Quantum Error Correction

Lecture notes of the Quantum Error Correction course by Prof. Kastoryano at University of Cologne, Wintersemester 2018/2019.

Contents

Introduction

At the beginning the theory of quantum error correction was a minor field inside quantum information and quantum computation. Physicists were mainly interested in abstract ideas of entanglement and some connections to thermodynamics. The development of quantum error correction was very slow and it was a fringe topic until Schor came out with the factoring algorithm. The factoring algorithm showed that a quantum computer can factor numbers in a polynomial time, while a classical computer takes exponential time. However, even with this result, physicists at that day did not believe that quantum computation would ever be possible because coherent quantum states were extremely fragile, and thus building a large scale, controllable, quantum system with a small error rate was a chimera. At the beginning of 1995, there were some proposals of codes that were able to correct quantum data. This was one of the major development in the early days in quantum computation and it was the starting point of convincing the physics community that quantum computation was possible. The importance of quantum error correction is easily understood by comparing classical and quantum error rates. In a classical computer the average error rate is 10^{-18}, while the best quantum computers that exist nowadays have an error rate of 10^{-4}. Actually, it is almost inconceivable that they will go beyond 10^{-7}. In other words, in quantum computation we will not be able to perform any relevant computation unless we can are able to perform error correction.

The first section of this lecture notes is about classical error correction. Concepts such as physical and logical bits and error rates will be explained. Then, we will focus on linear codes will and we will use the generator matrix to represent them. Moreover, the parity-check matrix, which is an equivalent representation for codes, will also be introduced. Afterwards, we will go through the decoding process and we will review what the distance of a code is. To finish the chapter, we will see a threshold that a code should fulfil in order to be considered a good code.

The second section is another necessary review before delving into quantum error correction (QEC). We go over the basics of classical and quantum information. It starts characterising the state of a classical system and introducing the concept of a classical bit. We explain that there exists only one single-bit operation, but that we can do computation with more than one bit. In this context and to complete the review about classical information, the concept of gate is introduced and some examples of gates and operations are given. The first element of quantum information that we introduce

is the qubit. We explain the possibility of representing it using the Bloch sphere. Then, quantum operations are described with particular attention to unitary operations and projective measurements. We introduce a useful decomposition of quantum operations called Kraus decomposition. In quantum information, the concepts of randomness and noise are different than in classical information. We see them in detail in this chapter. Finally, the potential issues that quantum information has to overcome are enumerated and explained.

The third section of these notes is delved into the Shor code. We explain how it can correct bit and phase errors and linear combinations of them. We also comment why it is not used in practise.

The forth, fifth and sixth sections are shorter sections that delve into concrete topics. We first review the Knill-Laflamme theorem, which gives conditions for a subspace to be a code space. The physical noise is considered in the fifth section, in particular under the assumption of independent and identically distributed noise. Then, we study continuous time errors and see how they can be discretised.

In section seven we explain the stabilizer formalism. The Pauli group is defined as starting point and then its tensor product is considered to build stabilizer codes. We explain several properties of them as well as we see them in the concrete example of the Shor code on nine qubits.

The eighth section is devoted to the toric code. We explain this relevant code introduced by Kitaev in 1998 presenting its stabilizers and logical operators. The toric code has a connection with many-body physics, which is seen in this section. Then, we consider errors in the toric code and three different decoders. The corresponding thresholds are viewed at the end of the section. In section nine, we estimate the maximal threshold of the toric code. This threshold is obtained by analysing the decoding problem as a classical statistical model.

Section ten is devoted to the relation between topological order and quantum error correction. For that, we consider and define commuting projector codes (CPC), which are a slightly more general scenario than the scenarios considered up to here. Then, topological order is defined from three different approaches. The last part of this section considers erasure noise and guides us on proving that local CPC cannot be ideal, in contrast to classical codes.

The eleventh section introduces thermal noise. It describes a model to characterise thermal noise in a classical and a quantum scenario. Classical thermodynamics is studied using Glauber dynamics and quantum thermodynamics is modelled by the Davies master equation. Then, thermodynamics is related to error correction. We use the Ising model to explain phase transitions, which lead to an interesting property of thermal noise known as self-correction. Self-correction is the ability of a code to create thermal noise and, at the same time, correct against it. At the end of the section, the no-go theorem is stated, which proves that a self-correcting quantum code with dimension less or equal to three is not possible. We also enumerate some strategies to avoid the no-go theorem.

In section twelve, we study the planar codes. Planar codes, as well as the toric code, are in the class of surface codes, but they have non-periodic boundary conditions. We remark the properties that the different boundary condition give to the planar code in comparison of the toric code, which has periodic boundary conditions. After that, a new quantum error correction code is describe: the colour code.

Up to here, we have only corrected errors produced when storing information. However, in quantum error correction errors can also appear when performing gates. Section thirteen addresses this issue introducing the field of fault tolerance, which takes care of performing quantum errors in an efficient way. We start approximating a general unitary operator by one- and two-qubit unitaries in a finite set of gates. Then, some strategies of fault tolerance are studied such as transversal gates, braiding, magic state distillation, kinks and lattice surgery. In this section we also define the Clifford group and see some properties and theorems about it.

1 Classical Error Correction

As the error rate in a classical computer is very small, it may seem that classical error correction is not an important field. It is true that this field is more fundamental in quantum error correction, but classical error correction has nevertheless some interesting applications in fields such as wireless networks, deep space communication an optical storage[1]. In this chapter we will see some basic concepts of the theory of error correction that will be useful during all the lecture. We will first decompose an error correcting code in four parts and study them. Then, the notion of physical and logical bits and error rates will be defined as well as the Hamming distance and the distance of a code. We will focus on linear codes and explain the generator matrix and the parity-check matrix, which are two equivalent representations of linear codes. We will close this chapter mentioning a threshold that every good code satisfies.

Every error correcting code can be broken up into four steps (see Fig. 1):

1. *Source*

[1]For example, most of the improvement of the capacity of a CD to the capacity of a DVD is mainly due to the introduction of a better error code.

The source, which can also be called logical information, is the information that we want to say or transmit.

2. *Encode*
   We want to encode the information that we want to transmit in a larger system in order to protect it.

3. *Noise*
   The noise, which is sometimes also called channel, will corrupt our information. The noise can be of all sorts of different natures.

4. *Decode*

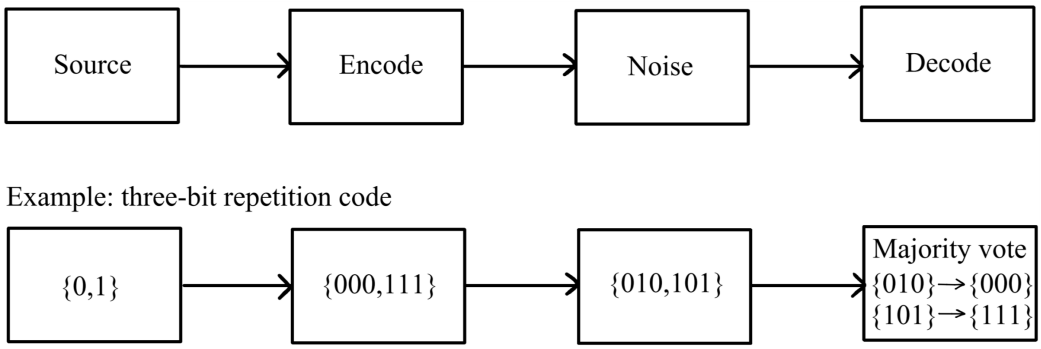

Figure 1: Every error correcting code can be broken up into four steps: source, encode, noise and decode. The three-bit repetition code is the simplest example of error correcting code.

In order to illustrate the decomposition of a code, let us consider the well-known the repetition code as an example

Example 1.1. *The four parts of the three-bit repetition code are (see Fig. 1):*

1. Source
   The simplest logical information consists of a single bit, $\{0, 1\}$.

2. Encode
   The simplest encoding of one bit is to encode it into three bits, i.e., $\{0, 1\} \to \{000, 111\}$. *Note that a bit spans an entire space,* $\mathbb{C}^2$, *while* $\{000, 111\}$ *forms only a subspace.*

3. Noise
We assume that the noise consists of a flip on the middle bit, i.e., it corrupts our information and gives $\{010, 101\}$.

4. Decode
The decoding should map $\{010, 101\}$ *back to* $\{000, 111\}$. *In this case we can do it by majority vote, i.e.,* 010 *is interpreted as* 000 *because it has more zeros than ones, and analogously for* 101.

As we will see later on, this code is denoted as $[3, 1]$.

One can naturally extend the three-bit repetition code to the n-bit repetition code. We obtain a two-dimensional subspace spanned by

$$\{\underbrace{0, \ldots, 0}_{n}, \underbrace{1, \ldots, 1}_{n}\} \in \left(\mathbb{C}^2\right)^{\otimes n}.$$

From these examples, we can see that the fundamental principle of error correction is redundancy. Note that the concept of code, $\mathcal{C}$ refers to the subspace, i.e., in the example we have $\mathcal{C} = \{000, 111\}$. Moreover, the strings that span the code are called codewords.

Classical error correction is a broad field and the goal of this chapter is far from being a complete review of classical error correction. In the following sections we will only cover some elements of classical error correction that will be useful for the chapters about quantum error correction, which is our main focus.

1.1 Physical error rate

The theory of error correction analyses the errors at the level of samples, i.e., individual codewords. In this section we will talk about specific type of errors and codewords. However, we have to keep in mind that the error process acts on the individual codeword in a certain probabilistic way. This means that, when we want study global logical errors, the type of analysis that we have to do is at the level of ensembles, instead of codewords.

Noise can occur in many different ways. For example, the errors caused by the optical fibre through which the information is transmitted will not be the same as the noise occurred while storing the information in a magnetic device or a CD. In general, the noise will depend on the physical support and the type of process we want to perform.

In classical computation there exists only the flip-error, i.e., the error that exchanges 0 and 1. We will assume identically independent distributed (iid) noise on each physical bit. In operational terms, this means that it each bit can individually flip with probability $p < \frac{1}{2}$. Usually, the noise process is going to be a continuous process, but we will break it up into discrete chunks. In every individual chunk, there is a certain probability that a bit is flipped. Note, actually, that the probability p does not represent a single flip, but the union of all odd number of flips because two flips in the same bit ends up in no error.

Definition 1.1. *A logical error is the probability that information is decoded incorrectly.*

Example 1.2. *Consider again the 3-bit repetition code. If we have a probability p to flip every single bit, the probability to flip two bits of $\{000, 111\}$ is $3p^2$. As soon as two bits are flipped, decoding by majority vote does not work anymore because the state 000 with two errors (e.g., 110) will be mapped to 111, and vice-versa.*

From the example above, we can see that, if there are too many flip-errors, the decoding processes will be incorrect, i.e., it will give a global error. Therefore, a logical error can be equivalently described as an error that happens at the end of the process of Fig. 1. Note that the notion of logical error completely depends on the description of the noise process and the choice of decoder.

The numbers of logical and physical bits are denoted by k and n, respectively. We will use the notation that a $[n, k]$ code encodes k logical bits and in n physical bits. For a code to be consider good, we would like to have $\frac{k}{n} \to \text{cnt}$ when $n \to \infty$. In general, k will depend on n.

1.2 Linear codes

There exist different types of classical error codes, but the most useful codes are inside the class of linear codes. In this section we will study this type of codes and see a possible representation called generator matrix.

Consider an n-bit codeword of logical bits, $\{x_1, x_2, \ldots, x_n\}$[2], where $x_j = \{0, 1\}\ \forall\ j$. Our goal is to encode these logical bits, x_j, into a code, i.e., we

[2]Note that $\{x_1, x_2, \ldots, x_n\}$ is not the classical analogy of a vector in a Hilbert space, but only a condensed representation of a specific codeword.

want to map $\{x_1, x_2, \ldots, x_n\}$ into a larger space. For this, we will typically use the so-called generator matrix, G. The generator matrix is an isometry that maps the logical information, x_j, onto the representation of the logical information in the physical space, y_j, i.e., $y_j = Gx_j$.

Example 1.3. *The generator matrix of the* $[3, 1]$ *repetition code is*

$$G = \left.\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}\right\} n.$$

Therefore, when we encode the information of a bit using G*, we get*

$$G[0] = \left.\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}\right\} n \quad \textit{and} \quad G[1] = \left.\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}\right\} n.$$

Note that the arithmetic is mod 2.

Example 1.4. *Consider now the* $[6, 2]$ *repetition code. The generator matrix has two map the following elements*

$$\begin{aligned} \{00\} &\to \{000000\}, \\ \{01\} &\to \{000111\}, \\ \{10\} &\to \{111000\}, \\ \{11\} &\to \{111111\}. \end{aligned}$$

Therefore, we write G *as*

$$G = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

in such a way that

$$G\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \qquad G\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \qquad G\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \qquad G\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

Note that the arithmetic is mod 2.

In general, the generator matrix has k columns and n rows, i.e.,

$$G = \underbrace{\begin{bmatrix} & \\ & \end{bmatrix}}_{k} \Big\} n. \tag{1}$$

From the general form of the generator matrix (Eq. (1)), we can an interesting property of the linear codes. If we have k logical bits, we can encode up to 2^k codewords. We may think that we would need n^k bits that represents the encoding, but the representation of linear codes are extremely efficient because, instead of using n^k bits to represent the codespace, we only use nk bits. On top of that, the encoding procedure is efficient as it only consists of matrix multiplication. Therefore, the generator matrix is extremely convenient to describe the encoding part of the process in Fig. 1. Nevertheless, it does not tell anything about decoding. We will see later on that classical linear codes have always a natural way of decoding[3], but before we need to introduce a different representation for linear codes.

1.3 Parity-check matrix

We have seen in the previous section that linear codes can be represented using the generator matrix. This is not the only possible representation. In this section we will introduce the parity-check matrix, which is an equivalent representation that can be more useful in certain situations.

The parity-check matrix, H, is representation for linear codes that consists of a $(n-k) \times n$ matrix such that

$$Hy = 0 \ \forall \ y \in \mathcal{C}, \tag{2}$$

where $\mathcal{C}$ is the codespace, i.e., the n-bit space. Therefore, the codespace is the kernel of H according to Eq. (2). The rows of H are linearly independent, while columns are linearly dependent.

Example 1.5. *The parity-check matrix of the* $[n, 1]$ *repetition code is*

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 1 & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \\ 0 & \cdots & 0 & 1 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 & 1 \end{pmatrix}$$

[3]This will not be true for quantum codes

Consider that we initially have the codeword $y_0 = \{0, \ldots, 0\}$ *and it occurs an error on the third bit,* $e = \{0, 0, 1, 0, \ldots, 0\}$*. Then, the parity-check matrix will detect the error as*

$$Hy_0 = 0$$

$$He = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Note that the capacity of H to detect errors relies on the fact that it is completely insensitive to the codewords by definition. Thus, it only picks up where the errors are.

There exists an equivalent representation of the parity-check matrix which is called the Tanner graph (see Fig. 2). The Tanner graph consists on lines of boxes where the upper line represents the bits and the lower line shows the parity of two neighbouring bits. If there is an error on the upper line, the boxes of the lower line connected to the box that contains the error will be activated. These "activations" are called error syndromes. They give information about where the errors are in the code (see Fig. 2), and thus they are crucial for decoding.

1.4 Decoding

Once the information we want to transmit has been encoded and corrupted, the work of decoding is to "remove errors" in an intelligent way using the syndrome information of the corrupted codeword. In this section we will see that linear codes have a natural way of decoding.

The first fact that it is important to note is that, if all zeros are flipped to ones and all ones are flipped to zeros, we get exactly the same syndrome information. The syndromes do not care about the original codeword. Thus, the decoding procedure should not depend on the codeword, but only on the error syndromes.

Consider that {0000000} is the initial codeword and that the information has been corrupted and we have six syndrome bits (see Fig. 3). We have absolutely no way of knowing whether to correct in one direction or the opposite

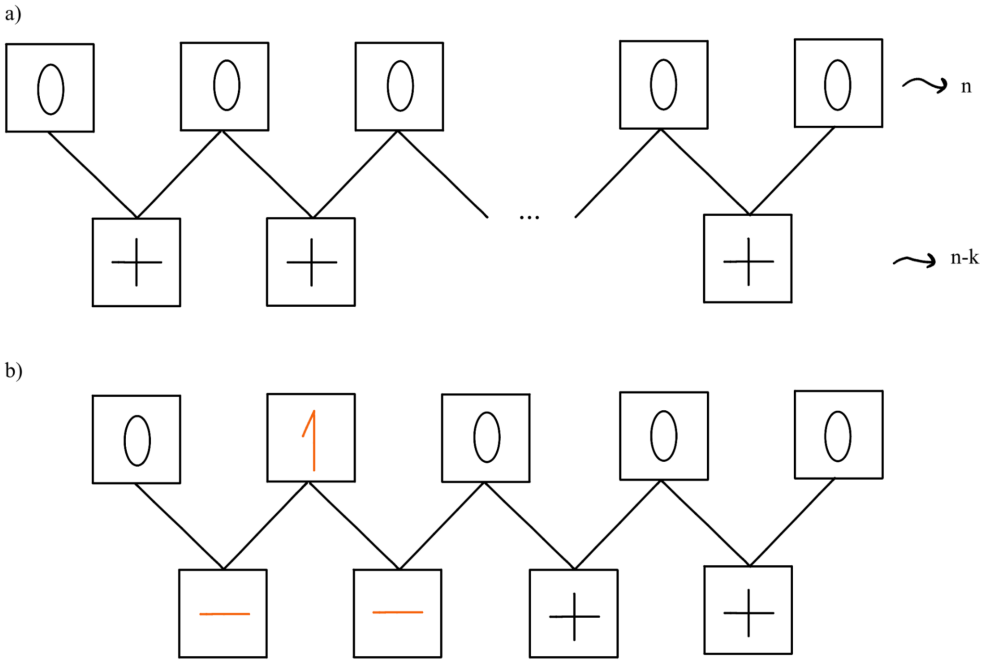

Figure 2: Tanner graph of a) the n-bit codeword $\{0, \ldots, 0\}$, and b) the five-bit codeword $\{00000\}$ with a flip error on the second bit.

because there are two possible parents of errors. The first one corresponds to the situation that three bits of the initial codeword have been flipped. However, an equivalent parent of errors is the one that hit the conjugate bits, and thus there have been four flip-errors. For these two situations, we would get exactly the same syndrome information. The decoder has to make a choice to correct into one direction or the other. The typical solution is to choose the most likely outcome. It is most probable to have three errors than to have four errors if a bit has on average an error with probability $p < \frac{1}{2}$. Obviously, every once on a while the decoder will make a mistake, and thus the information we will get is not the same information that was sent.

Example 1.6. *Consider the* $[3, 1]$ *repetition code and an initial string* $\{000\}$. *If the error probability of each individual bit is* $p < \frac{1}{2}$, *the probability of the initial string,* $\{000\}$, *to become a different codeword is the follwoing*

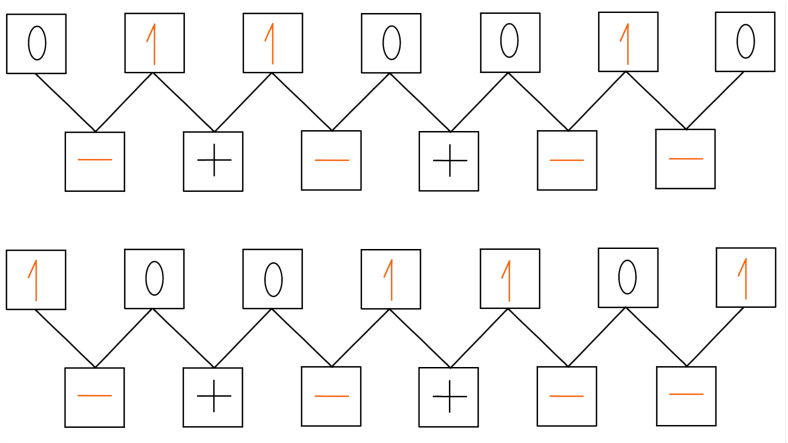

Figure 3: An error syndrome has two parents of errors. Given the initial codeword $\{0000000\}$, a flip-error on the second, third and sixth bit gives the same error syndrome than a flip-error on the first, forth, fifth and seventh bit. However, the situation with only three errors is more likely.

Codeword	*Probability*
$\{000\}$	$(1-p)^3$
$\{001\}$	$(1-p)^2p$
$\{010\}$	$(1-p)^2p$
$\{100\}$	$(1-p)^2p$
$\{011\}$	$(1-p)p^2$
$\{110\}$	$(1-p)p^2$
$\{101\}$	$(1-p)p^2$
$\{111\}$	p^3

As $p < \frac{1}{2}$, the probability of $\{000\}$ having no error is much bigger than the probability of having three errors, and thus becoming $\{111\}$.

As we are assuming iid errors, the decoder will always make the choice of the situation with the fewest number of errors. Note that this does not work if the errors are correlated.

1.5 Distance of a code

An important characteristic of an error correcting code is its robustness towards noise. In this section we define the distance of a code, which will give an idea of how robust a code is. For that, we first need the definition of the Hamming distance.

Definition 1.2 (Hamming distance)**.** *Given two codewords, y_1 and y_2, the Hamming distance, $d(y_1, y_2)$ is the minimum number of bits that must be flipped to transform y_1 into y_2.*

Example 1.7. *The distance between $y_1 = \{1100\}$ and $y_2 = \{1010\}$ is $d = 2$.*

Once we know what the Hamming distance is, we can define the distance of a code.

Definition 1.3 (Distance of a code)**.** *The distance of a code $\mathcal{C}$ is the minimal Hamming distance between to different codewords y_i and y_j, i.e.,*

$$d(\mathcal{C}) \equiv \inf_{\substack{y_i, y_j \\ y_i \neq y_j}} d(y_i, y_j).$$

The distance of a code gives an idea of how resilient the code is. However, in order to get the full idea, we should consider distributions and entropic factors. It is also worth noting that any $\lfloor \frac{d-1}{2} \rfloor$ errors of a linear code can be corrected. Actually, the distance of a code is such an important quantity that codes are usually identified with $[n, k, d]$, where d is the distance of the code and, as mentioned before, n and k are the number of physical and logical bits, respectively.

The goal of information theory is to understand the limits on the amount of information that can be transmitted through a channel. Information theory was developed in 1950, but the first codes that achieved maximal transmission of information through a channel were proposed only fifteen years ago. These codes are called constant-rate codes and they fulfil that

$$\frac{k}{n} \xrightarrow[n\to\infty]{} \text{cnt},$$

$$\frac{d}{n} \xrightarrow[n\to\infty]{} \text{cnt}.$$

The fact that both limits go to a constant means that, as n becomes higher and higher, we need to waste fewer and fewer physical bits in order to robustly encode an amount of information proportional to the amount of physical information. These codes exist in classical error correction, but not in quantum error correction.

The parameters n, k and d of a code are not completely free, i.e., there exist constraints on them such as[4]

[4]In the exercise class we will prove the last constraint and show some more.

- $n \geq k$
- $n \geq d$
- $n - k \geq d - 1$

1.6 Thresholds

In the last section we have weekly suggested the idea that, if the distance of the code is large, the code is robust. Here we will see that, on top of that, a code is considered a good code if it fulfils the threshold.

[Thereshold for a good code] *Given a code, $\mathcal{C}$, with n physical bits and a physical error rate p, it is considered a good code if there exists a probability threshold, $p_{th} \leq \frac{1}{2}$, such that the logical error rate, P_{log}, satisfies*

$$P_{\log}(n, p) \leq ce^{-\alpha d} \quad \forall \ p \leq p_{th}. \tag{3}$$

Here it is assumed that d scales with n.

As this threshold is a strong statement, the exponential decay is sometimes relaxed by only requiring that $P_{\log}(n, p)$ decays as a function $f(n)$ such that $f(n) \to 0$ when $n \to \infty$. On the contrary, the threshold error rate decays even faster for some codes. For example, the $[n, 1]$ repetition code has an error correction threshold of $p_{th} = \frac{1}{2}$, which is the highest possible[5]. Obviously, this is not the general case.

Example 1.8 (The Hamming code)**.** *The Hamming codes are a family of linear codes with $[2^r, 2^r - r, 3]$, where r is an integer such that $r \geq 1$. They are perfect codes, that is, they achieve the highest possible rate $\frac{k}{n}$ for codes with minimum distance of three. The party-check matrix of the Hamming code with $r = 3$ is*

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

Note that the rows are linearly independent, but not the columns. In this case, we have $n = 7$, $k = 4$ and $d = 3$, i.e., it is a $[7, 4, 3]$ code. In figure 4 we can see the Tanner graph of the Hamming code with $r = 3$. From this figure it is obvious that many errors will have the same error syndrome, and thus it will be difficult to know where the error is.

[5] We show that in the exercise class

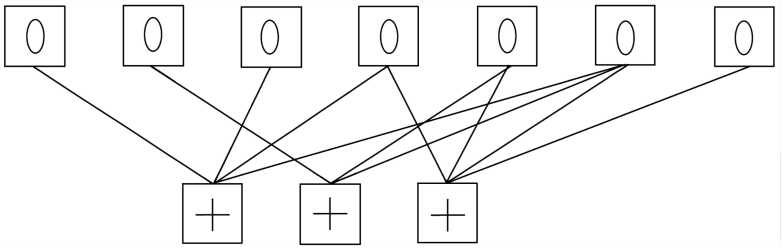

Figure 4: Tanner graph of the codeword $\{0000000\}$ of the Hamming code with $r = 3$.

2 Quantum mechanics of one qubit

Instead of directly delving into quantum error correction (QEC), in the previous section we have seen some elements of classical error correction. This section is also devoted to concepts that are needed before studying QED. Here we review the basics of classical and quantum information. We start explaining the concept of a classical bit and describing the two types of states of a classical system. Then, single-bit operation as well as gates are considered. Some examples are also given. When we move to quantum information, we introduce the qubit and emphasise its representation on the Bloch sphere. Operations in quantum mechanics are described. In particular, we pay attention to unitary operations and projective measurements. The Kraus decomposition is also introduced due to its interpretation in terms of error correction. We then differentiate between the two types of randomness that exist in quantum information, which is an important difference to classical information. After that, noise is characterised using the concept of quantum operations. The last explanation of this section is about the potential issues that we need to overcome in quantum information.

2.1 Classical information

In classical information, the fundamental unit of information is the bit, i.e., $\{0, 1\} \in \mathbb{Z}_2$. The physical state of the system can be:

- a certain state, i.e., $|0\rangle\langle 0|$ or $|1\rangle\langle 1|$
- an uncertain state, i.e., $q|0\rangle\langle 0| + (1-q)|1\rangle\langle 1|$ where $q \in \mathbb{R}$ with $0 < q \leq 1$. The system being in an uncertain state means that there exists a probability q to find the system in state $|0\rangle\langle 0|$ and a probability $(1-q)$ that it is in state $|1\rangle\langle 1|$. Therfore, the uncertainty reflects our knowledge of the system.

The only single-bit operation[6] in classical information is the bit-flip, which consists in

$$\begin{array}{ccc} 0 & \rightarrow & 1 \\ 1 & \rightarrow & 0 \end{array}$$

Noise in classical information will typically take the system from a certain state to an uncertain state.

Example 2.1. *Consider a noise consisting of a flip with probability $p < 1$, then the state of the system will undergo the following changes*

$$\begin{array}{ccc} |0\rangle\langle 0| & \rightarrow & p|1\rangle\langle 1| + (1-p)|0\rangle\langle 0| \\ |1\rangle\langle 1| & \rightarrow & p|0\rangle\langle 0| + (1-p)|1\rangle\langle 1| \end{array}$$

Note that an operation can be interpreted as the limit case of a noise where $p = 1$.

Computation is the process of taking several bits and mapping to them in a certain way. In other words, computation consist of operations acting on more than one bit. These operations are also known as gates

Example 2.2. *An example of a two-bit gate in classical information is the so-called NAND, which consists of*

$$\begin{array}{cc} 00 & 01 \\ 01 & 01 \\ 10 & 01 \\ 11 & 00 \end{array}$$

This gate is important in classical computation because it is a universal gate, i.e, once we are able to perform it, we can perform any other gate.

It is worth mentioning that in the formulation of computation we always represent operations going from a certain state to a certain state. However, in practise, we will always have an uncertain state, which will be mapped to another uncertain state.

2.2 Quantum information with one qubit

In this section, we introduce the basics of quantum information. The characterisation of a quantum system is first explained as well as how to operate on

[6]When we talk about operations, we always think about their action on certain states.

it. Then, we will explain the concepts of noise and randomness in quantum information emphasising the difference to classical information. Finally, the potential issues that have to be overcome to do quantum error correction are enumerated.

State of the quantum system

In quantum information the state of the system is a quantum state, i.e., a normalised vector of a two-dimensional Hilbert space, $\mathcal{H}_2$. Thus, we write $|\varphi\rangle \in \mathcal{H}_2$ such that $\langle\varphi|\varphi\rangle = 1$.

The typical physical basis of quantum information is the so-called computational basis, which consists of $\{|0\rangle, |1\rangle\}$. We can always write the state of the system as a linear combination of the physical basis such that

$$|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle, \text{ where } \alpha, \beta \in \mathbb{C} \text{ with } |\alpha|^2 + |\beta|^2 = 1.$$

As a global phase is not relevant in physics, we can choose α to be real and non-negative. This fact, together with $|\alpha|^2 + |\beta|^2 = 1$, allows to write the two-qubit state as

$$|\varphi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle,$$

where $0 \leq \theta \leq \pi$ and $0 \leq \phi \leq 2\pi$. The parameter θ and ϕ can be interpreted as spherical coordinates giving rise to a unit sphere in $\mathbb{R}^3$ known as Bloch sphere (see Fig. 5). Each point of the Bloch sphere, which can be characterised by the unit vector $\vec{n} \equiv (\sin\theta\cos\phi, \sin\theta\cos\phi, \cos\theta)$, specifies a two-qubit state. Note that two antipodal points of the Bloch sphere correspond to two orthogonal states.

Mixed states can also be represented using the Bloch sphere. Any two-dimensional density operator, ρ, can be written as

$$\rho = \frac{1}{2}\left(\mathbb{I} + \vec{a}\cdot\vec{\sigma}\right),$$

where $\mathbb{I}$ is the identity matrix, $\vec{a} = (a_x, a_y, a_z) \in \mathbb{R}^3$ and $\vec{\sigma} \equiv (\sigma_x, \sigma_y, \sigma_z)$ is a vector made of the Pauli matrices with

$$X \equiv \sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y \equiv i\sigma_y = i\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z \equiv \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \quad (4)$$

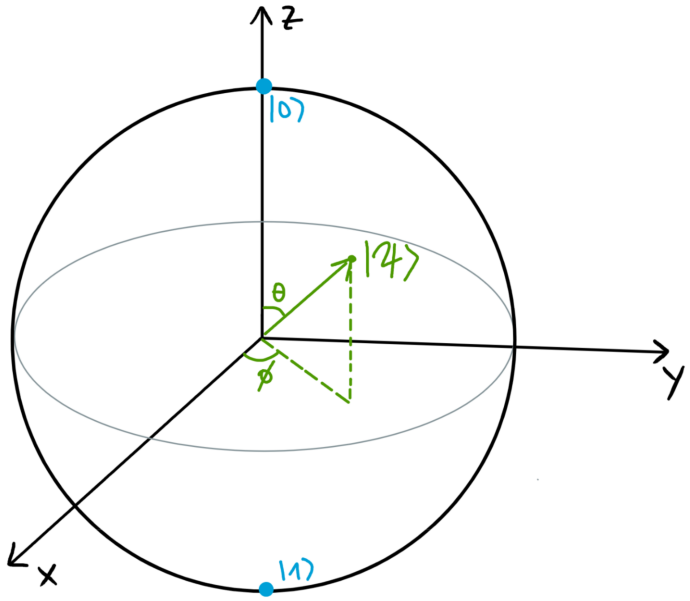

Figure 5: Bloch sphere

Due to normalisation of the density matrix, it is easily proven that $|\vec{a}|^2 \leq 1$ and $|\vec{a}|^2 = 1$ if and only if the density matrix is a pure state[7]. In other words, pure states lie on the surface of the Bloch sphere, while mixed state correspond to point in the interior.

Quantum operations

A system can undergo many different physical transformations. They are known as operations and represented by a map $\mathcal{E} : \mathcal{B}(\mathcal{H}_A) \to \mathcal{B}(\mathcal{H}_B)$ with the following properties. The map must be

(i) Linear, i.e., $\mathcal{E}\left[\sum_i p_i \rho_i\right] = \sum_i p_i \mathcal{E}(\rho_i)$,

(ii) Positive semidefinite, i.e., $\mathcal{E}(\rho) \geq 0 \ \ \forall \rho \geq 0$,

(iii) Completely positive, i.e., $(\mathcal{E}_A \otimes \mathbb{I}_C)\left[\rho_{AC}\right] \geq 0 \ \ \forall \rho_{AC} \geq 0$ and any Hilbert space $\mathcal{H}_C$, where $\rho_{AC} \in \mathcal{B}(\mathcal{H}_A \otimes \mathcal{H}_C)$.[8]

Note that (iii) implies (iii). The first two properties guarantee that the output of a quantum operation on a physical state is a physical state as well,

[7]Recall the density matrix of a pure state, $|\psi\rangle$, is $\rho = |\psi\rangle\langle\psi|$.

[8]In these notes we do not consider Hilbert spaces with infinite dimension.

while the third one ensures the state will still be physical even if the quantum operation applies only on a subsystem. In summary, a quantum operation is a completely positive (CP) map that describes the transformation of a physical system.

A particular class of quantum operations are unitary transformations. A unitary transformation is a map, U, such that $U|\varphi\rangle = |\psi\rangle$, where $UU^\dagger = \mathbb{I}$. It is easily proven that any unitary map U can be written as e^{iH} with H an hermitian operator, i.e., $H = H^\dagger$.

In quantum information, measurements are another important class of quantum operations. Measurements are observables, which implies that they are represented by hermitian operators. The simplest kind of measurements are the so-called projective measurements. A projective measurement, M, can be written as

$$M = \sum_k \nu_k P_k,$$

where P_k are projectors, i.e., $P_k^2 = P_k$ and $\nu_k = \pm 1$. Given an initial state $|\varphi\rangle$, the probability of obtaining the result m after the measurement M on $|\varphi\rangle$ is $p_m = \langle\varphi|P_m|\varphi\rangle$. The state of the system after the measurement is

$$|\varphi'\rangle = \frac{P_m|\varphi\rangle}{\sqrt{\langle\varphi|P_m|\varphi\rangle}}$$

Example 2.3. *In order to measure if the qubit is in the state $|0\rangle\langle 0|$ or in $|1\rangle\langle 1|$, we use the operaation $M = Z = |0\rangle\langle 0| - |1\rangle\langle 1| = P_0 + (-1)P_1$.*

Example 2.4. *Consider the measurement*

$$M = X = |+\rangle\langle +| - |-\rangle\langle -|, \text{ where } |\pm\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle \pm |1\rangle\right).$$

The probability of obtaining the result $\pm$ after measuring $M = X$ on a state $|\varphi\rangle$ is $p_\pm = |\langle\pm|\varphi\rangle|^2$.

Any quantum operation can be written as

$$T(\rho) = \sum_k E_k \rho E_k^\dagger,$$

where E_k are maps such that $\sum_k E_k^\dagger E_k = \mathbb{I}$. This decomposition is known as Kraus decomposition and the operators E_k are called Kraus operators. Due to linearity of the trace, it is easy to see that the Kraus decomposition guarantees the preservation of the trace. The Kraus decomposition can be easily interpreted in terms of error correction. Consider a state $\rho = |\varphi\rangle\langle\varphi|$, then error operator E_k occurs with probability $p_k = ||E_k|\varphi\rangle||^2$. For this reason, Kraus operators are also known as noise operators.

Another useful representation of operations in $\mathcal{H}_2$ consists in writing an operation, Ω, in the basis $\{\mathbb{I}, X, Y, Z\}$, i.e.,

$$\Omega = a_1\mathbb{I} + a_x X + a_y Y + a_z Z. \tag{5}$$

Randomness in quantum information

One of the most important difference between quantum and classical mechanics is the origin of randomness. Randomness in classical physics has to do with the ignorance about the system, while in quantum mechanics it has two forms:

1. *Uncertainty*
   When our knowledge of the system is limited, it is described by a mixed state because we do not know exactly the state of the system. For example, if the system is in the state

   $$\rho = \lambda_0|0\rangle\langle 0| + (1-\lambda_0)|1\rangle\langle 1|,$$

   we know that it is in state $|0\rangle\langle 0|$ with probability $p = \lambda_0$ and in state $|1\rangle\langle 1|$ with probability $q = 1 - \lambda_0$. This uncertainty introduces randomness which has its origin in lack of information. It is the same randomness that exists in classical information.

2. *Intrinsic*
   In quantum mechanics, even if we know exactly the state of the system, there is room for randomness. Consider a system in the state $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $|\alpha|^2 + |\beta|^2 = 1$. As we have said before, if we measure $|\varphi\rangle$, there is a certain probability that we get the outcome 0 and a certain probability for outcome 1. This introduces randomness in the system which comes from intrinsic properties of its state.

Note that if the state of a system is mixed, both kind of randomness can appear. Wdescarhen we perform a measurement, it is not always obvious to know which kind of randomness we are facing.

Noise in quantum information

In quantum information, noise is a general operation (i.e., anything that is physically allowed) between two quantum states. On qubits, this means an operation, T, that takes the system from a density matrix, ρ, to another density matrix, σ, i.e., $T(\rho) = \sigma$. In order T to be a physical operation, it must fulfil the following properties. Given a density matrix X, T has to be

- trace-preserving, i.e.,
$$\mathrm{tr}\,[T(X)] = \mathrm{tr}(X)$$
- complete-positivity preserving in such a way that the state modified by the noise remains as a physical state.

Example 2.5. *Consider a noise, T, consisting of a flip with probability $p \geq 0$. In other words, with probability $(1-p)$ the initial state, ρ, remains unchanged an with probability p one of its bits is flipped. The resulting state is*
$$T(\rho) = (1-p)\rho + pX\rho X,$$
where $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$.

Potential issues of quantum information

Recall that the simplest classical EC code is the three-qubit repetition code, where the correction is done by majority vote (see Example 1.1). In quantum mechanics, we would like to have an analogous code, but there exist some potential issues that we have to overcome. We have to face with

- *The no-cloning principle*
  It is well-known that in quantum mechanics there cannot exist a general operation that realises $|\varphi\rangle \to |\varphi\rangle|\varphi\rangle|\varphi\rangle$.
- *The collapse of the state*
  In order to correct the errors of a state, we need to measure each qubit. In quantum mechanics, however, measurements collapse the state of the system, and thus they may change it.

- *Continuous errors*
  We have previously seen that in classical information there exist only flip-bit errors. Nevertheless, in quantum mechanics there are more types of errors. Some of these errors are continuous, i.e., they are described by a continuous parameter. For example, a state could suffer a small rotation such that

$$|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle \rightarrow |\varphi'\rangle = \alpha|0\rangle + e^{i\phi}\beta|1\rangle,$$

  where $0 \leq \phi \leq 2\pi$.

3 The Shor code

The Shor code is a quantum error correction code that is able to protect against phase and bit errors. In this section, we first learn to correct bit errors and phase errors independently, and then we concatenate both codes to built the Shor code. Here, the Shor code is studied on nine qubits, but its generalisation to n qubits is straightforward. The Shor code can be interpreted as two classical repetition codes in two different levels. As we see below, the first level acts on individual qubits and corrects against bit errors, while the second l evel considers groups of three qubits in order to correct against phase errors.

The logical qubits of the Shor code are

$$|\bar{0}\rangle = \frac{1}{\sqrt{2^3}}\left(|000\rangle + |111\rangle\right)\left(|000\rangle + |111\rangle\right)\left(|000\rangle + |111\rangle\right), \tag{6}$$

$$|\bar{1}\rangle = \frac{1}{\sqrt{2^3}}\left(|000\rangle - |111\rangle\right)\left(|000\rangle - |111\rangle\right)\left(|000\rangle - |111\rangle\right). \tag{7}$$

It is easy to see that the structure of the logical qubits is three chunks of three qubits. If we focus on a single chunk, we can interpret it as a logical $\pm$ of the classical repetition code, i.e.,

$$|\bar{\pm}\rangle = \frac{1}{\sqrt{2}}\left(|000\rangle \pm |111\rangle\right), \tag{8}$$

where the states $|000\rangle$ and $|111\rangle$ play the role of the logical bits of a classical repetition code. This allows to write the logical operators of the Shor code as

$$|\bar{0}\rangle = \frac{1}{\sqrt{2^3}}\left(|000\rangle + |111\rangle\right)^{\otimes 3} = |\bar{+}_{(1)}\rangle^{\otimes 3}$$

$$|\bar{1}\rangle = \frac{1}{\sqrt{2^3}}\left(|000\rangle - |111\rangle\right)^{\otimes 3} = |\bar{-}_{(1)}\rangle^{\otimes 3}$$

Note now that $|\bar{0}\rangle$ can be interpreted at the same time as logical operators of another classical repetition code. In this second level of correction, we are able to against phase flips[9]. Therfore, the Shor code is the simplest example of a concatenated code where at the first level it protects against bit errors and at the second level it protects against phase errors.

How do we protect against bit errors and phase errors?

As we have seen in the classical repetition code, the decoding process uses the majority vote. Nevertheless, in quantum error codes we cannot decode using this strategy because the action of measuring the qubits to see which state predominates collapses the system in a post-measurement state. Instead of the majority vote, we quantun error correction decodes using parity measurements because they do not affect the logical information. Classicaly we have already seen the parity measures with the parity-check matrix (see Section 1.3) and the Tanner graph (see Section 1.4). A parity measurement measures if two consecutive qubits are in the state. If they are in the same state, we associate to the result of the measurement a "+" sign and say that we have "even parity". On contrary, if the state of the qubits is different, we associate to the result of the measurement a "-" sign and say that we have "odd parity".

In order to understand the decoding based on parity measurments, let us first consider a bit error and we assume that it happens in the first qubit. The parity measurements for bit errors are Z_1Z_2 and Z_2Z_3. We can write Z_1Z_2 in terms of projectors such that

$$\begin{aligned} Z_1Z_2 &= (|00\rangle\langle 00| + |11\rangle\langle 11|) - (|01\rangle\langle 01| + |10\rangle\langle 10|) \\ &= P_+ - P_-, \end{aligned}$$

where $P_+ \equiv |00\rangle\langle 00| + |11\rangle\langle 11|$ and $P_- \equiv |01\rangle\langle 01| + |10\rangle\langle 10|$ are the projectors on the even-parity space and odd-parity space, respectively. Consider that

[9]Recall that

$$\begin{array}{ll} X|0\rangle = |1\rangle & \\ X|1\rangle = |0\rangle & \end{array} \quad \text{and} \quad \begin{array}{ll} Z|+\rangle = |-\rangle & \\ Z|-\rangle = |+\rangle & . \end{array}$$

This means that, in order to perform the equivalence of the repetition code for phases, we have to do it in the basis made of $\{|\pm\rangle\}$.

the initial state, $|\psi\rangle$, gets corrupted by X_1, and thus we have

$$|\psi_{X_1}\rangle \equiv X_1|\psi\rangle = \frac{1}{\sqrt{2}}|100\rangle + |011\rangle.$$

The outcomes of measuring Z_1Z_2 are

- Even parity with probability $\langle\psi_{X_1}|P_+|\psi_{X_1}\rangle = 0$,
- Odd parity with probability $\langle\psi_{X_1}|P_-|\psi_{X_1}\rangle = 1$.

The state of the system after the parity measurement is

$$\frac{P_-|\psi_{X_1}\rangle}{\langle\psi_{X_1}|P_-|\psi_{X_1}\rangle} = P_-|\psi_{X_1}\rangle = |\psi_{X_1}\rangle$$

Thus, we have measured Z_1Z_2 on qubits one and two, we have obtained with certainty that they have odd parity and, in particular, the measurement has not changed the state. The next step is to measure the other parity measurement, Z_2Z_3. It is easy to check that in this case we obtain that qubits two and three are in strict even parity. The combination of both results allows to localise the error without changing the state. Now, we can simply apply X_1 to the corrupted state, $|\psi_{X_1}\rangle$, and we recover the initial state, i.e.,

$$X_1|\psi_{X_1}\rangle = X_1X_1|\psi\rangle = |\psi\rangle.$$

Doing the same procedure for all bit errors, we obtain the following recipe, which links the results of the parity measurements with the operation that we have to do to restore the corrupted state. The recipe for bit errors is

Result of Z_1Z_2	Result of Z_2Z_3	Restoring operation
+	+	$\mathbb{I}$
-	+	X_1
+	-	X_3
-	-	X_2

Note that this recipe is only valid if there is only one bit error.

In order to correct phase errors, we can use the same method as for bit errors, but we have to work on the basis made of $\{|\pm\rangle\}$. Consider that the initial state is $|\phi\rangle = |++\!+\rangle$ and that it has been corrupted by Z_2, i.e., we have $|\phi_{Z_2}\rangle = Z_2|\phi\rangle = |+\!-\!+\rangle$. The parity measurements of phase errors are X_1X_2 and X_2X_3. The operator X_1X_2 can be written as

$$\begin{aligned} X_1X_2 &= (|++\rangle\langle++| + |--\rangle\langle--|) - (|+-\rangle\langle+-| + |-+\rangle\langle-+|) \\ &= Q_+ - Q_-, \end{aligned}$$

where $Q_+ = |++\rangle\langle++| + |--\rangle\langle--|$ and $Q_- = |+-\rangle\langle+-| + |-+\rangle\langle-+|$ are the projectors on the even-parity space and odd-parity space of the X operator, respectively. The outcomes of measuring X_1X_2 are

- Even parity with probability $\langle\phi_{Z_2}|Q_+|\phi_{Z_2}\rangle = 0$.
- Odd parity with probability $\langle\phi_{Z_2}|Q_-|\phi_{Z_2}\rangle = 1$.

If we now measure X_2X_3, we get that the qubits are in strictly odd parity. Thus, we have localised the phase error on the second bit and we can corrected applying a Z_2 on $|\varphi_{Z_2}\rangle$. As for bit errors, we can proceed analogously for all phase-flips and construct the following recipe

Result of X_1X_2	Result of X_2X_3	Restoring operation
+	+	$\mathbb{I}$
-	+	Z_1
+	-	Z_3
-	-	Z_2

We have seen that with three qubits we are able to correct against bit or phase errors, but we cannot correct both at the same time because the restoring operations do not commute. The Shor code, however, solves this problem by using two levels of correction instead of one. For this purpose, it considers a total of nine qubits and, when correcting phase errors, it considers groups of three qubits instead of individual qubits. In other words, the Shor code uses the states $|\bar{\pm}\rangle$, i.e., logical qubits made of three qubits in such a way that we work at the second level of correction. The parity measurements become, then, $(X_1X_2X_3)(X_4X_5X_6)$ and $(X_4X_5X_6)(X_7X_8X_9)$. Note that $X_1X_2X_3$, $X_4X_5X_6$ and $X_7X_8X_9$ play respectively the role of $X_1^{(1)}$, $X_2^{(1)}$ and $X_3^{(1)}$ at the first level. Note further that $(X_1X_2X_3)(X_4X_5X_6)$ and $(X_4X_5X_6)(X_7X_8X_9)$ have eigenvalues ± 1, and thus they measure parity, but in the X basis of groups of three qubits. In order to see that, consider an initial state $|\varphi\rangle = |\bar{0}\rangle$. The state is corrupted with a bit-flip and a phase-flip on qubit one, i.e., we have

$$|\varphi'\rangle \equiv Z_1X_1|\varphi\rangle = (-|100\rangle + |011\rangle)\,|\bar{+}\rangle|\bar{+}\rangle.$$

After measuring Z_1Z_2 and Z_2Z_3 and using the recipe, we detect that the corrupted state has a bit-flip on the first qubit and we apply X_1 to correct it. We obtain

$$X_1|\varphi'\rangle = X_1Z_1X_1|\varphi\rangle = -Z_1|\varphi\rangle = (-|000\rangle + |111\rangle)|\bar{+}\rangle|\bar{+}\rangle.$$

Now, we measure the parity operators $X_1X_2X_3X_4X_5X_6$ and $X_4X_5X_6X_7X_8X_9$ and the results show that there is a phase-flip on the first logical qubit. We

can correct it by simply applying Z_1, Z_2 or Z_3 on the state. We apply, for example, Z_1 and obtain

$$Z_1(-Z_1|\varphi\rangle) = -|\varphi\rangle.$$

As global phases have no physical meaning, we have been able to correct both, a bit and a phase error. Note that this analysis also shows that errors given by Y can also be corrected due to $Y = iZX$. It is easy to see that the process is valid independently of which qubit the error acts on. In conclusion, any single-qubit phase or bit error can be corrected, i.e., there exists a correction operation that takes

$$X_i Z_j|\varphi\rangle \to e^{i\alpha}|\varphi\rangle \qquad \forall i,j \in [1,9] \text{ and } \forall|\varphi\rangle \in \mathcal{C}.$$

So far we have only considered pure errors, i.e., errors given by X, Y or Z. We now want to show that the Shor code can also correct linear combinations of $\{X_i, Y_j, Z_k\}_{i,j,k=1}^{9}$. Consider an error operator, E, given by

$$E = e_x X_1 + e_z Z_1,$$

where for simplicity we do not consider a Y operator. The initial state is $|\varphi\rangle = |\bar{0}\rangle$, and thus the corrupted state is $E|\varphi\rangle$. If we measure[10] Z_1Z_2, we obtain "even parity" with probability

$$\begin{aligned}\langle\varphi|E^\dagger P_+ E|\varphi\rangle &= \langle\bar{0}|(e_x^* X_1 + e_z^* Z_1)P_+(e_x X_1 + e_z Z_1)|\bar{0}\rangle \\ &= |e_x|^2\langle\bar{0}|X_1P_+X_1|\bar{0}\rangle + e_x^* e_z\langle\bar{0}|X_1P_+Z_1|\bar{0}\rangle + \\ &\qquad + e_x e_z^*\langle\bar{0}|Z_1P_+X_1|\bar{0}\rangle + |e_z|^2\langle\bar{0}|Z_1P_+Z_1|\bar{0}\rangle \\ &= |e_z|^2.\end{aligned}$$

The probability of "odd parity" is $\langle\varphi|E^\dagger P_- E|\varphi\rangle = |e_x|^2$. Assume without loss of generality that the measurement yields "even parity", then we know that the post-measurement state is

$$\frac{1}{|e_z|}P_+E|\varphi\rangle = \frac{e_z}{|e_z|}Z_1|\bar{0}\rangle,$$

where $\frac{e_z}{|e_z|}$ is a phase. Thus, the parity measurement has removed the bit error and we are left with a clean phase flip on $|\bar{0}\rangle$. Doing this analysis for all the parity measurements, we see that that the post-measurement state

[10]Here we know where the error is, and thus we only measure one parity measurement. In practise, however, one must measure all and then use the recipe.

is always a state of the set $\{|\varphi\rangle, X_j|\varphi\rangle, Z_j|\varphi\rangle, X_jZ_j|\varphi\rangle\}$. Note that we know how to correct all states of the set. Note further that the collapse of the state after a measurement is crucial to be able to correct errors. In conclusion, if the state has suffered an error which is a linear combination of errors that we know how to correct, we are able to correct it exactly.

We have just observed that the Shor code can correct against error given by any linear combination of $\{\mathbb{I}, X, Y, Z\}$. Moreover, in the previous chapter, we have seen that any operator can be written in the basis $\{\mathbb{I}, X, Y, Z\}$ (Eq. (5)). This implies that Shor code is able to correct against any arbitrary single-qubit error.

As we have mentioned at the beginning of the chapter, the Shor code on n^2 qubits is a concatenation of two $[n^2, 1, n^2]$ classical repetition codes. The error correction threshold is the same as in the classical case, and thus it becomes interesting when n is big (see Section 1.6). In practise, however, the Shor code is not use when n is large because of the following reason. The parity measurement at first level of the Shor code on n^2 are[11]

$$\begin{gathered} Z_1Z_2, Z_2Z_3, \ldots, Z_{n-1}Z_n, \\ Z_{n+1}Z_{n+2}, \ldots, Z_{2n-1}Z_{2n}, \\ \vdots \\ Z_{n^2-n+1}Z_{n^2-n+2}, \ldots, Z_{n^2-1}Z_{n^2}, \\ X_1 \cdots X_n, \ldots, X_n \cdots X_{2n}. \end{gathered}$$

Note that the parity measurements for phase errors imply to measure n qubits at the same time. This is a problem because nowadays we are able to apply at most three-qubit operations. Beyond this, measurements are too noisy. Therefore, the Shor code is not practical.

4 Quantum error correction conditions

Knill and Laflamme gave conditions for a subspace to be a code space. In this section, we want to review them and analyse an important consequence.

Given a Hilbert space[12], $\mathcal{H}$, a quantum error correcting code is is a subspace, $\mathcal{C}_n \in \mathcal{H}_2^{\otimes n}$, that protects against a quantum channel, $\mathcal{E}(\rho) = \sum_k E_k \rho E_k^\dagger$. In

[11] We will see in following chapters these operators are known as stabilizer operators.

[12] For convenience, we consider that the Hilbert space, $\mathcal{H}$, is embedded in a Hilbert space that characterises n qubits, $\mathcal{H}_2 \otimes \cdots \otimes \mathcal{H}_2 = \mathcal{H}_2^{\otimes n}$.

other words, if the error $\mathcal{E}$ happens on the system, there exists a recovery channel, $\mathcal{R}$, such that

$$\mathcal{R} \circ \mathcal{E}(\rho) = \rho \qquad \forall \rho : \mathcal{C} \to \mathcal{C}.$$

Theorem 4.1 (Knill-Laflame theorem)**.** *A subspace $\mathcal{C}$ is a quantum error code against $\mathcal{E}(\rho) = \sum_k E_k \rho E_k^\dagger$ if and only if*

$$P_{\mathcal{C}} E_i^\dagger E_j P_{\mathcal{C}} = \alpha_{ij} P_{\mathcal{C}},$$

where $P_{\mathcal{C}}$ is the projector on the code space and α_{ij} are the matrix elements of an hermitian matrix, i.e., $\alpha = \alpha^\dagger$.

Given a state $|\varphi\rangle \in \mathcal{C}$, the Knill-Laflame theorem says that an error might take a state out of $\mathcal{C}$, but then we are able to bring it back.

An important consequence of the Knill-Laflamme theorem is that any linear combination of errors that can be corrected is also correctable. This can be easily proven as follows. Suppose that we can correct against errors given by X_1 and Z_1. Then, according to the Knill-Laflamme theorem, it is satisfied that

$$P_{\mathcal{C}} X_1 Z_1 P_{\mathcal{C}} = \alpha_{12} P_{\mathcal{C}}.$$

If we now consider a linear combination such as $E = \alpha X_1 + \beta Z_1$, it can be corrected because

$$\begin{aligned} P_{\mathcal{C}} E^\dagger E P_{\mathcal{C}} &= P_{\mathcal{C}}(|\alpha|^2 X_1 X_1 + \alpha^* \beta X_1 Z_1 + \alpha \beta^* Z_1 X_1 + |\beta|^2 Z_1 Z_1) P_{\mathcal{C}} \\ &= \alpha_{11} P_{\mathcal{C}} + \alpha_{12} P_{\mathcal{C}} + \alpha_{21} P_{\mathcal{C}} + \alpha_{22} P_{\mathcal{C}} \\ &\propto P_{\mathcal{C}}. \end{aligned}$$

5 Physical noise

In this section we want to consider cases where noise affects to more than one qubit at the same time under the assumption of independent and identically distributed (idd) noise. This assumption considers that noise acts individually on each bit, and thus there is no correlation between noise on individual systems. This is not always a good assumption, but it is extensively used because it is simple.

Generically, the noise on a single qubit can be modelled by $\mathcal{E}(\rho) = \sum_k E_k \rho E_k^\dagger$. Then, considering iid noise, the noise on n qubits is

$$\underbrace{\mathcal{E} \otimes \cdots \otimes \mathcal{E}}_{n}(\rho).$$

Consider the Shor code on nine qubits and a single-qubit noise given by

$$\mathcal{E}(\rho) = (1-p)\rho + \frac{p}{2}\mathbb{I} = (1-p)\rho + \frac{p}{4}\left(\rho + X\rho X + Y\rho Y + Z\rho Z\right),$$

which does not change the state with probability $1-p$ and erases any information with probability p. If the error happens on each qubit, the global noise of the nine qubits is characterised by

$$\mathcal{E}^{\otimes 9}(\rho) = (1-p)^9\rho + (1-p)^8\frac{p}{3}\left(\sum_{i=1}^{9}\sum_{\alpha=1,x,y,z}\sigma^\alpha\rho\sigma^\alpha\right) + \mathcal{O}\left((1-p)^7p^2\right). \quad (9)$$

The first term of Eq. (9) carries no error, and thus we do not need to correct it. The second term of Eq. (9) contains single-qubit errors, which we have seen in the previous chapter that the Shor code can correct. The rest of the terms Eq. (9) correspond to errors on more than one qubit and we do not know a general recovery map for them[13]. This means that the probability with which we can protect against errors on every single-qubit is given by the remaining terms ($\mathcal{O}\left((1-p)^7p^2\right)$) in Eq. (9). If we consider the Shor code on n^2, the term $\mathcal{O}\left((1-p)^7p^2\right)$ is exponentially suppressed.

6 Continuous time errors

In this section we consider continuous time errors and we see that they can be discretised. Continuous time errors can be modelled by quantum dynamical semigroups. This means that we characterise the noise as a function of a continuous variable, t, as

$$\mathcal{E}_t(\rho) = e^{t\mathcal{L}}(\rho),$$

where $\mathcal{L}$ is generically given by

$$\mathcal{L}(\rho) = i\left[H,\rho\right] + \sum_k L_k\rho L_k^\dagger - \frac{1}{2}\left(L_k^\dagger L_k\rho + L_k^\dagger L_k\rho\right),$$

with H a Hamiltonian and L_k jump operators.

Consider the situation of a bit error on the first qubit, X_1, at rate γ. In this case, the Hamiltonian is zero and there is only one jump operator such that

$$\mathcal{L}_{X_1}(\rho) = X_1\rho X_1 - \rho.$$

[13]The Shor code is able to correct against two-qubit errors such as X_1Z_2, but it fails for errors of the form of X_1X_2.

We can expand the error operator as

$$\begin{aligned}
\mathcal{E}_t(\rho) &= e^{t\mathcal{L}_{X_1}}(\rho) \\
&= \rho + t\mathcal{L}_{X_1}(\rho) + \frac{t^2}{2!}\mathcal{L}^2_{X_1}(\rho) + \frac{t^3}{3!}\mathcal{L}^3_{X_1}(\rho) + \cdots \\
&= \rho + t(X_1\rho X_1 - \rho) + \frac{t^2}{2!}\mathcal{L}_{X_1}(X_1\rho X_1 - \rho) + \frac{t^3}{3!}\mathcal{L}^2_{X_1}(X_1\rho X_1 - \rho) + \cdots \\
&= \rho + t(X_1\rho X_1 - \rho) + \frac{t^2}{2!}2(X_1\rho X_1 - \rho) + \frac{t^3}{3!}3(X_1\rho X_1 - \rho) + \cdots \\
&= \rho\left(1 - t + 2\frac{t^2}{2!} - 3\frac{t^3}{3} + \cdots\right) + X_1\rho X_1\left(t - \frac{2t^2}{2!} + \frac{3t^3}{3!} + \cdots\right) \\
&= \rho\left(1 - te^{-t}\right) + X_1\rho X_1 te^{-t}.
\end{aligned}$$

Most errors on the physical world are continuous time errors, but they can be discretised as follows. In a laboratory, the measurements are performed at a certain speed. We can break the continuous time up into a bunch of discrete steps (see Fig. 6), where each individual step is the time required to apply all parity measurements. Then, in practise we can consider each step as a discrete error processes, where the error occurs with probability

$$p = \Delta t e^{-\Delta t}.$$

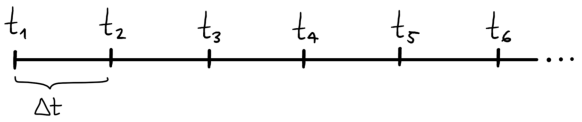

Figure 6: Discretisation of the time for continuous time errors.

7 Stabilizer codes

In the section devoted to the Shor code, we have seen that for a system of three qubits we can detect bit errors using the parity measurements Z_1Z_2 and Z_2Z_3. These two operators, Z_1Z_2 and Z_2Z_3, have the states $|000\rangle$ and $|111\rangle$ as common eigenstates with eigenvalue $+1$ and they also satisfy $\{X_1, Z_1Z_2\} = 0$ and $[X_1, Z_2Z_3] = 0$. Actually, when we measure the parity measurements on a code state, we are using these property since

$$Z_1Z_2X_1|000\rangle = -X_1Z_1Z_2|000\rangle = -X_1|000\rangle.$$

In this section we want to make use of this properties to construct a more general code on n qubits, the stabilizer code.

In order to develop the stabilizer formalism, we first need to define the Pauli group.

Definition 7.1. *The Pauli group, $\mathcal{P}_1$, is the group consisting of the 2×2 identity matrix, $\mathbb{I}$, and the Pauli matrices together with the product of these matrices with the factor -1, which are*

$$\mathcal{P}_1 \equiv \{\pm\mathbb{I}, \pm X, \pm Y, \pm Z\},$$

where X, Y and Z are defined in Eq. (4).

Note that the Pauli group has order eight, $|P_1| = 8$, which means that the group has eight elements. These elements are related by the commutation properties of the Pauli matrices, i.e.,

$$[X, Y] = 2Z, \qquad [X, Z] = -2Y, \qquad [Y, Z] = 2X.$$

If we consider the n-fold tensor product of the Pauli group, the resulting set of matrices is also a group. It is denoted by $\mathcal{P}_n$ and written as

$$\begin{aligned} \mathcal{P}_n &\equiv \{\pm\mathbb{I}, \pm X, \pm Y, \pm Z\}^{\otimes n} \\ &\equiv \{\pm G_{\vec{\alpha}}\}, \end{aligned} \tag{10}$$

where for a compact notation we define $G_{\vec{\alpha}} = \sigma_{\alpha_1} \otimes \cdots \otimes \sigma_{\alpha_n}$ with $\sigma_{\alpha_1} = \mathbb{I}$, $\sigma_{\alpha_2} = X$, $\sigma_{\alpha_3} = Y$ and $\sigma_{\alpha_4} = Z$. Some interesting properties of the group $\mathcal{P}_n$ are:

- It is a group of order $|\mathcal{P}_n| = 2 \cdot 4^n = 2^{2n+1}$.
- Any element, $G_{\vec{\alpha}} \in \mathcal{P}_n$, satisfies $G_{\vec{\alpha}}^2 = \mathbb{I}$ and $G_{\vec{\alpha}}^\dagger G_{\vec{\alpha}} = \mathbb{I}$.
- Given two different elements of the group, $G_{\vec{\alpha}}, G_{\vec{\beta}} \in \mathcal{P}_n$, they either commute, $[G_{\vec{\alpha}}, G_{\vec{\beta}}] = 0$, or anticommute, $\{G_{\vec{\alpha}}, G_{\vec{\beta}}\}$. Note that, in the case that the elements commute, they also share an eigenbasis.

Once we have seen the Pauli group and its generalisation to n qubits, we can define the stabilizer code.

Definition 7.2. *Let $\mathcal{S}$ be an abelian[14] subgroup of $\mathcal{P}_n$. Then, a stabilizer code, $\mathcal{C}$, is defined as $\mathcal{C} \equiv \{|\psi\rangle \mid S|\psi\rangle = |\psi\rangle \ \ \forall S \in \mathcal{S}\}$.*

[14] A group is abelian if all its elements commute, i.e., $[S_1, S_2] = 0 \ \forall S_i \in \mathcal{S}$.

We say that $\mathcal{S}$ is the stabilizer (group) of the code and that $S \in \mathcal{S}$ are stabilizer operators of the code. The stabilizer group fully characterises the code.

Stabilizer operators are not linearly independent in general. Note that the concept of linear independence is defined in a vector space, not in a group. Here, when we talk about linear independence, we formally mean that we map the elements of $\mathcal{P}_n$ to the vector space $(\mathbb{Z}_2)^{2n}$ using

$$\varphi : \left(\frac{\mathcal{P}_n}{\mathbb{Z}_2}; \cdot\right) \longrightarrow (\mathbb{Z}_2)^{2n}$$

and, then, we consider the concept of linear independence in $(\mathbb{Z}_2)^{2n}$. This translates in a simple way to the elements of the group $\mathcal{P}_n$, which is that two elements of $\mathcal{P}_n$ are linear independent if their product is not in $\mathcal{P}_n$. Note that then any product of the elements are also in the group. For convenience, we want to use the minimal number of elements that generate the stabilizer group, which we call generators of the stabilizer group. In other words, the generators of the stabilizer group are the operators $\{S_j\}_{j=1}^{s}$, where $S_j \in \mathcal{S}$, such that they are commuting and linearly independent. Then, there are $k = n - s$ logical qubits in the stabilizer code, i.e., $\mathcal{C}$ is 2^k-dimensional.

Example 7.1. *Consider the Shor code on nine qubits. Its stabilizer group is generated by the eight operators,* $\mathcal{S} =< \{S_k\}_{k=1}^{8} >$*, which can be written as*

$$\begin{aligned} S_1 = Z_1Z_2, \quad S_2 = Z_2Z_3, \quad S_3 = Z_4Z_5, \\ S_4 = Z_5Z_6, \quad S_5 = Z_7Z_8, \quad S_6 = Z_8Z_9, \\ S_7 = X_1X_2X_3X_4X_5X_6, \quad S_8 = X_4X_5X_6X_7X_8X_9. \end{aligned} \tag{11}$$

We can easily find other stabilizer operators by multiplying any two generators of the stabilizer group. For example,

$$\begin{aligned} S_1S_2|\varphi\rangle &= Z_1Z_3|\varphi\rangle \\ &= Z_1Z_3(\alpha|\bar{0}\rangle + \beta|\bar{1}\rangle) \\ &= Z_1Z_3(\alpha|++\rangle + \beta|--\rangle) \\ &= Z_1(\alpha|-+\rangle + \beta|+-\rangle) \\ &= (\alpha|++\rangle + \beta|--\rangle) \\ &= |\varphi\rangle, \end{aligned}$$

where $|\bar{0}\rangle$*,* $|\bar{1}\rangle$ *and* $|\pm\rangle$ *are defined in Eq. (6), Eq. (7) and Eq. (8). Thus, we have* $S_1S_2|\varphi\rangle = |\varphi\rangle$*, which implies that* $S_1S_2 \in \mathcal{S}$*.*

Consider a state in the code space, $|\varphi\rangle \in \mathcal{C}$, and an operator T that commutes with all stabilizer operators, i.e., $[T, S_k] = 0 \; \forall \; S_k \in \mathcal{S}$. Then, the state $T|\varphi\rangle$ is also in the code space because

$$T|\varphi\rangle = TS_k|\varphi\rangle = S_kT|\varphi\rangle \Rightarrow S_kT|\varphi\rangle = T|\varphi\rangle \Rightarrow T|\varphi\rangle \in \mathcal{C}.$$

Moreover, the operator T is called logical operator because it maps a state in the code space, $|\varphi\rangle \in \mathcal{C}$, to another state in the code space, $|\varphi'\rangle \in \mathcal{C}$ [15]. Note that this is not true in the case that T anticommutes with the stabilizer operators.

Example 7.2. *Consider the Shor code on nine qubits. Its logical operators are*

$$\bar{X} = X_1 \cdot X_9 = \prod_{j=1}^{9} X_j \qquad \textit{and} \qquad \bar{Z} = Z_1 \cdot Z_9 = \prod_{j=1}^{9} Z_j, \tag{12}$$

where $[\bar{X}, \bar{Z}] = 2\bar{Y}$*. We are interested in the effect of these operators on the logical qubits, which is*

$$\begin{aligned} \bar{X}|\bar{0}\rangle &= |\bar{1}\rangle & \qquad \bar{Z}|\bar{0}\rangle &= |\bar{0}\rangle \\ \bar{X}|\bar{1}\rangle &= |\bar{0}\rangle & \qquad \bar{Z}|\bar{1}\rangle &= -|\bar{1}\rangle \end{aligned}$$

where $|\bar{0}\rangle$ *and* $|\bar{1}\rangle$ *are defined in Eq. (6) and Eq. (7). Thus, the operators* $\bar{X}$ *and* $\bar{Z}$ *are the logical Pauli operators* X *and* Z*.*

Given $|\varphi\rangle \in \mathcal{C}$, any operator $\bar{X}S_k$ is also a logical operator since it maps $|\varphi\rangle \in \mathcal{C}$ to $|\varphi'\rangle \equiv \bar{X}|\varphi\rangle \in \mathcal{C}$, which we can easily see as follows

$$\bar{X}S_k|\varphi\rangle = \bar{X}|\varphi\rangle = |\varphi'\rangle \in \mathcal{C}.$$

This means that logical operators are not uniquely defined.

Example 7.3. *Consider the Shor code on nine qubits. More logical operators apart from* $\bar{X}$ *and* $\bar{Z}$ *(Eq. (12)) would be*

$$\begin{aligned} \bar{X}S_8 &= X_1X_2X_3, \\ \bar{X}S_2 &= X_1Y_2Y_3X_4X_5X_6X_7X_8X_9, \\ \bar{X}S_82S_8 &= X_1Y_2Y_3. \end{aligned}$$

[15] Stabilizer operators are not logical operators because they map a state in the code, $|\varphi\rangle$, to itself, not to another state.

It can be shown that the minimal length of all logical operators is the distance of the code. This make sense because, as we have seen, a logical operator maps a logical bit to another logical bit, and thus the minimal length of the logical operator means the minimal number of qubit operations that are necessary to map two different logical states. This is indeed the definition of the distance of the code (see Section 1.5).

Consider a general error, $E = e_x X + e_y Y + e_z Z$. As we have seen, we can correct error X, Y, Z individually. If the error is in the group $\mathcal{P}_n$, $E_\alpha \in \mathcal{P}_n$, each error commutes or anticommutes with the stabilizer operators. Then, if the error commutes, we have

$$SE_\alpha|\psi\rangle = E_\alpha S|\psi\rangle = E_\alpha|\psi\rangle,$$

and, if the error anticommutes, we have

$$SE_\alpha|\psi\rangle = -E_\alpha S|\psi\rangle = -E_\alpha|\psi\rangle.$$

In other words, given $\psi \in \mathcal{C}$, the state $E_\alpha|\psi\rangle$ is an eigenvector of the stabilizer operators with eigevalue $+1$ if $[E_\alpha, S] = 0$ and -1 if $\{E_\alpha, S\} = 0$. This means that the stabilizer can act as parity measurements to detect the errors. Note that the stabilizer formalism is a generalisation of the parity check matrices (see Section 1.3).

Let us remark that in the definition of the stabilizer code we have chosen the subspace with $+1$ eigenvalue. Nevertheless, we could take another fixed reference value (as it is done in the laboratory). For example, in the Shor code we could have taken $S_2 = -Z_2 Z_3$, and then the logical state would have been

$$\left[\frac{1}{\sqrt{2}}\left(|001\rangle \pm |110\rangle\right)\right]|+\rangle|+\rangle.$$

Another way to represent the stabilizer codes consists in splitting the stabilizers operators into two independent parity check matrices such that

$$\left(\begin{array}{c|c} H_z & 0 \\ \hline 0 & H_x \end{array}\right) \tag{13}$$

Example 7.4. *Consider the Schor code on nine qubits. The parity check*

matrices are

$$\left(\begin{array}{c|c} H_z & 0 \\ \hline 0 & H_x \end{array}\right) = \left(\begin{array}{ccccccccccc} 1 & 1 & 0 & & & & & & & & \\ 0 & 1 & 1 & & & & & & & & \\ \hline & & & 1 & 1 & 0 & & & & & \\ & & & 0 & 1 & 1 & & & & & \\ \hline & & & & & & 1 & 1 & 0 & & \\ & & & & & & 0 & 1 & 1 & & \\ \hline & & & & & & & & & & \\ & & & & & & & & & & \end{array}\right)$$

$$\left(\begin{array}{ccccccccc} & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right)$$

General CSS code, which is a method to go from two classical error correction codes to one quantum code, is another code that can be characterised with parity check matrices of the form of Eq. (13).

The minimal number of physical qubits that a code must have in order to be able to correct against single-qubit errors is five physical qubits. For this optimal code, the stabilizers and the logical operators are

$$\begin{aligned} S_1 &= X_1 Z_2 Z_3 X_4 \mathbb{I}_5 \\ S_2 &= \mathbb{I}_1 X_2 Z_3 Z_4 X_5 \\ S_3 &= X_1 \mathbb{I}_2 X_3 Z_4 Z_5 \\ S_4 &= Z_1 X_2 \mathbb{I}_3 Z_4 X_5 \end{aligned} \qquad \begin{aligned} \bar{X} &= X_1 \cdots X_5 \\ \bar{Z} &= Z_1 \cdots Z_5 \end{aligned} \tag{14}$$

8 Toric code

In this section we use the stabilizer formalism from the last section to explain the toric code, $\mathcal{C}_{\text{toric}}$. The toric code was introduced by Alexei Kitaev in 1998 and its relevance relies on the fact that it will probably be the code used on the first generation of quantum computers.

The name of the toric code comes from the fact that it is defined on a lattice with periodic boundary conditions, which form a torus. (see Fig. 7).
As we have already mentioned, the toric code is a stabilizer code and its stabilizers are

$$A_X \equiv X_u \otimes X_d \otimes X_l \otimes X_r,$$
$$B_Z \equiv Z_u \otimes Z_d \otimes Z_l \otimes Z_r,$$

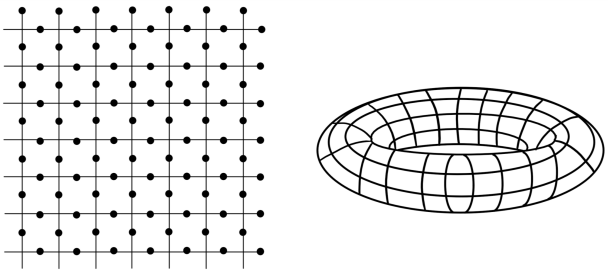

Figure 7: The topology considered for the toric code is a lattice with periodic boundary conditions, i.e., a torus.

where u, d, l, r stand for "up, down, left and right", respectively. The operators A_X and B_Z are defined on every single cross and plaquette of the lattice (see Fig. 8). We often call the operators A_X and B_Z themselves as cross and plaquette, respectively. Note that the stabilizers of the toric code are local, in contrast to the stabilizers of the Shor code (Eq. (11)). This property makes the toric code much more practical.For an $L \times L$ lattice, the toric code has $n = 2L^2$ physical qubits. There are L^2 plaquettes and L^2 crosses.

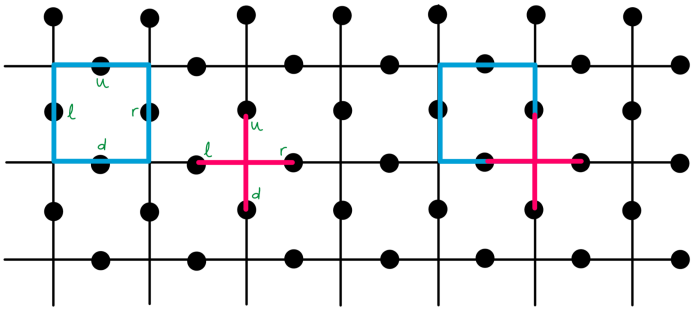

Figure 8: All plaquettes and crosses are stabilizers of the toric code. When a plaquette and a cross overlap, they do it always on two qubits.

We can easily verify that the stabilizers of the toric code commute. The case $[A_X, A_{X'}] = 0$ and $[B_Z, B_{Z'}] = 0$ are trivial because $Z^2 = X^2 = \mathbb{I}$. The case $[A_X, B_Z] = 0$ is also trivial if A_Z and B_Z do not overlap. In the case that they overlap, the operators A_Z and B_Z coincide in two qubits, and thus the phase produced by $XZ = -ZX$ cancels out (see Fig. 8).

The multiplication of two plaquettes can be easily understood via illustrations (see Fig. 9). Consider two plaquettes, B_{Z_1} and B_{Z_2} that overlap on the right qubit of the first plaquette, which is the left qubit of the second plaque-

tte. On this qubit two operators Z are applied, one from the first plaquette and one from the second. However, recall that $Z^2 = \mathbb{I}$, and thus the identity is actually applied on this qubit. This leads to a plaquette made of six Z operators (see Fig. 9), which is also a stibilizer. Note that the multiplication of plaquettes will always give open strings. Here we will mainly talk about the plaquettes operators, but the discussion with crosses can be done analogously.

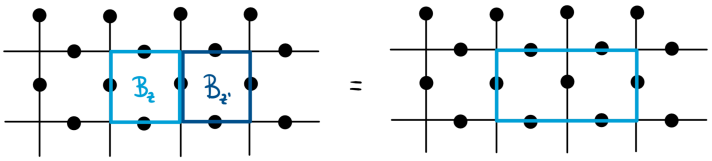

Figure 9: The multiplication of two plaquettes, B_Z and $B_{Z'}$, gives a bigger plaquette, which is also a stibilizer of the toric code.

In the section about stabilizer formalism, we have seen that the stabilizer generators must be linearly independent. We can easily check that crosses and plaquettes, A_X and B_Z, are not linearly independent since

$$\prod_{\text{all crosses}} A_X = \mathbb{I} \qquad \text{and} \qquad \prod_{\text{all plaquettes}} B_Z = \mathbb{I}. \tag{15}$$

In order to have a set of independent stabilizers of the toric code, we simply need to remove one plaquette and one cross. Thus, the toric code has $s = 2L^2 - 2$ independent stabilizers. As we have also explained in the previous section, the number of encoded qubits (logical qubits) in a stabilizer code is $k = n - s$. For the toric code, $k = 2$ logical qubits.

In order to complete the characterisation of the code space $\mathcal{C}_{\text{toric}}$, we need to identify the logical operators. Recall that these operators must commute with the stabilizers without being stabilizers themselves. The logical operators can be either product of Z or product of X. We have seen that multiplying plaquettes gives loops of different sizes, which are also stabilizers, but they never are open strings. Consider a product of Z along a whole horizontal string of the lattice, i.e., $\bar{Z}_1 \equiv Z_1 \otimes \cdots \otimes Z_L$. It commutes with B_Z and A_X due to the same reasons that have implied $[A_X, A_{X'}] = [B_Z, B_{Z'}] = [A_X, B_Z] = 0$ (see Fig. 10). Note that this is also true for a product of Z along a whole vertical string, $\bar{Z}_2$, and for a product of X along a whole horizontal and vertical string, $\bar{X}_1$ and $\bar{X}_2$ (see Fig. 10). Note further that $\{\bar{Z}_i, \bar{X}_i\} = 0$ and $[\bar{Z}_i, \bar{X}_j] = 0$ for $i \neq j$ and $i, j = 1, 2$. Defining $\{|\bar{0}_1\rangle, |\bar{1}_1\rangle\}$ as the eigenvectors

of $\bar{Z}_1$, we can easily check that, as expected, the logical operators satisfy

$$\begin{aligned} \bar{X}_1|\bar{0}_1\rangle &= |\bar{1}_1\rangle, & \bar{Z}_1|\bar{0}_1\rangle &= |\bar{0}_1\rangle, \\ \bar{X}_1|\bar{1}_1\rangle &= |\bar{0}_1\rangle & ,\bar{Z}_1|\bar{1}_1\rangle &= -|\bar{1}_1\rangle. \end{aligned}$$

We find analogous equalities for $\bar{X}_2$ and $\bar{Z}_2$.

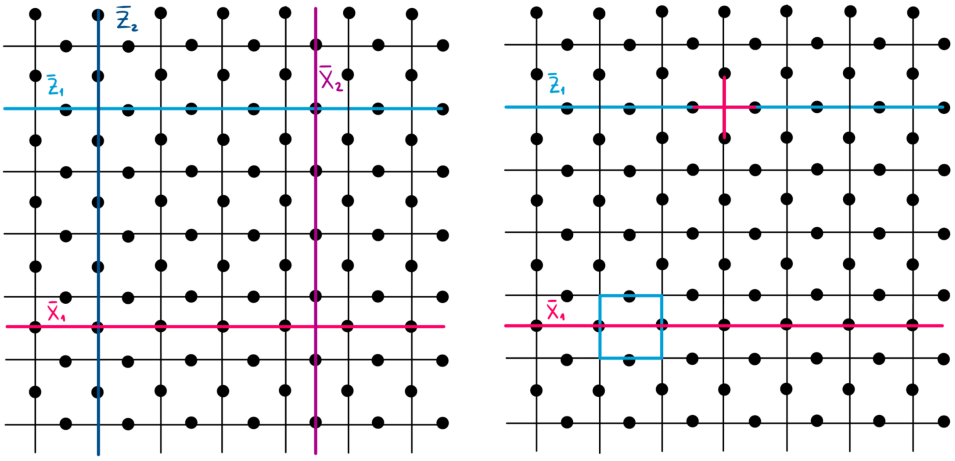

Figure 10: Logical operators of the toric code. When the logical operators $\bar{Z}_1$ and $\bar{Z}_2$ ($\bar{X}_1$ and $\bar{X}_2$) overlap with a cross (plaquette), they do it on two qubits.

Recall from the section 7 that logical operators do not have a unique representation. Indeed, we can obtain a new logical operator multiplying any operator $\bar{X}_1, \bar{X}_2, \bar{Z}_1, \bar{Z}_2$ by any stablizer operator, i.e., by any plaquette or cross. In the toric code, this property translate to the fact that $\bar{Z}_1$, which is a straight line, can be stretched several times and still represent the same logical operator (see Fig. 11). Note that the same is true for $\bar{X}_1, \bar{X}_2, \bar{Z}_1, \bar{Z}_2$. Therefore, the logical subspace is the subspace spanned by all strings with "the same topology".

The distance of a stabilizer code is the weight of the minimal representation of logical operators, as we have seen in the previous section. For the toric code we have $d_{\text{Toric}} = L$.

In summary, we can characterise the toric code as a $[2L^2, 2, L]$ code. Let us remark that the toric code has a topologic flavour because its encoded information is defined by objects that exists only on the topology of the space,

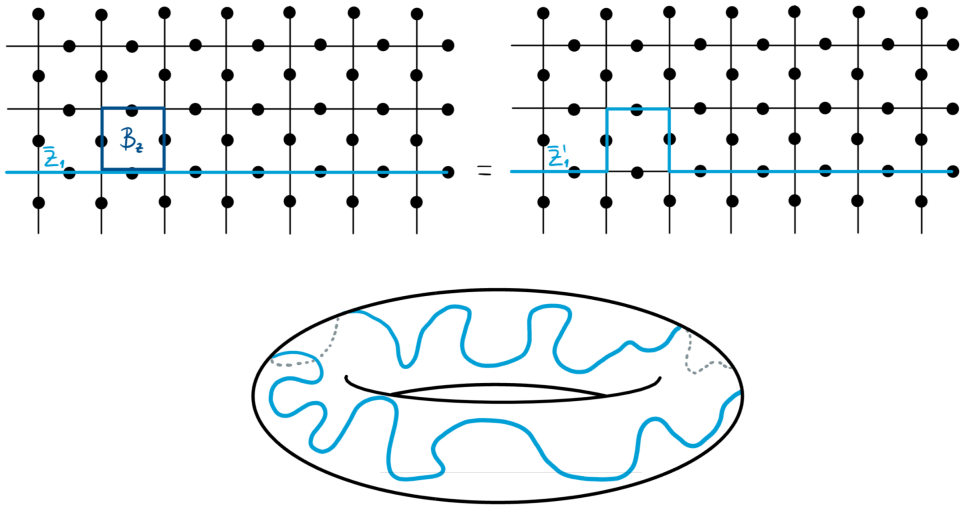

Figure 11: Multiplying a plaquette, B_Z, by a logical operator, e.g. $\bar{Z}_1$, gives new logical operator. Therefore any string around the torus is a logical operator.

i.e., of the torus.

8.1 Connection to many-body theory (quantum statistical mechanics)

In this section we see the connection between the toric code and the many-body theory. In order to do that, let us define a Hamiltonian, H, such that

$$H = -\sum_{\text{crosses}} A_X - \sum_{\text{plaquettes}} B_Z.$$

Note that this Hamiltonian, H, is made of a sum of local terms on a lattice, which are typical characteristics of Hamiltonians used in many-body theory. Since A_X and B_Z can only have ± 1 eigenvalues, the ground states of the Hamiltonian, H, are the stabilizer states because they all have $+1$ eigenvalue. In other words, the ground state subspace is $\mathcal{C}_{\text{Toric}}$ and the ground state energy is $-2L^2$.

This connection is not a property only of the toric code, but of all topological stabilizer codes. Topological stabilizer codes can always be defined as the ground space of a local commuting Hamiltonian. The toric code is the simplest example. This connects coding theory to many-body physics. In

particular, one of the most remarkable links is that error correction in the code picture corresponds to topological order in the many-body picture.

8.2 Errors on the toric code

Errors on the toric code are detected and corrected using the stabilizer formalism. In this section we consider independent and identically distributed (iid) noise produced by local Pauli matrices X and Z and explain how can be corrected. As we have seen in section 7, we can identify the X-errors with the Z stabilizers and the Z-errors with the X stabilizers and treat them independently. For simplicity, here we do the error analysis only for Z errors. Recall that the outcomes -1 of the stabilizer measurements are called syndromes.

Consider an X-error on a qubit. In order to detect it, we measure all plaquette stabilizers, and thus we obtain two syndromes on the plaquettes acting on the corrupted qubit. If we now consider two or more X-errors on the lattice forming a string, we see that we also get two syndromes and they are at the ends of the string error (see Fig. 12).

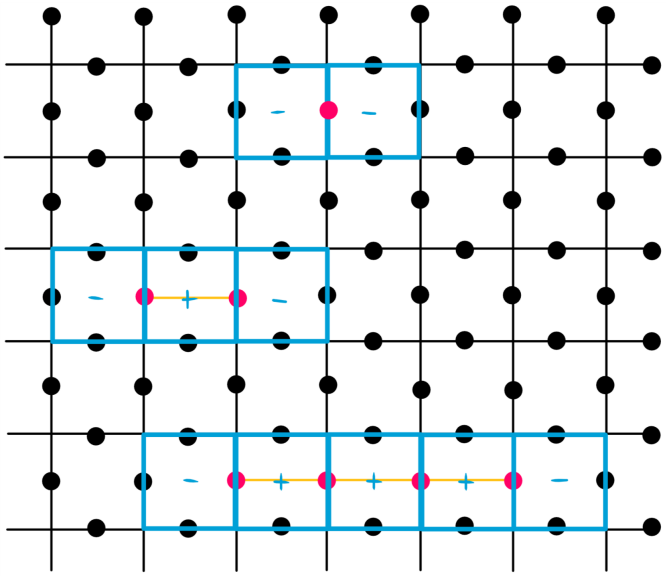

Figure 12: Any string error gives always a pair of syndromes at the ends of the string.

The relation between the error string and the syndrome is not unique, i.e.,

there exist differnt string errors with the same syndrome (see Fig. 13). Actually, whenever we have two syndromes, they can correspond to any string connecting them and, when we perform the stabilizer measurements, we have absolutely no way of knowing the correct string. As we have seen before, the code space is the ground space of all trivial loops, i.e., all loops that do not wrap around the torus. Therefore, even if we do not know the "real string", we simply need to assume that it is one of the shortest and correct it by closing the loop. The only problem comes if we choose the correction that extends the loop to a string around the torus since it creates a logical error (see Fig. 14). In other words, if apply a recovery map, $\mathcal{R}$, which closes the string error creating a trivial loop, we recover the initial state as

$$\mathcal{R}E|\psi\rangle = |\psi\rangle.$$

However, if the recovery map applied, $\mathcal{R}_{\text{wrong}}$, creates an string around the torus, we will have

$$\mathcal{R}_{\text{wrong}}E|\psi\rangle = O_{\text{logical}}|\psi\rangle \neq |\psi\rangle,$$

where O_{logical} is any logical operator, i.e., $O_{\text{logical}} \in \{\bar{X}_1, \bar{Z}_1, \bar{X}_2, \bar{Z}_2\}$.

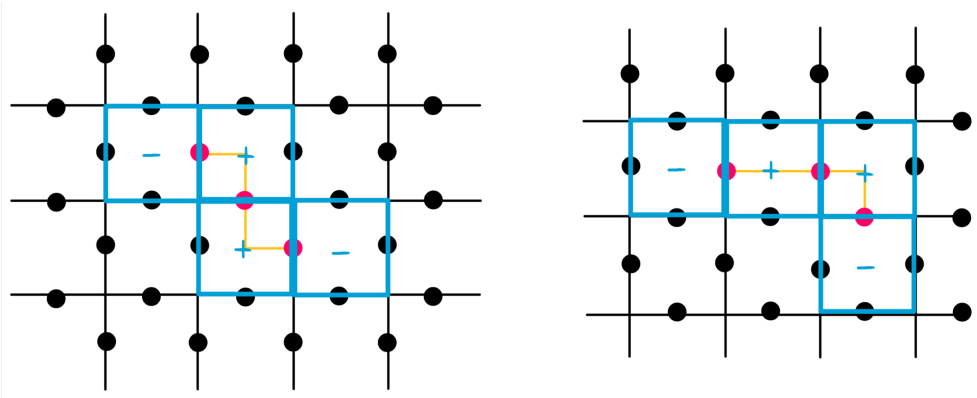

Figure 13: The correspondence between syndromes and string error is not unique, i.e., two different string error can give the same syndromes.

The toric code allows to correct $\lfloor \frac{d-1}{2} \rfloor$ errors. We can see this as follows. When we have more than $\lfloor \frac{d-1}{2} \rfloor$ errors along a line, which are the worst type of errors, the natural choice for correction, i.e., the shortest path to close the loop, gives a straight string. Thus, we get a logical error (see Fig. 15). In a sense this is completely inefficient because on average we have $p \cdot n$ errors for iid flip (phase) noise with qubit error rate $p<1/2$. Asymptotically, $n \sim L$

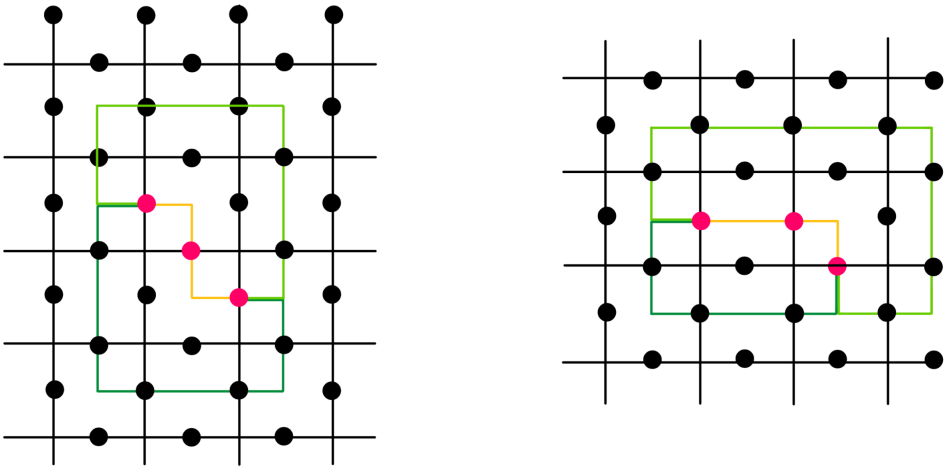

Figure 14: The correction operation corresponds on closing the string error to create a loop. Even if there exist several ways of closing the loop, we apply the shortest for convenience.

and $d \sim L$, thus even if we have asymptotically small p, but constant, we are always going to have more than $\lfloor \frac{d-1}{2} \rfloor$ errors. Then, the decoding task is to pair all syndromes such that we do not create a logical operator.

Consider the setting with iid noise with a fixed value of $p<1/2$. The probability to have $n = 0, 1, 2, \dots$ errors is summarized in the following table

n	0	1	$\cdots$
probability	$(1-p)^n$	$np(1-p)^{n-2}p^2$	$\cdots$

n	$d/2$	$d/2-1$	$\cdots$
probability	$(\cdots)(1-p)^{n-d/2}p^{d/2}$	$(\cdots)(1-p)^{n-d/2-1}p^{d/2+1}$	$\cdots$

n	$n/2$
probability	$\sim \frac{\text{cnt}}{n}$

where $(\cdots)$ represent factors that are not relevant in this discussion. Let us suppose that we have calculated all these probabilities and that we have a computer that can calculate the minimal distance between two syndromes. Then, it can be shown that there exists a function that gives the most probable source of error for any given syndrome. Although this is the optimal decoding process, it inefficient because we have to keep track of a factorial number of iterations and calculate all the probabilities. Here efficient means

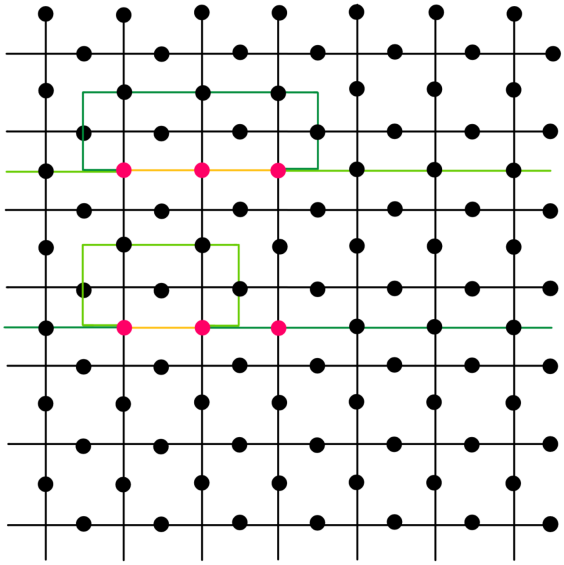

Figure 15: 7×7 lattice. When the string error has more than $\lfloor \frac{d-1}{2} \rfloor$, where $d = L$ for the toric code, the shortest way to close the loop creates a logical error.

that the function that assigns a correction operator to each syndrome is a efficient (i.e., linear or quadratic) function as a function of n; while optimal means that the function never assigns a wrong correction operator. In practise, we use non optimal, but efficient decoders.

8.2.1 Minimum weight perfect matching

The minimum weight perfect matching (MWPM), which is also called Edmonds algorithm, consists in pairing all syndromes. As we have seen, the most likely source of errors gives syndromes which are close. Therefore, the natural choice to bring syndromes together is minimising the total distance. This decoding procedure runs in time n^3.

8.2.2 Renormalisation

Consider a system with a certain number of syndromes and label them. The renormalisation decoder is an iterative processes consists in the following. For each error, we pair up all syndromes in a ball of radius $r = r_0$ with center in the error. If there was an even number of syndromes in the ball of radius r_0, there are none left; and, if there was an odd number, there is one syndrome left. Now we have a system with less syndromes and, in particular,

we now that all syndromes are in a distance bigger than r_0. We repeat the pairing using $r_1 > r_0$. After repeating the pairing enough times, we will end up either with no syndromes or two syndromes very far apart, which cannot be corrected. It can be shown, however, that the probability that the latter situation happens is exponentially suppressed.

8.3 Thresholds

As the distance of the toric code is $d = L$, one would expect that increasing the size of the lattice leads to a more robust code. Nevertheless, the number of errors also increase with L for a idd noise model. This implies that we have to find a trade off between these two phenomena, which is called the threshold.

In section 1.6, we have seen that a code $\mathcal{C}$ with decoder $\mathcal{R}$ have threshold, $p_{\text{th}} < 1/2$, if for $p < p_{\text{th}}$ the probability to have a logical error is exponentially small, i.e.,

$$P_{\text{logical}}(n, p) < ce^{-d\xi}.$$

If you have a small physical error, p, the probability that the decoder creates a logical error is exponentially small. However, if the physical error, p, is above the threshold, then the decoder will often apply a "wrong correction", i.e., it will create a logical error. Only in the first case with small values of physical error rates, increasing the size of the lattice will imply an exponential decay of the logical error rate.

Each decoder has a different error rate depending on the algorithm that it uses. In the case of the toric code, the theoretical optimal threshold is $p_{\text{th}}^{\text{optimal}} = 0,113$. For the decoders we have seen before, the thresholds are $p_{\text{th}}^{\text{MWPM}} = 0,11$ and $p_{\text{th}}^{\text{renormalisation}} = 0,88$.

For real error correction, we are interested not just in the error correction threshold, but also in fault tolerant threshold, which is when measurement error are also taken into account. The theoretical fault tolerant threshold is $p_{\text{th, FT}} \approx 0,02$. In practise, one has to perform measurements on two qubits, which requires that all gates are accurate to rates of 0,005 roughly. Once the fault tolerance threshold is achieved, the rest is making larger and larger codes, which is mainly an engineer problem.

9 Lower bound on the threshold

In this chapter, we show how to estimate the threshold of maximal threshold of the Toric code by analyzing the decoding problem as a classical statistical mechanics model: the random bond Ising model. This mapping was first identified by Dennis et al [**?**].
Before going into the exact mapping, we will examine why the decoding problem might be related to (classical) statistical mechanics.

9.1 Entropy and Energy

Consider a CSS code on n qubits. Since the X and Z sectors decouple, we can restrict our attention to the Z sector where errors are bit flips (X). We assume that each qubit is flipped with probability p. Then the logical failure rate is given by

$$\bar{P}(p,n) = \sum_{E \in \mathcal{F}} \pi(E), \tag{16}$$

where

$$\pi(E) = (1-p)^n \left(\frac{p}{1-p}\right)^{\omega}.$$

is the probability that error configuration E occurs, ω the weight of error E, i.e., $|E| = \omega$, and $\mathcal{F}$ is the set of error configurations leading to a failure for the optimal decoder. Clearly, the difficulty in estimating Eqn. (16) is that the set $\mathcal{F}$ is difficult to characterise exactly.

To make the connection to statistical mechanics more obvious, define an effectiv temperature

$$\beta \equiv \log\left(\frac{1-p}{p}\right) > 0 \qquad \text{for} \quad 1 > p > 0,$$

and rewrite $\bar{P}(p,n)$ as

$$\bar{P}(p,n) = (1-p)^n \sum_{\omega=d/2}^{n} N_{\text{fail}}(\omega) e^{-\beta\omega},$$

where the sum on failing configurations has been reorganised into errors of a given weight w. Note that the minimal failing error configuration has weight $d/2$, determining the lower index in the sum. $N_{\text{fail}}(\omega)$ accounts for the multiplicity of error configurations with a fixed weight w. Hence we have

shifted. The expression can now be reinterpreted as a statistical mechanics model with

$$\bar{P}(p,n) = (1-p)^n \sum_{\omega=d/2}^{n} e^{-\beta F(\omega)},$$

where $F(\omega) = \omega - \frac{S_{\text{fail}}(\omega)}{\beta}$ is a free energy with $S_{\text{fail}}(\omega) = \log(N_{\text{fail}}(\omega))$ the entropic contribution. The weight w of the error string can be understood as an energy.

9.2 Lower bound on the threshold

We now turn our attention back to the Toric code. We will provide an upper bound on the logical failure probability (and hence a lower bound on the threshold), by upper bounding the number of error configurations in $\mathcal{F}$.
Given two complementary errors, E and E', a loop can be represented by $L = EE'$ (multiplication of Pauli operators). Instead of the optimal decoder, we consider a decoder that for any specific error E chooses a correction E'. Then to each error E we can associate a loop L (note however that this converse is not true; each loop L is associated with many errors E). We get the following upper bound:

$$\bar{P}(p,n) \leq (1-p)^n \sum_{L} \sum_{u=\frac{|L|}{2}}^{|L|} \sum_{v=0}^{n-|L|} \binom{|L|}{u} \binom{n-|L|}{v} \left(\frac{p}{1-p}\right)^{u+v}. \tag{17}$$

The upper bound can be understood as follows. The first sum runs over all possible non-contractible loops L wrapping around the torus. The second sum, together with the first binomial factor, account for all the ways the errors can be distributed alone L leading to a failure. Any error E along L with $|E| \geq |L|/2$ will lead to a failing correction. The final sum accounts for all of the errors that are not on L, and do not lead to a non-trivial correction. The binomial factor accounts for all of the ways of distribution up to $n-|L|$ flip errors on the rest of the lattice. Again we will group the first sum into loops of a fixed length $l \geq d$ to get

$$\bar{P}(p,n) \leq (1-p)^n \sum_{l=d}^{n} N_{\text{con}}(l) \sum_{u=l/2}^{l} \sum_{v=0}^{n-l} \binom{l}{u} \binom{n-l}{v} \left(\frac{p}{1-p}\right)^{k+v}, \tag{18}$$

where $N_{\text{con}}(l)$ counts the number of non-intersecting loops of length l. Recall that

$$\binom{b}{a} = \frac{a!}{(a-b)!b!},$$

and note the following identities

$$\sum_{v=0}^{n-l} C_v^{n-l} \left(\frac{p}{1-p}\right)^v = (1-p)^{l-n},$$

$$\sum_{n=l/2}^{l} C_n^l \left(\frac{p}{1-p}\right)^n \leq 2^l \left(\frac{p}{1-p}\right)^{l/2}.$$

Thus, the expression can be upper bounded as

$$\bar{P}(p,n) \leq \sum_{l=d}^{n} N_{\text{con}}(l) 2^l p^{l/2} (1-p)^{l/2}. \tag{19}$$

This expression over-counts, primarily by associating certain failing error configurations to many different failing paths. Asymptotically, the number of non self-intersection paths is given by $N_{\text{con}}(l) \leq N_0 c^l$, where $c \approx 2.64$ is an expansion coefficient, allowing us to obtain the bound

$$\bar{P} \leq \sum_{l=d}^{n} N_0 \left(2c\sqrt{p(1-p)}\right)^l.$$

The series will be convergent, wheneer $2c\sqrt{p(1-p)} \leq 1$. Hence we can associate identify a lower bound on the threshold to any value of p satisfying this bound. The maximal such value gives us our best lower bound on the threshold: $p_{\text{th}} \approx .037$.

9.3 Estimating the optimal threshold

In order to compute the actual threshold, we need to resort to a different statistical mechanics mapping. The probability of an error configuration E can be written as

$$P(E) = \left[\prod_l (1-p)\right] \left[\prod_l \left(\frac{p}{1-p}\right)^{n_E(l)}\right],$$

where the products are over all of the edges of the lattice, and the function

$$n_E(l) = \begin{cases} 0 & \text{if there is no error on } l \\ 1 & \text{if there is an error on } l \end{cases}.$$

For a fixed E, we now seek to describe the probability distribution of errors E' that have the same boundary as E. We assume that the (optimal) decoder chooses the operation that maximises the likelihood of correcting to

the original homology class. If each path had the same entropic weight, the maximum likelyhood would be given by the minimum weight configuration, which is what the MWPM decoder is based on.
Any correction E' can be written as

$$E' = E + C,$$

where C is a loop (see Figure 16). We assume that the distribution of loops C is given by the natural distribution of loops on the lattice post-selected on the loops containing the boundary points of E. The edges l of C are given with probability

$$\left(\frac{p}{1-p}\right)^{n_C(l)},$$

when

$$\begin{cases} n_C(l) = 1 \\ n_E(l) = 0 \end{cases} \quad \text{when } l \text{ is occupied by } E'.$$

and with probability

$$\left(\frac{p}{1-p}\right)^{n_C(l)},$$

when

$$\begin{cases} n_c(l) = 1 \\ n_E(l) = 1 \end{cases} \quad \text{when } l \text{ is not occupied by } E'.$$

Thus, the chain $E' = E + C$ occurs with probability

$$P(E'|E) \propto \prod_l e^{J_l u_l},$$

where $u_l = 1 - 2n_C(l) \in \{-1, 1\}$ and

$$J_l = \begin{cases} \frac{p}{1-p} & \text{if } l \notin E \\ \\ \frac{1-p}{p} & \text{if } l \in E \end{cases}$$

Let us remark that the one-chain $\{l|u_l = -1\}$ is a cycle with a cycle condition that reads

$$\prod_{l \ni s} u_l = 1,$$

where s is a point in the dual lattice (see Fig. 17). There exists also a cycle condition for the dual lattice, which is

$$\prod_{l^* \in P^*} u_{l^*} = 1.$$

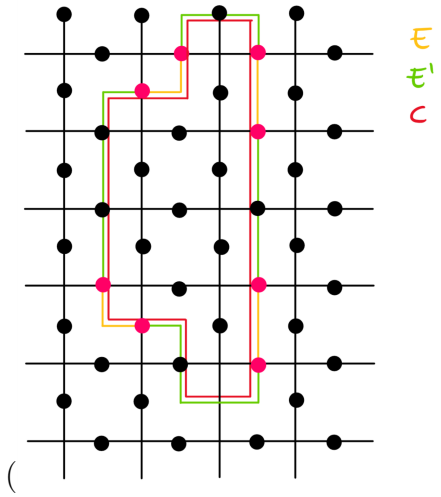

Figure 16: A loop C can be represented by two complentary errors, E and E' such that $C = EE'$.

It is easy to see that we can write this constraint as $u_{ij} = \sigma_i \sigma_j$. Thus, the fluctuation of the error chains E' that share a bound with E is described by

$$Z(J, \eta) = \sum_{\{\sigma_j\}} \exp \left[J \sum_{<ij>} \eta_{ij} \sigma_i \sigma_j \right],$$

with

$$e^{-2J} = \frac{p}{1-p} \qquad \text{and} \qquad \eta_l = \begin{cases} 1 & \text{if } l \notin E^* \\ -1 & \text{if } l \in E^* \end{cases}.$$

Another important observation is that, if E and E' are generated by sampling the same probability distribution, then the values of η'_l are chosen randomly subject to

$$\eta_l = \begin{cases} 1 & \text{with probability } (1-p) \\ -1 & \text{with probability } p \end{cases}.$$

The interpretation of this choice is

10 Topological order and QEC

In section 8, we have seen that the toric code has topological features, which show up in the fact that the logical operators of the Toric code can be represented as strings wrapping around one direction of the torus (see Fig. 10).

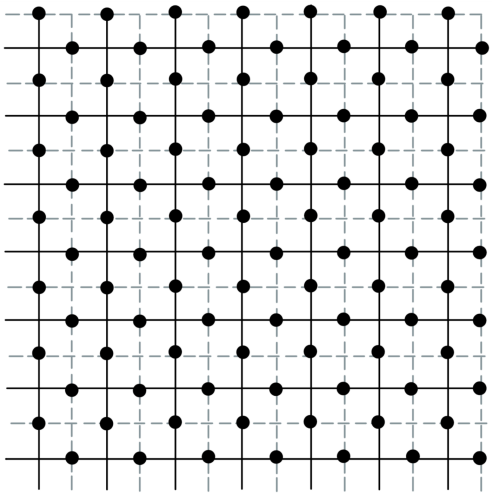

Figure 17: A dual lattice (dashed lines) can be defined for the toric code.

This is a very specific property of the toric code. In this section, we want to go to a more general system and explain how we can chacarterise topological order in a lattice system.

The systems that we study in this section are called commuting projector codes, which is defined as follows.

Definition 10.1. *Given* $\{P_j, P_k\}$ *such that* $[P_j, P_k] = 0$*, a commuting projector code (CPC) is defined as*

$$\mathcal{C} = \{|\psi\rangle \text{ such that } P_j|\psi\rangle = |\psi\rangle\}.$$

Note that this definition looks like the stabilizer code (see Definition 7.2), but here P_j are not required to be Pauli matrices. There exist plenty commuting projector codes that are not stabilizer codes, but it is not easy to write them down. If all projectors P_j are local, i.e., their support is a ball of finite radius, the CPC is called local CPC. In this section we deal with local CPCs.

A commuting projector code, $\mathcal{C}$, is the ground subspace of the Hamiltonian

$$H = \sum_j Q_j, \text{ where } Q_j = \frac{1}{2}\left(\mathbb{I} - P_j\right).$$

Note that Q_j are also projectors, and thus they satisfy $Q_j^2 = Q_j$. We can easily see that $\mathcal{C}$ is the subspace of H because the projectors Q_j annihilate the states in $\mathcal{C}$, i.e., $Q_j|\psi\rangle = 0\ \forall j$ and $\forall|\psi\rangle \in \mathcal{C}$. Moreover, the Hamiltonian,

H, is also frustration free[16]. The projector on the code space, i.e., on the subspace of H, can be written as

$$P_{\mathcal{C}} = \prod_j P_j. \tag{20}$$

10.1 Definition of topological order

Topology is a concept that comes up in different contexts and its definitions is different depending on the subfield of physics. In quantum information, there exists three definitions of topological order defined on large lattices. We explain them in the subsequent sections.

10.1.1 Topological order I: Local indistinguishability

Consider two states of the codespace, $|\psi_1\rangle$, $|\psi_2\rangle \in \mathcal{C}$, such that $\langle\psi_1|\psi_2\rangle = 0$. The topological order known as local indistinguishability says that, for any local operator O defined on the lattice, it is satisfied that $\langle\psi_1|O|\psi_1\rangle = \langle\psi_2|O|\psi_2\rangle$. In other words, the states in the code space have global properties, and thus they cannot be distinguished using local operators. We have already seen local indistinguishability in the toric code with the fact that their logical operators must be completely non-local.

Let us mention that that local indistinguishability can be also stated as

$$P_{\mathcal{C}} O P_{\mathcal{C}} = c(O) P_{\mathcal{C}} \qquad \text{with} \qquad c(O) = \frac{\mathrm{tr}(PO)}{\mathrm{tr}P}, \tag{21}$$

where O is a local operator, $P_{\mathcal{C}}$ is the projector on the code space $\mathcal{C}$ and $c(O)$ is a constant that depends on the local operator O [17]. Note that Eq. (21) reminds to the error correcting condition (Eq. (3)).

10.1.2 Topological order II: topological entanglement entropy

In order to define topological order as topological entanglement entropy, we first need some definitions.

Definition 10.2. *The entropy of a density matrix, ρ, is given by*

$$S(\rho) \equiv -tr\left(\rho \log \rho\right).$$

[16] A Hamiltonian, H, is a frustration free Hamiltonian if all its terms annihilate the ground subspace

[17] We will show Eq. (21) in the exercise class.

Definition 10.3. *For a pure state, $|\varphi\rangle$, defined on a lattice, the entropy of a region, A, of the lattice is*

$$S_\varphi(A) = -tr\left(\rho_A \log \rho_A\right),$$

where $\rho_A = tr_B(|\varphi\rangle\langle\varphi|)$.

Definition 10.4. *Given A, B and C disconnected regions of a lattice, Λ, and a state, $|\varphi\rangle$, defined on the lattice, the conditional mutual information is*

$$I_\varphi(A:C|B) = S_\varphi(AB) + S_\varphi(BC) - S_\varphi(B) - S_\varphi(ABC),$$

where $AB \equiv A \cup B$.

After these definitions, we are able to define topological order in the sense of entanglement entropy. Consider a lattice, Λ, and regions A, B and C such that B shields A from C and $A \cup B \cup C = \Lambda$ (see Fig. 18a). The system has topological entanglement entropy if $I_\rho(A:C|B) = 0 \ \forall \rho \in \mathcal{C}$. For the case where the regions are defined as in Figure 18b, the topological entanglement entropy exists if $I_\rho(A:C|B) = c_{\text{top}} \ \forall \rho \in \mathcal{C}$, where c_{top} is a topological constant.

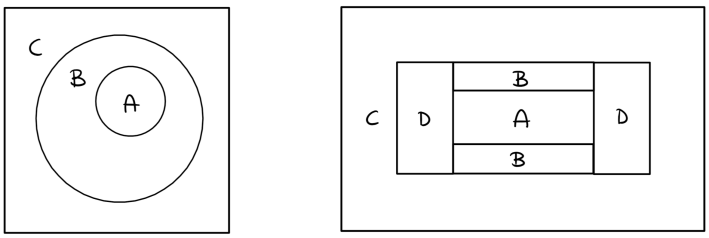

Figure 18: Lattice partitions used to define topological entanglement entropy.

10.1.3 Topological order III

The third and last notion of topological order needs the definition of local unitary.

Definition 10.5. *A unitary operator is called local unitary if it can be written as*

$$U = U_1 \cdots U_l,$$

where l is a constant and each factor U_j is local.

Now, we can state the third definition of topological order as follows. Given a state $|\varphi\rangle$, it has topological order if there exists no local unitary circuit, U, and no product state $|0\rangle^{\otimes\Lambda}$ such that $\varphi = U|0\rangle^{\otimes\Lambda}$.

All three definitions of topological order can be stated allowing for small errors. Even in this scenario, it is not known how to connect the different definitions. Some partial implications have been found, but not more. In this notes, we mainly use the first definition of topological order, i.e., the local indistinguishability, because it is related to quantum error correction as we will see.

10.2 Theorems, lemmas and facts on CPC

Up to here we have used arbitrary Pauli matrices as error model. Now, we change it and consider the erasure model[18]. Erasure noise is a process that, at some discrete interval of time, some qubits are erased. Namely, the noise channel that erases a single qubit is

$$\mathcal{N}_j(\rho) = \mathrm{tr}_j(\rho) \otimes \frac{1}{2}\mathbb{I} = \frac{1}{4}\left(\rho + X_j\rho X_j + Y_j\rho Y_j + Z_j\rho Z_j\right),$$

and the noise channel that erases a region A is

$$\mathcal{N}_A(\rho) = \mathrm{tr}_A(\rho) \otimes \frac{1}{d_A}\mathbb{I}, \tag{22}$$

where d_A is the dimension of region A.

There exists an important theorem on the erasure channel, which states the following.

Theorem 10.1. *A quantum error correcting code, $\mathcal{C}$, can protect against d erasure errors if and only if it can also protect against $\frac{d}{2}$ arbitrary errors*[19].

Proof. The idea of the proof is the following. The error correction condition requires that $P_{\mathcal{C}}E_iE_jP_{\mathcal{C}} = cP_{\mathcal{C}}$, where $P_{\mathcal{C}}$ is the projector on the code space (Eq. (20)), while the error detection condition is $P_{\mathcal{C}}EP_{\mathcal{C}} = cP_{\mathcal{C}}$. The difference on these conditions comes from the fact that detection only requires access to a single Kraus operator, but correction involves many of them.

[18] Even if in this section we only consider erasure errors, the statements that we make here reduce to general errors.

[19] Here, an arbitrary error means that we ignore where it has occurred, while we do know where the erasure errors are.

Clearly, if we have $\frac{d}{2}$ errors and we can take any two-combination of the errors, the E can be represented as having support on a maximum of d sites, and viceversa. □

Once we have defined and established the noise channel, we define the notion of error correction under the assumption of erasure noise, which is the following.

Definition 10.6. *Assume the erasure channel (Eq. (22)) and consider a code space, $\mathcal{C}$, defined on a lattice, Λ, and a region of the lattice, $A \subset \Lambda$. An erasure error is recoverable if there exists a recovery map, $\mathcal{R}$, such that*

$$\mathcal{R}(tr_A \rho) = \rho \qquad \forall \rho \in \mathcal{C} \text{ and } \forall A \subset \Lambda.$$

Definition 10.7 (Decoupling)**.** *Given a lattice, Λ, and three regions, A, B and C, such that B shields A from C and $ABC = \Lambda$ (see Fig. 18a), then there exists decoupling if for any state $\rho \in \mathcal{C}$ it is fulfilled that*

$$tr_A \rho = \rho_A \otimes \rho_C.$$

Note that decoupling is a non-trivial property since, in general, regions A and C are entangled.
In section 1.1, we have mentioned that the ideal situation for error correction is a code with $k(n) = \alpha n$ and $d(n) = \beta n$ in such a way that

$$\frac{k}{n} = \alpha \qquad \text{and} \qquad \frac{d}{n} = \beta, \tag{23}$$

where α, β are constants. This kind of codes exist in classical error correction, but nowadays no quantum code satisfying Eq.(23) is known. Indeed, a theorem for local commuting projector codes (see Theorem 10.1) states that the ideal scenario is not possible. In order to see and prove this theorem, we need some lemmas and facts about local commuting projector codes.

Lemma 10.1. *Consider a local commuting projector code and two disconnected regions, A and B, i.e., they are separated by a distance bigger than l^*, where l^* is the radius of the support of any commuting projector (see Fig. 19)*[20]*. Then, it is satisfied that*

$$\rho_A \otimes \rho_B = \rho_{AB} \qquad \forall \rho \in \mathcal{C}. \tag{24}$$

[20]Note that the required separation in Lemma 10.1 can be stated in other words saying that there exists no stabilizer operator that acts on both regions A and B.

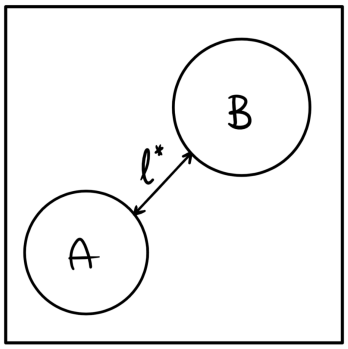

Figure 19: Lemma 10.1 requires two disconnected regions, A and B, that are separated by a distance bigger than l^*, where l^* is the radius of the support of any commuting projector.

Proof. Given a state ρ, the definition of the covariance between regions disconnected regions A and B is

$$\begin{aligned}\mathrm{Cov}(A:B) \equiv \sup\{\, &|\mathrm{tr}\left[X_A \otimes X_B \left(\rho_{AB} - \rho_A \otimes \rho_B\right)\right]| \\ &\text{such that } ||X_A|| = 1, ||X_B|| = 1, X_A = X_A^\dagger, X_B = X_B^\dagger\},\end{aligned}$$

where for operators it is used the infinite norm, i.e., $||X||$ equals to the largest eigenvalue of X. A very well-known result[21] bounds the covariance as

$$\frac{1}{\min\{d_A, d_B\}}||\rho_{AB} - \rho_A \otimes \rho_B||_1 \leq \mathrm{Cov}(A, B) \leq ||\rho_{AB} - \rho_A \otimes \rho_B||_1,$$

where for states it is used the trace norm, i.e., $||\cdot||_1 = \mathrm{tr}(\cdot)$. We will prove Eq. (24) by showing that the covariance between two regions of a locla CPC code is zero, and so its bounds.

Consider a state $\rho \in \mathcal{C}$ such that $\mathcal{C}$ is a local CPC and $\rho = \mathrm{tr}_R(|\psi\rangle\langle\psi|)$, where

[21]The upper bound on the covariance is obvious since we only need to replace the supremum over tensor operators by the supremum over operators that exist on AB. The lower bound would requires more work, which will not be done here.

R is a purification system. Then, we can write

$$\begin{aligned}
\mathrm{tr}\left[(X_A \otimes X_B)\rho_{AB}\right] &= \langle\psi|X_A \otimes X_B|\psi\rangle \\
&= \langle\psi|P_{\mathcal{C}} X_A \otimes X_B P_{\mathcal{C}}|\psi\rangle \\
&= \langle\psi|P_A P_{A^C} X_A X_B P_{B^C} P_B|\psi\rangle \\
&= \langle\psi|P_A P_{A^C}^2 X_A X_B P_{B^C}^2 P_B|\psi\rangle \\
&= \langle\psi|P_{\mathcal{C}} X_A P_{A^C} P_{B^C} X_B P_{\mathcal{C}}|\psi\rangle \\
&= \langle\psi|P_{\mathcal{C}} X_A P X_B P_{\mathcal{C}}|\psi\rangle \\
&= \langle\psi|P_{\mathcal{C}} X_A P_{\mathcal{C}} P_{\mathcal{C}} X_B P_{\mathcal{C}}|\psi\rangle \\
&= \langle\psi|P_{\mathcal{C}}|\psi\rangle c(X_A) c(X_B) \\
&= c(X_A) c(X_B) \\
&= \mathrm{tr}\left[X_A \rho_A\right] \mathrm{tr}\left[X_B \rho_B\right] \\
&= \mathrm{tr}\left[X_A \otimes X_B (\rho_A \otimes \rho_B)\right],
\end{aligned}$$

where we have used that $\mathcal{C}$ is a local CPC, and thus we can split $P_{\mathcal{C}}$ in local terms; that for any projector, P, it is satisfied that $P^2 = P$; and Eq. (21) with $c(X) = \langle\psi|X|\psi\rangle$. This result implies that the convariance and its bounds are zero. Therefore, we have obtained that $||\rho_{AB} - \rho_A \otimes \rho_B||_1 = 0$, which is only possible if Eq. (24) is satisfied. □

Lemma 10.2 (Union lemma)**.** *Consider a local commuting projector code and two disconnected regions, A and B, i.e., they are separated by a distance bigger than l^*, where l^* is the radius of the support of any commuting projector (see Fig. 19). If A and B are correctable, then $A \cup B$ is correctable.*

Proof. This proof will be given in an exercise class. □

Lemma 10.3 (Holographic lemma)**.** *Consider a local commuting projector code, $\mathcal{C}$, where l^* is the radius of the support of any commuting projector. Given A, B and C regions of $\mathcal{C}$ such that B shield A from C (see Fig. 20a) and the width of B is at least l^*, if A and B are correctable, $A \cup B$ is correctable.*

Proof. As regions A and B are correctable, by definition there exist recovery maps $\mathcal{R}_A$ and $\mathcal{R}_B$ such that

$$\mathcal{R}_A\left[\mathrm{tr}_A(\rho)\right] = \rho \qquad \text{and} \qquad \mathcal{R}_B\left[\mathrm{tr}_B(\rho)\right] = \rho.$$

Given these maps, we want to show that there exists the map $\mathcal{R}_{AB}$ that corrects $A \cup B$, i.e., that satisfies

$$\mathcal{R}_{AB}\left[\mathrm{tr}_{AB}(\rho)\right] = \rho.$$

In order to do this, we define the channel, T_A, acting only on C such that $T_A(\rho_C) = \rho_A \otimes \rho_C$. We can write

$$\rho = \mathcal{R}_B\left[\mathrm{tr}_B(\rho)\right] = \mathcal{R}_B(\rho_{AC}) = \mathcal{R}_B\left(\rho_A \otimes \rho_C\right) = \mathcal{R}_B\left[T_A(\rho)\right] = \mathcal{R}_B \circ T_A(\rho),$$

where we have used that A and C are disconnected by assumption, and thus $\rho_A \otimes \rho_C = \rho_{AC}$ according to Lemma 10.1. We have obtained the map $\mathcal{R}_B \circ T_A$, that acts on AB and recovers the state ρ. □

By definition of distance, we are able to correct any error configuration as long as it has at most d errors. However, it might exist specific error configurations that have many more errors and we are still able to correct them. For example, in the case of the toric code, the critic situation is when errors occur along a string, but, if they are distributed, it is not a problem. As a follow-up of lemma 10.3, we can wonder about the size of the largest correctable square. This is answered with the following fact.

Fact 10.1. *Given a local commuting projector code, $\mathcal{C}$, with distance d, the largest correctable square region is $d \times d$.*

Proof. The idea of the proof consists in constructing the largest correctable square. Let us start with a region A such that $|A| = d$, which implies that A is correctable by definition of the distance. Now, we add a region B (see Fig. 20b) around A with $|B| \leq d$. Lemma 10.3 says that AB is correctable. We iterate this as many times as possible until we will reach a situation where $|B|$ saturates to d. Then, we cannot continue increasing B because it will not be correctable anymore. Therefore, we conclude that squares of side length proportional to d are the largest correctable squares. □

Fact 10.2. *Given a local commuting projector code, $\mathcal{C}$, and a correctable region A, it is satisfied that*

$$S_\rho(AA^c) + S_\rho(A) = S_\rho(A^c) \qquad \forall \rho \in \mathcal{C}. \tag{25}$$

Proof. Consider a state of the code space, $\rho_{AA^c} \in \mathcal{C}$, and its purification, $\Psi_{AA^cR} = |\psi\rangle\langle\psi|$. We denote the erasure channel as

$$T[\cdot] \equiv \mathrm{tr}_A(\cdot).$$

As A is correctable by assumption, there exists a correctable operation, $\mathcal{R}$, such that $\mathcal{R} \circ T(\rho_{AA^c}) = \rho_{AA^c}$. This can be easily transformed to the same statement, but for the purification, i.e.,

$$\mathcal{R} \circ T(\Psi_{AA^cR}) = \Psi_{AA^cR}. \tag{26}$$

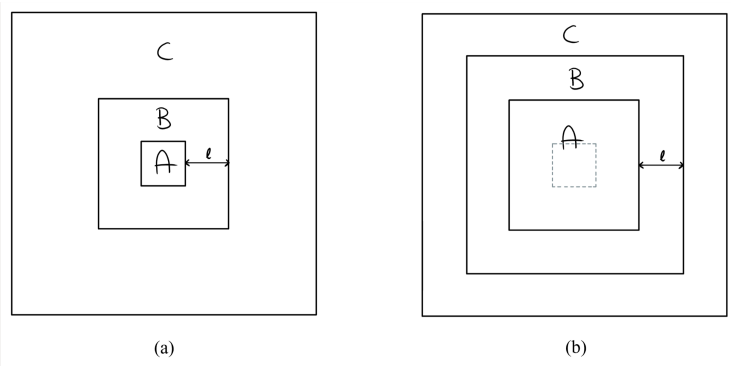

Figure 20: The largest correctable square region is $d \times d$. In order to prove it, we apply Lemma 10.3 iteratively until B is no longer correctable, i.e., $|B| \sim d$.

Recalling that entropy decreases under the action of any map and using Eq. (26), we get the following two inequalities

$$S\left(\Psi_{AA^cR}||\rho_{AA^c}\otimes\rho_R\right) \geq S\left[T(\Psi_{AA^cR})||T(\rho_{AA^c}\otimes\rho_R)\right] = S\left(\rho_{A^cR}||\rho_{A^c}\otimes\rho_R\right),$$
$$S\left(\Psi_{AA^cR}||\rho_{AA^c}\otimes\rho_R\right) = S\left[\mathcal{R}(\rho_{A^cR})||\mathcal{R}(\rho_{A^c}\otimes\rho_R)\right] \leq S\left(\rho_{A^cR}||\rho_{A^c}\otimes\rho_R\right).$$

Thus,

$$S\left(\Psi_{AA^cR}||\rho_{AA^c}\otimes\rho_R\right) = S\left(\rho_{A^cR}||\rho_{A^c}\otimes\rho_R\right).$$

Introducing the definition of conditional entropy, we can write

$$-S(AA^cR) + S(AA^c) + S(R) = -S(A^cR) + S(A^c) + S(R).$$

This implies Eq. (25) since Ψ_{AA^cR} is a pure state, and thus $S(AA^cR) = 0$ and $S(A^cR) = S(A)$. □

At this point of the section, we have all ingredients to prove a theorem that does not allow local commuting projector codes to be ideal, i.e., to satisfy Eq. (23). The theorem is stated as follows.

Theorem 10.2. *For a local commuting projector code $[n, k, d]$ on a two-dimensional lattice, it is satisfied that*

$$kd^2 \leq \alpha n, \tag{27}$$

where α is a constant.

Proof. Consider the state $\rho \in \mathcal{C}$ such that

$$\rho = \frac{\mathbb{I}_{\mathcal{C}}}{\mathrm{tr}(\mathbb{I}_{\mathcal{C}})} = \frac{\mathbb{I}_{\mathcal{C}}}{2^k},$$

where $k = S(ABC)$. Consider also the partition of the lattice in Figure 10.2 with regions A and B taken as large as possible. Fact 10.1 says that $|A|$ and $|B|$ can be at most proportional to d^2, and thus $R \sim d$. The union lemma (Lemma 10.2) states that the union of all regions A is correctable as well as the union of all regions B. Fact 10.2 says that Eq. (25) is fulfilled for regions A and for regions B individually. Thus, we have

$$S_\rho(ABC) + S_\rho(A) = S_\rho(BC),$$

$$S_\rho(ABC) + S_\rho(B) = S_\rho(AC).$$

Using the subadditivity of the entropy, these equations can also be written as

$$S_\rho(ABC) = S_\rho(BC) - S_\rho(A) \leq S_\rho(B) + S_\rho(C) - S_\rho(A),$$
$$S_\rho(ABC) = S_\rho(AC) - S_\rho(B) \leq S_\rho(A) + S_\rho(C) - S_\rho(B).$$

Adding both equations, we get

$$S_\rho(ABC) \leq S_\rho(C) \leq |C|.$$

Recall that $k = S(ABC) \propto |C|$. Now, we want to relate the $|C|$ with the size of the lattice. It is easy to see that

$$|C| \sim \frac{\sqrt{n}}{R}\frac{\sqrt{n}}{R} = \frac{n}{d^2} \geq ck,$$

where c the constant of proportionality. This implies Eq. (27). □

From Theorem 10.1, we conclude that, if we increase the code size, i.e., n, the ratio $\frac{kd^2}{n}$ is always upper bounded by a constant. In other word, the ideal scenario cannot happen since $kd^2 = (cnt)n^3 \geq (cnt)n$. Note that the toric code (see Section 8) saturates the bound of Eq. (27). There exists similar bounds for three-dimensional codes.

Theorem 10.1 not only give a bound, but also restricts the form of the logical operators. It might not be straightforward, but logical operators must live either in regions A or regions B and must go through regions C.

11 Thermal noise (self-correction)

In this section we want to introduce a new type of noise called thermal noise. A physical system can never be completely isolated, but the system is in contact with an environment. The environment and the system interact in such

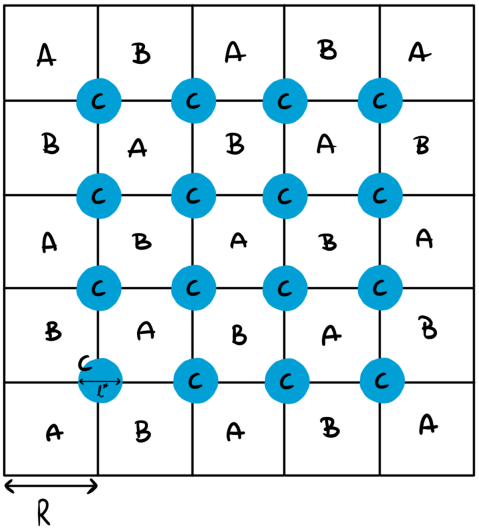

Figure 21: Partition of the lattice used to prove Theorem 10.1. Regions A and B are taken as large as possible and radius of C is at least $\frac{l^*}{2}$.

a way that the system increases its temperature. The gain of this temperature is known as thermal noise since it can cause loss of logical information. In this section, we review a model to describe thermal noise and we relate it to error correction.

In order to study thermal noise, we consider a scenario made of a system, S, and an environment, E (see Fig. 22). Initially the system is out of equilibrium and the environment is at equilibrium at temperature T. At long times, the system reaches thermal equilibrium at temperature T due to the interaction with the environment. For simplicity, we make two assumptions. First, we assume that the interaction between the system and the environment is local. In addition, we consider the limit of large environment, which says that the environment keeps no memory of the system for an infinitesimal amount of time.

As we have seen in previous sections, classical error correction consider a system composed of bits. The code space is defined as the ground space of a (classical) Hamiltonian. Since the system is classical, the eigenstates of the Hamiltonian are the canonical basis states. In other words, for a system of N spins, any state $|\sigma_1, \ldots, \sigma_N\rangle$, where $\sigma_j \in \{0, 1\}$, is an eigenstate of the (classical) Hamiltonian, H. We want to describe a dynamical process such that at long time system converges to a classical probability of distribution

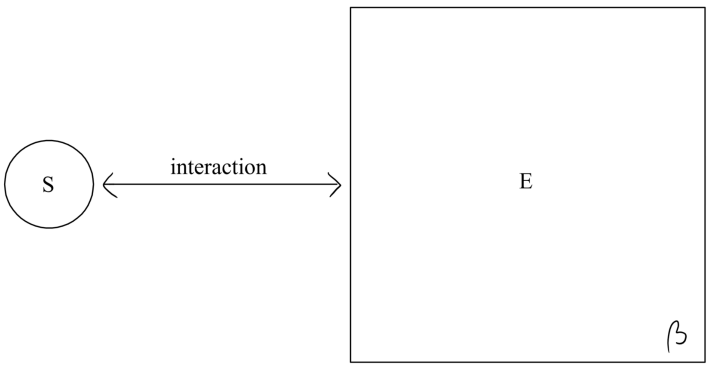

Figure 22: The scenario considered to model thermal noise consists of a system, S, that interacts with an environment, E, which is at equilibrium at temperature, $\beta = \frac{1}{T}$.

of the form

$$p \sim e^{-\beta H},$$

where $\beta = \frac{1}{T}$.[22]. In order to model the thermal dynamics we use the so-called Glauber dynamics. The Glauber dynamics can be understood as an algorithm which works with the following steps:

1. Pick a site at random.
2. Flip the bit at this site.
3. Evaluate the energy difference, i.e., the difference between the energy of the configuration after and before the flip, ΔE.
4. Keep the flip with probability

$$p_{\text{flip}} = \frac{1}{e^{-2\beta\Delta E} + 1}.$$

Theoretically, we consider that these four steps happen in an infinitesimal time, but, when we simulate the process in the computer, we take a discrete time. Note that the energy difference produced by a flip is local, i.e., it only depends on a small region around this site. This locality comes from the

[22]As an example we can consider the Hamiltonian of the Ising model, which is

$$H = -J \sum_{\langle i,j \rangle} \sigma_i \sigma_j + h \sum_j \sigma_j,$$

where $\langle i, j \rangle$ denote a sum over nearest neighbours and J and h are constants.

assumption of a local interaction between the system and the environment, which leads to a local Hamiltonian.

Completely generalising the classical scenario to quantum systems is difficult. A flip on a classical system implies that the system changes from one eigenstate to another. Therefore, the thermal dynamics of a classical system simply describes a random walk along a graph made of the eigenstates of the classical Hamiltonian. When we want to extend this dynamics to quantum systems, we have to deal with the eigenstates of a quantum Hamiltonian, which are in general non commuting[23]. These eigenstates can be large entangled objects, and thus they are often complicated to write down and computationally inefficient. There exist ways to overcome these difficulties and, actually, one of the most sophisticated algorithm is based on the Glauber dynamics. However, this field is still on research.

In the setting of error correction we have mainly seen stabilizers codes. We have also studied a slightly more general scenario, but restricting ourselves to commuting projector codes. The purpose of this restriction was to keep the ability of correcting X errors and Z errors independently. In fact, constructing non commuting codes is very demanding and only few ways are known[24]. In this section, we also consider the commuting scenario, i.e., we assume that the quantum Hamiltonian is commuting.

The thermal dynamics of a quantum system can be described using the so-called Davies master equation. Consider a system described by a Hamiltonian such that

$$H_{SE} = H_E + H_S + \varepsilon H_{\text{int}}, \qquad \text{where } H_{\text{int}} = \sum_\alpha S_\alpha \otimes E_\alpha.$$

Here, H_E, H_S and H_{int} respectively characterise the environment, the system and the interaction between them; S_α spans the operator algebra of the

[23]A Hamiltonian is said to be commuting if it can be expressed as a sum of local terms such that all these terms commute. Mathematically, given a lattice, Λ, and a subset of the lattice, A, a commuting Hamiltonian can be written as

$$H = \sum_{A \subset \Lambda} h_A,$$

where $h_A = 0$ for A larger than some ball and $[h_A, h_{A'}] = 0 \ \forall A, A' \subset \Lambda$.

[24]We will see some in further sections.

system[25]; E_α acts on the environment[26]; and ε is a constant. We consider the following assumptions on the scenario:

1. The interaction between the system and the environment is weak, i.e., $\varepsilon \to 0$ [27].

2. The environment is Markovian, i.e., it has no memory about the system.

3. The initial state is uncorralated, i.e., the initial state can be written as $\rho_0^S \otimes \rho_\beta^E$, where the system is in a random state, ρ_0^S, and the environment is in any Gibbs state, $\rho_\beta = \frac{1}{Z} e^{-\beta H_E}$.

The unitary evolution of the quantum system, which is initially in $\rho_0^S \otimes \rho_\beta^E$, is given by

$$U_t = e^{-itH_{SE}}.$$

Note that the system, S, and the environment, E, evolve together. If we trace the environment out, we get a quantum process, T, which only depends on the system, S, due to the markovian environment. Mathematically, the quantum process T is

$$T_t\left(\rho^S\right) = \mathrm{tr}_E\left[U_t \rho_0^S \otimes \rho_\beta^E U_t^\dagger\right].$$

Using the strong assumptions 1-3, it can be shown that the quantum process, T, can also be expressed as

$$T_t\left(\rho^S\right) = e^{t\mathcal{L}^0\left(\rho^S\right)}, \tag{28}$$

with

$$\mathcal{L}^0(\rho) = -i\left[H_s, \rho\right] + \sum_{\omega, \alpha(k)} \chi_{\alpha(k)}(\omega)\left[S_{\alpha(k)}(\omega)\rho S_{\alpha(k)}^\dagger(\omega) - \frac{1}{2}\left\{S_{\alpha(k)}^\dagger S_{\alpha(k)}, \rho\right\}\right],$$

[25]For example, if we are dealing with a stabilizer code, the operator S_α can be $S_\alpha = \sigma^{\alpha_1} \otimes \sigma^{\alpha_2} \otimes \cdots \otimes \sigma^{\alpha_n}$, where σ^{α_i} are Pauli matrices and α_i denote the spin on which σ^{α_i} acts.

[26]The operators E_α can be any operator acting on the environment different to the identity.

[27]Formally speaking, this assumption restricts ε to be smaller than the smallest eigenvalue difference in the spectrum of the system Hamiltonian. Note that this assumption is not reasonable for many-body Hamiltonians that are not commuting due to the following. If the Hamiltonian is a many-body and commuting Hamiltonian, its spectrum has constant energy differences. Nevertheless, if the many-body Hamiltonian is a non-commuting, it has 2^N eigenvalues and they are exponentially close, where N is the number of particles of the system. This implies that ε must be exponentially small for a many-body non-commuting Hamiltonian, which makes the assumption unreasonable. If the quantum system is small, i.e., N is small, the assumption is also valid.

where the summation runs over all ω and over all sites k in S, $\chi_{\alpha(k)}$ are scalar functions related to the two-point correlations of the environment, and $S_{\alpha(k)}(\omega)$ are the Fourier coefficients of the time evolved $S_{\alpha(k)}$ operator, i.e.,

$$e^{-itH_S} S_{\alpha(k)} e^{itH_S} = \sum_{\omega=-\infty}^{\infty} e^{it\omega} S_{\alpha(k)}(\omega).$$

If the Hamiltonian H_S is composed by local commuting terms, then $e^{\pm itH_S}$ break up as a product of terms. The terms that do not intersect with the given $\alpha(k)$ commute with $S_{\alpha(k)}$, and thus cancel out. This implies that each $S_{\alpha(k)}(\omega)$ is local around $\alpha(k)$. Moreover, the operators $S_{\alpha(k)}(\omega)$ are bounded because $e^{-itH_S} S_{\alpha(k)} e^{itH_S}$ is bounded.

In previous sections we have seen noise models consisting on flipping or erasing bits/qubits with certain probability. Imagine a situation where we can correct in the lab all flip and erase errors. As laboratories cannot get zero temperature, and thus experiments happen at a few kelvin, some thermal noise is always left. Assuming that the thermal noise is well characterised by Eq. (28), in the rest of this section we try to understand whether error correction procedures are good against thermal noise. We will see that thermal noise can be corrected and that, in addition, it has a special property called self-correction.

11.1 Phenomenology

In a course of statistical (classical) mechanics, one studies statical properties of the system. A typical problem consists on considering a Gibbs state of a certain system and asking if the system has a phase transition, i.e., whether there exists a critical temperature under which there is no trivial magnetisation. In other words, we consider a system under an external magnetic field that forces the state to a certain magnetisation. If we take the magnetic field to zero adiabatically, does the system preserve the magnetisation induced by the magnetic field or does it converge to an equilibrium state without magnetisation? At low temperatures, the system keeps the magnetisation and we say that the system is in a bistable phase. However, if the system is at high temperature, the system converges to an equilibrium state that has no magnetisation. The Ising model explains these physical processes as phase transitions.

In the Ising model for one dimensional systems, there are no phase transitions, i.e., for any temperature arbitrary close to zero the system does not

have any residual magnetisation. In order to understand this lack of phase transition, let us study the dynamical picture. Consider a system made of spins in a lattice with a certain configuration. The system is in contact with an environment at temperature β, which can be arbitrary small. We use the Metropolis algorithm to describe the dynamics of this scenario. The Metropolis algorithm consists of the following steps:

1. Pick a site randomly.
2. Flip the bit in this site.
3. If the energy of the new configuration decreases, keep the flip. If the energy of the new configuration increases, keep the flip with probability

$$p_{\text{flip}} \sim e^{-2\beta\Delta E}. \tag{29}$$

The energy of the system is proportional to the number of times that a spin in state zero is next to a spin in state one. Therefore, if the initial configuration of the system is all spins at state $|0\rangle$, the initial energy is zero (see Fig. 23). When we flip a bit, the energy increases $\Delta E = 2J$, i.e., one single flip costs energy. If the temperature of the environment, T, is arbitrary small, the probability that the flip remains is exponentially suppressed, but not zero (Eq. (29)). When a flip happens, i.e., the system has already one spin in state $|1\rangle$, flipping a neighbouring spin does not increase the energy of the system. In other words, the configuration with only one spin at state $|1\rangle$ has the same energy as the configuration with several spins at $|1\rangle$ as long as they are next to each other. These accumulations of states $|1\rangle$ are known as islands or droplets. We can conclude that, in the 1D Ising model, droplets take energy to create, but not to extend. This explains why there is no phase transition in the 1D Ising model. If we start with a system that has a well defined magnetisation, such as all spins at $|0\rangle$, the system can flip to a configuration made of all $|1\rangle$ in a constant amount of time no matter how low the temperature of the environment is.

Consider now the 2D Ising model with an initial configuration made of all spins in $|0\rangle$. Figure 24 explains that a single flip costs $\Delta E = 4J$ and one and two neighbouring flips increase the energy of the system to $6J$ and $8J$, respectively. When the system has a droplet of three $|1\rangle$, it does not cost any energy to extend the droplet to four spins at $|1\rangle$. In other words, the transformation from a non convex droplet into a convex droplet does not require energy. However, and in contrast to the 1D Ising model, extending a droplet does cost energy in the 2D Ising model. In particular, the probability

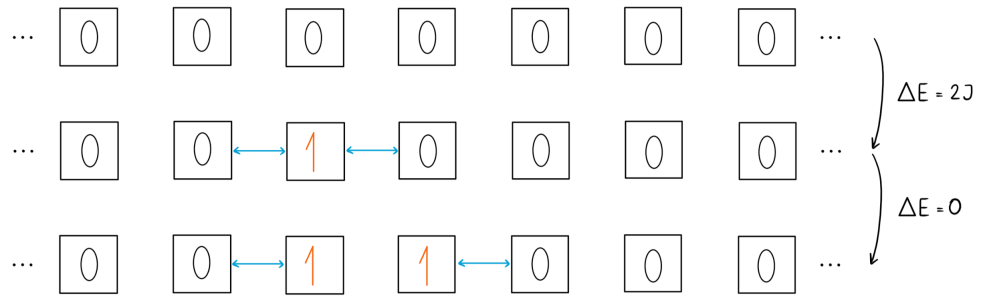

Figure 23: In the 1D Ising model, droplets take energy to create, but not to extend.

to create a droplet with area B is proportional to $e^{-\beta|\partial B|}$. This implies that the probability that the configuration of a system changes from all spins in $|0\rangle$ to all spins in $|1\rangle$ is exponentially small. In short, droplets in the 2D Ising model cost energy to create and also to extend, except when a non-convex droplet transform to a convex droplet.

At high temperatures the argument we have followed above still holds, i.e., creating each flip has still a probability smaller than one. Nevertheless, it is known that at high temperatures the 2D Ising model has a single grounf state. This is explained due to the entropy. If the environment is at high enough temperature, an exponential number of different types of droplets can be created. In other words, the entropic contribution beats the energy contribution and takes the system to a single state. Unfortunately, an analytic treatment of the entropy is very difficult.

Thermodynamics has an interesting property when it is though in terms of error correction. Magnets were one of the first forms of error correction for digital computing. We usually do not think about a magnet as error correcting systems because we do not apply any error correction, but, in fact, they use self-correction. Each magnetic domain is made of many spins pointing on average in one direction which encode a redundancy of the spin information. When the magnets are magnetised, they stay magnetised and this is exactly what self-correction means. Self-correction is a remarkable property of thermal noise that creates errors by flipping spins at a certain rate, but its dynamics also forces the system to keep the initial information. Otherwise speaking, thermodynamics allow to create droplets, but also prevent droplets of growing and spreading. This can be viewed as a dual effect of creating noise and correcting against noise at the same time.

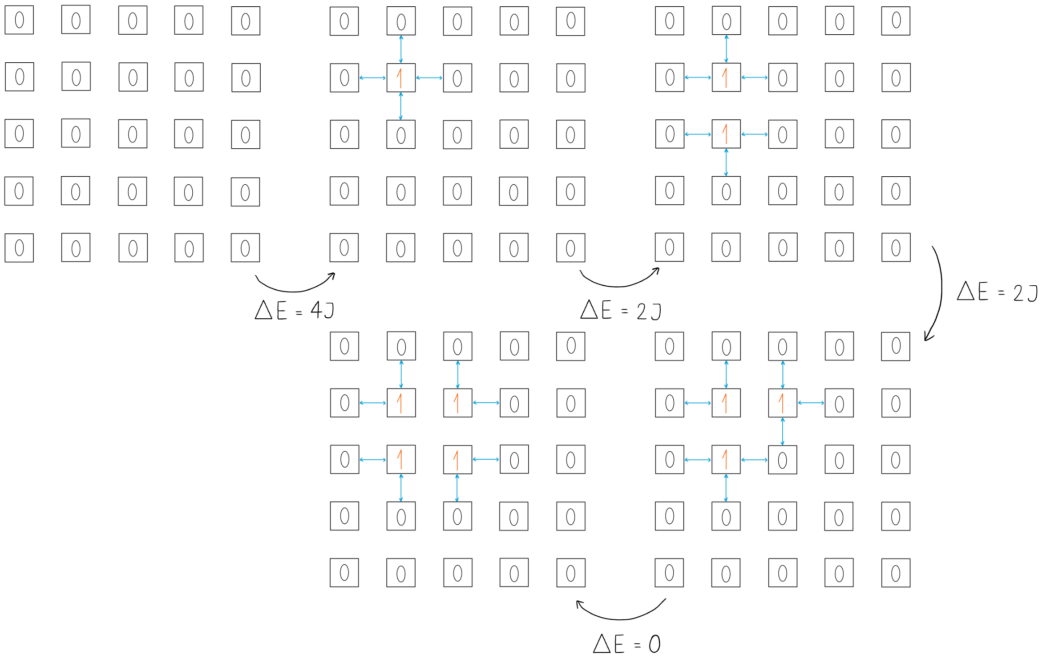

Figure 24: Droplets in the 2D Ising model cost energy to create and also to extend, except when a non-convex droplet transform to a convex droplet.

At low temperature (i.e., temperature below the phase-transition temperature) the system has two metastable states, ρ_β^0 and ρ_β^1 (see Fig. 25). Due to thermodynamics, jumps from one metastable state to the other are possible. In order a jump to happen, a droplet has to be created and it has to extend across the whole system. In particular, the energy barrier that a system has to overcome to go from one metastable state to the other is $\Delta E \sim L$, where L is the size of the lattice. The Arrhenius law says that this flip from ρ_β^0 to ρ_β^1 occurs in a time

$$\tau \sim e^{\beta \Delta E} \sim e^{\beta L/\xi}. \tag{30}$$

The Arrhenius law works for many different systems, but it is not the proper behaviour of a phase transition. The proper behaviour is

$$\tau \sim e^{\beta F/\xi},$$

where F is the free energy.

In the beginning of these notes, we argued the importance of error correction. We said that it adds an enormous overhead on to the physical necessities of doing quantum computation. If we had a self-correcting code, then we would

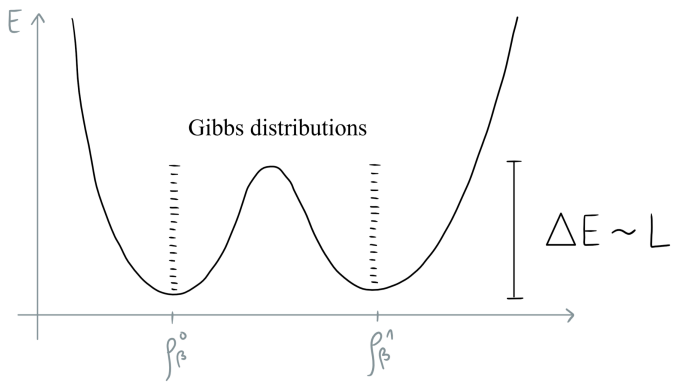

Figure 25: In the 2D Ising model at low temperature, there exist two metastable states, ρ_β^0 and ρ_β^1. Jumps from ρ_β^0 and ρ_β^1 (or viceversa) require to overcome the energy barrier, which is $\Delta E \sim L$, where L is the size of the lattice. The two metastable states are populated according a Gibbs distribution.

not need to implement any error correction because the system would do it for itself. As we have said in previous section, in quantum error correction the 2D toric code correspond to two copies of the 1D Ising model. The above discussion about the 1D Ising model has conclude that, even at very low temperature, creating a string on the toric code can happen with a certain probability. Then, the string takes no energy to extend, i.e., its extension happen in constant time. This argument implies that the 2D toric code does not have the self-correction property. Stated differently, if we do not actively correct thermal noise when using the 2D toric code, the logical information is lost in a constant[28] amount of time.

We can look for the self-correction property in higher dimensions. Two copies of the 2D Ising model correspond to the 4D toric code. Figure 26 depicts the stabilizer operators of the 3D and the 4D toric code. The logical operators of the 3D toric code are one string and one plane[29] which intersect in a single point (see Fig. 27). The logical operators of the 4D toric code are two planes that intersect in a single point. The 4D toric code would have the self-correction property because the 2D Ising model has it, but a 4D error correcting code is not a realistic approach.

The lack of self-correction in the 2D and 3D toric code is not specific prop-

[28] According to the system size.

[29] Either the Z logical operator is a plane and the X a string, or vice versa.

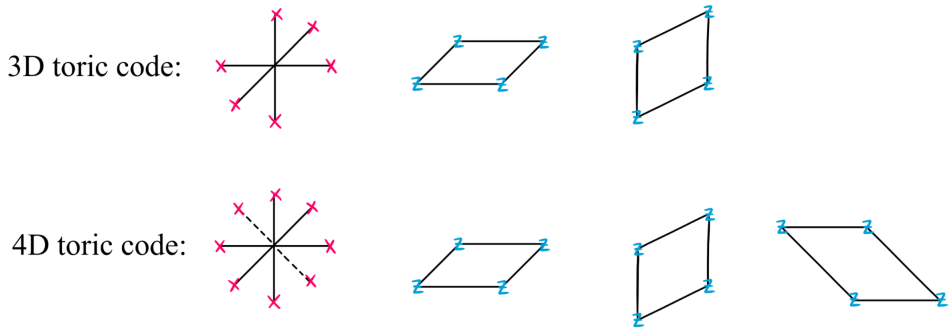

Figure 26: Stabilizer operators of the 3D and the 4D toric code. Note that a cube of Z operators cannot be a stabilizer operator because it does not commute with the X stabilizer operator.

erty of this code unfortunately. The following theorem proves that for any commuting projector code on three dimensions self-correction is not possible. Even more, this theorem has also been shown for some non-commuting codes.

Theorem 11.1 (No-go theorem)**.** *Consider a commuting projector code in D dimensions. If the X logical operator has dimension d_1, then the Z logical operator must have dimension $d_2 = D - d_1$. This implies that the energy barrier is $\Delta E = \min\{(c_1(d_1 - 1), c_2(d_2 - 1)\}$, where c_1 and c_2 are constants.*

For three dimensions, $D = 3$, the no-go theorem shows that there can be no quantum error correcting code that is self-correcting. When $D = 3$, the theorem implies that the Z logical operators are a plane and the X logical operators are a string (or vice versa) for all codes (see Fig. (27). Stated differently, either d_1 or d_2 is equal to one, and thus the energy barrier is always constant. This implies that the time at which a flip occurs is never exponentially suppressed, but it is a constant, which makes self-correction impossible.

Finding a self-correcting quantum error correcting code in less than four dimensions was the major open problem in quantum error correction research for a long time. In order to get around the no-go theorem, physicists have tried the following strategies:

1. *Interactions between the syndromes*
   One proposal that uses interactions between the syndromes consists in embedding a 2D toric code in a 3D liquid. When there is a syndrome,

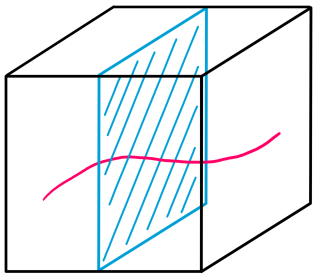

Figure 27: In three dimensions, $D = 3$, the no-go theorem says that the Z logical operators are a plane and the X logical operators are a string (or vice versa) for all codes.

the engineered interactions work such that the viscous force on the syndromes prevent them from moving away from each other.

2. *Fractal logical operators*
   The fractal logical operators have irrational dimension, which make the energy barrier growing with the logarithm of the system size. Unfortunately, when the entropic contribution plays a role, the fractal logical operator fail. The Haah code is an example of a code with fractal logical operators.

3. *Entropic doping*

4. *Fractal embeddings*
   We know that a 2D Ising model has a phase transition. This phase transition remains even if we cut little squares of the lattice. In fact, the maximal amount of lattice that we can cut corresponds to the Sierpinski carpet (see Fig. 28), which has a fractal dimension. With two cut 2D Ising model, we can construct a toric code with dimension 2.9. The problem of this approach is that, even if the effective dimension is smaller than three, the code still lives in a 4D space. Then, the 4D toric code must be embedded to three dimensions, but this embedding makes either the logical operators non-communting or the stabilizer operators non-local.

5. *Dissipative codes*
   If errors occur in the system, we only need to extract the entropy that has been added to the system in order to preserve the purity of the encoded subspace. Thermal noise does exactly this process. Interestingly, a dissipative code exists in 2D and it is realated to the phase transition in the cellular automaton in 1D.

6. *Non-commuting codes*
   At the moment, there exists no non-commuting code that works. In fact, analysis of non-commuting codes is very hard because the mathematical model of the thermal noise requires a commuting Hamiltonian.

The majority of these attempts either failed or still have not been proven to work. Some approaches theoretically succeed, but they were experimentally not practical. Nowadays, it is believed that self-correction for quantum computation is not going to be a useful option.

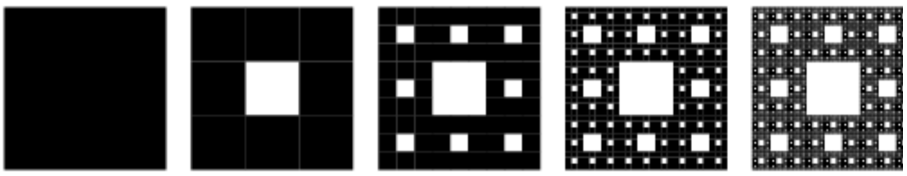

Figure 28: Sierpinski carpet. Picture taken from the website http://mathworld.wolfram.com/SierpinskiCarpet.html.

12 Surface codes

In previous sections we have seen two quantum error correcting codes: the Shor code and the toric code. Although both codes are stabilizer codes, they have a very different nature. The toric code is defined on a surface, while the Shor code does not consider any topology. In this section we first want to consider planar codes, which are codes defined on a surface with non-periodic boundary conditions, and compare them with the toric code, which have periodic boundary conditions. Then, we study another surface code known as the colour code.

12.1 Planar codes

The toric code is a specific instance of a class of codes called surface codes. In particular, the toric code is defined in any surface with periodic boundary conditions. Implementing periodic boundary conditions on a many body object in a lab can become very messy. Experimental physicists overcome this problem by using a planar code. A planar code is a stabilizer code implemented on a planar surface with non-periodic boundary conditions (see Fig. 29). For simplicity, we consider a quadrilateral surface.

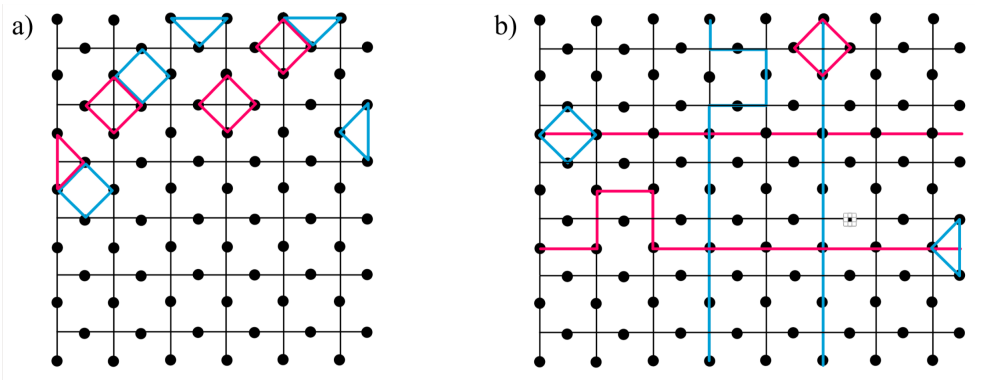

Figure 29: Planar codes are stabilizer codes implemented on a planar surface with non-periodic boundary conditions. a) The X (pink) and Z (blue) stabilizer operators in the bulk are plaquettes with a diamond form, whereas, at the edges, the stabilizers have a triangular form. Note that all stabilizers commute because they overlap at an even number of qubits. b) The X and Z logical operators of the planar code are strings of X and Z operators. Note that the logical operators can be deformed. Note further that not all triangular operators commute with the logical operators, and thus this triangular operators cannot be part of the stabilizer group.

The X and Z stabilizer operators in the bulk are plaquettes with a diamond form, whereas, at the edges, the stabilizers have a triangular form. We can easily note that all stabilizers commute because, when different type of stabilizers meet, they always overlap on two qubits. Recall that, in the toric code, the product of all X stabilizers and Z stabilizers (independently) is equal to the identity (Eq. 15). The non-periodic boundary conditions of the planar codes, however, break this property in such a way that the product of all X (Z) stabilizers leave a line of X (Z) on the top and bottom (right and left) boundaries (see Fig. 30). Therefore, we cannot use this property to compute the number of logical qubits of the planar code as we did for the toric code. We will do it with the logical operators.

The X and Z logical operators of the planar code are strings of X and Z operators, respectively (see Fig. 29). The obvious main difference between planar codes and the toric code are the boundary conditions. In the case of the toric code the logical operators had to wrap around itself, whereas on the case of the planar code the string only needs to be attached somewhere on the boundary. Logical operators can be deformed as shown in of Fig. 29. This deformation corresponds to a product of a straight string with a stabilizer, which gives a new logical operator.

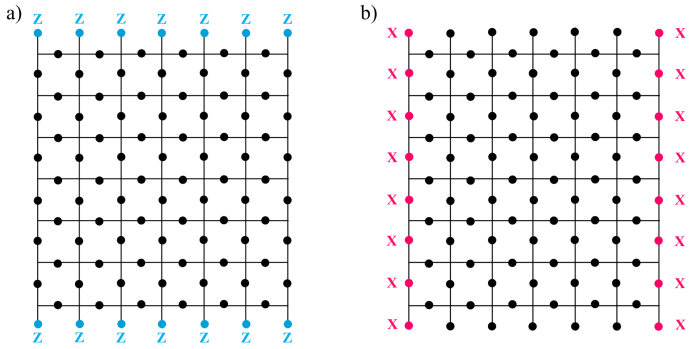

Figure 30: The product of all X (Z) stabilizers leave a line of X (Z) on the top and bottom (right and left) boundaries.

If we consider a logical operator made of a string of X operators, there is at least one Z stabilizer at the boundary that does not commute with the string of X. Thus, this stabilizer operator cannot be part of the stabilizer group (see Fig. 29). This gives us an intuition of the fact that quadrilateral planar codes only encode one logical qubit[30]. The number of logical qubits increases if we consider a surface code defined on a surface with more sides. For example, a planar code on an octagon encodes three logical qubits, which correspond to eight edges divided into two types of logical operators minus one logical qubit that is redundant (see Fig. 31).

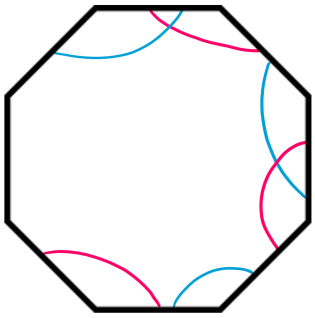

Figure 31: A planar code on an octagon encodes three logical qubits, which correspond to eight edges divided into two types of logical operators minus one logical qubit that is redundant.

We have described the surface codes in a square lattice, but they can be

[30]Here we do not want to go over the rigorous proof, but only give a feeling of its idea.

implemented on any lattice. Consider any lattice with one qubit placed in each edge. At every face we define a Z stabilizer in a form of a plaquette of Z operators and, at each vertex, we define a X stabilizer as a cross of X operators (see Fig. 32). In the next subsection we study a surface code which is not defined in a square lattice.

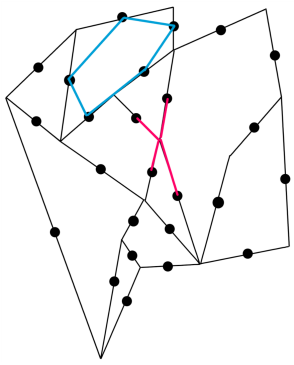

Figure 32: Surface codes can be implemented on any lattice. Qubits are placed on the edges, Z stabilizers (blue) are defined on faces and X stabilizers (pink) are defined on crosses.

12.2 Colour codes (2D)

Colour codes are codes implemented on a trivalent lattice (i.e., all vertices are connected to three edges) with 3-colourable faces (i.e., faces can be coloured in three different colour such that no two faces that share a boundary have the same colour). Edges can also be coloured and, typically, 3-coloured graphs are coloured in red (r), green (g) and blue (b). There is a qubit on each vertex of the lattice. For simplicity we assume periodic boundary conditions and the honeycomb lattice, which is the simplest example of a trivalent 3-colourable lattice (see Fig. 33).

The stabilizer operators of the colour codes act on qubits living on the same face. In particular, for every single face we define a X and a Z stabilizer operator as

$$X_{f_c} = \prod_{j \in F} X_j \qquad \text{and} \qquad Z_{f_c} = \prod_{j \in F} Z_j,$$

where F is a face and $c \in \{r, g, b\}$ denotes the colour of the face. Any two faces of the lattice meet at two vertices, and thus all stabilizers commute. Indeed, the set $\{X_{f_c}, Z_{f_c}\, \forall f_c\, \forall c\}$ form an over-complete set of stabilizers.

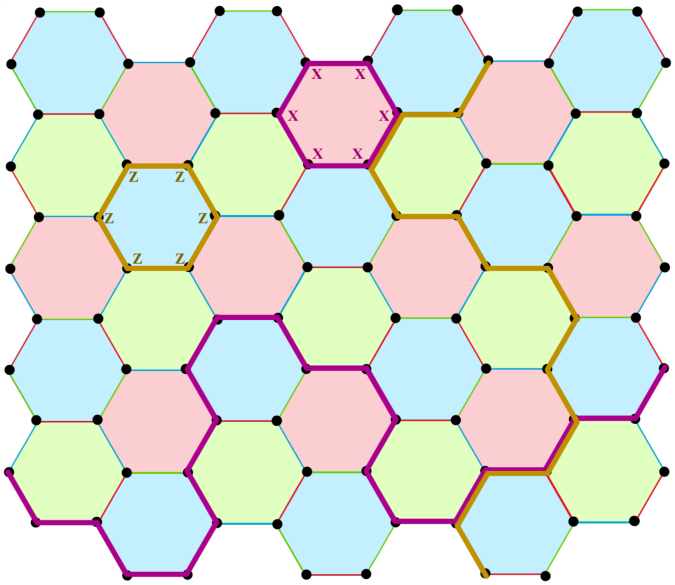

Figure 33: Colour codes are codes implemented on a trivalent lattice with 3-colourable faces. Qubits are placed on the vertices of the lattice and stabilizers operators are defined on each face. Logical operators are vertical and horizontal strings.

The product of all stabilizers defined on faces of the same colour satisfies

$$\prod_{f_r} X_f = \prod_{f_g} X_f = \prod_{f_b} X_f = \prod_{\text{full lattice}} X_f, \tag{31}$$

$$\prod_{f_r} Z_f = \prod_{f_g} Z_f = \prod_{f_b} Z_f = \prod_{\text{full lattice}} Z_f. \tag{32}$$

The equations (31) and (32) imply that a colour code on a torus encodes four logical qubits.

In order to completely describe the colour code, we need to construct its logical operators. Let us consider a string across the lattice. We can break the string up on sections of different colours and define operators on each section such that

$$X_\gamma^c = \prod_{j \in V_c} X_j \qquad\qquad Z_\gamma^c = \prod_{j \in V_c} Z_j,$$

where γ is a string and $c \in \{r, g, b\}$ denotes the colour of the section. The reason for this breaking is that every X_γ^c is an independent formulation of

the same string. In other words, since each X_γ^c of any c fully specifies the string, we only need to specify the operators of a single colour. We can easily see that each X_γ^c and each Z_γ^c commute with all stabilizers. For any string, it is satisfied that

$$X_\gamma^r X_\gamma^g X_\gamma^b = \mathbb{I} \qquad \text{and} \qquad Z_\gamma^r Z_\gamma^g Z_\gamma^b = \mathbb{I}. \tag{33}$$

This means that there are only two colours independent and the third is dependent. Therefore, the set of independent logical operators of the colour code is $\{X_\gamma^c, X_\gamma^{c'}, Z_\gamma^c, Z_\gamma^{c'}\}$, where $c \neq c'$. If we take a string in the vertical direction, even if it overlaps in several faces with the horizontal string, they overlap in an even number of qubits, and thus X and Z logical operators commute.

Logical operators of the colour code not only can be deformed as in the toric code, but they can also have bifurcations. As usual, a deformation of a logical operator is done by multiplying a string with a stabilizer. The bifurcations appear due to the colours and the property of Eq. (33). Consider a green logical operator. If we multiply the green logical operator with a blue logical operator, we get a red logical operator because the logical operators satisfy Eq. (33). Consider now the same green logical operator, but we multiply it with a red stabilizer. As the stabilizer formalism says, we get a new logical operator and it consists on a green string with a bifurcation (see Fig. 34). The bifurcations must close, i.e., the branches must get together again at some vertex.

13 Fault tolerance

In all previous sections we have addressed error correction from the point of view of errors that occur when storing information. Nevertheless, the application of gates on qubits are also a source of errors. A good error correction protocol must answer how to keep the logical information as well as as well as how to perform gates in an efficient way. Fault tolerance is the field of error correction that deals with performing gates. In classical computing, fault tolerance is only an academic exercise because the elements of a computer can perform gates with errors of the order of 10^{-18}. In contrast, fault tolerance is a crucial field in quantum error correction as we will see in this section.

A quantum algorithm consists on three steps:

1. State preparation of an initial state, $|\psi_0\rangle$. For simplicity, the initial

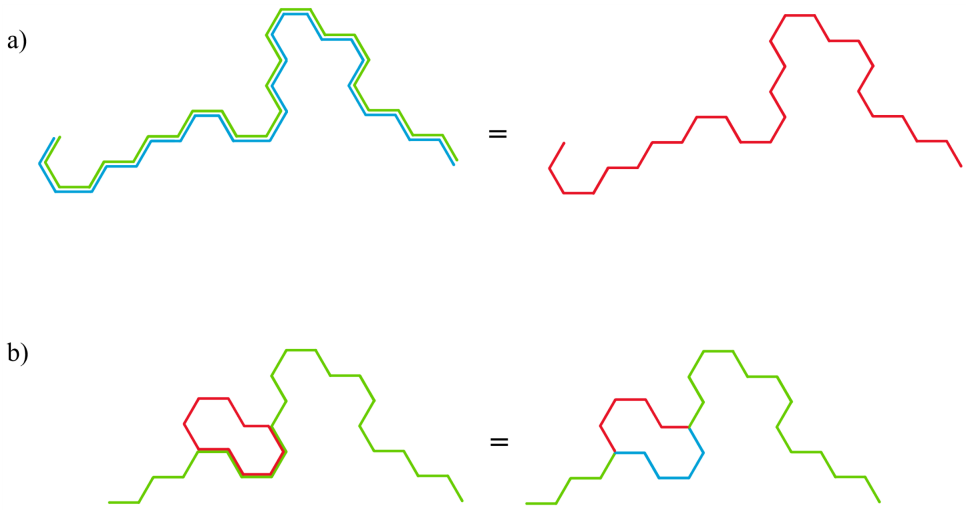

Figure 34: a) A blue logical operator multiplied by a green logical operator gives a red logical operator due to Eq. (33). b) Logical operators of the colour code can have bifurcations, but they must close.

state is often considered the product state $|\psi_0\rangle = |0\rangle^{\otimes N}$, where N is the number of qubits of the system.

2. State evolution, which is called quantum circuit and denoted by a unitary U.

3. Measurement.

Figure 35 shows a typical scheme to describe a quantum algorithm.

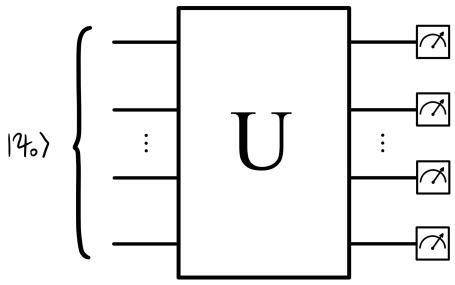

Figure 35: Scheme of a quantum algorithm, where $|\psi_0\rangle$ is the initial state and U the quantum circuit.

For any N-qubit state, $|\psi\rangle$, there exists a unitary, U, such that

$$|\psi\rangle = U|\psi_0\rangle,$$

where U is a $2^N \times 2^N$ matrix that can involve any type of correlations[31]. In the following we want to see that we can perform any unitary, U, from a finite set of two-qubit unitaries. Therefore, as long as we have access to this finite set of unitaries, we are able to perform any unitary operation on any system (up to some precision). For that, we consider the following three steps:

1. *Step 1*
   A N qubit unitary can always be decomposed in, at most, $\mathcal{O}(2^N)$ two-qubit unitary operators (see Fig. 36). Note that this a week statement because we need access to arbitrary two-qubit unitaries. Moreover, a quantum circuit made of an exponential number of gates is not efficient, and thus we are interested on quantum circuits which have a polynomial number of gates.

2. *Step 2*
   Any two-qubit unitary, U, can be written as two single-qubit unitaries, V and W, and a CNOT gate[32] (see Fig. 37). This step is more efficient than the first one since it goes from completely general two-qubit unitaries to only single-qubit unitaries and a CNOT, but still the single-qubit unitaries are arbitrary.

3. *Step 3*
   Any single-qubit unitary can be approximated by a product of gates in $\{X, Z, H, T\}$, where H and T are the Hadamard gate and the $\frac{\pi}{8}$-gate, respectively. Mathematically, we write

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad \text{and} \qquad T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}.$$

   In concrete, the theorem states the following.

   Theorem 13.1 (Solvay-Kitaev)**.** *We need a $\mathcal{O}[\log(1/\epsilon)]$ number of gates from $\{X, Z, H, T\}$ to approximate an arbitrary two-qubit unitary*

[31] In experiments, unitaries are typically restricted to, at most, three body interactions.

[32] Recall that a CNOT gate is a two-qubit gate that flips the second qubit (the target qubit) if and only if the first qubit (the control qubit) is in $|1\rangle$. Mathematically,

$$CNOT(1,2) = |0\rangle_1\langle 0| \otimes \mathbb{I}_2 + |1\rangle_1 1 \otimes X_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

where qubit 1 is the control qubit and qubit 2 is the target qubit.

with precision ϵ[33].

Putting all three steps together, we have seen that an arbitrary unitary operator, U, can be approximated by $\mathcal{O}[(2^N)\log(1/\epsilon)]$ number of gates in $G \equiv \{X, Z, H, T, CNOT\}$, which are one- and two-qubit gates. Indeed, the set $G \equiv \{X, Z, H, T, CNOT\}$ is a universal set of gates.

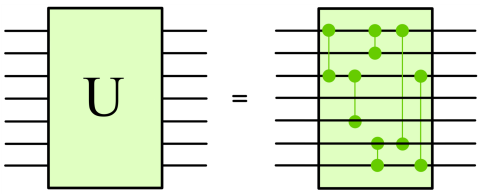

Figure 36: A N qubit unitary can always be decomposed in, at most, $\mathcal{O}(2^N)$ two-qubit unitary operators.

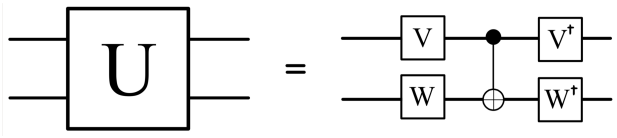

Figure 37: Any two-qubit unitary, U, can be written as two single-qubit unitaries, V and W, and a CNOT gate.

13.1 Transversal gates

We have been able to decompose this computation as a product of CNOTs and single-qubit gates. We do this in such a way that the answer that we get out is always encoded in the last qubit. Then, to get the answer we only need to measure the last qubit. Now, we wonder what happens if there are errors in the quantum circuit. We can easily convince ourselves that an odd number of errors, and thus, in particular, a single error, would give a wrong answer. Stated differently, any computation is going to fail in a constant amount of time[34] if we do not perform error correction. Fault tolerance (FT)

[33]Precision ϵ means that

$$||U - \tilde{U}|| \leq \epsilon,$$

where U is a two-qubit unitary and $\tilde{U}$ is its approximation consisting on only unitaries in $\{X, Z, H, T\}$.

[34]A failure in a constant amount of time cannot be avoided adding more qubits to the system.

is a method of performing gates in the encoded space without decoding at every step of the quantum circuit. In other words, FT performs gates on the encoded qubits instead of the physical qubits (see Fig. 38). This is not a trivial change because of the following idea. Consider an input qubit on which we perform some gate. If a physical qubits of this implementation have an error, the act of performing gates will create more errors. Therefore, an implementation of a gate can take the system from a state with a tolerable number of errors to a state with too many errors for a quantum error correcting code. Otherwise speaking, since errors get multiplied by gates, logical errors can be created. The goal of fault tolerance is to design gates that do not multiply errors.

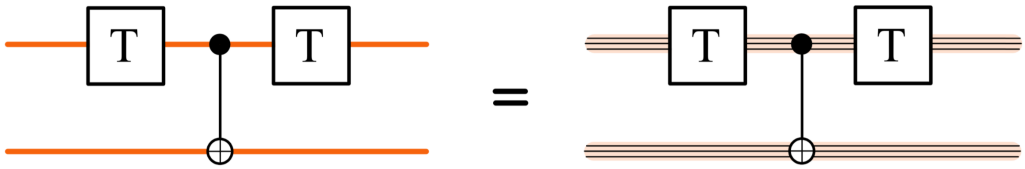

Figure 38: Fault tolerance (FT) is a method of performing gates on encoded qubits (orange lines) instead of the physical qubits (black lines).

The theorem below establishes a threshold for a computation to be performed in terms of the probability of creating a logical error.

Theorem 13.2 (Threshold theorem)**.** *[1] Consider q the probability of creating one or more logical errors with any fault tolerant gate in $\{X, Z, H, T\}$. There exists p_{th} such that $p \leq p_{th}$ arbitrary long computation can be performed.*

Let us remark that the nature of theorem 13.2 is atypical for threshold theorems on quantum error correcting codes. Usually thresholds for QEC codes are derived by adding more physical qubits to make the computation error better. In the case of theorem 13.2, however, adding more gates does not work.

Transversal gates are a way of performing FT gates for CSS codes. In order to explain them, let us state the proper definition of transversal gates and CSS codes.

Definition 13.1. *A transversal gate, U, is a gate such that*

$$U_T = \bigotimes_{\alpha} V_\alpha,$$

where all V_α act on individual physical qubits.

Definition 13.2. *A CSS code is a stabilizer code that satisfies two conditions:*

1. *X stabilizers and Z stabilizers consists only on X operators and Z operators, respectively.*
2. *There exists a X logical operator and a Z logical operator that are product of only X operators and Z operators, respectively.*

The logical Hadamard gate, the logical X gate, the logical Z gate and the logical CNOT consist on a product of physical Hadamard gates, X operations, Z operations and CNOT gates, respectively (see Fig. 39). Mathematically, we write

$$H_L = \bigotimes_\alpha H_\alpha, \qquad X_L = \bigotimes_\alpha X_\alpha, \qquad Z_L = \bigotimes_\alpha Z_\alpha,$$

$$CNOT_L(\alpha, \beta) = \bigotimes_{\alpha_i} CNOT(\alpha_i, \beta_i).$$

This is not true for the logical T gate.

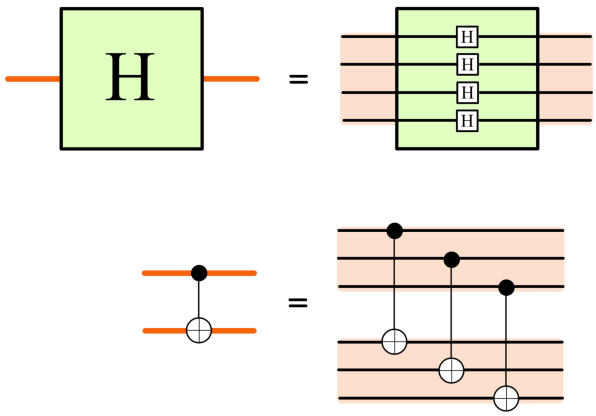

Figure 39: The logical Hadamard gate (above) and the logical CNOT (below) are transversal gates, i.e., they consist on a product of physical Hadamard gates and CNOT gates, respectively. The logical X gate and the logical Z gate are also transversal and their scheme is analogous to the scheme of the logical Hadamard gate.

Transversal operators are convenient for two reasons. First, transversal operations are naturally fault tolerant because they act only on single qubits,

and thus they do not propagate errors to other physical qubits. Otherwise speaking, if there are errors on the system, transversal operations keep them localised. The second reason is that transversal operations can be performed in parallel.
The Hadamard gate maps the X to a Z, and viceversa, i.e.,

$$HZH = X \qquad \text{and} \qquad HXH = Z \tag{34}$$

Therefore, the Hadamard gate on the toric codes maps a Z stabilizer to an X stabilizer and a Z logical operator to an X logical operator, and viceversa. The toric code is then defined on the dual lattice since X stabilizers act on crosses and Z stabilizers on squares (see Fig. 40). This map between the primal and the dual lattice due to the Hadamard gate only makes sense for a selfdual codes. A selfdual code is a code that can be defined identically in the primal and the dual lattice[35].

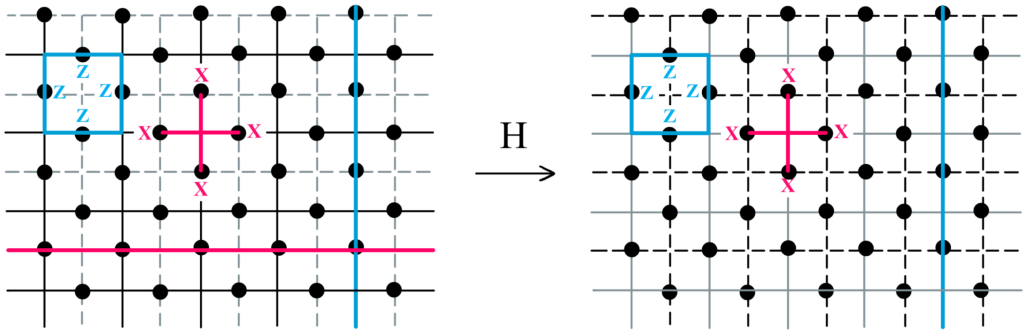

Figure 40: The Hadamard gate on a toric code maps a Z stabilizer to an X stabilizer and a Z logical operator to an X logical operator, and viceversa. The toric code is then defined on the dual lattice.

We have seen that the set of gates $G = \{X, Z, H, T, \text{CNOT}\}$ is a universal gate set. For selfdual codes, the Hadamard gate is a transversal gate and, as we have mentioned before, for CSS codes, CNOT is always a transversal because it never maps between two different Pauli operators. The X and Z gates are transversal by construction. Therfore, all gates in G except for the T gate are transversal. However, the square of the T gate, which is known as the phase gate, is transversal. The phase gate is denoted by S and written

[35]Obviously, a quantum error correcting codes is not selfdual in general. For example, the five-qubit code, which we have seen in Section 7 and its stabilizers and logical operators are in Eq. (14)), is not selfdual.

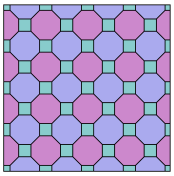

Figure 41: Truncated square lattice. Image taken from https://zh.wikipedia.org.

in matrix notation as

$$S = T^2 = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}.$$

The effect of the phase gate on single Pauli matrices is

$$SZS^\dagger = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & -i \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = Z, \tag{35}$$

$$SXS^\dagger = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = Y = iXZ \tag{36}$$

Now, we can see how the phase gate transforms logical operators, Z_L and X_L, which are strings of Z and X operators. The mapping established by the phase gate is

$$\prod_j S_j Z_L \prod_j S_j^\dagger = Z_L,$$

$$\prod_j S_j X_L \prod_j S_j^\dagger = (i)^L X_L Z_L,$$

where L is the length of the logical operators. We clearly see that in order to get the right action on the logical operators with a transversal phase gate, the length, L, of the logical operations is very important. In particular, the length must be 1 modulus 4. Toric code logical operators have even length, and thus a transversal S gate is not possible on the toric code in general. Some colour codes[36] can implement the phase gate, such as the colour code defined on a truncated square lattice (see Fig. 41) because the number of faces are a multiple of four.

In the next subsection we study a strategy to perform $CNOT$ without using transversal gates. Later on, we also see methods to implement T gates and S.

[36] not the colour code on a honeycomb lattice

13.2 Braiding

For topological codes, operations on single qubits can be performed transversally in an easy way, but a two-qubit transversal gate is not useful. To implement more than one-qubit transversal gates on topological codes, for example to implement a CNOT, we need to connect two lattices, which is not very practical. Fortunately, there exist other ways to implement a CNOT. In the following, we want to explain how to perform a CNOT gate using a strategy called braiding. For that, we first need to introduce holes in the lattice.

Consider a large toric code lattice[37]. On the lattice in Figure 42, we represent the Z and X stabilizers by light blue and light pink diamonds, respectively, and the Z and X logical operators by blue and pink strings, respectively. Now, we remove a Z stabilizer, and thus the lattice has a hole (green in Fig. 42). Recall that logical operators can freely propagate on the lattice by multiplying them by stabilizers. The removal of a stabilizer, however, implies that the logical operators cannot propagate across the hole any more. In addition, the existence of a hole creates a new logical element, i.e., a non-contractable loop that commutes with all elements in the stabilizer group. Consider a product of Z operators around the hole. Usually this element can be contracted, but the removal of the stabilizer makes it not possible any more. In other words, the hole creates a new logical element. As the number of encoded qubits is the number of logical operators, the creation of a new logical element implies that we have a new encoded qubit (see below). Analogously, we can create X holes.

Encoded qubits created by holes are referred as X-cuts and Z-cuts and they are defined as follows.

Definition 13.3. *We define the Z-cut qubit as the encoded qubit formed by removing two Z stabilizers. Its logical operators are X_L, which is a product of X operators from a boundary of one Z-hole to the other Z-hole, and Z_L, which is product of Z operators surrounding the Z-hole (see Fig. 43)*[38].

Definition 13.4. *We define the X-cut qubit as the encoded qubit formed by removing two X stabilizers. Its logical operators are Z_L, which is a product of Z operators from a boundary of one X-hole to the other X-hole, and X_L, which is product of X operators surrounding the X-hole (see Fig. 43).*

[37] As we have discussed in the previous section, the code considered in labs is the planar code because of the open boundary conditions. For the academic purpose of these notes, considering the toric code is enough.

[38] Note that each hole creates a new encoded qubit. For simplicity, here we only consider one of them.

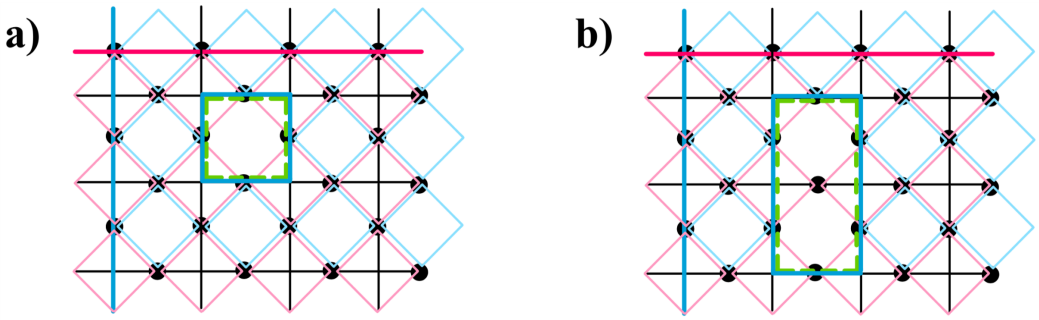

Figure 42: a) The removal of a Z stabilizer creates a Z-hole on the code. It blocks string logical operators to move across the hole, but it creates a new encoded qubit. b) Holes can be extended multiplying the logical operator consisting of a loop around the hole by a neighbouring Z stabilizer and then removing this stabilizer.

The definition of X- and Z-cut qubits is a way of initialising states. Now, given a combination of $|0\rangle$ and $|1\rangle$, we only need to perform X on the different lines connecting holes.

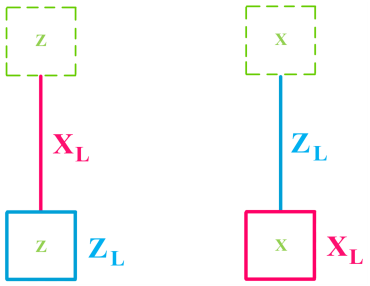

Figure 43: Z-cut (right) and X-cut (left) are encoded qubits created by the existence of holes in the lattice.

In the section about stabilizer formalism (Section 7), we have seen that distance of a code is the length of the shorter logical operator[39]. Note that holes can be easily extended. If we multiply the logical operator consisting of a loop around the hole by a neighbouring Z stabilizer and then remove this stabilizer, we get an larger hole with a larger logical operator (see Fig. 42b). Since the quality of the code is determined by the distance, in general we would prefer holes to be large and far away from each other.

[39]Recall that, in the stabilizer formalism, the length of a logical operator, which is a product of X or Z operators, is measured by the number of operators that it consists of.

An important operation involving cut qubits is called braiding. A topological braid transformation consists on moving one hole of a cut qubit between the two holes of a second cut qubit (see Fig. 44). Note that braiding does not decrease the distance of the code. We want to show that braiding an X-cut qubit with a Z-cut qubit performs a CNOT gate. For that, we first need to explain how braiding is implemented, i.e., how we move holes.

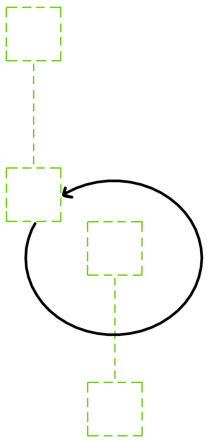

Figure 44: A topological braid transformation consists on moving one hole of a cut qubit between the two holes of a second cut qubit.

A hole can be moved around the lattice by multiplying logical operators with stabilizers and removing and adding stabilizers. In particular, consider a Z-cut qubit with logical operators $X_L^{(a)} = X_1X_2X_3$ and $Z_L^{(a)} = Z_3Z_4Z_5Z_6$ (see Fig. 45). We want to vertically move the bottom hole one cell down. For that, we multiply $Z_L^{(a)}$ by the Z stabilizer below, $Z_6Z_7Z_8Z_9$, and get a new logical operator $Z_L^{(b)} = Z_3Z_4Z_5Z_7Z_8Z_9$. We remove the stabilizer $Z_6Z_7Z_8Z_9$ and apply an X operator on qubit 6, which resets the Pauli basis. Then, we extend $X_L^{(a)}$ by multiplying it by X_6 and get $X_L^{(c)} = X_1X_2X_3X_6$. We turn on the stabilizer $Z_3Z_4Z_5Z_6$ and define a new logical operator as $Z_L^{(c)} = Z_3Z_4Z_5Z_6Z_L^{(b)} = Z_6Z_7Z_8Z_9$.

In order to see that braiding performs the CNOT gate, we will look at the effect of the CNOT on the following operators:

a) $\text{CNOT}(\mathbb{I} \otimes X_L)\text{CNOT} = \mathbb{I} \otimes X_L$

b) $\text{CNOT}(X_L \otimes \mathbb{I})\text{CNOT} = X_L \otimes X_L$

c) $\text{CNOT}(\mathbb{I} \otimes Z_L)\text{CNOT} = Z_L \otimes Z_L$

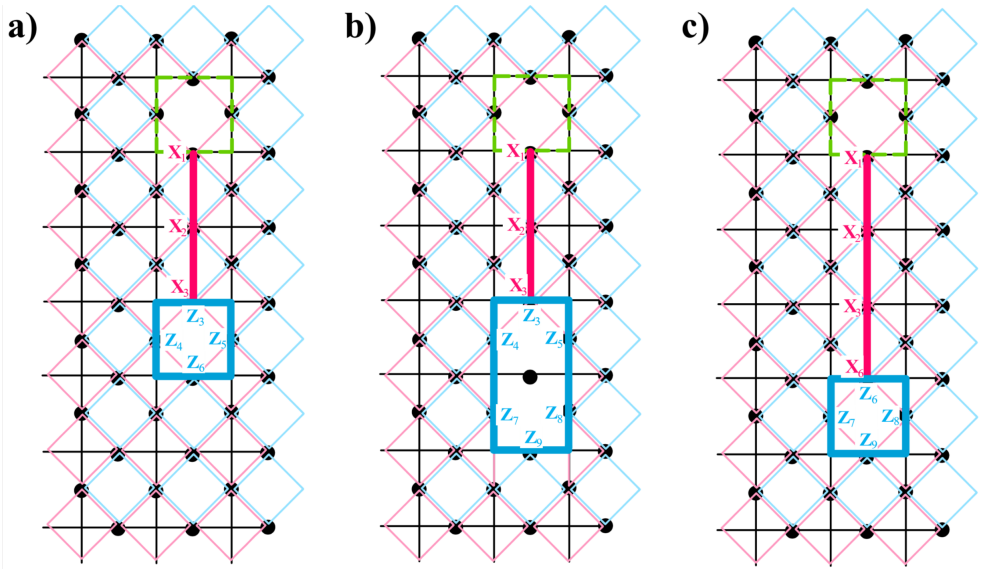

Figure 45: a) We want to move one cell down the bottom hole of the Z-cut qubit with logical operators $X_L^{(a)} = X_1X_2X_3$ and $Z_L^{(a)} = Z_3Z_4Z_5Z_6$. b) First, we multiply $Z_L^{(a)}$ by the Z stabilizer below, $Z_6Z_7Z_8Z_9$, and get a new logical operator $Z_L^{(b)} = Z_3Z_4Z_5Z_7Z_8Z_9$. We remove the stabilizer $Z_6Z_7Z_8Z_9$. c) Then, we apply an X operator on qubit 6, which resets the Pauli basis. We extend $X_L^{(a)}$ by multiplying it by X_6 and get $X_L^{(c)} = X_1X_2X_3X_6$. We finally turn on the stabilizer $Z_3Z_4Z_5Z_6$ and define a new logical operator as $Z_L^{(c)} = Z_3Z_4Z_5Z_6Z_L^{(b)} = Z_6Z_7Z_8Z_9$.

d) $\mathrm{CNOT}(Z_L \otimes \mathbb{I})\mathrm{CNOT} = Z_L \otimes \mathbb{I}$

Taking linear combinations of these four elements, we can create any two-qubit logical operations involving Pauli matrices. We want to check these four equalities directly with the braiding of an X-cut with a Z-cut. If the equalities are fulfilled, then it guaranties that the braiding performs a CNOT gate. Figure 46 shows pictorically that braiding an X-hole with a Z-hole encodes the CNOT gate on the logical information. This method of performing a CNOT is fault tolerant because it does not propagate errors. Let us remark that neither with braiding operations nor with transversal gate we can perform a T gate on a toric code. As we will see later on, we need measurement Z-feedbacks.

13.3 Clifford operations

In this section, we want to define study Clifford operations, which are operations map Pauli operations to Pauli operations. One important property is that they can be simulated efficiently, and thus, we need to go beyond them in order to make quantum computation powerful.

The definition of the Clifford group is the following.

Definition 13.5. *The Clifford group, $\mathcal{W}_1$, consists of all operations that map Pauli matrices to Pauli matrices via conjugation without considering possible added phases. Mathematically, we write*

$$\mathcal{W} \equiv \{W \text{ such that } WPW \in \mathcal{P}^* \,\forall\, P \in \mathcal{P}^*\},$$

where $\mathcal{P}^* \equiv \mathcal{P}\backslash\{\pm\mathbb{I}\} \equiv \{\pm\mathbb{I}, \pm X, \pm Y, \pm Z\}$.

Analogously to the Clifford group, we can define the n-qubit Clifford group, $\mathcal{W}_n$, based on $\mathcal{P}_n^* = \mathcal{P}_n\backslash\mathcal{P}_n$, where $\mathcal{P}_n$ is the n-qubit Pauli group defined in Eq. (10). The Hadamard gate, the phase gate, the CNOT gate are generators of the n-qubit Clifford group, i.e., $\mathcal{W}_n = \langle\{H, S, \mathrm{CNOT}\}\rangle$. Note that the generators of the Clifford group can be performed transversally.

Consider single-qubit case, where the Clifford group is generated by the Hadamard and the phase gate, $\mathcal{W}_1 = \langle\{H, S\}\rangle$. Recalling Eq. (34), (35) and (36), single-qubit Clifford gates can be thought as rotations on the Bloch sphere around the different axes. In particular, when we apply H or S to a qubit represented on one concrete point of the Bloch sphere, we rotate the qubit to the mirror point on the Bloch sphere with respect to the corresponding axis. Since we can only rotate to the mirror points, all possible

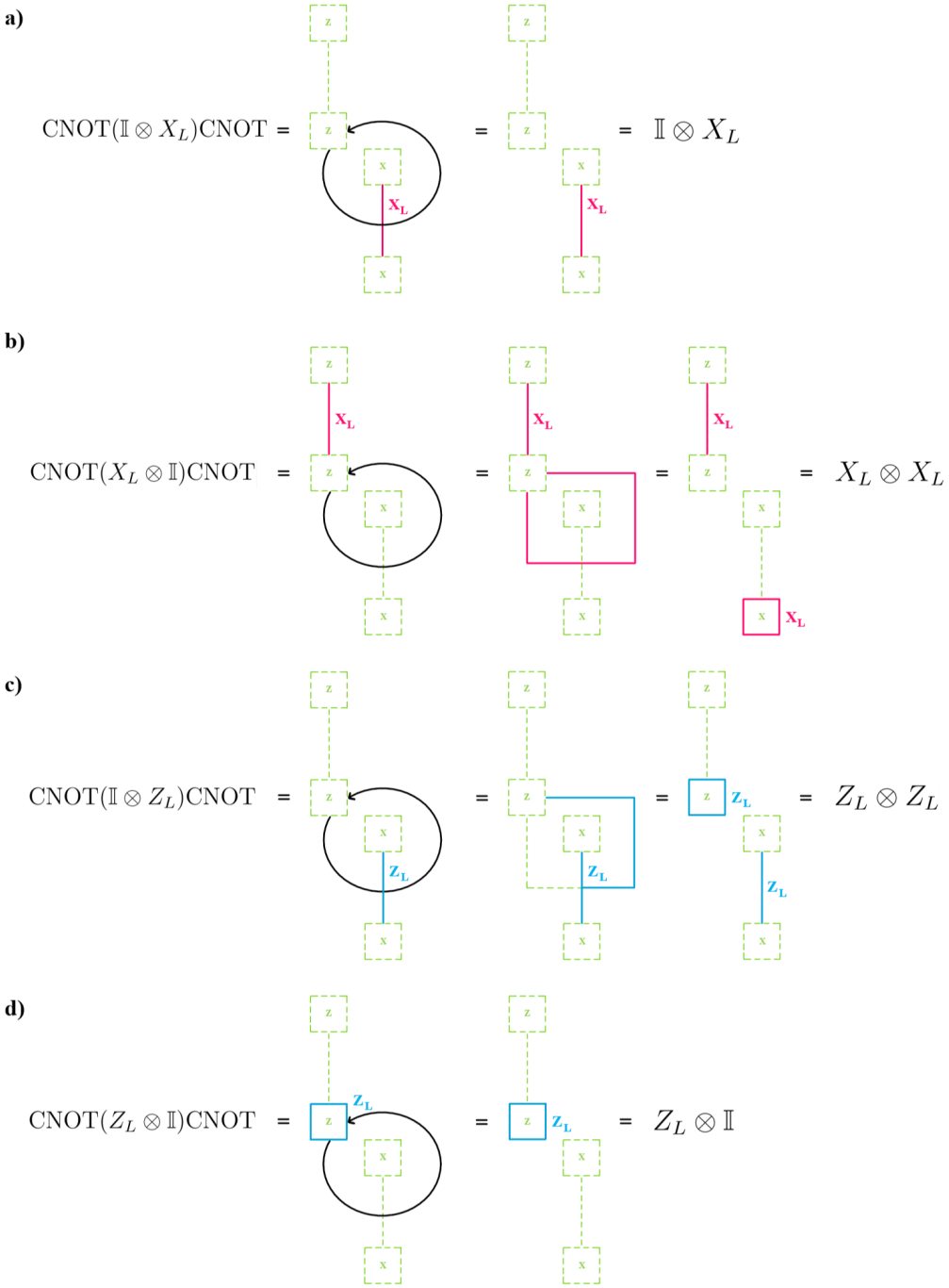

Figure 46: Braiding an X-cut qubit with a Z-cut qubit performs a CNOT gate. In a), note that a Z-hole always meet an X line at two points, and thus they commute. The same occurs for the case of X-holes and Z-lines. In b), the cycle would be trivial if there was no hole. Since there is an X-hole, the cycle cannot be completely contracted, but only shrunk on to the border of the X-hole.

combinations of the Hadamard and the phase gates give access to a finite number of rotations. At the beginning of the section, however, we have argued that with a finite number of gates we are able to perform any unitary operation with a certain precision. Thus, we need another element tp achieve this; the T gate.

In the exercise class we have seen that the T gate and the product HTH are rotations of $\frac{\pi}{8}$ around the $\hat{z}$ axis and the $\hat{x}$ axis, respectively (see Fig. 47). Mathematically, we write

$$T \propto e^{i\frac{\pi}{8}Z} \qquad \text{and} \qquad HTH \propto e^{i\frac{\pi}{8}X}. \tag{37}$$

One can also show that $THTH$ is a rotation by an angle $\theta = 2\arccos\left(\cos^2\frac{\pi}{8}\right)$ around the axis

$$\hat{n} = \frac{2}{\sqrt{6+\sqrt{2}}}(\cos\tfrac{\pi}{8}, \sin\tfrac{\pi}{8}, \cos\tfrac{\pi}{8}).$$

The crucial property of this rotation is that the angle θ is an irrational number multiple of π. To see this, consider a rational rotation multiple of π, i.e., a rotation of an angle

$$\theta = \frac{p}{q}2\pi,$$

where p and q are integer numbers. If we perform q times this rational rotation, we are back to the starting point. Therefore, whenever the rotation that we are able to perform is rational, we have access only to a discrete number of points. Conversely, there exists a theorem that states that, given a starting point, the iteration of an irrational rotation covers the unit circle. This process is efficient, i.e., given a point on the Bloch sphere, we can always reach it with a finite number of rotations which is proportional to the precision by which we want to reach this point. In particular, if we iterate the operator $THTH$ a sufficient number of times on the Bloch sphere, we can reach any point on the circle that is spanned by the rotations in Eq. (37) (see Fig. 47). The operator $HTHT$ is also an irrational rotation, but around an axis orthogonal to $\hat{n}$. Using $THTH$ and $HTHT$ we can apply rotation as the following theorem states.

Theorem 13.3. *Given two orthogonal unit vectors, $\hat{n}$ and $\hat{m}$, any rotation, R, can be written as*

$$R = R_{\hat{n}}(\alpha)R_{\hat{m}}(\beta)R_{\hat{n}}(\gamma),$$

where $\alpha, \beta, \gamma \in [0, 2\pi)$.

In summary, as long as we have access to Hadamard gate and T gate, we are able to perform any rotation on the Bloch sphere. A direct consequence of

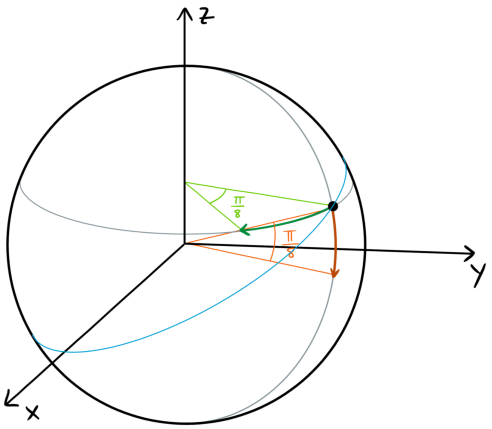

Figure 47: The T gate and the product HTH are rotations of $\frac{\pi}{8}$ around the $\hat{z}$ axis (green rotationi) and the $\hat{x}$ axis (orange rotation), respectively. If we iterate the operator $THTH$ a sufficient number of times, we can reach any point on the blue circle, which is the unit circle spanned by the rotations made by T and HTH.

this theorem is that any rotation can be decomposed as a product of H and T gates. Recall, however, that the T gate cannot be perform transversally, and thus we still need a strategy to perform it. Note that irrational rotation have been the key element to to go from a continuous set of single-qubit gates to a finite set.

Whether quantum computers will be able to overcome classical computers is not yet clear. Even more, the origin of the possible improvements that quantum computations can achieve it is unknown. In 1998, Gottesmann published the theorem below, which clearly states that this supposed power does not come from Clifford gates.

Theorem 13.4 (Gottesmann-Knill)**.** *Any computation based on Clifford operations (i.e., combinations of* $\{H, S, X, Z, CNOT\}$*), state preparation in any Pauli basis and measurements in any Pauli basis can be efficiently simulated on a classical computer.*

This theorem means that, if we have a long unitary circuit that only involves Clifford gates, U_{Clifford}, there exists a classical program which is able to perform the same task as U_{Clifford}. Moreover, the number of classical gates that the classical program need is only a polynomial times the number of quantum gates that form U_{Clifford}. An immediate consequence of the Gottesmann-Knill theorem is that we need to find a way to perform the T gate, which is a non-

Clifford gate, in order quantum computation to be useful.

If Gottesmann and Knill showed that Clifford gates are not enough to overcome classical computation, Eastin and Knill proved that we also need to go beyond transversal operations. Their theorem states the following.

Theorem 13.5 (Eastin-Knill)**.** *For any local stabilizer code, where local means that the stabilizer operators have a finite number of Paulis matrices, the set of transversal operations is not a universal set.*

As we have commented previously in these notes, in the lab we can only deal with local quantum error correcting codes. Nevertheless, the Eastin-Knill theorem say, if the code is local, we need to be able to perform non-transversal gates in order to have a universal set of gates. In the following subsection, we see a method to perform the T gate, which is a non-transversal gate.

13.4 Magic state distillation

Magic state distillation is a technique that uses measurements to go beyond Clifford and transversal gates, i.e., to get around the Gottesmann-Knill theorem and the Eastin-Knill theorem.

Suppose we can prepare the state $|A\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\frac{\pi}{4}}|1\rangle)$, which is known as magic state, then we can perform the T gate on an arbitrary qubit, $|\psi\rangle$, by implementing the circuit in Figure 48. Mathematically, we write

$$T|\psi\rangle|A\rangle = C(S,Z)\text{CNOT}|\psi\rangle|A\rangle, \tag{38}$$

where $C(S,Z)$ is an S gate controlled by a Z measurement, i.e., the S gate is applied only if the measurement result is 1; otherwise, nothing happens. We can easily prove Eq. (38) by straight computation. Given an arbitrary

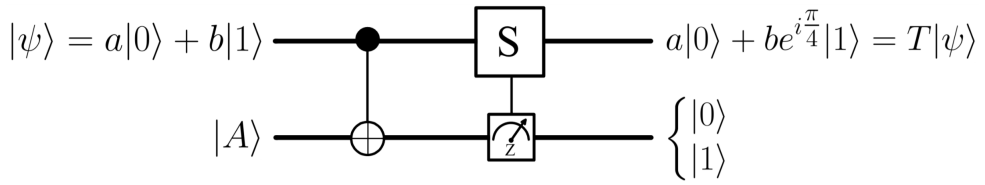

Figure 48: Circuit that implements the T gate on an arbitrary qubit $|\psi\rangle = a|0\rangle + b|1\rangle$ with a magic state $|A\rangle = \frac{1}{2}(|0\rangle + e^{i\frac{\pi}{4}}|1\rangle)$.

qubit, $|\psi\rangle = a|0\rangle + b|1\rangle$, we have

$$
\begin{aligned}
C(S,Z)\mathrm{CNOT}|\psi\rangle|A\rangle &= \frac{1}{\sqrt{2}}C(S,Z)\mathrm{CNOT}(a|0\rangle + b|1\rangle)(|0\rangle + e^{i\frac{\pi}{4}}|1\rangle), \\
&= \frac{1}{\sqrt{2}}C(S,Z)\mathrm{CNOT}(a|00\rangle + ae^{i\frac{\pi}{4}}|01\rangle + b|10\rangle + be^{i\frac{\pi}{4}}|11\rangle), \\
&= \tfrac{1}{\sqrt{2}}C(S,Z)(a|00\rangle + ae^{i\frac{\pi}{4}}|01\rangle + b|11\rangle + be^{i\frac{\pi}{4}}|10\rangle), \\
&= \begin{cases} \mathbb{I}\otimes|0\rangle\langle 0|(a|00\rangle + ae^{i\frac{\pi}{4}}|01\rangle + b|11\rangle + be^{i\frac{\pi}{4}}|10\rangle) & \text{if result is 0,} \\ S\otimes|1\rangle\langle 1|(a|00\rangle + ae^{i\frac{\pi}{4}}|01\rangle + b|11\rangle + be^{i\frac{\pi}{4}}|10\rangle) & \text{if result is 1,} \end{cases} \\
&= \begin{cases} (a|00\rangle + be^{i\frac{\pi}{4}}|1\rangle)|0\rangle & \text{if result is 0,} \\ (ae^{i\frac{\pi}{4}}|0\rangle + be^{i\frac{\pi}{2}}|1\rangle)|1\rangle & \text{if result is 1,} \end{cases} \\
&= \begin{cases} T|\psi\rangle|0\rangle & \text{if result is 0,} \\ T|\psi\rangle|1\rangle & \text{if result is 1,} \end{cases}
\end{aligned}
$$

where we have omitted the global phase because it is not physically relevant. Note that the CNOT gate as well as the S gate controlled by a Z measurement are Clifford operations. Therefore, the key point that allow us to perform the T gate is the ability of preparing the magic state, $|A\rangle$. Actually, there exist more states that are as powerful as state $|A\rangle$. All these states are known as magic states and they can be obtained from $|A\rangle$ via Clifford operations. In the following, we want to study a strategy for magic state preparation.

There exists a simple circuit that allow us to encode a state in the code subspace. In particular, two subsequent CNOT gates (see Fig. 49) transform an arbitrary state $|\psi\rangle = a|0\rangle + b|1\rangle$ to the encoded state $|\psi_L\rangle = a|000\rangle + b|111\rangle$ such that

$$|\psi_L\rangle_{123} = \mathrm{CNOT}(1,2)\mathrm{CNOT}(1,3)|\psi\rangle_1|0\rangle_2|0\rangle_3.$$

This circuit corresponds to the encoding of the repetition code. More generally, there is a way of going from stabilizer operators to general construction

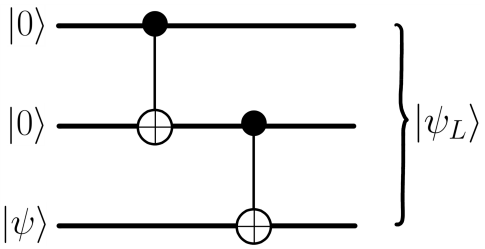

Figure 49: Encoding circuit of the repetition code.

of the unitary operation corresponding to the quantum code, $U_{\mathcal{C}}$ (see Fig. 50). The quantum circuit $U_{\mathcal{C}}$ takes a state ρ that we want to encode to an encoded subspace. Then, one might expect that, if we perform the inverse operation, $U_{\mathcal{C}}^{\dagger}$, the map identifies the elements in the code space and it projects them into a single state, ρ' (see Fig. 51). Indeed, if we can prepare ρ such that $\langle T_0|\rho|T_0\rangle \leq 1-\epsilon_c$, where $|T_0\rangle$ is a magic state and $\epsilon_c \approx 0.2$, then applying $U_{\mathcal{C}}^{\dagger}$ on N copies of ρ we get $\langle T_0|\rho'|T_0\rangle \leq 1-\epsilon'$, where $\epsilon' << \epsilon$ with $\epsilon' \approx \epsilon^N$. This process is known as a (magic state) distillation. In words, if we are able to prepare states close to a magic state, then we can flow them into an inverse quantum error correcting code and distil them to obtain a single-copy, but with higher fidelity. However, there is a trade-off. In the distillation procedure, we need to construct the $|0\rangle$ states, i.e., we need to perform a measurement on each of the individual qubits and post-select on the result $000\cdots0$. If there are too many states, the probability of obtaining the result $000\cdots0$ is exponentially suppress. Therefore, we should use a code that is not too large (five- or seven-qubit code). Measurements will fail most of the times, but every once in a while, we will get the result $000\cdots0$ and obtain am almost magic state. Once we have this almost magic state, we can send it to the circuit that performs a T gate (Eq. (38)). Therefore, quantum computations need "magic state factories" that are constantly measuring and, when they obtain an almost magic state, they send it to the main circuit to perform a T gate. This procedure implies a high cost on time and on the number of qubits. For toric code, quantum computation is even more expensive because we need another distillation procedure to perform S gates, which are required in the T-gate circuit (Eq. (38)). Therefore, to quantum compute on the toric code we need two levels of distillation: one to perform the T gate and another to implement the S gate.

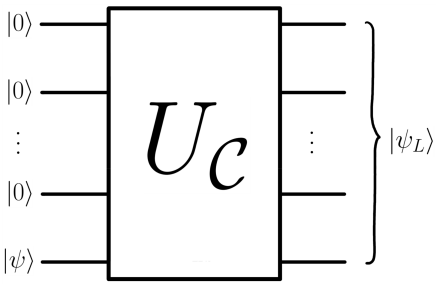

Figure 50: General encoding circuit.

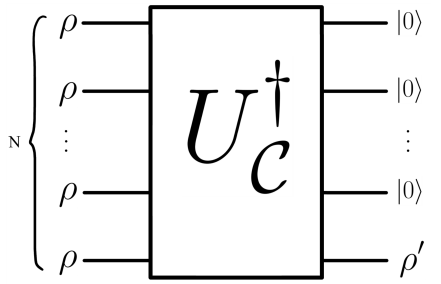

Figure 51: Distillation process.

13.5 Coupling of two surface codes without transversal operations

As we have mentioned earlier in this section, performing many-qubit gates on surface codes using transversal gates is not efficient. It requires a direct interaction between two surfaces in order to couple the qubits and this produces a lot of loses. We have studied braiding as a possible alternative, but, indeed, braiding can also be understood as a transversal gate. Other alternatives are known and here we want to see some.

The first method to perform logical gates without transversal gates consists in introducing a kink in the lattice. A kink deforms the lattice replacing one four-qubit stabilizer to a five-qubit stabilizer and another four-qubit stabilizer to a three-qubit stabilizer as Figure 52 shows. The effect of a kink is that, when a Z logical operator moves across the kink, it becomes a X logical operator, and viceversa. It turns out that combining holes and kinks, S gates are possible.

Another alternative that avoids transversal gates is called lattice surgery. Lattice surgery combines small chunks of a planar code and treats each chunk

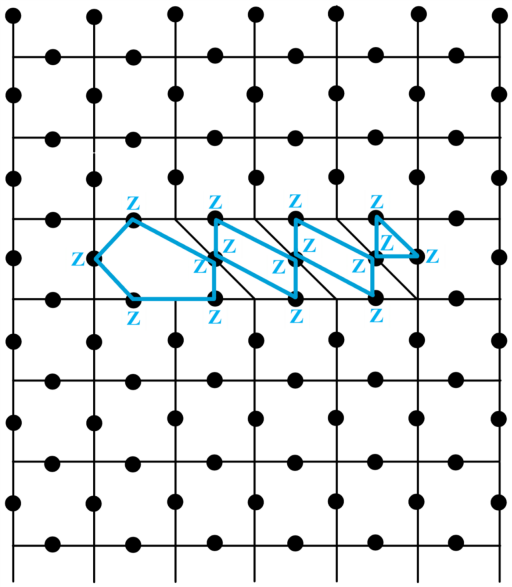

Figure 52: A kink deforms the lattice replacing one four-qubit stabilizer to a five-qubit stabilizer and another four-qubit stabilizer to a three-qubit stabilizer.

as a qubit (see Fig. 53). If chunks are only one line away, it turns out that there exists an efficient way of measuring the chunks that performs gates. This might be the optimal strategy of performing gates nowadays.

In the case of colour codes, there exists a third alternative. If you put a colour code in a three-dimensional lattice that has a concrete structure and boundaries, then it is possible to construct the code such that we can perform T gates, but not Hadamard gates. The trick is then to use a two-dimensional colour code to perform Hadamard gates and a three-dimensional colour code to implement T gates. The quantum computation switches between both

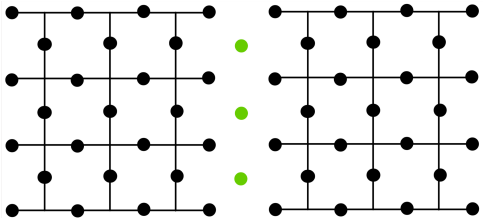

Figure 53: Lattice surgery combines small chunks of a planar code and treats each chunk as a qubit.

colour codes.

For a quantum computation on a toric code using lattice surgery, kinks and magic state distillation, estimates say that we will need $10^9 - 10^{10}$ physical qubits and 10^{12} operations for factoring a hundred-digit co-prime.

14 Subsystem codes

Almost all quantum error correcting codes that we have seeen in these notes are stabilizer codes. This section wants to go beyond stabilizer formalism by considering the so-called subsystem codes. Subsystem codes are a set of codes combine decoherence free subspaces, noiseless subsystems, and quantum error correcting codes. This combination provides subsystem codes interesting characteristics, such as simplified syndrome calculation and a easily implementable fault-tolerant operations.

When we defined the stabilizer codes (see Sec. 7), we divided the Hilbert space, $\mathcal{H}$, into the code space, $\mathcal{C}$, and its orthogonal space, i.e., $\mathcal{H} = \mathcal{C} \oplus \mathcal{C}^\perp$. This decomposition is not the most general decomposition of a Hilbert space. A Hilbert space, $\mathcal{H}$, can be decocomposed as

$$\mathcal{H} = \bigoplus_j \mathcal{A}_j \otimes \mathcal{B}_j.$$

The stabilizer codes are the particular case where $\mathcal{B}_1$ is trivial, and thus we have $\mathcal{C} = \mathcal{A}_1$ and $\mathcal{C}^\perp = \bigoplus_{j>1} A_j$, i.e.,

$$\mathcal{H}_{\text{Stabilizer codes}} = \mathcal{A}_1 \bigoplus_{j>1} \mathcal{A}_j.$$

For subsystems codes, $\mathcal{B}$ is not trivial any more, which implies that the Hilbert space decomposes as

$$\mathcal{H}_{\text{Subsytem codes}} = (\mathcal{A} \otimes \mathcal{B}) \oplus \mathcal{C}^\perp,$$

where the code subspace is $\mathcal{C} = \mathcal{A} \otimes \mathcal{B}$ with $\mathcal{B}$ non trivial. Here, the subspace $\mathcal{A}$ encodes the logical information, and thus $\mathcal{A}$ is called logical subsystem; and $\mathcal{B}$ is known as gauge subsystem and it contains the gauge information, i.e., the degrees of freedom that are not logical information.

Subsystem codes have a code space defined analogously as the code space of stabilizer codes, i.e.,

$$\mathcal{C} = \{|\psi\rangle \text{ such that } S_j|\psi\rangle = |\psi\rangle \ \ \forall S_j \in \mathcal{S}\},$$

where $\mathcal{S}$ is the stabilizer group. In addition to the stabilizer group, subsystem codes have a logical group, $\mathcal{L}$, and a gauge group, $\mathcal{G}$, which are subspaces of the Pauli group, i.e., $\mathcal{L}$, $\mathcal{G}$, $\mathcal{S} \leq \mathcal{P}_n$, with $\mathcal{P}_n$ the n-qubit Pauli group. Consequently, all elements of $\mathcal{L}$, $\mathcal{G}$, $\mathcal{S}$ are products of Pauli operators. These groups have the following interesting properties:

- Elements in $\mathcal{G}$ are referred to as gauge operators.
- As long as $\mathcal{L}$ and $\mathcal{G}$ are non-trivial, only $\mathcal{S}$ is abelian.
- Elements of $\mathcal{L}$, $\mathcal{G}$ and $\mathcal{S}$ commute among each other, i.e.,

$$\forall\, G \in \mathcal{G}, \forall\, L \in \mathcal{L} \text{ and } \forall\, S \in \mathcal{S}, \ \ [G, L] = 0 = [S, L] = [G, S]. \qquad (39)$$

- The stabilizer group is a subgroup of the gauge group, i.e., $\mathcal{S} \leq \mathcal{G}$.

In subsystem codes, we have the extra degrees of freedom that form the gauge group, but we do not care about the information they encode. Thus, we do not need to protect this information neither care if measurements effect it.

Logical information which is contained in $\mathcal{A}$ have different representations. Consider a state $|\psi\rangle \in \mathcal{C}$ and interpret that $G|\psi\rangle$ for any $G \in \mathcal{G}$ as representing the same information. This implies that logical operation have also many representations.

Definition 14.1. *A logical operator, $L \in \mathcal{C}$, is called a bare logical operator if it can be written as*

$$L = L_A \otimes \mathbb{I}_B,$$

where $L_A \in \mathcal{L}$.

Definition 14.2. *A logical operator, $L \in \mathcal{C}$, is called a dressed logical operator if it can be written as*

$$L = L_A \otimes L_B,$$

where $L_A \in \mathcal{L}$ and $L_B \in \mathcal{G}$ such that $L_B \neq \mathbb{I}_B$.

Note that this different representation for logical operators of subsystem codes is not the same as the different logical operators we can have for stabilizer codes. In particular, all logical operators of stabilizer codes are bare logical operators since $\mathcal{G}$ is trivial.

For an n-qubit stabilizer code, we had k logical operators and we completed the rest of the code space with $n-k$ stabilizers such that

Stabilizer operators				Logical operators		
S_1^X	S_2^X	$\cdots$	S_{n-k}^X	$\bar{X}_1$	$\cdots$	$\bar{X}_k$
S_1^Z	S_2^Z	$\cdots$	S_{n-k}^Z	$\bar{Z}_1$	$\cdots$	$\bar{Z}_k$

where $\{S_i^X, S_i^Z\}$ and $\{\bar{X}_i, \bar{Z}_i\}$ are stabilizer and logical operators, respectively. For subsystem codes, we still have a certain number of stabilizer and logical operators and, moreover, we have a number of gauge qubits such that

Stabilizer operators			Gauge operators			Logical operators		
S_1^X	$\cdots$	S_{n-k-r}^X	$\tilde{X}_1$	$\cdots$	$\tilde{X}_r$	$\bar{X}_1$	$\cdots$	$\bar{X}_k$
S_1^Z	$\cdots$	S_{n-k-r}^Z	$\tilde{Z}_1$	$\cdots$	$\tilde{Z}_r$	$\bar{Z}_1$	$\cdots$	$\bar{Z}_k$

where $\{S_i^X, S_i^Z\}$, $\{\bar{X}_i, \bar{Z}_i\}$ and $\{\tilde{X}_i, \tilde{Z}_i\}$ are stabilizer, logical and gauge operators, respectively. A convenient way to understand gauge operators is as gauge qubits which we encode in the space, but we do not care about protecting them against noise. Stated differently, we engineer the system such that we can measure the stabilizers and effectively detect and correct errors on the logical qubits without paying attention on the effect on the gauge qubits.

As for stabilizer codes, logical operations of subsystem codes can be multiplied by stabilizers operators and we get new logical operators. The minimal length of the logical operators is the distance of the code.

14.1 Shor code

In previous chapters, we have studied the Shor code using the stabilizer formalism. Now, we want to see the Shor code as a subsystem code.

Recall that, in the stabilizer formalism, the stabilizer group of the Shor code on nine qubits is

$$\mathcal{S} = \{Z_1Z_2, Z_2Z_3, Z_3Z_4, Z_4Z_5, Z_5Z_6, Z_6Z_7, Z_7Z_8, Z_8Z_9, X_1\cdots X_6, X_4\cdots X_9\},$$

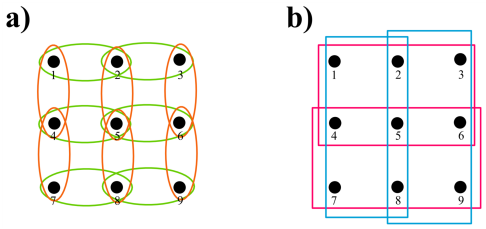

Figure 54: a) X (orange) and Z (green) gauge operators of the subsystem Shor code, b) X (pink) and Z (blue) stabilizer operators of the subsystem Shor code.

where all $S \in \mathcal{S}$ are independent. The Shor code encodes one logical qubit, which is defined by the logical operators

$$\bar{X}_1 = \prod_{i=1}^{9} X_i \qquad \text{and} \qquad \bar{Z}_1 = \prod_{i=1}^{9} Z_i. \tag{40}$$

In order to build the subsystem Shor code, we take four independent stabilizer operators that form the stabilizer group[40] and we have one logical qubit, then we still have four extra degrees of freedom, which constitute the gauge qubits. Thus, the subsystem Shor code has four guage operators (of each type). Indeed, the gauge group of the subsystem Shor code (see Fig. 54) is an overcomplete[41] group given by

$$\begin{aligned}\mathcal{G} = \{&Z_1Z_2, Z_2Z_3, Z_3Z_4, Z_4Z_5, Z_5Z_6, Z_6Z_7, Z_7Z_8, Z_8Z_9,\\ &X_1X_4, X_2X_5, X_3X_6, X_4X_7, X_5X_8, X_6X_9\}.\end{aligned}$$

Note that gauge operators do not commute in general. The stabilizer group of the subsystem Shor code is

$$\mathcal{S} = \{Z_1Z_2Z_4Z_5Z_7Z_8, Z_2Z_3Z_5Z_6Z_8Z_9, X_1X_2X_3X_4X_5X_6, X_4X_5X_6X_7X_8X_9\}.$$

The logical operators of the subsystem Shor code are the same as the logical operators of the stabilizer Shor code (Eq. (40)). Summarizing, we have

Stabilizer operators		Gauge operators			
$X_1X_2X_3X_4X_5X_6$	$X_4X_5X_6X_7X_8X_9$	X_1Z_7	X_3X_9	X_4X_7	X_6X_9
$Z_1Z_2Z_4Z_5Z_7Z_8$	$Z_2Z_3Z_5Z_6Z_8Z_9$	Z_1Z_6	Z_2Z_3	Z_5Z_6	Z_4Z_5

[40] Elements of the stabilizer group of a subsystem code are sometimes denoted as check operators.

[41] i.e., elements are not independent

Logical operators
$X_1 \cdots X_9$
$Z_1 \cdots X_9$

Note that all these operators satisfy the constraint given by Eq. (39). Multiplying a Z logical operator by a Z stabilizer we see that the distance of the subsystem Shor code is three.

An important difference between stabilizers and gauge operators is that gauge operators are local, which is an important property to implement the code in the lab. For a subsystem code implemented in the lab, we measure gauge operators instead of stabilizer operators and, from the gauge measurements, we infer the values of the stabilizers. Recall that gauge operators do not commute, and thus the result of the gauge measurements depends on the ordering of these measurements. However, the values of the stabilizers, which are products of gauge measurements outcomes, are independent of the ordering of gauge measurements as long as property in Eq. (39) is fulfilled.

The subsystem Shor code can correct against one error, which can be a flip or a phase error, as well as the stabilizer Shor code does. Recall that the stabilizer Shor code additionally protects against flip errors in blocks of three qubits. This extra protection property is not possible in the subsystem Shor code any more due to the reduction of the number of stabilizers.

14.2 Bacon-Shor code

The Bacon-Shor code is the natural extension of the Shor code to a lattice of $N \times N$ qubits. Label the qubits of the lattice with the subindices (i, j), where $i = 1, \ldots, N$ denotes the row and $j = 1, \ldots, N$ denotes the column (see Fig. 55).

The gauge group is

$$\mathcal{G} = \{\{Z_{ij}Z_{(i+1)j} \ \forall i = 1, \ldots, N-1 \text{ and } \forall j = 1, \ldots, N\}, \tag{41}$$
$$\{X_{ij}X_{i(j+1)} \ \forall i = 1, \ldots, N \text{ and } \forall j = 1, \ldots, N-1\}\} \tag{42}$$

The Z logical operators are vertical strings of Z operators and the X logical operators are horizontal strings of X operators. For a fix i and j, we write

$$\bar{X}_i = X_{i1} \cdots X_{iN} \qquad \text{and} \qquad \bar{Z}_j = Z_{1j} \cdots Z_{Nj}.$$

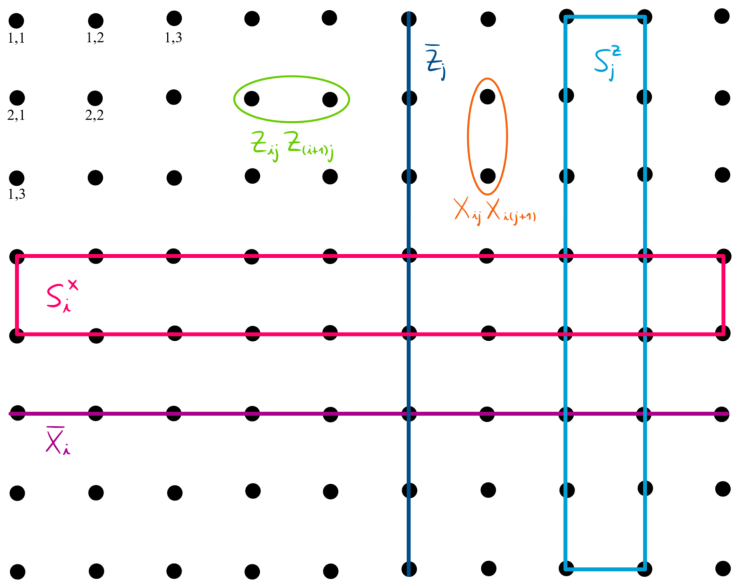

Figure 55: The Bacon-Shor code, where the X and Z gauge operators are in represented in orange and green, respectively, the X and Z logical operators are in represented in purple and dark blue, respectively, and the X and Z stabilizer operators are in represented in pink and light blue, respectively.

The Z stabilizer operators are two vertical strings of Z operators and the X stabilizer operators are two horizontal strings of X operators. Given a fix $i < N$ and $j < N$, we have

$$S_i^X = X_{i1} \cdots X_{iN} X_{(i+1)1} \cdots X_{(i+1)N} \quad \text{and} \quad Z_j^Z = Z_{1j} \cdots Z_{Nj} Z_{1(j+1)} \cdots Z_{N(j+1)}.$$

Note that, as required, the logical operators commute with all stabilizer operators and gauge operators as well as stabilizer and gauge operators also commute.

In the Bacon-Shor code, logical operators can be translated by multiplying them with stabilizer operators, but, unlike the toric code, we are not allowed to bend logical operators.

A disadvantage of the Bacon-Shor is that it is not resistant to $\{X, Z\}$ independent and identically distributed noise. Even considering perfect measurements, the Bacon-Shor code has no threshold. If measurements are not perfect, the code fails because we need to perform N-measurements, and thus the larger N gets, the bigger is the error corresponding to the measurement of a stabilizer is going to be worse and worse. If we do have perfect measurements, the Bacon-Shor code has no threshold either because, roughly

speaking, we do not have enough stabilizer measurements to distinguish errors. Stated differently, we are able to protect against strings that extend in one direction, but not in both.

References

[1] M. A. Nielsen, and I. L. Chuang, (Cambrige University Press, 2011), *Quantum computation and quantum information*, 10th edn.