^\dagger = r_i tr A_i^\dagger \rho_\beta A_i$ where r_i is the rate to decrease energy. This sort of time evolution is known as a Davies map [7, 70, 79]. The time evolution equation is given by

$$\frac{\partial \rho}{\partial t} = -\mathcal{L}(\rho) = -i[H, \rho] + \tag{4.2}$$

$$\sum_i e^{-\beta\omega_i} r_i \left(A_i \rho A_i^\dagger - \frac{1}{2}\{A_i^\dagger A_i, \rho\} \right) + r_i \left(A_i^\dagger \rho A_i - \frac{1}{2}\{A_i A_i^\dagger, \rho\} \right). \tag{4.3}$$

The full time evolution is given by $\rho(t) = e^{-\mathcal{L}t}\rho$. We say that the state is good if the probability to measure an error corrected observable has not changed much.

Definition The memory lifetime is the smallest time t such that $|tr(\Pi_{\text{ec}}\rho(0)) - tr(\Pi_{\text{ec}}\rho(t))| < \epsilon$ for a positive constant ϵ and an error corrected observable Π_{ec} is an error corrected observable.

If the memory lifetime scales unboundedly with the system size, we say that the Hamiltonian H is self-correcting.

It would be interesting to know which classes of Lindbladians have long lifetimes? For instance, if all of the jump operators act only on a local neighborhood of qubits and if the Hamiltonian is self-correcting, will it continue to be self-correcting for different jump operators that also act on a local neighborhood? In section 4.4 we prove that the criteria does not depend on the details of the jump operators, just that they are local.

4.2 Classical self-correction:Ising model

The 2-d Ising model is a classical self-correcting memory. It has an exponential lifetime as a function of the system size according to Glauber dynamics [58] which is the discrete version of a Davies map. The Hamiltonian for the Ising model is given by

$$H = \sum_{(i,j)\in E} (1 - Z_i Z_j) \tag{4.4}$$

where E is the set of edges of the 2-d lattice. The terms are just the parity checks of the Ising code. We will consider jump operators that are bit-flips on individual qubits where the flip is more likely by a factor of $e^{\beta\omega}$ if the flip decreases the energy by ω. Let s denote the syndrome string. It is just a list of measurement outcomes of the generating set $s = ((Z_i Z_j)' : (i,j) \in E)$ according to some ordering of the measurement outcomes. Let P_s be the projection operator onto the subspace with a syndrome string s. Let Δ_k be the change in the syndrome string if we were to flip the kth qubit. A simple jump operator that satisfies equation 4.1 is given by

$$A_k(\omega) = \sum_{s:\omega = E(s\oplus\Delta_k) - E(s)} P_{s\oplus\Delta_k} X_k P_s \,. \tag{4.5}$$

This is equivalent to measuring all of the syndromes and only applying a pauli X if it increases the energy by ω. The time evolution equation is

$$\frac{\partial \bar{\rho}}{\partial t} = -i[H, \rho] + \sum_{k,\omega} e^{-\frac{\beta\omega}{2}} r_i \left(A_k(\omega) \bar{\rho} A_k^\dagger(\omega) - \frac{1}{2}\{A_k^\dagger(\omega) A_k(\omega), \bar{\rho}\} \right) . \tag{4.6}$$

By definition the operator $A_k(\omega)$ satisfies $[H, A_i] = \omega_i A_i$. Thus detailed balance is satisfied and ρ_β is the equilibrium distribution.

How do we encode and decode? For the Ising model, we will choose to do error correction and decoding by taking the majority vote of all of the spins and flipping spins that do not agree with the majority vote. The error-corrected logical operator is not always easy to write down in closed form as it often involves performing an algorithm. However the error-corrected logical operator for the Ising model is particularly simple. If we imagine each qubit to be a spin, as the Ising model was originally imagined, then the error corrected logical operator is just the sign of the magnetization

$$\bar{Z}_{\text{ec}} = \frac{\sum_i Z_i}{|\sum_i Z_i|} . \tag{4.7}$$

We choose an encoding map ϕ as in [26] so that all of the degrees of freedom other than the error corrected logical operator are thermalized. This assumption simplifies the proof for the criteria of the memory lifetime in section 4.4. Let η be the density matrix of the quantum state that we intend to encode in the Ising code. We assume that η is diagonal in the Z basis so that it represents a classical probability distribution. Hence

$$\eta = \frac{1 + \Delta p Z}{2} . \tag{4.8}$$

Define $\bar{\eta}_{\text{ec}} = \frac{1+\Delta p \bar{Z}_{\text{ec}}}{2}$. We choose the encoding map to be

$$\bar{\rho} = \phi(\eta) = \frac{2e^{-\beta H}\bar{\eta}_{\text{ec}}}{tre^{-\beta H}} \,. \tag{4.9}$$

Essentially this is the thermal density matrix that has been projected onto a state with net magnetization + or -1 with a probability of $\frac{1\pm\Delta p}{2}$.

Below a critical temperature, the thermal density matrix of the Ising model has a phase transition which means that the probability to find the state of the system with a magnetization very near zero is very small. However, in order for the local jump operators to change the net magnetization, the net magnetization very likely has to go through a state with net magnetization zero. States with net magnetization zero have a high energy on average and are thus suppressed. The minimum such energy is $2L$ where L^2 is the number of qubits in the 2-d grid. This minimum energy is called the energy barrier.

The physical interpretation is that the dynamics tend to shrink contractible domains of spins. A domain is an island of spins that share the same value, 0 or 1. A domain is contractible when its boundary can be shrunk to a point with local deformations. The dynamics do this by doing a local majority vote. The dynamic are complicated but roughly, they tend to erode kinks in the domain causing them to tend to shrink. This because the the energy penalty of a domain is proportional to the length of the boundary. Thus larger domains are suppressed. They are complicated because a pure majority vote wouldn't work unless it failed with some probability. We can see this by considering a square island of 0s surrounded by 1s. If you flip a corner from 0 to 1, then the energy is the same and thus such a transition is not preferred However, the kink at the corner will tend to do a random walk and eventually reach the other side. For instance, the four leftmost diagrams of figure 4.1 all have the same energy. Only the rightmost diagram has less energy and is preferred.

If we have next to nearest neighbor terms or if we went to three dimensions, there would be no ambiguity about how to minimize the energy locally.

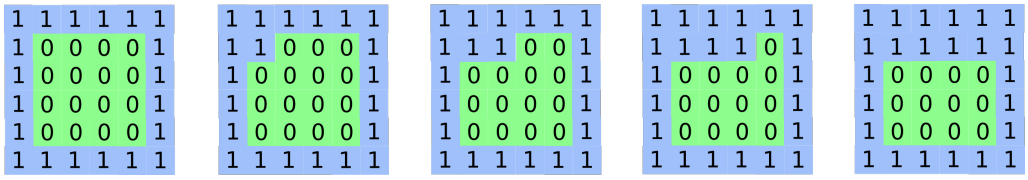

Figure 4.1: Minimizing the energy for the Ising model does not provide a definite path for the configurations to be annihilated. Some randomness is needed. Only on the rightmost diagram does the energy decrease.

We will prove a theorem in section 4.4 that says that the memory lifetime scales as

$$t \sim \frac{1}{poly(n) tr(P_{m=0}\rho_\beta)} \tag{4.10}$$

where m is the net magnetization and $P_{m=0}$ is the projector onto states with a magnetization of zero. Hence if the probability to have a net magnetization zero is very small, say exponentially small, then the memory lifetime will be exponentially big.

It is very important to distinguish the dynamics from the equilibrium distribution when talking about the stability of a self-correcting memory. For instance, the 2-d Ising model has a partition function which exhibits a phase transition. However, the lifetime of the Ising model depends very heavily on what sort of transitions are allowed in the dynamics. If the transition that corresponds to flipping every spin, $X^{\otimes n}$, is allowed with a finite probability, then the Ising model would have a constant life-time no matter what the system size even though it has a phase transition. This is because the all zeros state would mix with the all ones state in a constant time. We can evade this problem by choosing jump operators that are local.

Now that we have a feeling for how the dynamics work, we'll go back to the magnetization

of the Ising model and see why it's hard to maintain a superposition of it. The magnetization is local in nature. We can get a good idea of what the average magnetization is by looking at only one spin; in particular by measuring Z_i on the ith spin. It will be more likely to point in the direction of the average magnetization than not. The information is not hidden at all.

We'll have problems if we try to make a superposition of two magnetizations. First I show how this works when all of the spins point in the same direction and then what happens when it is only approximate. Say we were able to prepare the state $\alpha|00...0\rangle + \beta|11...1\rangle$. If we measured a single bit, the state will collapse to one or the other magnetization. If there are N spins total and each spin is being measured with a probability p, then the probability for no errors is $(1-p)^N$. That's exponentially small in the number of spins. So for a system with a very large number of spins, it will collapse right away.

The same argument goes through even if we have a superposition of less than perfect magnetization. Only then, making a measurement on a single bit collapses the global state with a finite probability, since each bit isn't necessarily entangled. For example a state such as $|0\rangle(\alpha|0...0\rangle + \beta|1...1\rangle)$ is still a superposition of net magnetization but measuring the first bit wouldn't collapse the state. Still the probability for the superposition to stay coherent would be exponentially small in the number of coherent spins.

There is yet another problem when you try to take the superposition of states with definite net magnetization. A phase operation Z on a single qubit can change the global phase completely. A single phase, a Z_i operator on spin i, for any i, would operate as follows on the state. $Z_i\,(\alpha|0...0\rangle + \beta|1...1\rangle)$ to $\alpha|0...0\rangle - \beta|1...1\rangle$. Hence the density matrix would quickly become diagonal in the Z basis.

To summarize, the properties of a 2d Ising model, what makes it a good classical memory and what prevents it from being a good quantum memory.

1. The magnetization is robust against local bit flips.

2. The magnetization is very sensitive to local phase errors.

3. The magnetization is not hidden. You can get information about it from a single spin.

In the next section I discuss how we might circumvent these issues.

4.3 Quantum self-correction

We will be considering a Hamiltonian where the terms are the local generators of a topological stabilizer code

$$H = \sum_{h \in R} -h \tag{4.11}$$

where $\langle R \rangle = S$.

We will split the stochastic part of the master equation into jumps corresponding to single qubit X-operators and jumps corresponding to single qubit Z operators. Let s be the syndrome string where $s = (h : h \in R)$ for some ordering of the generators. Let P_s be the projection operator onto the subspace with syndrome strings s. Let Δ_{X_k} be the change in the syndrome string with an application of the single-qubit Pauli-operator X_k. Similarly let Δ_{Z_k} denote the change in the syndrome string with an application of the single qubit Pauli-operator Z_k.

$$A_{X_k(\omega)} = \sum_{s:\omega = E(s \oplus \Delta_{X_k}) - E(s)} P_{s \oplus \Delta_{X_k}} X_k P_s \tag{4.12}$$

$$A_{Z_k(\omega)} = \sum_{s:\omega = E(s \oplus \Delta_{Z_k}) - E(s)} P_{s \oplus \Delta_{Z_k}} Z_k P_s \,. \tag{4.13}$$

The time evolution equation will be

$$\frac{\partial \bar{\rho}}{\partial t} = -i[H, \bar{\rho}] \tag{4.14}$$

$$+\sum_{k,\omega} e^{-\frac{\beta\omega}{2}} \left(A_{X_k}(\omega)\bar{\rho}A_{X_k}(\omega)^\dagger - \frac{1}{2}\{A_{X_k}(\omega)^\dagger A_{X_k}(\omega), \rho\} \right) \tag{4.15}$$

$$+\sum_{k,\omega} e^{-\frac{\beta\omega}{2}} \left(A_{Z_k}(\omega)\bar{\rho}A_{Z_k}(\omega)^\dagger - \frac{1}{2}\{A_{Z_k}(\omega)^\dagger A_{Z_k}(\omega), \rho\} \right) . \tag{4.16}$$

How do we encode and decode? Very similarly to how we did it for the Ising model. We have error corrected logical operators $\bar{X}_{\text{ec}}$, $\bar{Y}_{\text{ec}}$ and $\bar{Z}_{\text{ec}}$ where $\bar{Y}_{\text{ec}} = -i\bar{Z}_{\text{ec}}\bar{X}_{\text{ec}}$. These error corrected logical operators commute with the Hamiltonian and so constitute and operators on a subsystem. If we want to encode the state $\rho = 1 + aX + bY + cZ$, we choose encoding map ϕ to thermalize all other degrees of freedom except the error-corrected logical operators. This allows us to focus on how quickly just the encoded state thermalizes as all other degrees of freedom are already thermalized.

$$\bar{\rho} = \phi(\rho) = \frac{2e^{-\beta H}}{tr(e^{-\beta H})} \left(1 + a\bar{X}_{\text{ec}} + b\bar{Y}_{\text{ec}} + c\bar{Z}_{\text{ec}}\right) . \tag{4.17}$$

Annealing decoder With the Ising model, the majority vote was an obvious candidate for doing error correction. It is not so obvious for quantum codes. I am interested in topological codes that are stable to thermalizing dynamics. This means that the error-corrected operators have the same statistics for a very long time if they were to be measured. If we turn down the temperature, we expect that the error-corrected logical operators will be even more stable. Turning down the temperature to zero brings the state into the code space. Hence lowering the temperature to zero is itself a decoder. I call this decoding by annealing. Of course we can never completely turn down the temperature. Instead, decoding by annealing is a conceptual tool that allows us to compare different Hamiltonians for self-correction.

There are states for which a single X operator or a single Z operator changes the error-corrected state. These states are the bottle neck for our stochastic master equation. We

denote the projection operator onto these states with $P_\perp$. In the next section we will show the memory lifetime scales as

$$t \sim \frac{1}{Poly(n) tr(P_\perp \rho_\beta)} \tag{4.18}$$

where ρ_β is the thermal density matrix corresponding to the Gibbs distribution. Hence if the probability for the state to be in the subspace $P_\perp$ is low, then the memory lifetime will be high.

Physically, self-correcting memories rely on there being a way to minimize the energy by local measurements and local operations in order to do error correction.

4.4 Stability criteria and memory lifetime

In what follows I derive some conditions for when we can ignore the thermalizing dynamics and use the equilibrium distribution, i.e. the Gibbs distribution, to predict how long an encoded state will stay close to the original. The basic idea can be illustrated with the stability requirements for a ferromagnet. To transition from one net magnetization to another with random local changes to the magnetization, the state of the system will likely have to be very close to zero magnetization at some point in between. If we wanted to preserve the net magnetization of a ferromagnet then we would want the probability for zero net magnetization in the equilibrium distribution to be very low. The theorem I prove makes this intuition rigorous and applies equally to quantum states encoded using topological stabilizer codes. The basic idea is that one condition for an encoded quantum state to be stable is that the equilibrium distribution is very unlikely to be in a state where a single operation by the environment changes the decoded state. This theorem is a generalization of the beautiful theorem in [26].

The version I present differs in three way. First, the original theorem assumes a stabilizer or a subsystem code Hamiltonian and I assume any Hamiltonian with a ground state de-

generacy. Second, the original theorem assumed that error correction happens by measuring stabilizers and applying Pauli operators to correct errors. I assume measurements of energy eigenstates followed by any error correction operation. Finally, the original result was for Davies maps with single qubit jump operators. I assume only a Lindblad equation that has the thermal density matrix as the equilibrium distribution. Otherwise, the two proofs are identical and much of the language is the same.

I assume that the thermalizing dynamics follow Lindbladian evolution: $\frac{\partial \rho}{\partial t} = \mathcal{L}(\rho)$ where

$$\mathcal{L}(\rho) = -i[H, \rho] + \sum_i L_i \rho L_i^\dagger - \frac{1}{2}\{L_i^\dagger L_i, \rho\} \tag{4.19}$$

where H is the Hamiltonian, ρ is the quantum state and L_i are Lindblad operators. Furthermore I assume that the Lindblad evolution has the unique equilibrium distribution ρ_β.

Definition We also have a decoding map ϕ_{out} that maps any density matrix into a density matrix in the ground state subspace. I assume a particular form for the decoding map. We first measure the energy eigenstate, labeled by s, which projects the state onto the subspace P_s. We then apply an error correcting operation $C(s)$.

$$\phi_{\text{out}}(\rho) = \sum_i C(s) P_s \rho P_s C(s)^\dagger \,. \tag{4.20}$$

The error corrected logical operators are much the same as for stabilizer code Hamiltonians except now P_s may be a more general measurement; not only a measurement of stabilizers.

Definition For decoding map $\phi_{\text{out}}(\rho) = \sum_i C(s) P_s \rho P_s C(s)^\dagger$ and a ground state observable

G, we define the error corrected observable

$$G_{\text{ec}} = \sum_s P_s C(s)^\dagger G C(s) P_s \,. \tag{4.21}$$

The error corrected operator is guaranteed to commute with the Hamiltonian, by definition. This means that we can split the Hilbert space into a tensor product and find a basis in which we have two labels, one for the energy eigenstate and one for the encoded quantum state.

Next we define a projector P onto a subspace where an application of the Hamiltonian or one of the Lindblad operators does not change the decoded state.

Definition Let P be a projector onto a subspace that is protected from a set of observables $\mathcal{A}$. That is, $P|\psi\rangle = |\psi\rangle$ if and only if $\phi_{\text{out}}(A|\psi\rangle\langle\psi|A^\dagger) \propto \phi_{\text{out}}(|\psi\rangle\langle\psi|)$ for $A \in \mathcal{A}$. We use a proportionality symbol because the operator A may not be unitary. We can define $P_\perp = 1 - P$. States in $P_\perp$ have the property that a single application of A_i changes the error corrected state. I call $P_\perp$ the ridge of correctability or the ridge of the energy landscape when $\mathcal{A}$ are generators from a local Davies map, i.e. local jump operators that correspond to measuring energy eigenstates and jumping to a higher or lower energy eigenstate.

The memory fails when the probability to make a measurement on the error corrected state $C(\bar{\rho}(t))$ differs from the original state by more than ϵ. We can bound this value with

the one norm $\|(\bar{\rho}(t) - \bar{\rho}(0)\|_1$. We prove this as follows:

$$\begin{aligned} & |tr(\Pi_{\text{ec}}\bar{\rho}(t)) - tr(\Pi_{\text{ec}}\bar{\rho}(0))| \\ &= |tr(\Pi_{\text{ec}}(\bar{\rho}(t) - \bar{\rho}(0))| \\ &\leq \|\Pi_{\text{ec}}(\bar{\rho}(t) - \bar{\rho}(0)\|_1 \\ &\leq \|\Pi_{\text{ec}}\|_\infty \|(\bar{\rho}(t) - \bar{\rho}(0)\|_1 \\ &= \|(\bar{\rho}(t) - \bar{\rho}(0)\|_1 \end{aligned} \quad .$$

Theorem 4.4.1 *For the Lindblad operator $\mathcal{L}$ with local*[1] *jump operators $\mathcal{A} = \{A_k\}$ with the unique stationary distribution $\rho_\beta \sim e^{-\beta H}$ and for a state stored in the subsystem of error corrected logical operators there exists an encoding and decoding map such that*

$$\|\rho(t) - \rho(0)\|_1 \leq 2dt\|\mathcal{L}\|_1 tr((1-P)\rho_\beta) \tag{4.22}$$

where d is the degeneracy of the ground state, $\rho(0) \sim \eta_{ec}e^{-\beta H}$, η_{ec} the error corrected state that we would like to encode $\eta_{ec} = \sum_i a_i G_{i\,ec}$, P is a projector onto a subspace where local jump operators do not change the expectation of error corrected operators and $\|\mathcal{L}\|_1 = \max_{F:\|F\|_1=1} \|\mathcal{L}(F)\|_1$. The encoding and decoding maps Φ_{in} and Φ_{out} are as follows: for a single qubit density matrix η, $\Phi_{in}(\eta) = 2\bar{\eta}_{ec}\rho_\beta$ and $\Phi_{out}(\bar{\eta}_{ec}I_{ec} \otimes \sigma) = \eta$.

I want to note that this theorem is essentially the same as the original [26]. My contribution is to generalize the conditions for when it holds.

What this theorem says is that if the state we start out in has all degrees of freedom thermalized except the encoded state, then the decoded state takes a very long time to change so long as the probability to be in a transition state is very low.

In order to show that a particular system has a long memory lifetime we pick a maximum

[1]By local I mean operators that act nontrivially only within a ball of constant radius.

difference in probability ϵ that we are willing to tolerate between measurements on the intended encoded state and the final error corrected state. The time it takes for this to happen is lower bounded by $t \geq \frac{\epsilon}{2d\|\mathcal{L}\|_1 tr((1-P)\rho_\beta)}$. For a Lindbladian $\mathcal{L}$ with only local operations of bounded strength the number of terms is a polynomial of the volume: $\|\mathcal{L}\|_1 = poly(V)$ where $poly(V)$ is some polynomial of the volume. If all of the intermediate states, states in the ridge of correctability, $P_\perp$, have very large energies then the memory time will be very long. This is why the energy barrier is of paramount importance because if the energy barrier is large, then these intermediate states will have energies of at least the energy barrier and thus a low probability. A high energy barrier, however, is not enough to guarantee that the encoded state is stable for there could be very many intermediate states. However, a high energy barrier is a necessary condition for a long memory lifetime as was shown in [80].

This theorem goes only one way, you can prove that a particular system has a long memory lifetime but it doesn't help you prove that the memory lifetime is short. There you would have to prove that for any initial encoding, the memory time is short. However, in our proof, we assumed a particular encoding, where everything except the encoded state is thermalized.

The proof of this theorem relies on the following lemma.

Definition Let ϕ_s be the time evolution operator for a time s: $\phi_s(\rho) = \rho(s)$.

Lemma 4.4.2 *Let F be Hermitian operator. Thermalizing dynamics can only decrease the 1-norm of the state: $\|\phi_s(F)\|_1 \leq \|F\|_1$.*

Proof Decompose F into the parts with positive and negative eigenvalues. $F = F_+ - F_-$ where $F_\pm \geq 0$. $\|F\|_1 = tr(F_+ + F_-)$.

Let $\{|\alpha\rangle\}$ be a basis for which $\phi_s(F)$ is diagonal.

$$\|\phi_s(F)\|_1 = \sum_\alpha |\langle\alpha|(\phi_s(F_+) - \phi_s(F_-))|\alpha\rangle| \tag{4.23}$$

$$\leq \sum_\alpha \langle\alpha|\phi_s(F_+)|\alpha\rangle + \langle\alpha|\phi_s(F_-)|\alpha\rangle \tag{4.24}$$

$$= \sum_\alpha \langle\alpha|\phi_s(F_+ + F_-)|\alpha\rangle = \|F\|_1 \,. \tag{4.25}$$

This last line is true because the time evolution preserves the total probability of a density matrix and $F_+ + F_1$ is a positive operator. $\square$

Here is the proof of the main theorem of this section.

Proof I state the proof line by line and then explain each line after.

$$
\begin{aligned}
&\|\rho(t)-\rho(0)\|_1 && (4.26)\\
=&\|\int ds\dot{\rho}(s)\|_1 && (4.27)\\
\leq&\int ds\|\dot{\rho}(s)\|_1 && (4.28)\\
t=&\int ds\|\mathcal{L}(\phi_s(\rho(0)))\|_1 && (4.29)\\
=&\int ds\|\phi_s(\mathcal{L}(\rho(0)))\|_1 && (4.30)\\
\leq&t\|\mathcal{L}(\rho(0))\|_1 && (4.31)\\
=&t\|\mathcal{L}(d\bar{\eta}_{\mathrm{ec}}\rho_\beta))\|_1 && (4.32)\\
=&t\|\mathcal{L}(d\bar{\eta}_{\mathrm{ec}}(P+(1-P))\rho_\beta))\|_1 && (4.33)\\
\leq&t\|\mathcal{L}(d\bar{\eta}_{\mathrm{ec}}P\rho_\beta)\|_1+t\|\mathcal{L}(d\bar{\eta}_{\mathrm{ec}}(1-P)\rho_\beta)\|_1 && (4.34)\\
=&t\|d\bar{\eta}_{\mathrm{ec}}\mathcal{L}(P\rho_\beta)\|_1+t\|\mathcal{L}(d\bar{\eta}_{\mathrm{ec}}(1-P)\rho_\beta)\|_1 && (4.35)\\
=&t\|d\bar{\eta}_{\mathrm{ec}}\mathcal{L}((1-(1-P))\rho_\beta)\|_1+t\|\mathcal{L}(d\bar{\eta}_{\mathrm{ec}}(1-P)\rho_\beta)\|_1 && (4.36)\\
=&t\|d\bar{\eta}_{\mathrm{ec}}\mathcal{L}((1-P)\rho_\beta)\|_1+t\|\mathcal{L}(d\bar{\eta}_{\mathrm{ec}}(1-P)\rho_\beta)\|_1 && (4.37)\\
\leq&td\|\bar{\eta}_{\mathrm{ec}}\|_\infty\|\mathcal{L}((1-P)\rho_\beta)\|_1+t\|\mathcal{L}(d\bar{\eta}_{\mathrm{ec}}(1-P)\rho_\beta)\|_1 && (4.38)\\
=&td\|\bar{\eta}_{\mathrm{ec}}\|_\infty\|\mathcal{L}(\frac{(1-P)\rho_\beta}{\|((1-P)\rho_\beta)\|_1})\|_1\|((1-P)\rho_\beta)\|_1 && (4.39)\\
+&td\|\mathcal{L}(\frac{\bar{\eta}_{\mathrm{ec}}(1-P)\rho_\beta}{\|\bar{\eta}_{\mathrm{ec}}(1-P)\rho_\beta\|_1})\|_1\|\bar{\eta}_{\mathrm{ec}}(1-P)\rho_\beta\|_1 && (4.40)\\
\leq&td\|\bar{\eta}_{\mathrm{ec}}\|_\infty\|\mathcal{L}\|_1\|((1-P)\rho_\beta)\|_1+td\|\mathcal{L}\|_1\|\bar{\eta}_{\mathrm{ec}}(1-P)\rho_\beta\|_1 && (4.41)\\
\leq&2td\|\bar{\eta}_{\mathrm{ec}}\|_\infty\|\mathcal{L}\|_1\|((1-P)\rho_\beta)\|_1 && (4.42)\\
\leq&2td\|\mathcal{L}\|_1\|((1-P)\rho_\beta)\|_1 && (4.43)\\
& && (4.44)
\end{aligned}
$$

Line 4.28 follows from the triangle inequality. Line 4.30 follows from line 4.29 because $\phi_s(\rho) = e^{\mathcal{L}}\rho$. Line 4.31 follows by lemma 4.4.2. Line 4.34 follows from the triangle inequality. Line 4.35 follows from line 4.34 because the Lindblad operators commute with error corrected operators on the subspace defined by the projector P. This is because on this subspace, the Lindblad operators do not change the encoded quantum state. Line 4.37 follows from line 4.36 because ρ_β is the equilibrium state of the time evolution operator and hence $\mathcal{L}(\rho_\beta) = 0$. Line 4.38 follows from line 4.37 because of a property of the one norm that $\|AB\|_1 \leq \|A\|_\infty \|B\|_1$. Line 4.41 follows from the definition of $\|\mathcal{L}\|_1$. This is used again to get line 4.42 from line 4.41. Line 4.43 follows from line 4.42 because the operator norm of a density matrix is bounded by one. □

4.5 Energy barrier

My research is mostly concerned with finding codes with large energy barriers. So to be clear, I want to define the energy barrier rigorously at least once. We saw that for the Ising model, it is the minimum energy necessary to flip spins one by one from the all zeros state to the all ones state or vice-versa. For stabilizer code Hamiltonians the energy barrier is the minimum energy necessary to enact a nontrivial logical operator with a sequence of single qubit operations.

The lifetime of a self-correcting quantum memory depends on the full dynamics which includes both the energy barrier and the number of local sequences that lead to a logical operator. However, an energy barrier is a necessary condition for a long memory lifetime. If there is a constant energy barrier, then there is at least one Pauli sequence that can apply a logical operator with no more than a constant energy penalty. This means that thermalizing dynamics would have no bias against performing such a sequence. In contrast, if there is an energy barrier, then thermalizing dynamics would have a bias against enacting this Pauli-

sequence. The sequence necessarily increases energy up to the energy barrier and each such increase is less probable than a decrease in energy. Hence, the existence of an energy barrier already implies some amount of self-correction.

Consider a stabilizer code Hamiltonian with a generating set R. For topological codes in a dimension d, R is chosen to have the maximum number of local generators. For instance, many of the terms are redundant in the 2-d Ising model, a classical topological code. Without these redundant terms, the 2-d Ising model would not be stable against thermalization.

We imagine that we apply a single qubit operation X, Y or Z one at a time in order to enact a logical operator. This results in a Pauli sequence $K_n = (p_1, ..., p_n)$. The operator associated with this sequence is given by $L(K_a) = \prod_{p_i \in K_a} p_i$ where a is the ath step in the sequence.

For each operation in the sequence, the energy changes. We define the energy after the ath Pauli-operator to be $\epsilon(L(K_a)) = \langle\psi|L(K_a)HL(K_a)^\dagger|\psi\rangle$ where $|\psi\rangle$ is in the code space

Definition Finally, we define the energy barrier of a sequence to be $\Delta\epsilon(K_n)$ to be

$$\Delta\epsilon(K_n) = max_{a\in\{1,...,n\}}\epsilon(L(K_a))\,. \tag{4.45}$$

The energy barrier of a code is the minimum energy barrier over all sequences K such that $L(K)$ is a non-trivial logical operator of the code.

The way to imagine the energy barrier is in terms of an energy landscape. In this landscape there are different regions that have zero energy. Each region corresponds to a logical operator. There is an energy ridge between these regions. The lowest energy along this ridge is the energy barrier.

To be more clear, we define the graph G with vertices labeled by the elements of the Pauli group G. I use the same symbol for both the group and the graph and use context to

distinguish the two cases. For each operator h in the Pauli group G there is a vertex also denoted with h. Each vertex is connected to another by an edge if there is a single qubit Pauli operator that we can apply to change one vertex into another. More precisely, $E = \{(h, g) : h, g \in G, \exists b \in \{X_1, Y_1, Z_1, ..., X_n, Y_n, Z_n\}\ s.t.\ bh = g\}$. Each vertex has an energy $\epsilon(h) = \langle\psi|hHh|\psi\rangle$ where $|\psi\rangle$ is any state in the code space This is the energy landscape. The energy barrier is the lowest energy point along the ridge between two inequivalent logical operators.

The shape of the logical operator is likely enough to tell us what the energy barrier is. Suppose that we have a applied a sequence of single spin Pauli operators resulting in the operator E being applied to a state that is in the ground state subspace. If a local neighborhood of spins of the global error E looks like a logical operator, then the terms that are completely contained in that neighborhood are not violated. This is because the terms are local and so if a logical operator incurs no energy penalty in that region, then neither will the error E. We can lower bound the energy barrier by looking at the minimum number of disagreements. We split all of the spins into local neighborhoods, one for each term in the Hamiltonian. The energy barrier for a particular Pauli sequence is lower bounded by the number of such neighborhoods that do not match any logical operator. So for instance, a logical operator that is a membrane has an energy barrier proportional to the width of the membrane. This is because to grow a membrane there will be an energy penalty proportional to the boundary of the membrane.

There are several open questions here. How do we characterize the shape of the logical operators and how do we make the connection between the shape and energy barrier stronger?

4.5.1 The stabilizer graph

This chapter is about the quasi-particle picture of the energy barrier. The quasi-particle picture is a way to visualize different energy eigenstates of a stabilizer code Hamiltonian. [2] We imagine that each term $h \in R$, where R is a generating set, labels a quasi-particle on a vertex v_h. If h has a negative eigenvalue, $h|\psi\rangle = -|\psi\rangle$, then we say that there is a quasi-particle at the vertex v_h or just at h for short. If there is a positive eigenvalue for h, then we say that there is no quasi-particle present. Under this interpretation, code states of the stabilizer code $\langle R \rangle$ have no quasi-particle excitations.

We can express a generating set of a stabilizer group graphically. There are two types of vertices, one for each stabilizer and one for each qubit. The different colors express different Pauli operators: X, Y or Z. There is one stabilizer for each stabilizer vertex that acts on all of the qubits connected to it with the type of edge that connects the two. This is shown in figure 4.2.

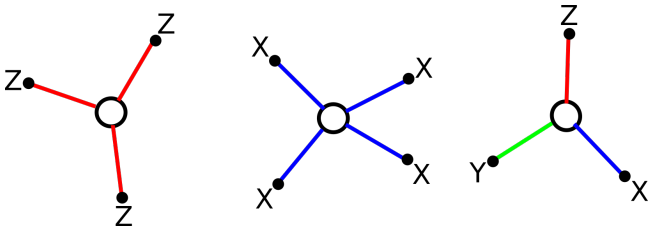

Figure 4.2: Different vertices of a stabilizer graph. A circle represents a stabilizer and a dot represents a qubit. The different edges in the quasi-particle graph represent different single qubit Pauli-operators. A blue edge represents an X operator, a red edge represents a Z operator and a green edge represents a Y operator on the qubit connected to that edge. For each stabilizer vertex, there is a stabilizer corresponding to the product of all the edges that connect to it.

---

[2] Terms in R may be redundant in the sense that we may be able to multiply a subset of them and get identity. That is, it maybe that for $F \subset R$, $\prod_{h \in F} h = I$ for some F. Indeed this is a feature of the 3-d toric code[3], the cubic code and the welded codes. I conjecture that it is necessary for the Hamiltonian to be self-correcting.

If we perform a Pauli-sequence that leads to a logical operator then in general there will be some quasi-particles created along the way. The energy of a state is just the total number of quasi-particles present.

The terms are local so applying an X or Z operator to a qubit may annihilate a local group of quasi-particles and potentially create new quasi-particles at neighboring sites. For instance, a Pauli-sequence of X operators on the toric code would lead to pairs of plaquette quasi-particles(or X-type quasi-particles) or equivalently it would make two quasi-particles move between neighboring sites.

The toric code is often visually displayed as consisting of one type of vertex that labels the stabilizer and an edge that labels the qubit. This picture, however, is too restrictive for general topological codes. This is because sometimes applying a single qubit Pauli-operator may create 3 or more quasi-particles. An edge only connects 2 quasi-particles and so does not suffice to describe the general case.

I prefer to visualize each qubit as one type of vertex and each stabilizer as a second type of vertex for each qubit. This helps me keep track of how quasi-particles move throughout a Pauli sequence. Then I visualize three types of edges. These edges connect stabilizers to qubits. The three types of edges correspond to whether there is a stabilizer that applies a single qubit X, Y or Z operator at that qubit.

The surface code is a topological code that can be displayed with the quasi-particle graph. Because qubits only have two terms per qubit, we keep the vertex between each stabilizer vertex implicit, i.e. we do not draw a dot.

For CSS codes we can imagine two graphs separately; one for the X-type edges and one for the Z-type edges. These are called the dual and primal graphs [51]. I am primarily concerned with CSS codes in this thesis so I use these. See figure 4.3.

Furthermore, we can associate an energy barrier separately for the dual and primal lat-

tice. There is one energy barrier for Pauli-sequences of X operators and another for Pauli-sequences of Z operators.

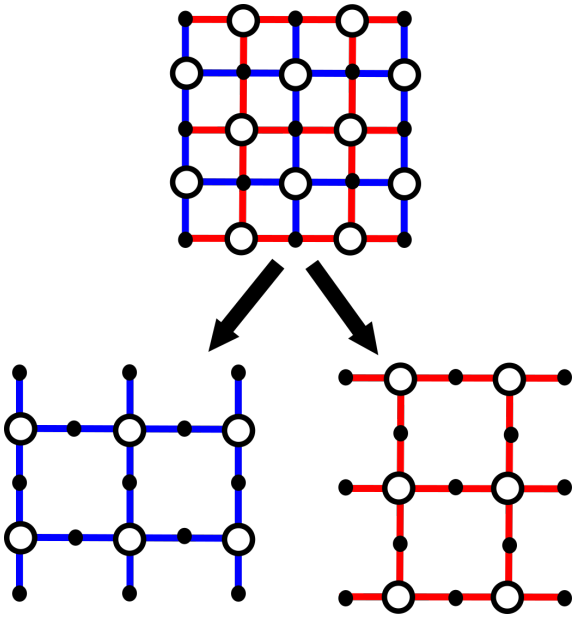

Figure 4.3: I denote a qubit with a dot and a stabilizer with a circle. For instance, the surface code is shown above. The surface code graph split into its dual and primal graph.

Codes with energy barriers must have a quasi-particle graph that have 3 or more edges of a particular type(X or Z) per qubit. We can visualize vertices as points where if a quasi-particle moves through that vertex then it splits into several quasi-particles at neighboring sites. The energy is proportional to the number of quasi-particles and so it is important that the quasi-particles split as they move around so that the energy increases at some point in a Pauli-sequence.

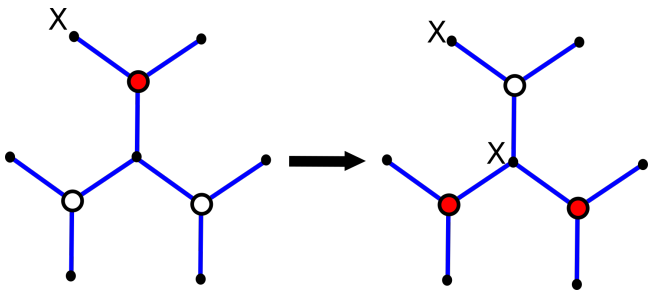

Figure 4.4: A quasi-particle moves and splits into two upon the application of an X operator. A red circle indicates a quasi-particle and an empty circle indicates no quasi-particle.

4.6 The solid code

4.6.1 Motivation

The toric code(and surface code) has a constant energy barrier. We can see this by noting that applying an X(or Z) operator creates a pair of defects, destroys one defect while creating one nearby or destroys a pair of defects. Thus once a pair of quasi-particles are created, they can move around without creating any additional quasi-particles. In particular, they can move around a non-trivial loop, resulting in a non-trivial logical operator.

This constant energy barrier suggests that the toric code Hamiltonian does not serve as a very good self-correcting memory. Indeed there is a theoretical study [5] that shows this for Davies maps.

My central contribution is to have invented a topological stabilizer code Hamiltonian with an energy barrier that scales as a power-law, $\sim L^a$ for a system of qubits of width L. I call this code the welded solid code. At present it is the highest energy barrier in 3-d.

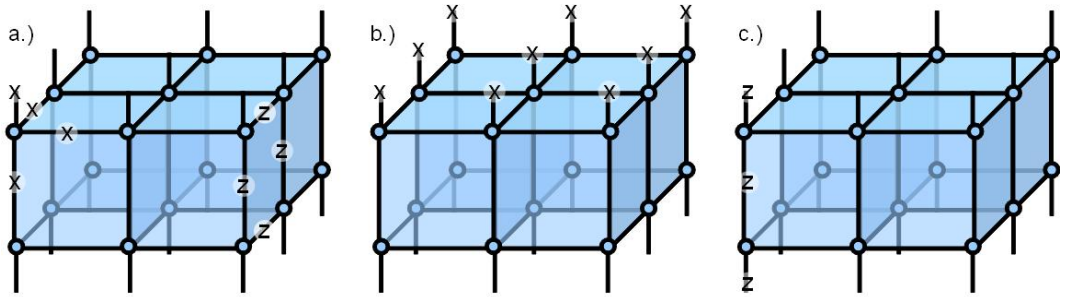

Figure 4.5: The graph of a small solid code with qubits on the edges. a.) An X-star operator and a Z-plaquette operator are shown. b.) The $\bar{X}$-membrane operator is shown. c.) The $\bar{Z}$-string operator is shown.

I now introduce the solid code, a topological code of central importance to the welded solid codes, the code that achieves the high energy barrier. I describe this code in three way, as I did with the toric code. First, I just define the generating set. Next, I describe it in terms of non-local spin operators in analogy to the Ising model. Finally, I describe it in terms of quasi-particles.

I also show that if we restrict to a subset of qubits and treat the rest of the qubits as an external spin bath, then the effective Hamiltonian has an energy barrier for the plaquette operators.

We define the solid code by analogy to the surface code [30], i.e. a 3-d toric code[4] [25,85] with rough and smooth boundaries. These boundary conditions determine the logical operators, one of which has a constant energy barrier. This means that the solid code cannot be used as a self-correcting quantum memory. However, the solid code has an energy barrier for one of its logical operators. It is the first quantum code that we have discussed that has such a logical operator.

[4]A $Z2$ lattice gauge theory [54] originally invented by Wegner [85].

4.6.2 The stabilizer picture

We define the generators of the solid code with respect to a graph, shown in figure 4.5 (qubits are labeled by edges)[5]. The graph is mostly a cubic lattice with $d \times d \times d$ primitive cells except that horizontal edges are missing from the top and bottom boundaries of the graph. We call these boundaries rough. Terms in the Hamiltonian are labeled by vertices of the graph and faces of the primitive cells. We refer to the faces as plaquettes. The vertices are:

$$V = \{\boldsymbol{v} = (v_1, v_2, v_3) : v_i \in \{1, ..., N\}\}\,. \tag{4.46}$$

Using the unit vectors $\boldsymbol{n_1} = (1, 0, 0)$, $\boldsymbol{n_2} = (0, 1, 0)$ and $\boldsymbol{n_3} = (0, 0, 1)$, the edges are:

$$\begin{aligned} E = &\{\{\boldsymbol{v}, \boldsymbol{v} + \boldsymbol{n_3}\} : \boldsymbol{v} \in V, v_3 \neq N\} \\ \cup &\{\{\boldsymbol{v}, \boldsymbol{v} + \boldsymbol{n_2}\} : \boldsymbol{v} \in V, v_3 \neq 1, v_3 \neq N\} \\ \cup &\{\{\boldsymbol{v}, \boldsymbol{v} + \boldsymbol{n_1}\} : \boldsymbol{v} \in V, v_3 \neq 1, v_3 \neq N\}\,. \end{aligned} \tag{4.47}$$

Let $\Gamma(\boldsymbol{v})$ be the set of edges that neighbor a vertex $\boldsymbol{v}$. For each $\boldsymbol{v}$ with $|\Gamma(\boldsymbol{v})| > 1$ define the term $h^X_{\boldsymbol{v}} = \prod_{e \in \Gamma(\boldsymbol{v})} X_e$. Let ∂f be the set of edges on the boundary of a plaquette f. For each plaquette f define a term $h^Z_f = \prod_{e \in \partial f} Z_e$ except that plaquettes at the top and bottom of the lattice are missing an edge. The Hamiltonian is a sum over elements of the set of vertices V and the set of faces F:

$$H = -\sum_{\boldsymbol{v} \in V : |\Gamma(\boldsymbol{v})| > 1} h^X_{\boldsymbol{v}} \quad + \quad -\sum_{f \in F} h^Z_f\,. \tag{4.48}$$

The X and Z terms commute and so generate a stabilizer group. The ground state subspace of the Hamiltonian is exactly the code space of this stabilizer group.

---

[5] Additionally we imagine a qubit vertex intersecting each edge

The logical operator $\bar{X}$ of the solid code resembles an open membrane. Multiplying vertex operators generates closed membranes of qubits and horizontal pairs of open membranes. These pairs can be made to be far apart so that each membrane overlaps with a disjoint set of plaquettes. Hence each membrane commutes with each term in the Hamiltonian, yet is not generated by them. We conclude that the logical operator $\bar{X}$ is a tensor product of single qubit X operators on a single horizontal membrane.

The logical operator $\bar{Z}$ resembles an open string. Multiplying plaquette operators generates closed strings, strings starting and ending on the same rough boundary and pairs of strings extending between opposite rough boundaries. These pairs of open strings can be made to be far apart so that each string overlaps with a disjoint set of vertex operators. Hence each open string commutes with each term of the Hamiltonian, yet is not generated by them. We conclude that the logical operator $\bar{Z}$ is a tensor product of single qubit Z operators on a single string extending between opposite rough boundaries. See figure 4.5.

The set of logical operators completely determines the ground state degeneracy. There are two distinct non-trivial logical operator, $\bar{X}$ and $\bar{Z}$. Because they anti-commute, we can only diagonalize one of them at a time. Hence the ground state degeneracy is 2.

The logical operator $\bar{Z}$ has a constant energy barrier. Understanding why is key to doing better. Applying a Z operator on a single qubit violates either one or two terms. We call these defects. By flipping a qubits in a line starting and ending on opposite rough boundaries, we move a single defect across the lattice with no more than a constant energy penalty. One way to create a large energy barrier is to force this string of bit flips to split many times. To do this we need qubits such that errors on them create three or more defects. This is the case for the logical $\bar{X}$ operator. If we apply a single qubit X operator, we create 4 quasi-particles.

4.6.3 Non-local spin-operators picture

The logical X operators for the solid code are of membranes that extend horizontally and the logical Z operators are strings that extend vertically. The terms in the Hamiltonian are the local deformations of these operators. Thus the X-stabilizers are mostly closed surfaces and the Z-stabilizers are mostly closed loops. There is an exception at the boundary of the lattice of qubits where we get half closed surfaces and half loops.

The Hamiltonian takes the form

$$H = -\sum_{(A,B)\in E_Z} Z_A Z_B - \sum_{(I,J)\in E_X} X_I X_J \tag{4.49}$$

where E_Z(or E_X) is the set of all pairs of logical Z(or X) operators that are related by a local deformation. i.e. a Pauli operator that acts on a set of qubits that are within a constant distance of each other. When all of the terms are minimized, we are in the code space and all of the different non-local fields agree. It is striking to note how similar this Hamiltonian is to the Ising model.

4.6.4 The quasi-particle picture

Although the solid code has a constant energy barrier, one of its logical operators has an energy barrier that scales as $O(L)$ where L is the width of the lattice of qubits.

Applying a single qubit X-operator either creates new loops or else changes the shape of a loop of defects. These loops are the boundaries of 2-dimensional domains of X-operators. In order to achieve a logical operator, these domains have to be grown until this loop reaches the boundaries of the lattice. In order to do this with a Pauli-sequence of X operators we will need to create at least $O(L)$ quasi-particles. Hence the energy barrier is $O(L)$ for this logical operator. The energy landscape is much like the 2-d Ising model. We grow membranes of

X-type operators. These are like the domains in the Ising model where the logical operator is again a product of X operators. However with the solid code, the membrane can bend and fold in three dimensions.

We can prove a lower bound on the energy barrier of the membrane operator by noting that if we remove the horizontal plaquettes, then the remaining set of stabilizers still generate the horizontal plaquettes. Now quasi-particles can move up and down without creating new quasi-particles. We now focus on the energy landscape of these vertical regions. It is exactly the same as for the 2-d Ising model. The graph of this landscape is composed of a vertex for each vertical strip of qubits and an edge between vertices if a single X operator creates a quasi-particle in each region. Hence in order to create a membrane of X operators, the energy must be at least $O(L)$ at some point, as with the 2-d Ising model. Putting the horizontal plaquettes back in the Hamiltonian can only increase the energy barrier.

However, the logical Z operator has only a constant energy barrier. Like the toric code, Z-type quasi-particles are free to move around without creating new quasi-particles. We can create a single Z-type quasi-particle at the bottom boundary of qubits and move it to the top boundary without creating any additional quasi-particles. Hence the energy barrier is a constant.

Chapter 5

WELDED CODES

5.1 Introduction

I first came up with the idea of welding by thinking about how to force two topological codes to share a subset of qubits. The rationale was that an error on these shared qubits would generate quasi-particles in each of the codes. This would then increase the energy barrier whenever a logical operator has support on one of the shared qubits. This led me to the idea of welding solid codes together. This code has the highest energy barrier to date.

It may be helpful to visualize the welded codes in terms of the shape of their logical operators, as with the non-local spin picture, not in terms for the stabilizer group. This shape fully determines the energy barrier of the code without having to look at the details of the generating set.

I still present the welded solid code in terms of the theory of welding for two reasons. First, it is historically the concept that lead to the welded solid code and second, it is interesting in its own right.

The idea of welding is a simple one as I show with the following example. Define two codes with with generating sets R_1 and R_2:

$$R_1 = \begin{matrix} \langle Z & Z & I & I & I, \\ I & Z & Z & I & I, \\ X & X & X & I & I \rangle \end{matrix}$$

$$R_2 = \begin{matrix} \langle I & I & Z & Z & I, \\ I & I & I & Z & Z, \\ I & I & X & X & X\rangle \end{matrix} .$$

Suppose now that we combine the two codes into a single code by taking the union of the generating sets. The generators do not commute as the X type stabilizers of the second code do not commute with the Z type stabilizers of the first code and vice-versa. However, if we combine the X type operators, $XXXII$ and $IIXXX$, from both codes into the operator $XXXXX$, then the set commutes and generates a stabilizer code. Let R be the welded code.

$$R = \begin{matrix} \langle Z & Z & I & I & I, \\ I & Z & Z & I & I, \\ I & I & Z & Z & I, \\ I & I & I & Z & Z, \\ X & X & X & X & X\rangle \end{matrix} .$$

This is the essential idea of welding. We force two codes to share a subset of qubits, keep one set of mutually commuting stabilizers from each group and then combine the rest of the stabilizers until the final group commutes.

5.2 Welding

I would like to define welding precisely, but first we have to develop some notation.

Definition Let $Q(S)$ be the qubits that the stabilizer group S has support on for at least one operator $h \in S$. For instance, if $S = \langle ZZII, IZIZ\rangle$ then $Q(S) = \{1, 2, 4\}$ since S only has support on the first second and fourth qubits.

Next we'll define the restriction onto a set of qubits.

Definition Let $\theta_Q : G_n \to G_n$ be the restriction of a Pauli operator onto qubits Q. For instance if $S = \langle ZZII, IZIZ \rangle$ and $h = XXXX$ then $\theta_{Q(S)}(h) = XXIX$.

We'll be primarily concerned with the restriction onto qubits of two codes S_1 and S_2. We will use the shorthand $\theta_i = \theta_{Q(S_i)}$ and $W = \theta_{Q(S_1) \cap Q(S_2)}$.

Definition Let S_i^X and S_i^Z be the set for the X and Z-type stabilizers of a CSS code S_i. If we do an X-type weld of the stabilizer groups S_1 and S_2 then the resulting stabilizer group is generated by the following sets

$$S^Z = S_1^Z \cup S_2^Z \tag{5.1}$$

$$S^X = \{hgW(h) : \forall h \in S_1^X \; and \; \forall g \in S_2^X \; s.t. \; W(h) = W(g)\}. \tag{5.2}$$

Of course, dealing with the entire stabilizer group can be inconvenient given that there are 2^n stabilizers in the group and a generating set only has n operators for a minimal generating set[1] with n elements. However, there are conditions under which two generating sets can be welded together. These conditions are called well matched and linear independent on the weld.

Well Matched: When generating sets R_1 and R_2 of S_1 and S_2, respectively, are in standard CSS form[2] and match up on the weld, this is called well matched. More specifically, R_1 and R_2 are well matched if for every operator $h \in R_1^X$ such that $W(h) \neq I$ there exists $g \in R_2^X$ such that $W(h) = W(g)$. Similarly for every $h \in R_2^X$ such that $W(h) \neq I$ there exists $g \in R_1^X$ such that $W(h) = W(g)$.

---

[1] A minimal generating set is one for which removing any element of the set results in a set that does not generate all of the stabilizer group.

[2] A generating set is in standard CSS form if it only has X and Z-type operators and no mixture of both.

Linearly independence on the weld: R_k is linearly independent on the weld if R_k is in standard CSS form and for any independent subset of X-type stabilizers $A = \{h\} \subset R_k^X$, if $W(h) \neq I \ \forall h \in A$ then $W(\prod_{h \in A} h) \neq I$.

Linear independence on the weld basically means that you can't multiply elements that act non-trivially on the weld in order to get an element that acts trivially on the weld.

We now state a theorem about when we can weld generating sets of two stabilizer groups of the CSS type.

Theorem 5.2.1 *Let R_i^Z (or R_i^Z) be a generating set of the X-type(or Z-type) stabilizer of the stabilizer group S_i such that R_1^X and R_2^X are well matched and linearly independent on the weld. The following generating sets generate all the X and Z type stabilizers for the welded code between groups S_1 and S_2.*

$$R^Z = R_1^Z \cup R_2^Z \tag{5.3}$$

$$R^X = \{hgW(h) : h \in R_1^X, g \in R_2^X, W(h) = W(g)\}. \tag{5.4}$$

Proof Trivially, all Z-type operators are generated. If g is a X-type operator of the welded code S then we can find an operator h, generated by R such that $\theta_1(hg) = I$ because of the well matched condition. $hg \in S_2$ and $W(hg) = I$ and by linear independence on the weld, hg must be generated by elements of R_2 that are identity on the weld, which are also in R. □

One nice feature of welding is that the logical operators weld in the same way that the generators do.

Theorem 5.2.2 *If we perform an X-type weld on two generating sets and $\bar{X}_1$ and $\bar{X}_2$ are two generating sets of each code respectively such that $W(\bar{X}_1) = W(\bar{X}_2)$ then $\bar{X} = \bar{X}_1\bar{X}_2W(\bar{X}_1)$ is a logical operator of the welded code.*

Logical operators such as in theorem 5.2.2 may not account for all of the logical operators of the welded code. You can also have a logical operator weld onto a stabilizer, so long as they match up on the overlapping qubits.

You can also have a situation where welding two codes introduces a new type of logical operator which was not present before. For instance, if we weld two surface codes into a tube, call it the tube code, and then we weld two tubes into a torus, then welding logical operators of the two tubes only accounts for one encoded qubit even though the toric code has two encoded qubits. This new logical operator is a string of Z operators that wraps around the tube. This string was generated by Z stabilizers on the ends of the tube code. After welding, these stabilizers get welded together so that they can only generate pairs of loops, no longer individual loops. Now a single loop of Zs is no longer contractible.

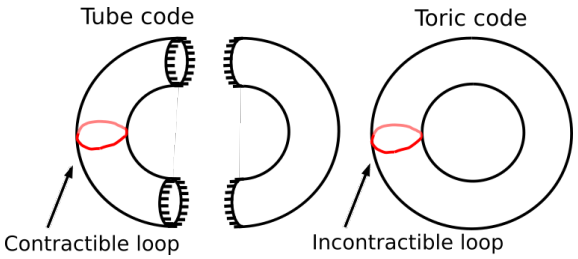

Figure 5.1: Welding two tube codes into a torus.

We can ensure that welding does not produce new logical operators by ensuring that our generating sets are well matched and linearly independent on the weld.

Theorem 5.2.3 *Let R_1^X and R_2^X be X type generating sets of the CSS codes S_1 and S_2, respectively, such that they are well-matched and linearly independent on the weld. Let $\bar{X}$ be a logical X operator of the stabilizer group S, where S is the welded code of S_1 and S_2. $\theta_1(\bar{X})$ and $\theta_2(\bar{X})$ cannot both be trivial logical operators of S_1 and S_2.*

Proof The proof is by contradiction. Suppose $\theta_1(\bar{X})$ is a trivial logical operator. Then we can find $g \in S^X$ such that g is generated by generators of R^X with support only on qubits Q_1. The operator $g\bar{X}$ is completely supported by qubits $Q_2 \backslash Q_1$. The operator $g\bar{X}$ is still a trivial logical operator of S_2 since

$$\theta_2(g\bar{X}) = \theta_2(g)\theta_2(\bar{X}) = \theta_2(\bar{X}) = g\bar{X}\,. \tag{5.5}$$

By linear independence on the weld $g\bar{X}$ is generated by R_2^X with no support on qubits $Q_1 \cap Q_2$. These generators are also in R^X and hence we conclude that $g\bar{X}$ is a trivial operator of S in contradiction to our assumption that $\bar{X}$ was nontrivial. $\square$

There is a corollary of theorem 5.2.3 that all logical X operators can be found by welding logical operators.

Corollary 5.2.4 *We can find all X-type logical operators by considering all welded X-type logical operators. That is, the following set generates all X-type logical operators*

$$\{\bar{X}_1\bar{X}_2W(\bar{X}_1) : \bar{X}_1 \in \mathcal{L}_1^X, \bar{X}_2 \in \mathcal{L}_2^X \ s.t.\ W(\bar{X}_1) = W(\bar{X}_2)\} \tag{5.6}$$

where $\mathcal{L}_i^X$ denotes the set of nontrivial logical X operators of the stabilizer group S_i.

Hence, if we know the logical operators of the codes S_1 and S_2, then we know the logical operators of the welded codes; they are just welded operators.

Remark 1 *Suppose R is a generating set which is a product of doing an X weld between standard CSS form, well matched, linearly independent on the weld generating sets R_1 and R_2. If $h_i \in R_i^X$ and t is a Z-type operator that anti-commutes with h_i and commutes with all other operators in R_i then t will only anti-commute with a welded operators of the form $h \in R^X$ where $\theta_i(h) = h_i$.*

Remark 1 says that if an operator anti-commutes with a stabilizer generator before welding, it will anti-commute with it after it is welded. This is important because after we have a generating set for the welded code we will choose an operator from the generating set to promote to a logical operator and this remark tells us how to quickly identify anti-commuting logical operators.

Like concatenation, (see [36] for a discussion) welding can be used to combine two codes, S_1 and S_2, to produce a third code, S. Whereas concatenation produces a code that acts on $N = n_1 n_2$ qubits when S_1 and S_2 act on n_1 and n_2 qubits respectively, welding produces a code that acts on $N < n_1 + n_2$ qubits.

Not only that, but welding also preserves the locality of the generating set, something that concatenation does not do. For instance, if two generating sets are well-matched and linearly independent on the weld and arranged in 3-d such that they are local and not too dense, then the weight of the welded generating set is no more than double the weight of the highest weight generator in each code. Furthermore, if the generators match up without having to bend the lattice of qubits too much, then the generators are local as well. We consider this type of welding in the next few sections.

5.2.1 Welding is a universal language for stabilizer codes

Every stabilizer code of the CSS type can be expressed as a welded code. Let R^Z and R^X be generating sets for the Z and X-stabilizers of a stabilizer code respectively. Let R_1^Z and R_2^Z be subsets such that $R_1^Z \cup R_2^Z = R$. Define the stabilizer groups $S_1 = \langle \theta_1(R^X) \cup R_1^Z \rangle$ and $S_2 = \langle \theta_2(R^X) \cup R_2^Z \rangle$. In fact we can weld S_1 and S_2 together to get S again. Thus every CSS code is a welded code.

We can use this argument to generalize welding to non-CSS codes so that we can say that every stabilizer code is a welded code. Let R be a minimum generating set of a stabilizer

group. Pick two subsets $R_1 \subset R$ and $R_2 \subset R$. Now $R_1 \cup \theta_1(R \backslash R_1)$ and $R_2 \cup \theta_2(R \backslash R_2)$ are both stabilizer groups and we can view the stabilizer group $S = \langle R \rangle$ as the welded version of the groups $\langle R_1 \rangle$ and $\langle R_2 \rangle$. In this sense, welding is a universal language for describing stabilizer codes. In particular for topological stabilizer codes, it gives us a way of representing the code as very many overlapping local patches of code welded together in a particular order.

5.3 Surface Codes and Welding

5.3.1 Surface Codes

To illustrate the process of welding, we generate the surface code [30] by welding only groups of the form $G = \langle XX, ZZ \rangle$ together. We have the two groups $S_1 = \langle XX, ZZ \rangle$ and $S_2 = \langle XX, ZZ \rangle$ and we identify qubit two of S_1 with qubit one of S_2 and we do a Z-type weld; we include all X-type operators and update the Z-type operators to commute with the X-type operators. After identifying the left and right qubits we have $S_1 = \langle XXI, ZZI \rangle$ and $S_2 = \langle IXX, IZZ \rangle$. We get the group $S_1 \boxplus S_2 = \langle XXI, IXX, ZZZ \rangle$ where $ZZZ = (ZZI)(IZZ)W(ZZI)$. This is shown diagrammatically in figure 5.2.

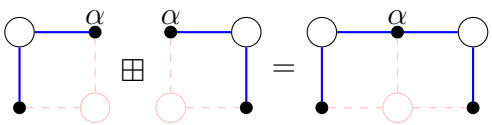

Figure 5.2: Welding two, two qubit repetition codes together. Black dots are qubits and circles are stabilizers. Each stabilizer vertex acts on adjacent qubit vertices with a Pauli operator corresponding to the type of edge that connects the two. A solid edge indicates a Pauli-X operator being applied and a dashed edge indicates a Pauli-Z operator being applied. α indicates which qubits are being identified.

Continuing on diagrammatically, we weld two such codes together, via an X-type weld, as in figure 5.3.

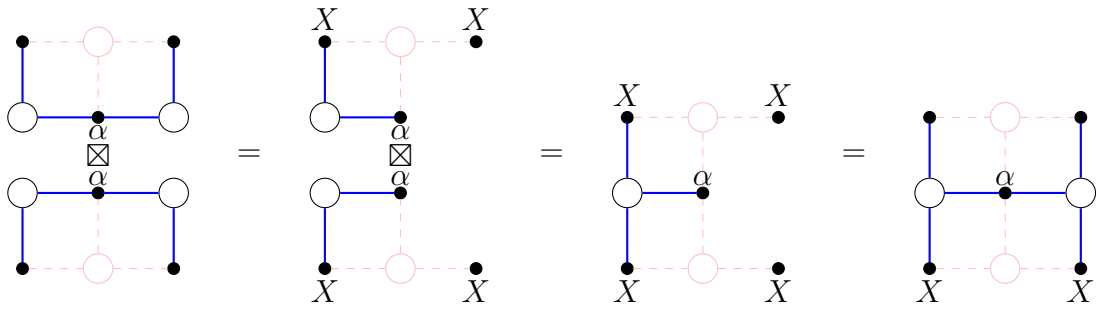

Figure 5.3: α identifies the two qubits to be welded. Requiring linear independence on the weld invokes a horizontal string of X operators to be in the stabilizer group after the weld. After the second step we have specified two horizontal X-type string operators by placing an X on the qubit vertex of the stabilizer. The reason for doing this is so that it is in the same form as an arbitrary surface code which has much longer string-like operators that can't conveniently be written as a stabilizer vertex.

After welding we have a 5 qubit surface code. But the string-like operator, the horizontal XX operator, is not yet three qubits long and so would not have protection from Z errors if it were promoted to a logical operator. In the next diagram we continue welding these tiny surface codes together, extending the length of the string-like operators. We find that welding two 5 qubit surface codes creates a 7 qubit surface code.

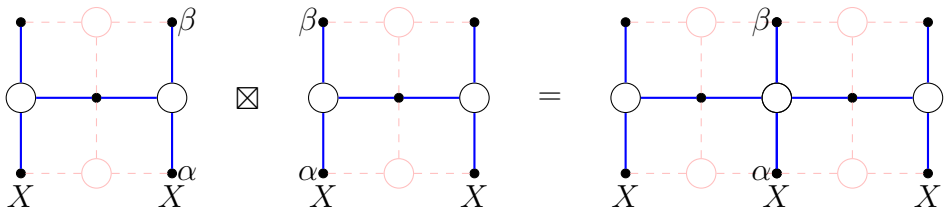

Figure 5.4: Welding two surface codes together on their smooth edge. The strings of Xs become welded together and promoted to a logical operator.

Using remark 1 we see that a vertical ZZ string anti-commutes with the horizontal string of three Xs in figure 5.4. Switching these in the stabilizer group generating set, represented

by the graph, we can do a Z-type weld between two strips of surface code to arrive at a 13 qubit surface code that has string logical operators that protect against an arbitrary error on a single qubit as in figure 5.5.

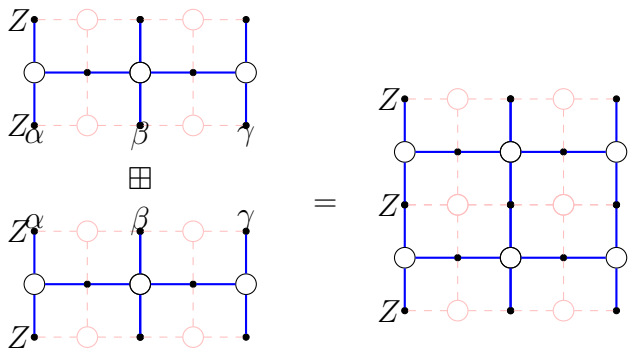

Figure 5.5: Welding two surface codes together on their rough edge also welds the logical Z operators from each surface code together, creating a longer string of Zs.

By first welding a thin strip of surface code and then welding those strips together, we can generate a surface code of an arbitrary size.

In order for logical operators to commute with the welded stabilizer code, they must get welded together as well. They behave the same way as generators. If we would like, we could add them to the generating set, weld and then promote them to logical operators.

It should be noted that the top and bottom edges of the surface code in figure 5.5 are called rough edges and that the left and right edges are called smooth edges. Rough edges are where a single X-type quasi-particles can be created from a single Z error and smooth edges are where a single Z-type quasi-particle can be created from a single X error.

5.3.2 Welded Surface Codes

In this section I specialize to welding boundaries of surface codes. Specifically, smooth edges to smooth edges with X-type welds and rough edges to rough edges with Z-type welds. We

show examples of how to create a high energy barrier and then show how regions where quasi-particles move around freely are used to put a lower bound on the energy barrier.

In the previous section we showed how to weld two surface codes to produce a larger surface code, however, the larger surface code still has string logical operators. Let's consider what happens if instead we welded three surface codes together along a rough edge with a Z-type weld as in figure 5.6. More precisely, we first weld the first two surface codes together and then weld the third surface code to the product. It will be useful to describe the rough and smooth boundaries of a given surface code after it has been welded.

Definition A rough weld describes the set of qubits along the rough edge of a surface code after it has been welded on that rough edge via a Z-type weld. A smooth weld is defined similarly where it describes the set of identified qubits of an X-type weld on a smooth edge of a surface code.

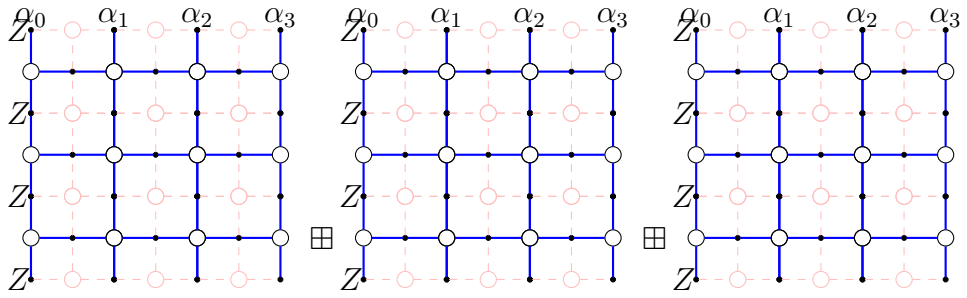

Figure 5.6: Three surface codes welded together by their rough edge via a Z-type weld. α_i label the identified qubits that will be welded together.

Before the weld we include, into the stabilizer group, the logical operator composed of a string of Z operators extending between opposite rough edges. We are then welding three codes with zero logical qubits together to produce a code with zero logical qubits. One of the generators of the welded code will be the three strings of Z operators joined on the line

of αs, which we'll call $\bar{Z}$. If we promote $\bar{Z}$ to a logical operator, we will then have a code that encodes a single qubit and $\bar{Z}$ has an energy barrier of 2.

The three logical Z operators get welded together on the rough boundary. It resembles three strings that are attached at a point.

Notice that on a particular surface, if there is an odd number of Z errors on the rough boundaries, i.e. rough welds and rough edges, of that surface, then there will be at least one quasi-particle in that surface. Also notice that $\bar{Z}$ has only one Pauli-Z operator on each rough boundary of each surface and that stabilizers of our welded code act with even numbers of Pauli Zs on the rough boundaries so that any representation of $\bar{Z}$ will have an odd number of Pauli Zs on their rough boundaries. For the example in figure 5.6, at least one Pauli Z has to be applied to the rough weld, hence there is no way to produce $\bar{Z}$, via a Pauli-Z sequence, without creating at least two quasi-particles. This bound can be saturated by considering an creating an X-type quasi-particle at a rough edge and traveling through the rough weld where it splits into two quasi-particles. Both quasi-particles then annihilate at their respective rough edges.

What about Z-type quasi-particles? Applying an X error on one of the smooth edges creates a single Z-type quasi-particle but everywhere else Z-type quasi-particles get created in pairs. This is an important observation because it means that Z-type quasi-particles can pass through Z-type welds on rough edges without creating new quasi-particles.

Similarly we can do X-type welds on smooth edges of surface codes as in figure 5.7 in order to increase the energy barrier for the $\bar{Z}$ operator of the encoded qubit. Actually, we could first weld three surface codes via a rough edge as in figure 5.6, and then weld three such codes together via an X-type weld on a common smooth boundary in order to create a code that has an energy barrier of 2 for both $\bar{X}$ and $\bar{Z}$.

This simple example illustrates the principle behind how to generate energy barriers using

welding. We make boundaries where errors create three or more quasi-particles and then those quasi-particles must travel through some region to annihilate at a different boundary.

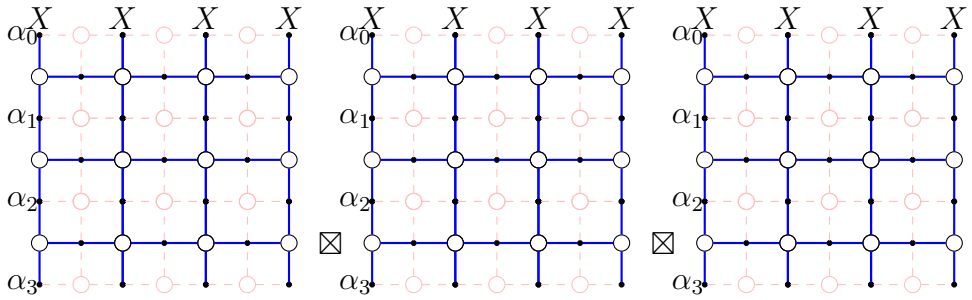

Figure 5.7: Three surface codes welded together on a smooth edge with an X-type weld.

Some key facts to remember about Z-type welds on rough edges and X-type welds on smooth edges are as follows:

- An X-type quasi-particle splits in going past a rough weld.
- Z-type quasi-particles pass through rough welds without splitting.
- A Z-type quasi-particle splits in going past a smooth weld.
- X-type quasi-particles pass through smooth welds without splitting.

These four rules allow us to find regions where quasi-particles can move around without creating new quasi-particles or annihilating.

Definition Regions of qubits where X and Z-type quasi-particles can move around without creating new quasi-particles will be called flat-X regions and flat-Z regions respectively.

If we only did Z-type welds between rough edges of surface codes then a flat-X region of our code, call it A, will have rough boundaries, i.e. a set of rough welds, for which a Z

error on a rough weld will create a single X-type quasi-particle in region A. If there are an odd number of Z errors on all the rough boundaries of A then there will be at least one X-type quasi-particle in A for otherwise an even number of quasi-particles could annihilate. In general we can split our code into several flat-X regions, call them A_i, that will be connected via rough welds where a Pauli-Z error on a rough weld creates a single excitation in each adjacent region A_i. If a particular representation of a logical Z operator has an odd number of single qubit Z operators on its rough welds, then throughout a Pauli sequence generating $\bar{Z}$, for each moment when A_i has an odd number of Z errors on it's rough welds there will be at least one quasi-particle in A_i. We can encode this information via a graph where each flat-X region is a vertex that connects to a "rough" vertex via an edge. A similar graph can be formed for flat-Z regions and smooth boundaries.

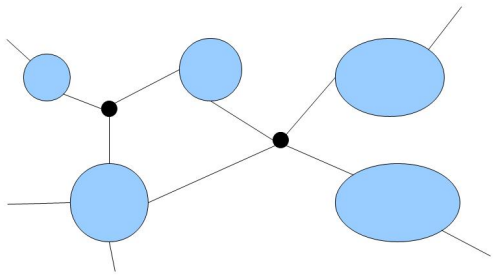

Figure 5.8: An abstract diagram of flat-X regions, represented as large nodes, connected by rough boundaries, represented as dots in black. A Z error on a rough boundary creates an X-type quasi-particle in each connected flat-X region.

Lemma 5.3.1 (Parity lower bound) *For a code that is the result of welding surface codes together via X-type welds on smooth boundaries and Z-type welds on rough boundaries, let $\{A_i\}$ be the set of flat X regions of the code and p_i be the parity of the number of Z errors on the rough boundaries of A_i. For a Pauli-Z walk[3], the total number of quasi-particles*

[3]A sequence of single qubit Pauli-Z operations.

$|F_k| \geq \sum_i p_i$ where $|F_k|$ is the total number of X-type quasi-particles at step k of the Pauli-Z walk. A similar statement holds for Z-type quasi-particles and flat-Z regions.

Notice that the sum is not taken modulo 2, that is, if three regions have parity 1 then the sum will be at least 3. The parity lower bound lemma tells us how to put a lower bound on the energy barrier of welded codes and in fact, this bound is saturated if pairs of quasi-particles in the same regions annihilate as soon as they are created in those regions.

Proof of the parity lower bound: An X-type quasi-particle can travel through X-type welds on smooth edges without creating new quasi-particles and so if an even number of X-type quasi-particles exists within a given flat X region, they can annihilate whereas if an odd number of errors occur on the rough boundary of a flat X region, an odd number of X-type quasi-particles must exist within that flat X region with a minimum of one Z-type quasi-particle existing. Summing over all such regions we find that the total number of quasi-particles satisfies $|F_k| \geq \sum_i p_i$ where p_i is the parity of Z errors on the boundaries the ith flat-X region. The proof of the parity lower bound for flat-Z regions follows similarly. □

5.4 Solid Codes and Welding

5.4.1 Solid Codes

I defined the solid code in section 4.6. However, I discuss it again using welding to construct it and use the parity lower bound to lower bound the energy barriers of its logical operators. I do this because I construct the welded solid code similarly. To remind you, we define the graph of a solid code by a cubic lattice. Next we remove all horizontal edges at the top and bottom boundary of the cube. Qubits live on edges. The vertices of the cubic lattice are X-type stabilizers where for each vertex the stabilizer is the product of Pauli-X operators for each edge connected to that vertex and identity elsewhere. We do not include the vertices at

the top and bottom rough boundaries as stabilizers. For each plaquette of this graph include a Z-type stabilizer that acts on the qubits of that face. On the roughened boundary we have half plaquettes, like for the surface code, where the Z-type stabilizer is the product of three Pauli-Z operators. A small version of a solid code is shown in figure 4.5.

Theorem 5.4.1 *There is one encoded qubit for the solid code with logical operators that are a membrane of Pauli-X operators and a string of Pauli-Z operators.*

Proof We'll prove this by constructing the solid code by welding surface codes together and then promoting a single stabilizer to a logical operator to arrive at the solid code. To start, consider a tall thin surface code as in figure 5.9 with a horizontal-XX operator in the generating set. This strip can be as tall as we wish. Label the left and right smooth edge with i and j and label the surface code with (i, j). Now do an X-type weld on these smooth edges such that the smooth edges and surface codes form a graph $G = (V, E)$ of a square grid where $V = \{i\}$ and $E = \{(i, j)\}$. The tall thin surface code with a horizontal-XX string operator encodes zero qubits because the tall thin surface codes are well matched and linearly independent on the weld so that welding them together in a square grid also produces a code with zero qubits. The horizontal-XX operators get welded together to form a membrane of Pauli-X operators, which we call $\bar{X}$. Promoting $\bar{X}$ to a logical operator gives a code with a single encoded qubit.

The logical operator that anti-commutes with $\bar{X}$, that is $\bar{Z}$, is the string of Zs going between the top and bottom of each surface code. We can see this as a consequence of remark 1.

The given code does not have horizontal plaquette operators but notice that the horizontal Z-plaquette operators are in the stabilizer group because we can multiply four half plaquettes at a rough boundary to get a plaquette operator. We can then multiply this horizontal

plaquette by four adjacent vertical plaquettes to move this horizontal plaquette up or down. □

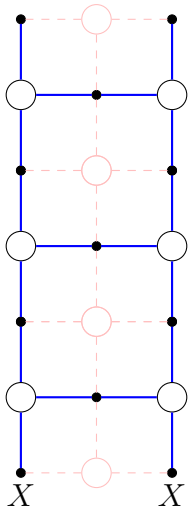

Figure 5.9: An example of a tall thin surface code.

Next we will find the energy barrier of $\bar{X}$ and $\bar{Z}$. First notice that we can produce $\bar{Z}$ via a Pauli sequence with a constant energy barrier because X-type quasi-particles can move without creating new quasi-particles when away from the rough boundaries, that is, $\bar{Z}$ is a string operator.

In order to put a lower bound on the energy barrier of the $\bar{X}$-membrane operator we'll remove the horizontal Z-plaquette operators from the generating set as in the construction via welding. Each tall thin surface $(i, j) \in E$ then becomes a flat-Z region where Z-type quasi-particles can move around without creating new Z-type quasi-particles. We can use the parity lower bound lemma to lower bound the energy barrier. The flat-Z regions and smooth boundaries are shown graphically in figure 5.10. The lower bound is given by the 2-d Ising model which means that:

Lemma 5.4.2 *For a solid code that is $O(d)$ qubits wide, the energy barrier for $\bar{X}$ is $O(d)$.*

Proof We use the parity lower bound on the flat-X regions of the solid code with horizontal plaquettes removed. The flat-X regions are in the shape of a 2-d Ising model. This gives an energy barrier of $O(d)$. This bound can be saturated if pairs of Z type quasi-particles are annihilated as soon as they are created in each flat-Z region. Adding horizontal plaquettes, as with the 3-d toric code[4], could only increase the energy and even then, keeping the membrane operator completely horizontal, the bound $O(d)$ can be saturated. □

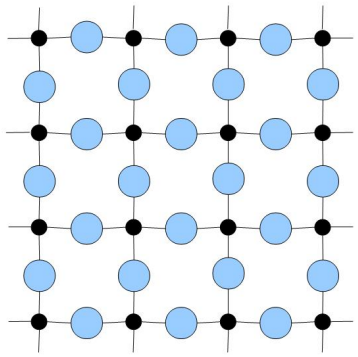

Figure 5.10: The blue regions are flat-Z regions and black vertices are smooth edges. The energy barrier is bounded below by the energy barrier of the Ising model corresponding to the same graph as this figure with the reinterpretation that the black dots are qubits and the blue circles are ZZ operators in the Hamiltonian $H = \sum_{(i,j)\in E} -Z_i Z_j$.

5.4.2 Welded Solid Code

We can reach an energy barrier that is a power of the total number of physical qubits by welding solid codes together. I call this the welded solid code.

Theorem 5.4.3 *There exists a local stabilizer code Hamiltonian in three dimensions with qubits that fit inside of a box with side lengths L with a bounded density of qubits with respect to a box of minimum length l, that has an energy barrier of $O(L^{\frac{2}{3}})$.*

[4]This same proof technique, via flat regions, applies equally to the 3-d toric code [25, 85] to provide a lower bound for the energy barrier of $O(d)$ for a 3-d toric code with $O(d^3)$ qubits.

As a first example we'll weld three solid codes together by their rough boundaries, discuss the generalization of this and then analyze the energy barrier of the resulting code.

As a simple example consider welding the three solid codes in figure 5.11 by their rough boundaries with a Z-type weld. After welding, the $\bar{Z}$ operator becomes three strings of Zs welded together at the rough boundary. $\bar{Z}$ will have a minimum energy barrier of 2 for the same reason that three surface codes welded on a rough edge have an energy barrier of 2. An X-type quasi-particle created at a rough boundary of one of the solids must annihilate on a different rough boundary and so must pass through the rough weld and create an additional X-type quasi-particle.

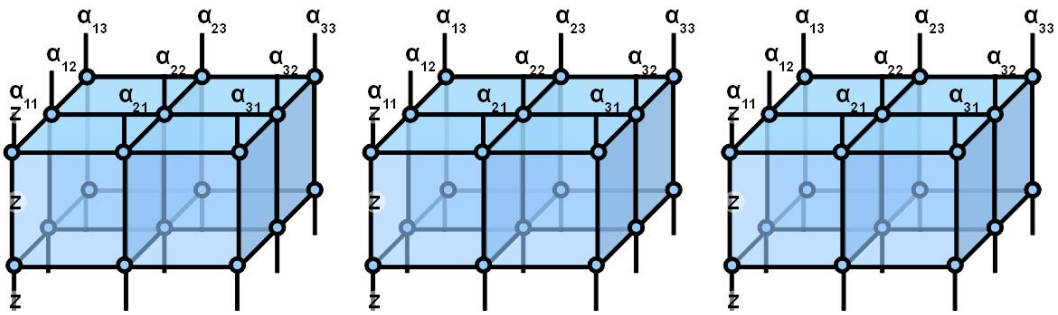

Figure 5.11: Three small solid codes with a Z-type weld between their upper rough boundary. The α_{ij} symbols label identical qubits.

Notice that the $\bar{X}$ operator does not change after welding as can be seen from remark 1. It is still a membrane in one of the solids. The graph of flat-Z regions does not change after the welding because Z-type quasi-particles can propagate through Z-type welds on rough boundaries without creating new Z-type quasi-particles. Hence the energy barrier for the $\bar{X}$ operator stays the same before and after the weld.

In analogy to welding surface codes together on smooth edges to make a solid code with an increased energy barrier for the $\bar{X}$ operator, we can weld solid codes together on rough boundaries to increase the energy barrier of the $\bar{Z}$ operator. To start we label the rough

boundaries of a solid code by i and j and then label the solid by its pair of boundaries (i, j). We can then weld the solid (i, j) with the solid (n, k) by identifying qubits of the boundary j with qubits of the boundary n and then doing a Z-type weld. We are left with solids (i, j) and (j, k) welded on the boundary j. In this way we can weld solid codes to form any graph $G = (V, E)$. Figure 5.12 shows a 2-d square graph of solids welded together. A 3-d cubic graph of solids gives a better energy barrier but is harder to show graphically.

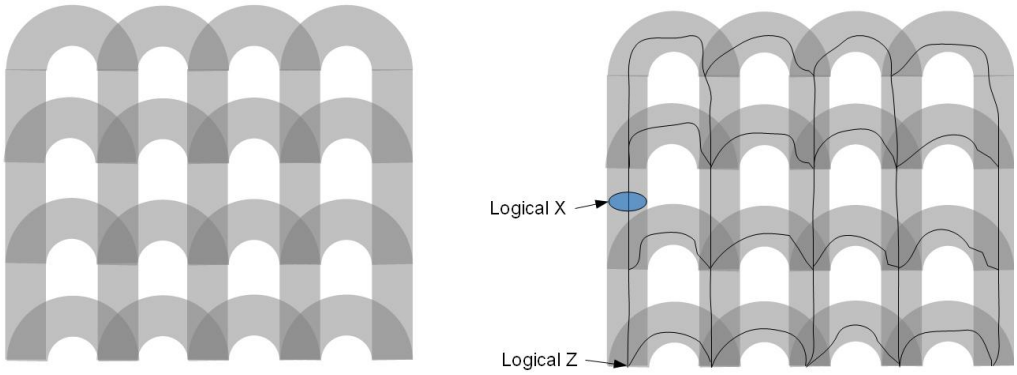

Figure 5.12: Welded solid code with solids welded together in a 2d square lattice. Notice that the object as a whole is three dimensional.

Like with welding three solids together on a rough edge, welding any number of solids on rough boundaries to form the graph G, does not change the energy barrier of the $\bar{X}$ operator.

Lemma 5.4.4 *Doing Z-type welds between rough boundaries $V = \{i\}$ of solid codes $E = \{(i, j)\}$ in a graph $G = (V, E)$ does not change the energy barrier of the membrane operator.*

Proof The graph of flat-Z regions does not change because Z-type quasi-particles can pass through rough welds without creating new quasi-particles while simultaneously not passing through smooth welds where additional quasi-particles are created. Using the parity lower

bound, we get the same energy barrier, which can be saturated if quasi-particles annihilate as soon as they are created. □

Corollary 5.4.5 *By lemma 5.4.4 we can conclude that the welded solid code G has an energy barrier of $O(d)$ for $\bar{X}$, for any solid code that is $O(d)$ qubits wide.*

Notice that each solid (i, j) is a region where X-type quasi-particles can move around without creating new quasi-particles. That is, (i, j) labels flat-X regions and i and j label rough boundaries where new quasi-particles are created. By the parity lower bound, the energy barrier of the welded solid code G is lower bounded by the Ising model corresponding to the same graph G with the reinterpretation that vertices V are qubits and edges E are ZZ interactions. This is precisely how we construct the code with a power-law energy barrier that proves theorem 5.4.3, namely that there exists a 3-d code, with side length $O(L)$, with an energy barrier of $O(L^{\frac{2}{3}})$.

Proof of theorem 5.4.3: We find the energy barrier of $\bar{X}$ and $\bar{Z}$ and then tune them to maximize the energy barrier. We make the graph of solids, G, a 3-d cubic lattice that is R solids wide. By the parity lower bound the energy barrier of $\bar{Z}$ will be lower bounded by $O(R^2)$, like for the Ising model in three-dimensions. This bound can be saturated if in each solid we annihilate pairs of X-type quasi-particles as soon as they are created. Simultaneously the energy barrier for the membrane operator in the solid code is $O(d)$, by lemma 5.4.2, and does not change after welding into the graph G by lemma 5.4.4. If the entire code is L qubits wide, each solid code is d qubits wide and the welded solid code is R solids wide then the number of qubits is $N = O(L^3) = O(d^3R^3)$. The logical-Z and logical-X operators have energy barriers of $O(R^2)$ and $O(d)$ respectively with distances of $O(R^3d)$ and $O(d^2)$ respectively. If we relate the two scales d and R by requiring that $d = O(R^\alpha)$ then the energy barriers of the logical-X and logical-Z operators will be $O(R^2) = O(N^{\frac{2}{3(1+\alpha)}})$

and $O(d) = L(N^{\frac{\alpha}{3(\alpha+1)}})$ respectively. The energy barrier is maximized when the two energy barriers are equal since they have opposite slopes with respect to α. The maximum happens when $\alpha = 2$. Hence the minimum energy barrier for the welded solid code can be tuned to be $O(N^{\frac{2}{9}}) = O(L^{\frac{2}{3}})$ with a minimum distance of $\min(d^2, dR^3) = O(L^{\frac{4}{3}})$.

The last thing to prove is that the density of qubits within a box of length l is bounded. To this end, notice that in the cubic graph G, there are never more than six solids welded together so that since each solid code has a bounded density of qubits, the welded solid code cannot have more than six times that density of qubits. □

5.5 Generalized solid code and welded solid code

In section 5 I welded several solid codes together into a graph where each edge of the graph represented a solid code. However, just like the surface code, the solid code can have more than 2 distinct patches of roughness. For instance, we can have 3 patches of roughness as in figure 5.13.

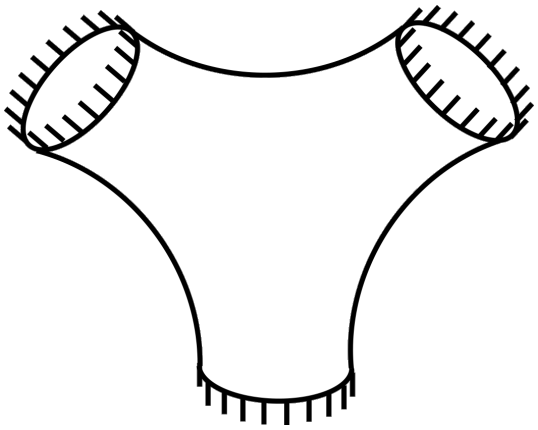

Figure 5.13: Generalized solid code

To describe welding these sorts of solid codes together we must use a generalized notion of a graph. Edges will connect not only two edges, but 3, 4 or more. The graph is still a pair

of sets, the vertices and the edges, but now an edge could be composed of several vertices. We can construct a generalized welded solid code with a set of boundaries V set of edges such that for each edge $e = \{i_1, ..., i_n\} \in E$, we have a solid code with boundaries on $i_1, ..., i_n$ and that if two solid codes share the same boundary, then they are welded together.

There is a strong connection between this topological code and the following linear code. For each generalized edge $e = \{i_1, ..., i_n\}$ do parity check $Z_{i_1}...Z_{i_n}$. Both codes have identical energy barriers and the same number of encoded qubits. Indeed the logical X-operators resemble each other. For the classical code, if we violate a stabilizer by doing a bit flip, then we must flip another bit of that stabilizer in order to satisfy that stabilizer. Similarly, if we flip a qubit on a boundary of solid code, we create a quasi-particle within that solid. If we annihilate that quasi-particle on the same boundary then that Pauli-sequence cannot result in a logical operator because that half-loop is contractible within that solid. Like with the classical code, we must annihilate that quasi-particle on another boundary of that code.

Furthermore, if we flip a qubit(bit) or the boundary(bit) i_1, then each solid(stabilizer) connected to that qubit(bit) will have a quasi-particle.

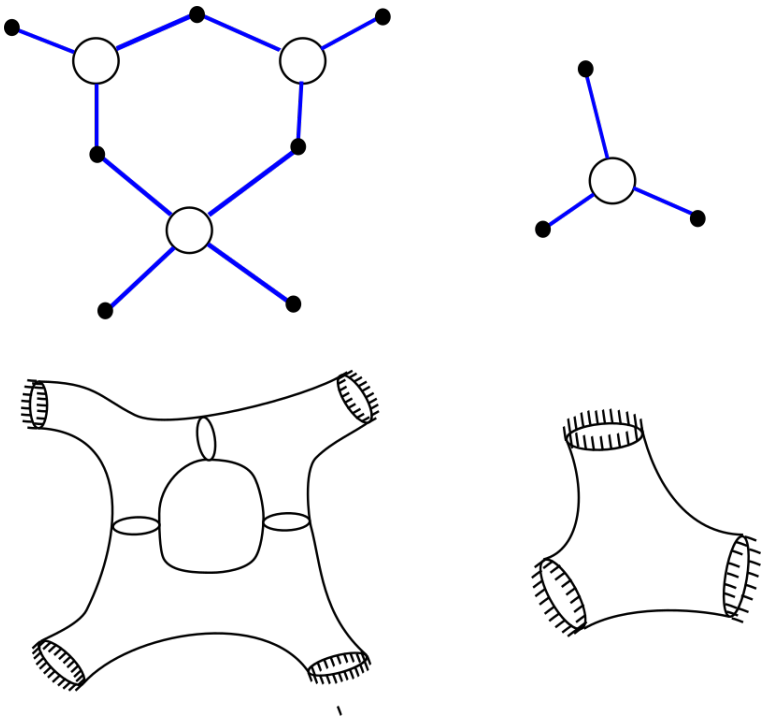

Figure 5.14: Generalized welded solid code. For each classical stabilizer, there is a corresponding solid code.

Why is the connection between welded codes and linear codes useful? It means that all of our knowledge about classical coding can be used in the quantum regime while at the same time preserving spatial locality. So if we had a spatially local classical code that encoded very many qubits, then we can make a quantum code that encoded the very same number of qubits.

5.6 Welded subsystem codes and the subsystem welded solid code.

A subsystem code is a generalization of a stabilizer code where we ignore certain logical operators of the stabilizer group. That is, we do not care about the state of a certain subsets. Logical operators on these qubits represent gauge degrees of freedom. The stabilizer group with these gauge operators is called the gauge group. The gauge group can be expressed as

$H = \langle S, \bar{Z}_{k+1}, \bar{X}_{k+1}, ..., \bar{Z}_{k+a}, \bar{X}_{k+a} \rangle$ where $\{\bar{Z}_{k+1}, \bar{X}_{k+1}, ..., \bar{Z}_{k+a}, \bar{X}_{k+a}\}$ are logical operators of the gauge qubits. When we weld the stabilizer group we must also weld the logical operators. Hence we weld two gauge groups H_1 and H_2 the same way we weld a stabilizer group. We include all of the Z-type gauge operators and weld all of the X-type gauge operators and logical operators. Thus if we have generating sets R_1 and R_2 that are in CSS form, then we simply weld the generators together. There is a subsystem version of the of the surface code with gauge operators of weight 3. In contrast, the surface code has weight 4 stabilizers in the generating set. Lower weight operators is an advantage because it means that we need only measure 3-qubit operators in order to do error correction. We can weld the subsystem surface code into a subsystem solid code and then weld the subsystem solid code together into a subsystem welded solid code. The subsystem welded solid code has generators of weight t. That is one less than the stabilizer version of the code. This result is part of work done by myself and Jonas Anderson [60].

Chapter 6

THERMAL PROPERTIES OF SELF-CORRECTING QUANTUM MEMORIES

6.1 Storage time of the welded solid code

The welded solid code likely does not act as a scalable self-correcting quantum memory for arbitrary systems sizes. However, it does have an increase in lifetime for an increase in system size up to a maximum system size. This is much like the cubic code [20]. I show this for a particular decoder: the annealing decoder. An upper bound is given in [80] where it is shown that the maximum memory lifetimes scales as $\beta N e^{2\beta E_b}$ where E_b is the largest energy barrier over all logical operators. When all logical operators have the same energy barrier, E_b is just the energy barrier.

I want to use theorem 4.4.1 to lower bound the memory lifetime. We can break up decoding into two steps:

1 Within each solid code, annihilate all pairs of defects until there is zero or one defect left.

2 Treat each solid code as if it were a link in an Ising model with the same "fat" graph as the welded solid code. A link is violated if it has a single quasi-particle in it. We then imagine how we would do a round of error correction on the corresponding Ising model. If we were to flip a qubit in the Ising model to do error correction then we would flip a qubit on the corresponding welded boundary. We then annihilate any pairs of qubits within each solid so that each solid has only zero or one quasi-particles. We continue

this way until there are no X-type quasi-particles left.

Would it be any better to consider what happens in each solid more carefully than to just annihilate all pairs? Probably not. We expect that like the 2d toric code, the inside of each solid will thermalize in a constant time [6]. Likely the details of the bulk of each solid code won't matter.

We must find the relative number of ways to arrange an even and odd number of defects within a single solid. This will correspond to the probability for a single violated link within an Ising model and thus an effective temperature. There are two states for each syndrome string s, since there is a single encoded qubit and states with different syndrome strings are orthogonal. What's more is that each star operator within a solid is independent. In order to calculate the thermal density matrix, ρ_β, we calculate the number of ways to have an even or odd number of quasi-particles in each solid. An odd number corresponds to a block with a single quasi-particle after decoding.

$$P_{\text{even}} \sim \sum_{i=0}^{\lfloor \frac{N}{2} \rfloor} \binom{N}{2i} e^{-\beta 2i} = \sum_{m=0}^{N} \binom{N}{m} \frac{1 + e^{\frac{im\pi}{4}}}{2} e^{-\beta m} \tag{6.1}$$

$$P_{\text{odd}} \sim \sum_{i=0}^{\lfloor \frac{N}{2} \rfloor} \binom{N}{2i+1} e^{-\beta} e^{-\beta 2i} = \sum_{m=0}^{N} \binom{N}{m} \frac{1 - e^{\frac{im\pi}{4}}}{2} e^{-\beta m} \,. \tag{6.2}$$

We can solve these exactly by noting that $(1 + e^x)^N = \sum_{m=0}^{N} \binom{N}{m} e^x$.

$$P_{\text{even}} = \frac{1}{2}(1 + (\tanh \frac{\beta}{2})^N) \tag{6.3}$$

$$P_{\text{odd}} = \frac{1}{2}(1 - (\tanh \frac{\beta}{2})^N) \,. \tag{6.4}$$

We conclude that the thermal density matrix of the welded solid code is the same as the

3-d Ising model with an effective temperature given by the relations $\tanh \beta_{\text{eff}}/2 = \tanh \beta/2^N$. Hence in the thermodynamic limit, as $N \to \infty$, $\beta_{\text{eff}} \to 0$. Thus the thermal density matrix of the solid code is the infinite temperature density matrix for the Ising model. Hence, we cannot prove a increasing storage time with increasing system size using theorem 4.4.1.

Still there are some finite size effects that are useful. If the temperature is small enough then the effective temperature will be below the threshold temperature. We expect that in this regime, increasing the system size leads to an increasing memory lifetime. Indeed, Bravyi and Haah [42] derive a useful equation that says that the memory lifetime scales as $t \sim e^{c\beta\delta E}$ where δE is the energy barrier and c is a positive constant. This result holds for any topological stabilizer code with a threshold for quantum communication and when $e^{-\beta} \ll \frac{1}{N}$ where N is the number of physical qubits in the code. This leads to a maximum memory lifetime of $e^{c\beta e^{\beta 3}}$. This is exponentially higher than the maximum lifetime for the cubic code which is $e^{c\beta^2}$. The difference is entirely in the exponentially higher energy barrier for the welded solid code.

6.2 Order parameter for a topologically ordered phase at non-zero temperature

I want to mention what an order parameter would look like for a topologically ordered state at finite temperature. What I mean by an order parameter is some observable that changes abruptly as a function of temperature. This could be taken as the memory lifetime. There the observable is the projector onto the energy ridge. This may be a complicated computation. I conjecture an easier formula, a simple expectation value. I predict that whenever the expectation value takes a particular form, that there will be a long memory lifetime for some decoder. What follows is a generalization of the order parameter proposed by Wegner [54, 85] to stabilizer code Hamiltonians.

For the Ising model, the order parameter is the correlation function $\langle Z_i Z_j \rangle$. Notice that

Z_iZ_j is in the stabilizer group of the Ising code. The order parameter for a topological code can only depend on expectation values of stabilizers. Only stabilizers can have a non-zero expectation value. This is because the thermal density matrix $\rho_\beta \sim e^{-\beta H}$ is a sum over stabilizers and the trace of a Pauli matrix is zero. So if w is a Pauli matrix then $tr(we^{-\beta H}) = 0$. Like the Ising model, I expect it to be a non-local stabilizer. However, the Ising model has non-local stabilizers with finite weight. This makes it so that they, the two point correlation functions, can have a constant expectation value, which they do in the ordered phase. In contrast, every non-local stabilizer of a topological code has a macroscopic weight. This makes it so that the expectation value is at best exponentially decreasing with the weight. I predict the expectation value to be in the ordered phase: $\langle h \rangle \sim e^{-w(h)}$ where q is real number satisfying $1 > q > 0$ and $w(h)$ is the weight of the stabilizer h. I predict that when this is satisfied that the underlying Hamiltonian is self-correcting.

To make this more clear, consider the toric code. What is the probability that a large loop operator is satisfied if the state is thermal? It is the probability of their being an even number of quasi-particles within the loop. If quasi-particles are unbound to each other then this probability will be about $\sim \frac{1+(\tanh \beta)^A}{2}$ where A is the area of the loop. However, when quasi-particles are bound to each other, that is when they pair up, we need not consider the likelihood of an even number of quasi-particles in the interior. We can restrict our attention to quasi-particles near the boundary of the loop. This is because every quasi-particle in the interior necessarily has a partner near by. The only way for an odd number of quasi-particles to be in the loop is if a pair straddles the boundary of the loop. Thus the probability of there being an odd number of defects in the interior of the loop will scale as $\sim \frac{1+e^{-w(h)}}{2}$. The decay of the expectation value of a large loop tells us how confined the quasi-particles are.

The 3-d toric code should have a similar relationship. Here we can think of domains of Xs producing boundaries of quasi-particles on plaquettes. A loop of Z operators, which is

the same as a product of plaquettes, will have a $+1$ eigenvalue if the loop of Zs punctures an even number of these domains and -1 if there are an odd number of such domains. Again, in the ordered phase, these domains should be small, however in the unordered phase, they could be any size. We can count the flux of such domain walls penetrating the loop. In the disordered phase a domain wall, itself a loop, is likely to go around the loop of Zs, like two rings stuck together, because the size of the domain wall is arbitrarily large. Hence I expect that even for the toric code that the expectation value of a loop of ZS goes like $e^{-w(h)}$ in the ordered phase as $\sim e^{-A(h)}$, where A is the area of the loop, in the disordered phase.

I conjecture that any topologically stabilizer code Hamiltonian that has a long lifetime will have quasi-particles that bunch up closely together in neutral clusters and thus that the expectation value of a stabilizer should scale as $\sim e^{-w(h)}$ in the ordered phase and $e^{-\beta f(h)}$ in the disordered phase where $f(h)$ is much bigger than $w(h)$. I am being a bit vague about what $f(h)$ is exactly because it may not be an area as in the 3-d toric code. It depends on the shape of the stabilizers. For instance, in a six dimensional version of the toric code, the logical X and Z operators could be a three dimensional. In that case $f(h)$ would be a volume rather than an area.

6.3 Mediated interactions the effective energy barrier.

Might we be able to improve the energy barrier by using an external system to mediate an attractive interaction between quasi-particles?

The memory lifetime theorem 4.4.1, says that what matters is that the density matrix has very little overlap with the unprotected states on the ridge of the energy landscape. However, we may consider a code that lives on the boundary of a solid and has coupling between it and the solid. In that case the reduced density matrix on the code qubits alone is enough to calculate a lower bound of the memory lifetime. If the reduced density matrix has

an exponentially small probability of being in the unprotected subspace, then the memory will have an exponential lifetime.

In the next two sections I show first that if the external system is a spin system and couples locally to stabilizers, then the probability for a state with syndrome string s to be occupied is at best improved to p_s^c where c is a real number greater than one. The log of this probability is what I call the effective energy barrier. Hence, the effective energy barrier is only improved by a constant factor when coupling to an external spin lattice.

There has been a suggestion that the toric code may be self-correcting if the plaquette and star operators are coupled to a bosonic bath [67]. These operators have an unbounded strength as there are an infinite number of levels for each boson. We prove that when the operators are of a finite dimension and of a bounded strength, then the effective energy barrier is no more than a constant factor bigger then with no bath. This implies that if the topological stabilizer Hamiltonian was not self correcting before coupling, then it will not be afterward.

Definition The Hamiltonian takes the form $H = H_A \otimes I_B + I_A \otimes H_B + V$, V is an interaction between system A and system B. $H_A = \sum_i -h_i$ and $V = \sum_i h_i \otimes v_i$ where h_i is a local generator of the toric code and v_i is an operator of bounded strength, $\|v_i\|_\infty < constant$.

Theorem 6.3.1 *Let H be a Hamiltonian such that H_A is a local stabilizer Hamiltonian and V has couplings of the form $V = \sum_i h_i \otimes v_i$ where h_i is a local generator of a stabilizer group and $h_i \otimes v_i$ is of bounded strength. The effective energy barrier can only be a multiplicative constant factor bigger than the energy barrier of H_A.*

Proof Let P_s be a subspace of the ridge of correctability: $P_s P_\perp = P_s$. Let r be an operator such that if $|\psi\rangle$ is a code state then $r|\psi\rangle \in P_s$. The relative probability for the thermal state

ρ_β to be in the subspace P_s versus the code space P_0 is

$$\frac{p_s}{p_0} = \frac{tr\rho_\beta r P_0 r}{tr\rho_\beta P_0} = \frac{tr(e^{-\beta rHr} P_0)}{tr(e^{-\beta(rHr+\Delta H)} P_0)} \tag{6.5}$$

where $\Delta H = H - rHr$. Now if r violates n stabilizers then

$$\frac{p_s}{p_0} \geq \frac{1}{\|e^{-\beta \Delta H}\|_\infty} \sim e^{-O(n)}\,. \tag{6.6}$$

This follows because rHr and ΔH commute and for any positive operators A and B, $tr(AB) \leq \|A\|_\infty tr(B)$. We conclude that the effective energy of a state in the subspace P_s can only increase by a constant factor when terms in a stabilizer Hamiltonian are coupled to a local spin Hamiltonian with terms of bounded strength. $\square$

Finally, to show that the overlap with the unprotected subspace is small, we note that if the coupled system is self correcting for some temperature, then we can just lower the temperature for the uncoupled stabilizer until the unprotected subspace is less probable than for the coupled system. This would imply that the uncoupled stabilizer code didn't have a small overlap with the unprotected subspace, then neither does the coupled system.

In particular, the previous theorem applies to the toric code. It says that coupling stabilizers of the toric code still has a constant energy barrier even if there are interactions that are mediated by a spin bath. This result was derived by myself and Aram Harrow in unpublished work [61].

However, if the coupling V does not commute with the stabilizer Hamiltonian H_A, then it is open as to whether coupling to a an external system can increase the energy barrier. However, I think it is unlikely that such a model will work. The reason is that if a coupling term anti-commutes with a stabilizer, then it will have an expectation value of zero in the ground state subspace, the code space, of H_A. However, this means that states in the code

space would have no energy barrier at all from such terms. That is, a Pauli-sequence would have no contribution from such terms.

Chapter 7

CELLULAR AUTOMATA DECODERS

In chapter 4 I mentioned that if we find a Hamiltonian that leads to a self-correcting memory, then we can find a local constant depth circuit that performs error corrections. A cellular automata(CA) decoder is another class of error correcting circuits that can be done with a local constant depth circuit. A cellular automata is essentially a lattice of finite sized computers, called cells that can only communicate with nearest neighbors. The idea is to embed this lattice of cells in a lattice of qubits. The lattice of qubits encodes a quantum state using a stabilizer code. The purpose of the cells is to make measurements, communicate with each other and do local operations.

A classical example of a CA decoder is Toom's rule [81]. It is a local majority vote decoder where a particular bit is changed to the majority vote of itself, its northern neighbor and its eastern neighbor. Each bit is updated in parallel in each round of error correction. Noise happens between error corrections steps. Toom's rule has a threshold [38] for i.i.d. bit flip noise, i.e. where each bit is independently flipped with a probability p at each time step. That is, if the system starts out with mostly 0s(or 1s), then Toom's rule will keep the system this way for a very long time on average. Not only that, but it is even stable for a non-local field, i.e. when there is a bias towards or against flipping to the 0 state. In contrast, doing the majority vote of all neighbors, the Ising model, is only meta-stable against a bias to flip towards a particular value, e.g. a magnetic field.

There is even numerical evidence [65] that a Toom-like rule can be used to make a 4-d toric code stable. The difficulty is to do this in 3-d for codes where there is no energy

barrier. The CA must mediate a long distance interaction as there may be a large distance between quasi-particles. This does not happen in the Ising code and the 4-d toric code. Quasi-particles always have neighbors a constant distance away.

I show numerical results for a different cellular automata that is based on signaling. It indicates that the cellular automata is only partially self-correcting. Ref. [47] simulates a force by using a cellular automata that is inspired by Gauss's law. Although their cellular automata fails to be a stable memory, it does work as a decoder.

There is an alternative approach to cellular automata that has not received much attention. It is to simulate concatenated fault-tolerant circuits using cellular automata. This is what Gac does in his stable 1-d cellular automata [34]. It is a stable 1-d classical memory. I believe those same principles can be used to design a stable quantum memory. Some of these principles are elucidated in [37].

7.1 Simulating a force with a CA

How do we simulate a force with a cellular automata? I use a cellular automata that does this by signaling. Each quasi-particle sends out a signal and signals bounce off of each other and back to where they originated. Thus they signal which way a quasi-particle should move to annihilate with a nearby quasi-particle. I do a simulation that is broken up into four pieces: apply errors, update the CA, do error-correction and finally check if the resulting error-corrected state encodes the same information as initially. The cellular automata that I present is not fully self-correcting. It has an increase in storage time with an increase in the system size up to a maximum system size. After this, the memory time flattens out. However, lifetime is very much higher than with no CA. We focus just on the plaquette operators, i.e. the Z-stabilizers of the toric code. If we can find a CA that corrects X errors by measuring Z-stabilizers, then we can find a CA that corrects Z errors by measuring

X-stabilizers. This is because both X and Z-type quasi-particles have the same behavior. This means that we are simulating a classical error correcting code with local parity checks where the logical operator that we care about is defined by loops around the torus. In the simulation we have a lattice of $2n^2$ bits and each set to 0 at the beginning of the simulation. The simulation then flips each bit with a probability p during each round. When every a bit is flipped, it creates two quasi-particles, annihilates two quasi-particles or moves one quasi-particle around. There are essentially two problems to solve in designing a local CA that does error correction. How do we send signals between close by quasi-particles and how do we get rid of those signals once the quasi-particles have annihilated? First I discuss the way I propose to do signaling. Imagine that you are sitting on a cell and you can shout things to people occupying neighboring cells. If you see a quasi-particle in your cell, you shout, "north, east, south and west". This tells your northern neighbor that there is something in the south, the western neighbor that something is in the east and so on. If you hear from the northern neighbor "south", then you know that there is a signal propagating from the north from a quasi-particle. Thus you move the quasi-particle to your neighbors cells and so on for each direction. Quasi-particles may not be directly north or directly west of each other. So there has to be a way of signaling. After some time, each quasi-particle will have a cross pattern of signals propagating outward, and a line of east facing arrows propagating eastward and similarly for each direction. When two perpendicular lines of signals intersect, they bounce off of each other. So for instance, if you had no quasi-particle at your site and you hear "east" from your "western" neighbor and "north" from your southern neighbor, you will shout, "west and south" at the next time step. This sends the signal back to the cell where the signals originated and thus signals the person at the cell to move their quasi-particle towards the signal and hopefully towards another quasi-particle.

How do we get rid of the signal once quasi-particles have collided? So far we haven't

specified a rule for when to stop shouting. The first rule is that when we move a quasi-particle towards the direction of a signal, we stop conveying that signal to our neighbors. So if our eastern neighbor shouts "west" and we move our quasi-particle east, we do not shout "west"; the signal dies. There is still a problem though, after the quasi-particles annihilate, their signal is still being propagated. To get around this, every time perpendicular signals collide, they get marked so that there is a different between your southern neighbor shouting "north" and "reflected north". After two collisions, they disappear. This tends to reduce the number of stray signals and thus noise from stray arrows.

Figures 7.1 and 7.2 show time slices of the CA decoder at work at a time $t = 20$. You can see from the figures that there are far more quasi-particles, red dots, when the CA is turned off.

These rules could probably be improved upon. The two basic problems are to get rid of stray arrows and to make sure that the arrows propagate a long distance faithfully; if a quasi-particle moves, it will have dodged the signal.

One solution is to propagate a signal with a fixed probability. Then the range of a signal becomes finite as the probability for a signal to survive after several time steps decays exponentially with the number of time steps. The survival probability would need to be system dependent and for large systems, it would need to be arbitrarily small. There are problems with this system dependent probability. How would we simulate it with a finite number of bits of storage in each cell? It would take $O(\log 1/p)$ coin tosses to get a probability of p. We can simulate a probability of 2^{-n} by tossing a coin n times and only accepting if all coins come out heads. This wouldn't properly be a finite cellular automata anymore if n is allowed to be arbitrarily large. Moreover, in a realistic setting, the CA itself will be noisy with a constant rate of faults. Thus propagation rate is actually bounded by the fault rate for small enough propagation rates. That is, an arrow is more likely to have noise than to

be destroyed by the cellular automata rule. We have to have CA rules that both makes the signal more reliable against faults and cleans up stray signal when quasi-particles annihilate.

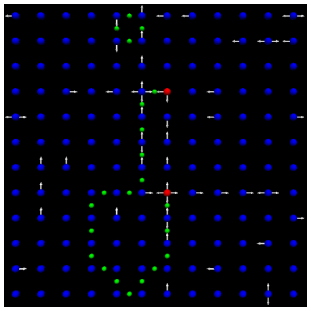

Figure 7.1: A time slice of a simulation at the 20th iteration and an error probability of 0.0005. The green dots indicate bit flips, blue dots indicate satisfied stabilizers, red dots indicate violated stabilizers and the arrows represent the signal traveling in the direction of the arrow.

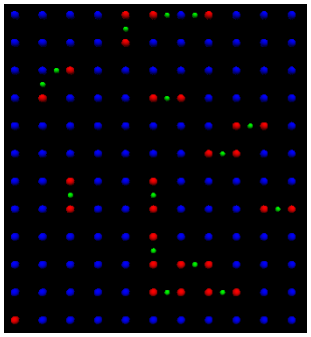

Figure 7.2: A time slice of a simulation at the 20th iteration and an error probability of 0.0005 this time without the CA decoder.

7.1.1 Cellular automata as a decoder

The first test that I do is to see if the cellular automata serves as a decoder against a single round of noise. Here I apply errors only once and then repeat the cellular automata until there are no quasi-particles left. Then I check if there was a logical error. I repeated this 1000 times and graphed the proportion of failures vs the error rate. To ensure that the CA

would annihilate all of the quasi-particles, I added the rule that the arrows only propagate with a fixed probability of $1/L$ where L^2 is the maximum number of quasi-particles. This makes it so that the probability for an arrow to go all the way around the torus to be about e^{-1}. This is assuming it did not collide with another arrow. The graph is shown in figure 7.3. The need for a probabilistic update rule is something I hope can be improved upon.

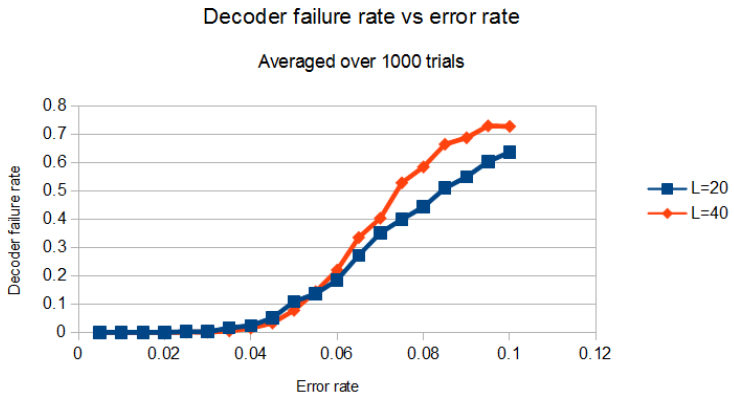

Figure 7.3: Decoding failure rate vs. system size averaged over 1000 trials for systems size $L = 20$ and $L = 40$.

We can see that the CA works quite well for error rates of less than 4%. However, there is not much improvement between system sizes. That is, the slope does not increase much. This indicates that this decoder does not have threshold. However at low error rates, below 4%, the larger system, $L = 40$, consistently has a lower failure rate.

The graph is peculiar in that I would expect that for a large enough error rate, the failure rate should approach 50%. However, it goes over 70% for $L = 40$. For high enough error rates, we would be better off assuming a logical error occurred and undoing it. In this regime, the CA behaves more as a logical operator than a decoder.

7.1.2 Performing error correction with a cellular automata.

Next I tested if the CA can increase the memory lifetime of the toric code by actively doing error correction after each round of noise. For a particular error rate and system size I first, apply errors, update the cellular automata, perform error correction and then test to see if the error corrected state is correct or not. To do this I apply the RG decoder and check if there is a logical error in the error corrected state. I chose the RG decoder [42] because it is simple to implement and has threshold for quantum communication.

The way the decoder works, is that quasi-particles are put into groups. For every quasi-particle, we search a distance 1 away to see if there are any quasi particles nearby. If so, put them in a group. Do this for every quasi-particle. If a group has an even number of quasi-particles in it, then annihilate them together. For all of the remaining quasi-particles repeat the procedure again but now doubling the maximum distance between quasi-particles. Continue doing this until all the quasi-particles are gone. At the very end, we check the parity of bit flips along a loop logical operator. If the parity is odd, then the decoder has failed. Otherwise it has succeeded.

One of the difficulties of testing whether a cellular automata does a good job decoding the toric code is that if it is doing a good job, it takes a very long time for a decoding error. So we have to restrict ourselves to small systems sizes when running numerical tests. Two graphs are shown below. One is the memory lifetime, time for a decoding error, vs system size for various error rates and the other is the memory lifetime vs error rate for a system size of $L = 66$. We see that for low enough error rates, as the system size increases the memory lifetime goes up but then eventually goes down. This indicates that while this cellular automata increases the lifetime of the memory, it does not have a threshold. We can see from figure 7.5 that as the error rate goes down, the memory lifetime goes up.

Despite not having a threshold, this CA seems to improve the memory lifetime quite a

bit. From a theoretical standpoint, it is a bit disappointing that the CA does not have a threshold. However from a practical standpoint, for someone trying to simulate the toric code in the lab, improving the the memory lifetime several thousand times is a huge gain.

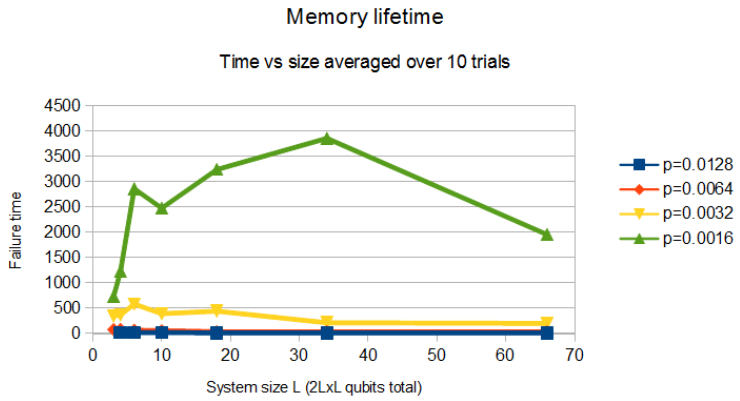

Figure 7.4: System size vs. memory lifetime averaged over 10 trials for various error rates p.

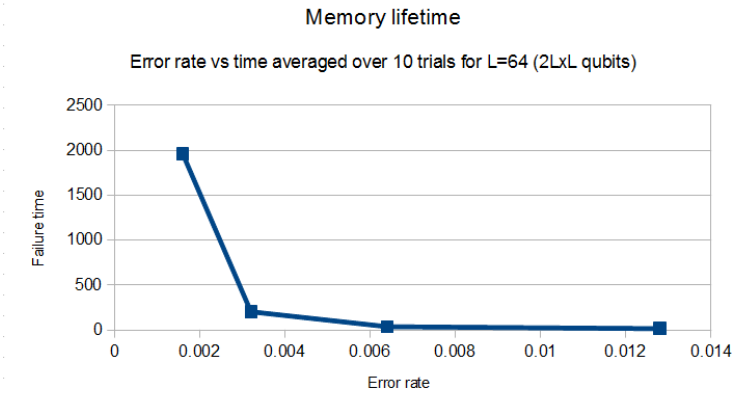

Figure 7.5: Error rate vs memory lifetime averaged over 10 trials for a system size of $L = 66$.

7.1.3 Summary

The signaling CA does seem to work as a decoder when repeated on a state that suffers a single round of i.i.d noise. The signaling CA does indeed improve the memory lifetime but the data indicates that it does so only by a constant factor. However, the data indicates that this constant factor goes up unboundedly as the error rate goes down to zero.

Chapter 8

CONCLUSION

I have discussed self-correcting memories based on topological stabilizer codes. I have discussed several ways of thinking about topological codes: stabilizers, non-local Ising model, quasi-particles and welding. It is my hope that these alternative ways of picturing and thinking about topological codes lead to codes with higher energy barriers, larger memory lifetimes or no-go theorems in 3-d.

Picturing things differently lead me to discover a way of constructing a code, the welded solid code, with the highest known energy barrier. The typical way that I saw people represent the toric code was by a graph where qubits were represented by edges of a graph and stabilizers by vertices and plaquettes. This representation limits the energy barrier by construction since there are at most two vertex stabilizers per qubit. To get an energy barrier you need three or more vertex operators per qubit and thus qubits have to be represented as vertices themselves. This is the stabilizer graph picture that I introduced.

Without visualizing things in this way, I would have never invented welding or the welded solid code. I invented the technique of welding because I needed to know how logical operators and stabilizers changed when I added new terms into the stabilizer group. A new stabilizer may not commute with the operators of a stabilizer group or the logical operators. The resolution was that these operator had to be welded together.

The shape of the logical operator determines the energy barrier and so by using welding, I was able to choose the shape so that it was large.

I later discovered that welding was in fact a universal language for describing stabilizer

codes. You can partition the qubits into overlapping sets and define stabilizer groups for each set such that welding these groups together leads to the original stabilizer groups. This can help to simplify thinking about topological codes.

I then discovered that you can ignore the fine details of these codes and just look at the shape of the logical operators and how they deform. For instance, the welded code can be described as a net of strings. You can deform the strings and move the location where they are attached, but you cannot get rid of any nodes and you cannot get rid of any strings. What is more is that the shape predicts the energy barrier. Quasi-particles have to travel along these strings and bifurcate at the nodes. The challenge is to come up with two sets of string nets, both with high energy barriers, that always intersect at an odd number of locations no matter how you deform them.

I searched in vain for a topological code which had a better energy barrier than the one that I invented. I moved my efforts to error correction by cellular automata. Cellular automata share an important property with self-correcting memories. Both of them can be simulated with a constant depth quantum circuit. In that sense, they are both self correcting.

There have yet to be any CA decoders for topological stabilizer codes which exhibit a long memory lifetime. So like thermal self-correcting memories, the problem is still open. However, it is very likely that such a CA decoder exists. I believe this because of a result by Gac [34]. There the author constructs a 1-d cellular automata that works as a classical memory. He uses concatenated linear codes. I believe that the same construction can be adapted to doing error correction on a concatenated stabilizer code of the CSS type. This is because a CSS code is essentially two classical codes, one for X-type errors and one for Z-type errors.

In the paper [67] the authors couple a toric code to a lattice in order to mediate an interaction. I wondered to what extent this could increase the energy barrier? However, I

found that if the coupling operators are of bounded strength, that you can only increase the energy barrier by a constant factor.

When reading the paper [26] on the criteria for self-correcting memories, I noticed that the theorem could be generalized to cases other than stabilizer codes and subsystem codes. However, the basic proof technique that I use is essentially the same. This theorem is remarkable in that it says that we need only prove certain properties of the equilibrium distribution of the thermalizing dynamics in order to prove that the dynamics take a long time to destroy the contents of the memory. This gives justification for talking about the stability of models like the Ising model based on the Gibbs distribution without reference to any dynamics.

There are several no-go theorems in 2-d that say that a Hamiltonian cannot both exhibit topological order and have an energy barrier [18, 50, 56]. The no-go theorems in 3-d are much more limited. We know that a translation invariant Hamiltonian can have at most an energy barrier of $\log L$ where the number of qubits is $\sim L^3$ [41, 88]. This bound is saturated by the Cubic code. The welded solid code has a much larger energy barrier: $\sim L^{\frac{2}{3}}$. This suggests that there are much richer possibilities for topological codes that are not translation invariant.

Finally, I want to mention future avenues of research. We can look for better topological codes with higher energy barriers. We can consider topological Hamiltonians that are not based on stabilizer codes; perhaps a frustrated model. We can design better quantum memories based on cellular automata. Or we can prove better no-go results in 3-d and focus on quantum memories based on concatenated codes and fault-tolerant circuits; something we already know how to do.

BIBLIOGRAPHY

[1] Dorit Aharonov and Michael Ben-Or. Fault-tolerant quantum computation with constant error. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 176–188. ACM, 1997.

[2] Abbas Al-Shimary, James R Wootton, and Jiannis K Pachos. Lifetime of topological quantum memories in thermal environment. *New Journal of Physics*, 15(2):025027, 2013.

[3] R Alicki, M Fannes, and M Horodecki. A statistical mechanics view on kitaev's proposal for quantum memories. *Journal of Physics A: Mathematical and Theoretical*, 40(24):6451, 2007.

[4] R. Alicki, M. Horodecki, P. Horodecki, and R. Horodecki. On thermal stability of topological qubit in kitaev's 4d model. *Arxiv preprint arXiv:0811.0033*, 2008.

[5] Robert Alicki, Mark Fannes, and Michal Horodecki. On thermalization in kitaev's 2d model. *Journal of Physics A: Mathematical and Theoretical*, 42(6):065303, 2009.

[6] Robert Alicki and Michal Horodecki. Can one build a quantum hard drive? a no-go theorem for storing quantum information in equilibrium systems. *arXiv preprint quant-ph/0603260*, 2006.

[7] Robert Alicki, Daniel A Lidar, and Paolo Zanardi. Internal consistency of fault-tolerant quantum error correction in light of rigorous derivations of the quantum markovian limit. *Physical Review A*, 73(5):052311, 2006.

[8] Panos Aliferis and Andrew W Cross. Subsystem fault tolerance with the bacon-shor code. *Physical review letters*, 98(22):220502, 2007.

[9] Panos Aliferis, Daniel Gottesman, and John Preskill. Quantum accuracy threshold for concatenated distance-3 codes. *arXiv preprint quant-ph/0504218*, 2005.

[10] D. Bacon. Operator quantum error-correcting subsystems for self-correcting quantum memories. *Physical Review A*, 73(1):012340, 2006.

[11] Charles H Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM journal on Computing*, 26(5):1510–1523, 1997.

[12] Elwyn R Berlekamp, Robert J McEliece, and Henk CA Van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.

[13] H Bombin, RW Chhajlany, M Horodecki, and MA Martin-Delgado. Self-correcting quantum computers. *New Journal of Physics*, 15(5):055023, 2013.

[14] H. Bombin and MA Martin-Delgado. Quantum measurements and gates by code deformation. *Journal of Physics A: Mathematical and Theoretical*, 42:095302, 2009. arXiv:quant-ph/0605094.

[15] Hector Bombin and Miguel Angel Martin-Delgado. Topological quantum distillation. *Physical review letters*, 97(18):180501, 2006.

[16] S. Bravyi and J. Haah. Analytic and numerical demonstration of quantum self-correction in the 3d cubic code. *Arxiv preprint arXiv:1112.3252*, 2011.

[17] S. Bravyi and J. Haah. Energy landscape of 3d spin hamiltonians with topological order. *Physical Review Letters*, 107(15):150504, 2011. arXiv:1105.4159v1.

[18] S. Bravyi and B. Terhal. A no-go theorem for a two-dimensional self-correcting quantum memory based on stabilizer codes. *New Journal of Physics*, 11:043029, 2009. arXiv:0810.1983v2.

[19] Sergey Bravyi, Guillaume Duclos-Cianci, David Poulin, and Martin Suchara. Subsystem surface codes with three-qubit check operators. *Quantum Information & Computation*, 13(11-12):963–985, 2013.

[20] Sergey Bravyi and Jeongwan Haah. Quantum self-correction in the 3d cubic code model. *Physical review letters*, 111(20):200501, 2013.

[21] Sergey Bravyi, Matthew B Hastings, and Spyridon Michalakis. Topological quantum order: stability under local perturbations. *Journal of Mathematical Physics*, 51(9):093512, 2010.

[22] Benjamin J Brown, Daniel Loss, Jiannis K Pachos, Chris N Self, and James R Wootton. Quantum memories at finite temperature. *arXiv preprint arXiv:1411.6643*, 2014.

[23] A.R. Calderbank, E.M. Rains, PM Shor, and N.J.A. Sloane. Quantum error correction via codes over gf (4). *Information Theory, IEEE Transactions on*, 44(4):1369–1387, 1998. arXiv:quant-ph/9608006v5.

[24] A.R. Calderbank and P.W. Shor. Good quantum error-correcting codes exist. *Physical Review A*, 54(2):1098, 1996. arXiv:quant-ph/9512032v2.

[25] C. Castelnovo and C. Chamon. Entanglement and topological entropy of the toric code at finite temperature. *Physical Review B*, 76(18):184442, 2007. arXiv:0804.3591v2.

[26] Stefano Chesi, Daniel Loss, Sergey Bravyi, and Barbara M Terhal. Thermodynamic stability criteria for a quantum memory based on stabilizer and subsystem codes. *New Journal of Physics*, 12(2):025013, 2010.

[27] Stefano Chesi, Beat Röthlisberger, and Daniel Loss. Self-correcting quantum memory in a thermal environment. *arXiv preprint arXiv:0908.4264*, 2009.

[28] William Cook and Andre Rohe. Computing minimum-weight perfect matchings. *INFORMS Journal on Computing*, 11(2):138–148, 1999.

[29] Andrew W Cross, David P DiVincenzo, and Barbara M Terhal. A comparative code study for quantum fault-tolerance. *arXiv preprint arXiv:0711.1556*, 2007.

[30] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill. Topological quantum memory. *Journal of Mathematical Physics*, 43:4452, 2002. arXiv:quant-ph/0110143v1.

[31] JM Deutsch. Quantum statistical mechanics in a closed system. *Physical Review A*, 43(4):2046, 1991.

[32] Guillaume Duclos-Cianci and David Poulin. Fast decoders for topological quantum codes. *Physical review letters*, 104(5):050504, 2010.

[33] Jack Edmonds. Maximum matching and a polyhedron with 0, l-vertices. *J. Res. Nat. Bur. Standards B*, 69(1965):125–130, 1965.

[34] Peter Gács. Reliable computation with cellular automata. *Journal of Computer and System Sciences*, 32(1):15–78, 1986.

[35] D. Gottesman. Pasting quantum codes. *Arxiv preprint quant-ph/9607027*, 1996.

[36] D. Gottesman. Stabilizer codes and quantum error correction. *Arxiv preprint quant-ph/9705052*, 1997.

[37] Daniel Gottesman. Fault-tolerant quantum computation with local gates. *Journal of Modern Optics*, 47(2-3):333–345, 2000.

[38] Geoffrey Grinstein. Can complex structures be generically stable in a noisy world? *IBM Journal of Research and Development*, 48(1):5–12, 2004.

[39] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219. ACM, 1996.

[40] J. Haah. Local stabilizer codes in three dimensions without string logical operators. *Physical Review A*, 83(4):042330, 2011. arXiv:1101.1962v2.

[41] J. Haah. Commuting pauli hamiltonians as maps between free modules. *Arxiv preprint arXiv:1204.1063*, 2012.

[42] J. Haah and S. Bravyi. Quantum memory on topological spin glass. *Bulletin of the American Physical Society*, 2012.

[43] J. Haah and J. Preskill. Logical operator tradeoff for local quantum codes. *Arxiv preprint arXiv:1011.3529*, 2010.

[44] Alioscia Hamma, Claudio Castelnovo, and Claudio Chamon. Toric-boson model: Toward a topological quantum memory at finite temperature. *Physical Review B*, 79(24):245122, 2009.

[45] Richard W Hamming. Error detecting and error correcting codes. *Bell System technical journal*, 29(2):147–160, 1950.

[46] Matthew B Hastings, Grant H Watson, and Roger G Melko. Self-correcting quantum memories beyond the percolation threshold. *Physical review letters*, 112(7):070501, 2014.

[47] Michael Herold, Earl T Campbell, Jens Eisert, and Michael J Kastoryano. Cellular-automaton decoders for topological quantum memories. *arXiv preprint arXiv:1406.2338*, 2014.

[48] Adrian Hutter, James R Wootton, Beat Röthlisberger, and Daniel Loss. Self-correcting quantum memory with a boundary. *Physical Review A*, 86(5):052340, 2012.

[49] Alastair Kay. Nonequilibrium reliability of quantum memories. *Physical review letters*, 102(7):070503, 2009.

[50] Alastair Kay and Roger Colbeck. Quantum self-correcting stabilizer codes. *arXiv preprint arXiv:0810.3557*, 2008.

[51] A.Y. Kitaev. Fault-tolerant quantum computation by anyons. *Annals of Physics*, 303(1):2–30, 2003. arXiv:quant-ph/9707021v1.

[52] A.Y. Kitaev, O. Hirota, AS Holevo, and CM Caves. Proceedings of the third international conference of quantum communication and measurement. 1997.

[53] Emanuel Knill, Raymond Laflamme, and Wojciech H Zurek. Resilient quantum computation: error models and thresholds. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 454(1969):365–384, 1998.

[54] John B Kogut. An introduction to lattice gauge theory and spin systems. *Reviews of Modern Physics*, 51(4):659, 1979.

[55] Lev Davidovich Landau, JS Bell, MJ Kearsley, LP Pitaevskii, EM Lifshitz, and JB Sykes. *Electrodynamics of continuous media*, volume 8. elsevier, 1984.

[56] Olivier Landon-Cardinal and David Poulin. Local topological order inhibits thermal stability in 2d. *Physical review letters*, 110(9):090502, 2013.

[57] Florence Jessie MacWilliams and Neil James Alexander Sloane. *The theory of error correcting codes*, volume 16. Elsevier, 1977.

[58] Fabio Martinelli. Lectures on glauber dynamics for discrete spin models. In *Lectures on probability theory and statistics*, pages 93–191. Springer, 1999.

[59] Spyridon Michalakis and Justyna P Zwolak. Stability of frustration-free hamiltonians. *Communications in Mathematical Physics*, 322(2):277–302, 2013.

[60] K. Michnicki and J. Anderson. In conversation, 2013.

[61] K. Michnicki and Harrow A. W. In conversation', 2014.

[62] Kamil P Michnicki. 3d topological quantum memory with a power-law energy barrier. *Physical review letters*, 113(13):130501, 2014.

[63] M.A. Nielsen, I. Chuang, and L.K. Grover. Quantum computation and quantum information. *American Journal of Physics*, 70:558, 2002.

[64] NIST. Quantum algorithm zoo, 2015.

[65] Fernando Pastawski, Lucas Clemente, and Juan Ignacio Cirac. Quantum memories based on engineered dissipation. *Physical Review A*, 83(1):012304, 2011.

[66] Fernando Pastawski and Beni Yoshida. Fault-tolerant logical gates in quantum error-correcting codes. *Physical Review A*, 91(1):012305, 2015.

[67] Fabio L Pedrocchi, Adrian Hutter, James R Wootton, and Daniel Loss. Enhanced thermal stability of the toric code through coupling to a bosonic bath. *Physical Review A*, 88(6):062313, 2013.

[68] Carl Pomerance. A tale of two sieves. *Biscuits of Number Theory*, 85, 2008.

[69] John Preskill. Reliable quantum computers. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 454(1969):385–410, 1998.

[70] Wojciech Roga, Mark Fannes, and Karol Życzkowski. Davies maps for qubits and qutrits. *Reports on Mathematical Physics*, 66(3):311–329, 2010.

[71] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[72] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 124–134. IEEE, 1994.

[73] Peter W Shor. Scheme for reducing decoherence in quantum computer memory. *Physical review A*, 52(4):R2493, 1995.

[74] Peter W Shor. Fault-tolerant quantum computation. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*, pages 56–65. IEEE, 1996.

[75] Mark Srednicki. Chaos and quantum thermalization. *Physical Review E*, 50(2):888, 1994.

[76] Cyril Stark, Lode Pollet, Ataç Imamoğlu, and Renato Renner. Localization of toric code defects. *Physical review letters*, 107(3):030504, 2011.

[77] A. Steane. Multiple-particle interference and quantum error correction. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 452(1954):2551–2577, 1996. arXiv:quant-ph/9601029v3.

[78] Andrew M Steane. Error correcting codes in quantum theory. *Physical Review Letters*, 77(5):793, 1996.

[79] Kristan Temme. Lower bounds to the spectral gap of davies generators. *Journal of Mathematical Physics*, 54(12):122110, 2013.

[80] Kristan Temme. Thermalization time bounds for pauli stabilizer hamiltonians. *arXiv preprint arXiv:1412.2858*, 2014.

[81] Andrei L Toom. Stable and attractive trajectories in multicomponent systems. *Advances in Probability*, 6(1):549–575, 1980.

[82] Daniel C Tsui, Horst L Stormer, and Arthur C Gossard. Two-dimensional magneto-transport in the extreme quantum limit. *Physical Review Letters*, 48(22):1559, 1982.

[83] Alexander Vardy. The intractability of computing the minimum distance of a code. *IEEE Transactions on Information Theory*, 43(6):1757–1766, 1997.

[84] David S Wang, Austin G Fowler, Ashley M Stephens, and Lloyd Christopher L Hollenberg. Threshold error rates for the toric and surface codes. *arXiv preprint arXiv:0905.0531*, 2009.

[85] Franz J Wegner. Duality in generalized ising models and phase transitions without local order parameters. *Journal of Mathematical Physics*, 12(10):2259–2272, 1971.

[86] X.-G. Wen. An introduction of topological order.

[87] Xiao-Gang Wen. Vacuum degeneracy of chiral spin states in compactified space. *Physical Review B*, 40(10):7387, 1989.

[88] B. Yoshida. On the feasibility of self-correcting quantum memory. *Annals of Physics*, 2011. arXiv:1103.1885v3.