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.

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*
   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*

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, $\{0, 1\}$.

[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.

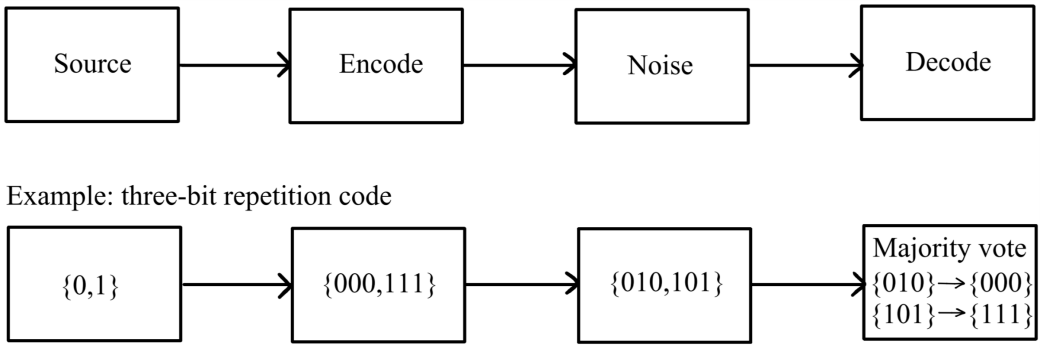

Figure 1: Every error correcting code can be broken up into four steps: source, encode, noise and decode. The three-bit repetition code is the simplest example of error correcting code.

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 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*

$$\{00\} \to \{000000\},$$

[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.

$$\{01\} \to \{000111\},$$
$$\{10\} \to \{111000\},$$
$$\{11\} \to \{111111\}.$$

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.,

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

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.

[3]This will not be true for quantum 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*

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

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.

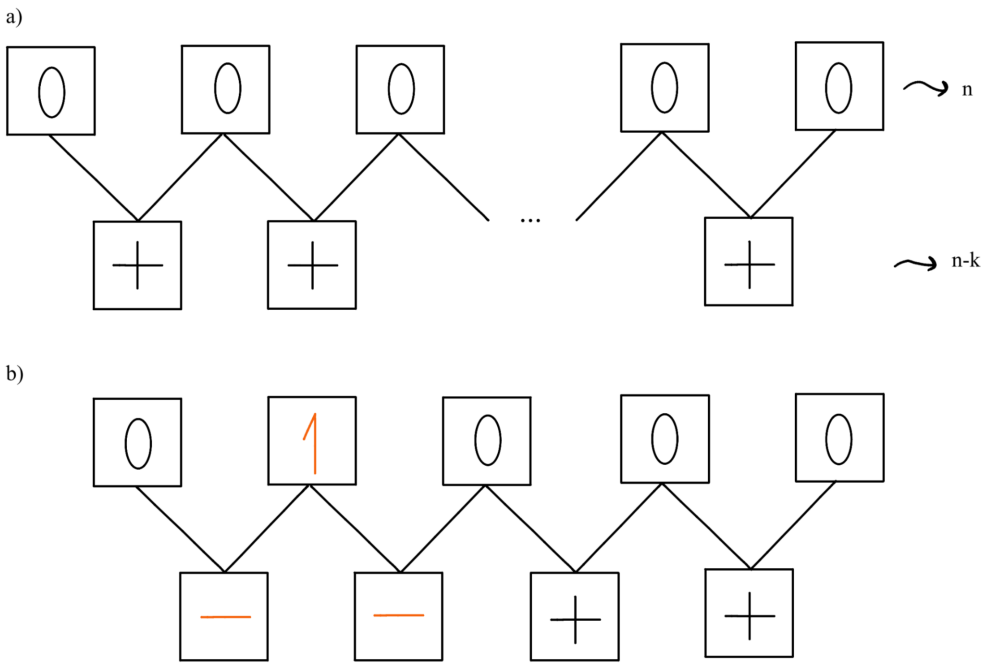

Figure 2: Tanner graph of a) the n-bit codeword $\{0, \ldots, 0\}$, and b) the five-bit codeword $\{00000\}$ with a flip error on the second bit.

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 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.

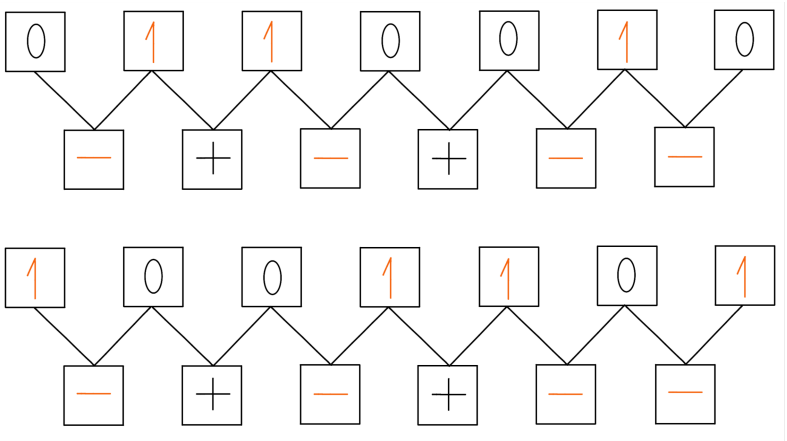

Figure 3: An error syndrome has two parents of errors. Given the initial codeword {0000000}, a flip-error on the second, third and sixth bit gives the same error syndrome than a flip-error on the first, forth, fifth and seventh bit. However, the situation with only three errors is more likely.

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*

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]

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

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.

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

[**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.

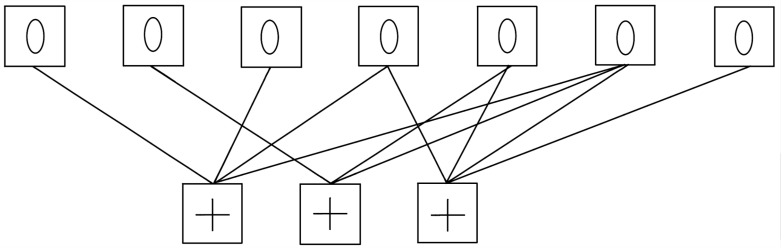

Figure 4: Tanner graph of the codeword $\{0000000\}$ of the Hamming code with $r = 3$.

[5]We show that in the exercise class

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

$$\begin{aligned} 0 &\rightarrow 1 \\ 1 &\rightarrow 0 \end{aligned}$$

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

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

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*

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

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*

00	01
01	01
10	01
11	00

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 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.

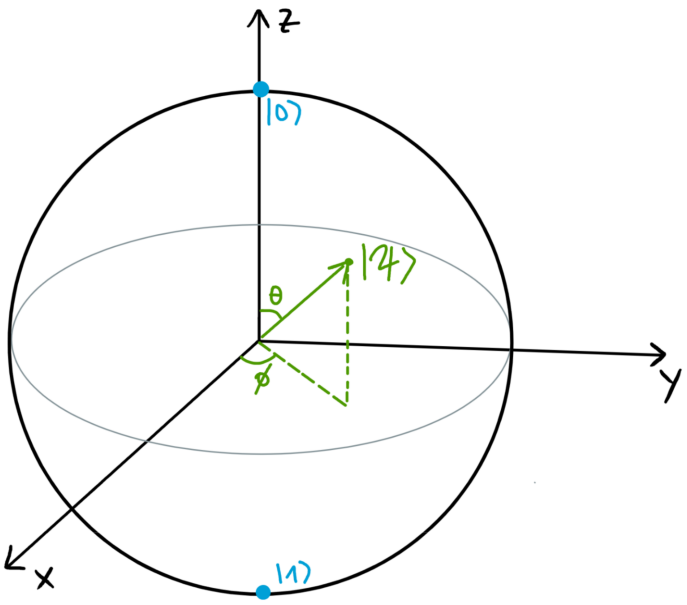

Figure 5: Bloch sphere

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

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

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, 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

[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.

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.,
$$\mathrm{tr}\left[T(X)\right] = \mathrm{tr}(X)$$
- 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 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

[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 $\{|\pm\rangle\}$.

- 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] $Z_1 Z_2$, we obtain "even parity" with probability

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

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 is always a state of the set $\{|\varphi\rangle, X_j|\varphi\rangle, Z_j|\varphi\rangle, X_j Z_j|\varphi\rangle\}$. 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

[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.

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

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

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

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

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

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

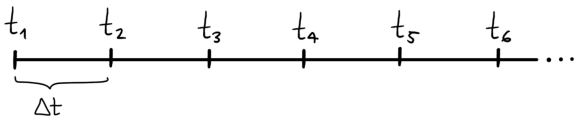

Figure 6: Discretisation of the time for continuous time errors.

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

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

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 $\mathcal{C} \equiv \{|\psi\rangle \mid S|\psi\rangle = |\psi\rangle \quad \forall S \in \mathcal{S}\}$.*

[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*

$$\begin{aligned} S_1 = Z_1Z_2, \quad S_2 = Z_2Z_3, \quad S_3 = Z_4Z_5, \\ S_4 = Z_5Z_6, \quad S_5 = Z_7Z_8, \quad S_6 = Z_8Z_9, \\ S_7 = X_1X_2X_3X_4X_5X_6, \quad S_8 = X_4X_5X_6X_7X_8X_9. \end{aligned} \tag{10}$$

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*

$$\bar{X} = X_1 \cdot X_9 = \prod_{j=1}^{9} X_j \qquad \textit{and} \qquad \bar{Z} = Z_1 \cdot Z_9 = \prod_{j=1}^{9} Z_j, \tag{11}$$

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. (11)) 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

$$\left(\begin{array}{c|c} H_z & 0 \\ \hline 0 & H_x \end{array}\right) \tag{12}$$

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

matrices are

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

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

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. (12).

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

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

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

$$A_X \equiv X_u \otimes X_d \otimes X_l \otimes X_r,$$

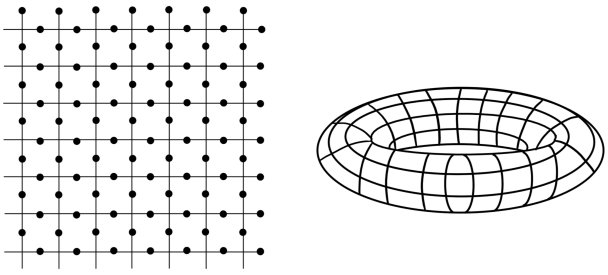

Figure 7: The topology considered for the toric code is a lattice with periodic boundary conditions, i.e., a torus.

$$B_Z \equiv Z_u \otimes Z_d \otimes Z_l \otimes Z_r,$$

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. (10)). 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.

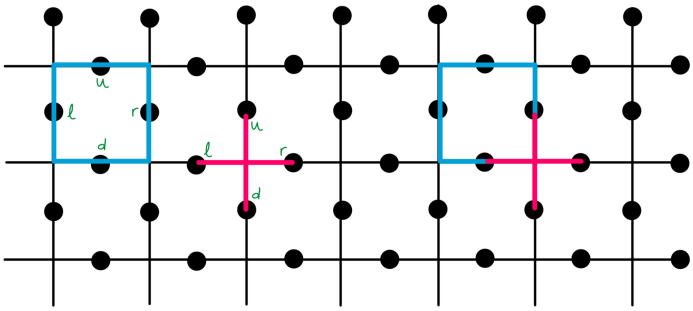

Figure 8: All plaquettes and crosses are stabilizers of the toric code. When a plaquette and a cross overlap, they do it always on two qubits.

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 illustra-

tions (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 plaquette. 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.

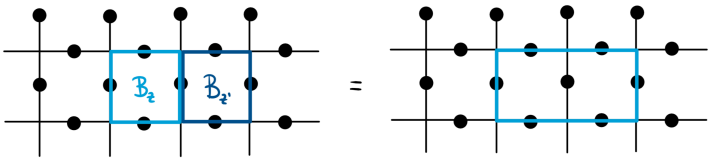

Figure 9: The multiplication of two plaquettes, B_Z and $B_{Z'}$, gives a bigger plaquette, which is also a stibilizer of the toric code.

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

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

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 $\bar{Z}_2$.

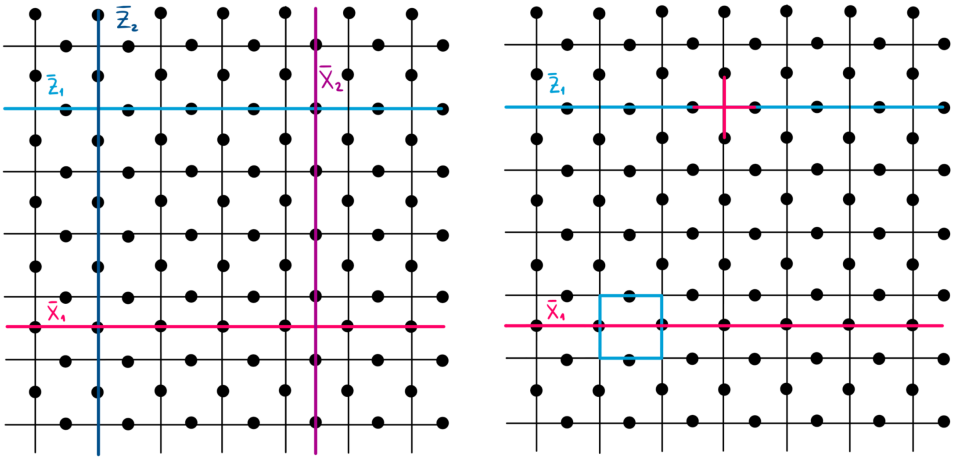

Figure 10: Logical operators of the toric code. When the logical operators $\bar{Z}_1$ and $\bar{Z}_2$ ($\bar{X}_1$ and $\bar{X}_2$) overlap with a cross (plaquette), they do it on two qubits.

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,

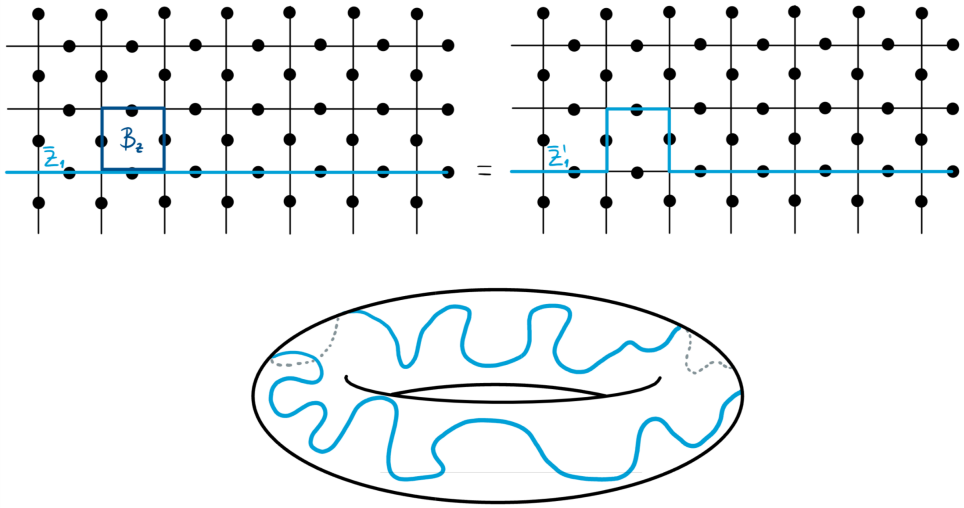

Figure 11: Multiplying a plaquette, B_Z, by a logical operator, e.g. $\bar{Z}_1$, gives new logical operator. Therefore any string around the torus is a logical operator.

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).

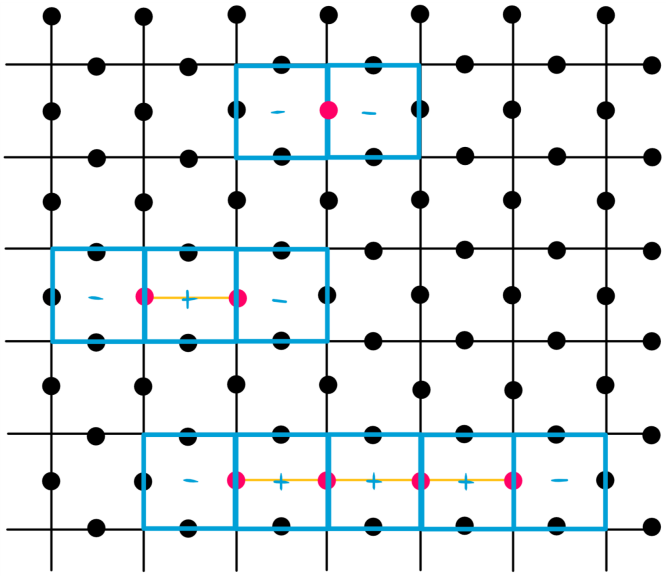

Figure 12: Any string error gives always a pair of syndromes at the ends of the string.

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., $O_{\text{logical}} \in \{\bar{X}_1, \bar{Z}_1, \bar{X}_2, \bar{Z}_2\}$.

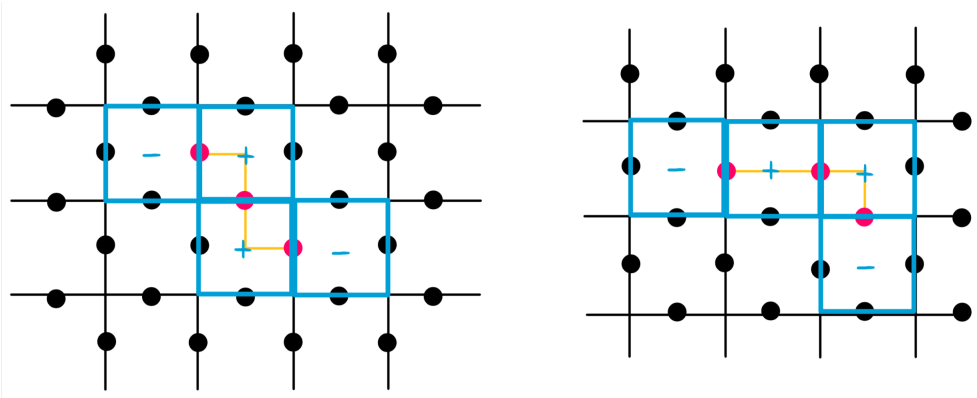

Figure 13: The correspondence between syndromes and string error is not unique, i.e., two different string error can give the same syndromes.

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, $n \sim L$

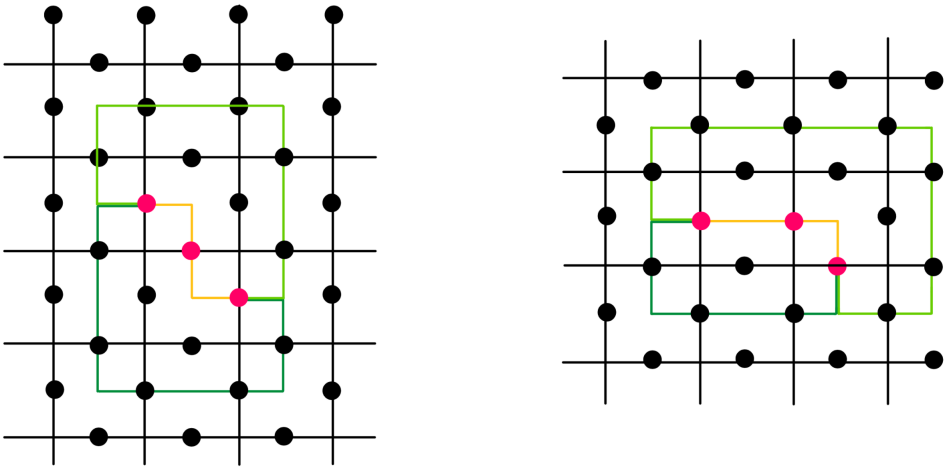

Figure 14: The correction operation corresponds on closing the string error to create a loop. Even if there exist several ways of closing the loop, we apply the shortest for convenience.

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

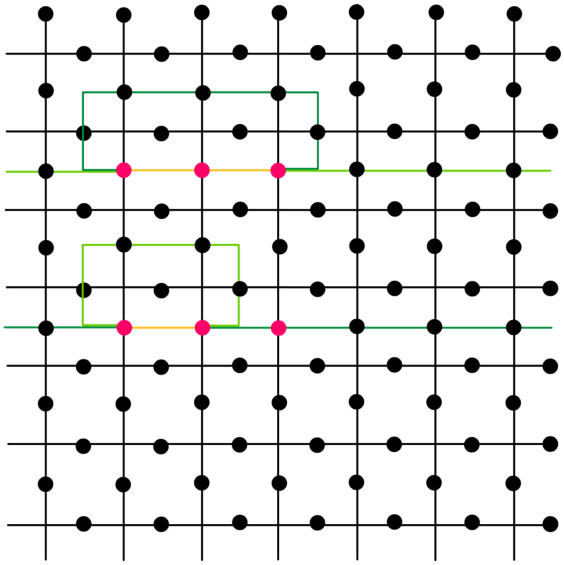

Figure 15: 7×7 lattice. When the string error has more than $\lfloor \frac{d-1}{2} \rfloor$, where $d = L$ for the toric code, the shortest way to close the loop creates a logical error.

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

$$\bar{P}(p,n) = \sum_{E \in \mathcal{F}} \pi(E), \tag{13}$$

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. (13) 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:

$$\bar{P}(p,n) \leq (1-p)^n \sum_{L} \sum_{u=\frac{|L|}{2}}^{|L|} \sum_{v=0}^{n-|L|} \binom{|L|}{u} \binom{n-|L|}{v} \left(\frac{p}{1-p}\right)^{u+v}. \tag{14}$$

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

$$\bar{P}(p,n) \leq (1-p)^n \sum_{l=d}^{n} N_{\text{con}}(l) \sum_{u=l/2}^{l} \sum_{v=0}^{n-l} \binom{l}{u} \binom{n-l}{v} \left(\frac{p}{1-p}\right)^{k+v}, \tag{15}$$

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

$$\bar{P}(p,n) \leq \sum_{l=d}^{n} N_{\text{con}}(l) 2^l p^{l/2} (1-p)^{l/2}. \tag{16}$$

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

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

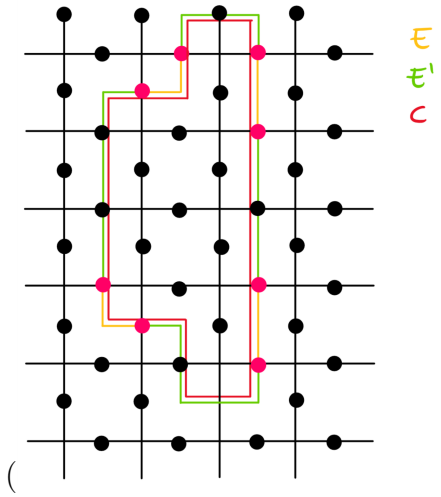

Figure 16: A loop C can be represented by two complentary errors, E and E' such that $C = EE'$.

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 saw that the toric code had some topological features. For instance, the logical operators of the Toric code can be represented as flexible strings wrapping around one direction of the torus (see Fig. 10). This is a

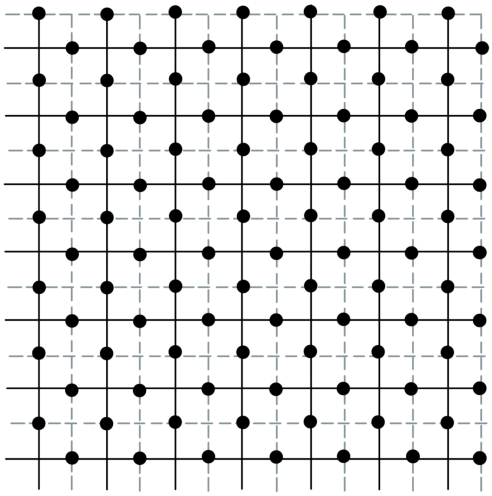

Figure 17: A dual lattice (dashed lines) can be defined for the toric code.

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

We will consider an extension of Stabilizer codes called commuting projector codes, defined as follows.

Definition 10.1. *Given projectors* $\{P_j\}$ *such that* $[P_j, P_k] = 0$ *for all* j, k*, 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 products of Pauli matrices. There exist plenty of 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

$$P_{\mathcal{C}} = \prod_j P_j. \tag{17}$$

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

$$P_{\mathcal{C}} O P_{\mathcal{C}} = c(O) P_{\mathcal{C}} \qquad \text{with} \qquad c(O) = \frac{\mathrm{tr}(PO)}{\mathrm{tr}P}, \tag{18}$$

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. (18) 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. (18) 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.

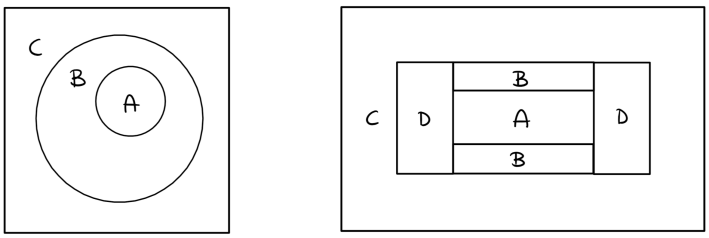

Figure 18: Lattice partitions used to define topological entanglement entropy.

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 The Bravyi-Poulin-Terhal (BPT) bound

In this section we derive a beautiful result by Bravyi, Poulin and Terhal, showing that for a local commuting projector code in two dimensions, the parameters of the code satisfy:

$$\mathcal{O}(\frac{n}{kd^2}) = 1 \tag{19}$$

This immediately tells us that constant rate and constant distance local topological CP codes do not exist. Recall that a constant rate code has $\mathcal{O}(k/n) = 1$ and a constant distance code has $\mathcal{O}(d/n) = 1$.
Up to here we have used arbitrary Pauli matrices as error model. Now, we change it and consider the erasure error model. 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

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

and the noise channel that erases a region A is

$$\mathcal{N}_A(\rho) = \mathrm{tr}_A(\rho) \otimes \frac{1}{d_A}\mathbb{I}_A, \tag{20}$$

where $d_A = 2^{|A|}$ is the dimension of the Hilbert space on region A. The primary difference between an arbitrary error and an erasure error is that we know where the erasure occurred.

It turns out that protection against arbitrary errors can be reduced to protection against erasure errors (and visa versa) via the following theorem:

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.*

Proof. The idea of the proof is the following. The error correction condition requires that $P_{\mathcal{C}} E_i E_j P_{\mathcal{C}} = c P_{\mathcal{C}}$, where $P_{\mathcal{C}}$ is the projector on the code space (Eq. (17)), while the error detection condition is $P_{\mathcal{C}} E P_{\mathcal{C}} = c P_{\mathcal{C}}$. 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. 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 restricted ourselves to erasure errors, we can naturally define the notion of correctability for a region of the lattice:

Definition 10.6 (correctability)**.** *Assume the erasure channel (Eq. (20)) and consider a code space, $\mathcal{C}$, defined on a lattice, Λ, and a region of the lattice, $A \subset \Lambda$. The region A is recoverable (correctable) if there exists a recovery map, $\mathcal{R}$, such that*

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

We will also make use of a notion closely related to correctability by a local channel; that certain subsystems on the lattice completely decouple:

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 A and C are decoupled on $\mathcal{C}$ if for any state $\rho \in \mathcal{C}$,*

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

Note that decoupling is a non-trivial property since, in general, regions A and C are entangled.
In order to prove the statement in Eqn. (19), we need to collect some 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)*[18]*. Then, it is satisfied that*

$$\rho_A \otimes \rho_B = \rho_{AB} \qquad \forall \rho \in \mathcal{C}. \tag{21}$$

[18]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.

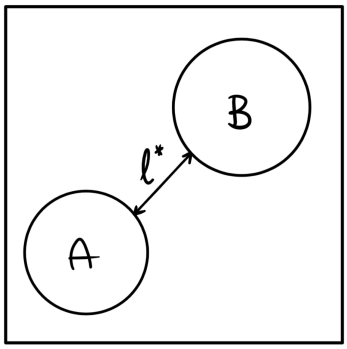

Figure 19: Lemma 10.1 requires two disconnected regions, A and B, that are separated by a distance bigger than l^*, where l^* is the radius of the support of any commuting projector.

Proof. Given a state ρ, the definition of the covariance between 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 the operator norm $||X||$ equals to the largest eigenvalue of X. The covariance can be related to the trace norm ($||\cdot||_1 = \mathrm{tr}(\cdot)$) as

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

The upper bound 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 follows from the equivalence of finite dimensional Shatten norms

$$\frac{1}{d_X}||X||_1 \leq ||X||_\infty \leq ||X||_1, \tag{22}$$

where d_X is the dimension of the matrix X. We will prove Eq. (21) by showing that the covariance between two regions of a local CPC code is zero.

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

R is a purification system. Then, we can write

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

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. (18) 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. (21) 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*

$$S_\rho(AA^c) + S_\rho(A) = S_\rho(A^c) \qquad \forall \rho \in \mathcal{C}. \tag{23}$$

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.,

$$\mathcal{R} \circ T(\Psi_{AA^cR}) = \Psi_{AA^cR}. \tag{24}$$

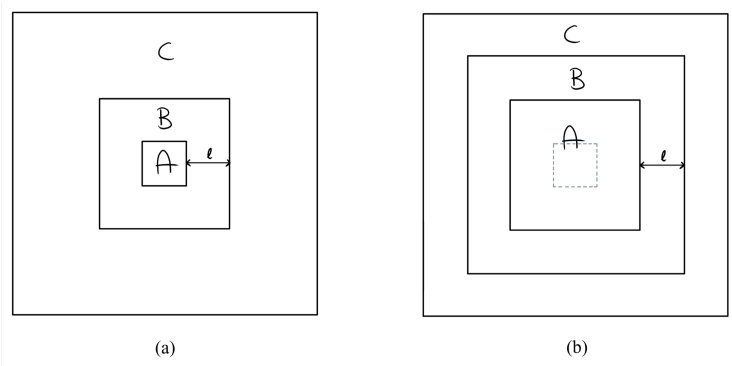

Figure 20: The largest correctable square region is $d \times d$. In order to prove it, we apply Lemma 10.3 iteratively until B is no longer correctable, i.e., $|B| \sim d$.

Recalling that entropy decreases under the action of any map and using Eq. (24), 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. (23) 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. (**??**). 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*

$$kd^2 \leq \alpha n, \tag{25}$$

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. (23) 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. (25). □

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. (25). 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.

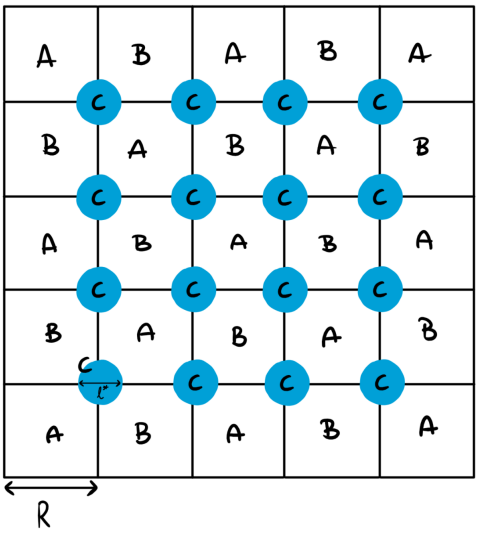

Figure 21: Partition of the lattice used to prove Theorem 10.1. Regions A and B are taken as large as possible and radius of C is at least $\frac{l^*}{2}$.