

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.

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 and optical storage¹. 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. *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.

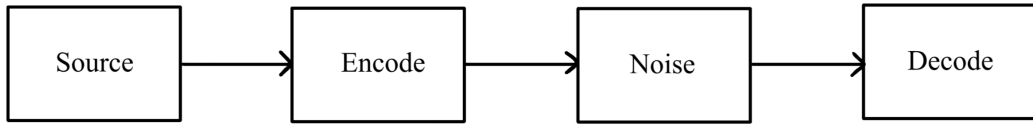
4. *Decode*

The decoding consists on taking the corrupted information and bringing it back to uncorrupted information.

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. ??):*

¹For 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.



Example: three-bit repetition code

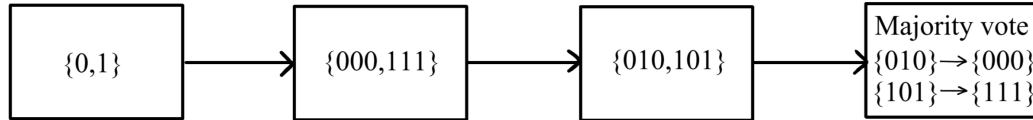


Figure 1: Every error correcting code can be broken up into four steps: source, encode, noise and decode. The three-bit repetition code is the simplest example of error correcting code.

1. Source

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

2. Encode

The simplest encoding of one bit is to encode it into three bits, i.e., $\{0, 1\} \rightarrow \{000, 111\}$. 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

$$\underbrace{\{0, \dots, 0\}}_n, \underbrace{\{1, \dots, 1\}}_n \in (\mathbb{C}^2)^{\otimes n}.$$

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. ???. 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 considered good, we would like to have $\frac{k}{n} \rightarrow \text{const}$ when $n \rightarrow \infty$. 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, $\{x_1, x_2, \dots, x_n\}$ ², where $x_j \in \{0, 1\} \forall j$. Our goal is to encode these logical bits, x_j , into a code, i.e., we want to map $\{x_1, x_2, \dots, x_n\}$ 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*

$$G = \left[\begin{array}{c} 1 \\ \vdots \\ 1 \end{array} \right] n.$$

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

$$G[0] = \left[\begin{array}{c} 0 \\ \vdots \\ 0 \end{array} \right] n \quad \text{and} \quad G[1] = \left[\begin{array}{c} 1 \\ \vdots \\ 1 \end{array} \right] n.$$

Note that the arithmetic is mod 2.

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

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

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

Therefore, we write G as

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

in such a way that

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

Note that the arithmetic is mod 2.

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

$$G = \underbrace{\left[\begin{array}{c} \\ \\ \\ \\ \\ \end{array} \right]}_k \Bigg\} n. \quad (1)$$

From the general form of the generator matrix (Eq. (??)), 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. ?? . Nevertheless,

it does not tell anything about decoding. We will see later on that classical linear codes have always a natural way of decoding³, but before we need to introduce a different representation for linear codes.

1.3 Parity-check matrix

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

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

$$Hy = 0 \quad \forall y \in \mathcal{C}, \quad (2)$$

where \mathcal{C} is the codespace, i.e., the n -bit space. Therefore, the codespace is the kernel of H according to Eq. (??). 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 $y_0 = \{0, \dots, 0\}$ and it occurs an error on the third bit, $e = \{0, 0, 1, 0, \dots, 0\}$. Then, the parity-check matrix will detect the error as

$$Hy_0 = 0$$

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

³This will not be true for quantum codes

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. ??). 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. ??), and thus they are crucial for decoding.

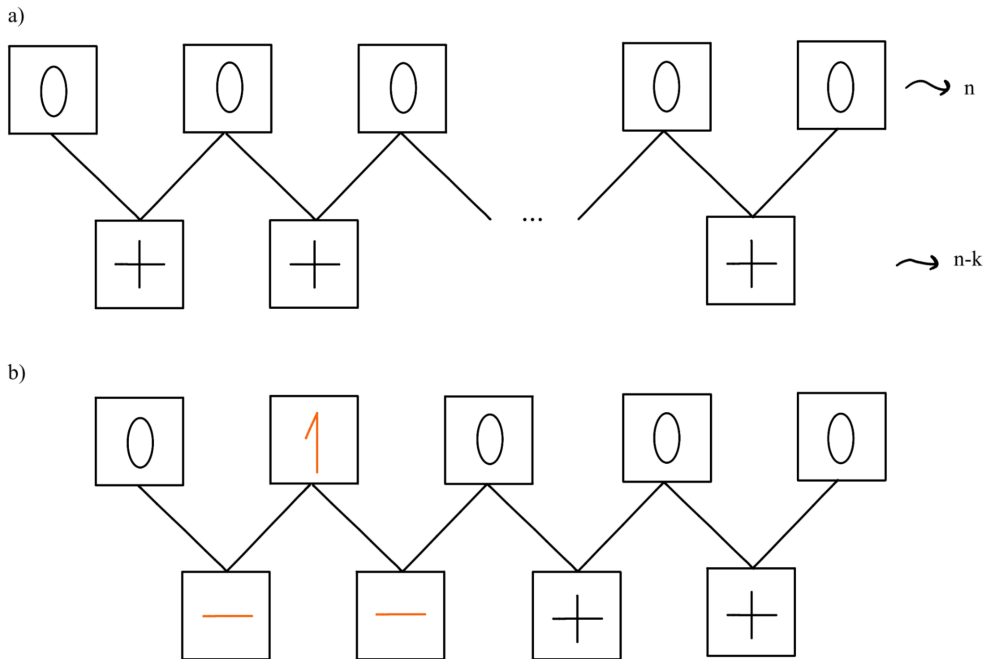


Figure 2: Tanner graph of a) the n -bit codeword $\{0, \dots, 0\}$, and b) the five-bit codeword $\{00000\}$ with a flip error on the second bit.

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. ??). 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.

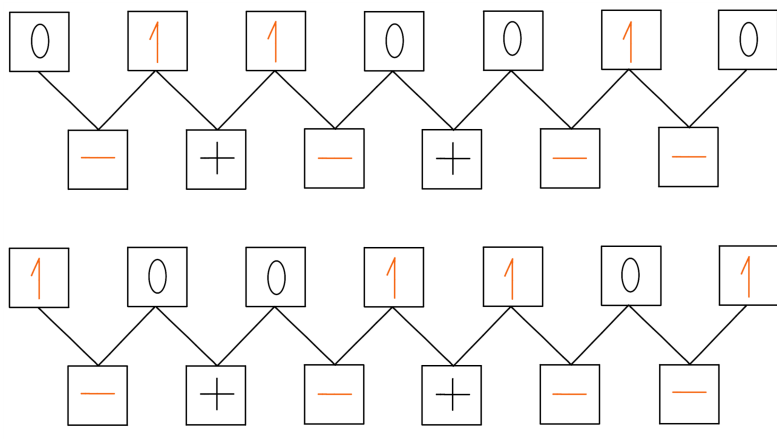


Figure 3: An error syndrome has two parents of errors. Given the initial codeword $\{0000000\}$, a flip-error on the second, third and sixth bit gives the same error syndrome than a flip-error on the first, fourth, fifth and seventh bit. However, the situation with only three errors is more likely.

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 following

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

$$\frac{k}{n} \xrightarrow{n \rightarrow \infty} \text{cnt},$$

$$\frac{d}{n} \xrightarrow{n \rightarrow \infty} \text{cnt}.$$

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 as⁴

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

1.6 Thresholds

In the last section we have weakly 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.

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

[Threshold 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

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

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 $f(n) \rightarrow 0$ when $n \rightarrow \infty$. 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 possible⁵. Obviously, this is not the general case.

Example 1.8 (The Hamming code) The Hamming codes are a family of linear codes with $[2^r - 1, 2^r - r - 1, 3]$, 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 parity-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 ?? 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.

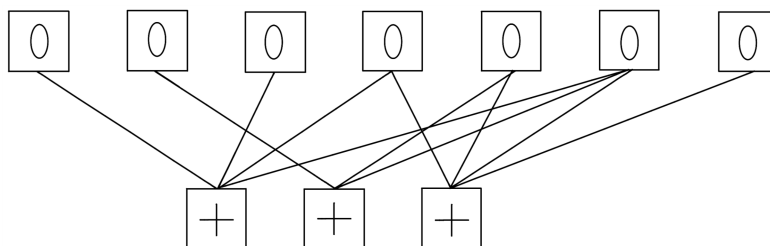


Figure 4: Tanner graph of the codeword $\{0000000\}$ of the Hamming code with $r = 3$.

⁵We 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 $|1\rangle\langle 1|$. Therefore, the uncertainty reflects our knowledge of the system.

The only single-bit operation⁶ in classical information is the bit-flip, which consists in

$$\begin{array}{l} 0 \rightarrow 1 \\ 1 \rightarrow 0 \end{array}$$

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

⁶When 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

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

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

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

2.2 Quantum information with one qubit

In this section, we introduce the basics of quantum information. The characterisation of a quantum system is first explained as well as how to operate on 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. ??). Each point of the Bloch sphere, which can be characterised by the unit vector $\vec{n} \equiv (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$, specifies a two-qubit state. Note that two antipodal points of the Bloch sphere correspond to two orthogonal states.

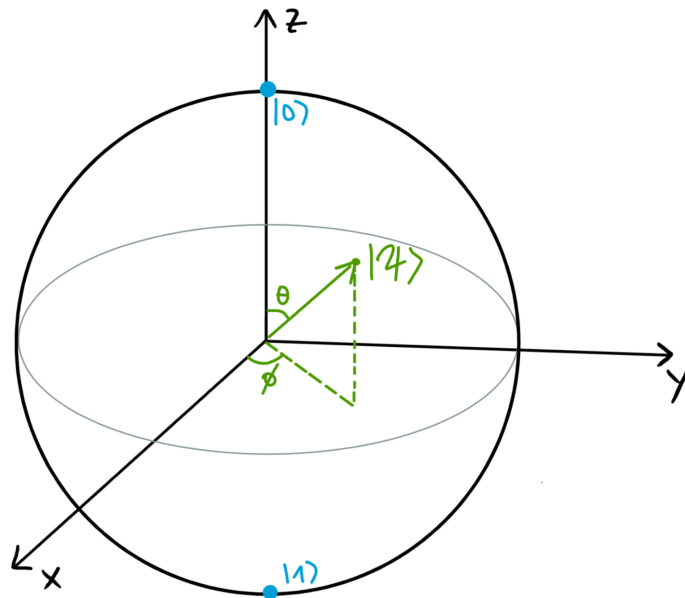


Figure 5: Bloch sphere

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

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

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

$$X \equiv \sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y \equiv i\sigma_y = i \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z \equiv \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \quad (3)$$

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 state⁷. 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 $\mathcal{E} : \mathcal{B}(\mathcal{H}_A) \rightarrow \mathcal{B}(\mathcal{H}_B)$ with the following properties. The map must be

- (i) Linear, i.e., $\mathcal{E} [\sum_i p_i \rho_i] = \sum_i p_i \mathcal{E}(\rho_i)$,
- (ii) Positive semidefinite, i.e., $\mathcal{E}(\rho) \geq 0 \quad \forall \rho \geq 0$,
- (iii) Completely positive, i.e., $(\mathcal{E}_A \otimes \mathbb{I}_C) [\rho_{AC}] \geq 0 \quad \forall \rho_{AC} \geq 0$ 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 (ii). 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

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

⁸In 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 operation $M = Z = |0\rangle\langle 0| - |1\rangle\langle 1| = P_0 + (-1)P_1$.*

Example 2.4 *Consider the measurement*

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

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 $p_k = \|\langle E_k|\varphi\rangle\|^2$. 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.,

$$\Omega = a_1\mathbb{I} + a_xX + a_yY + a_zZ. \quad (4)$$

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.,

$$\text{tr}[T(X)] = \text{tr}(X)$$

- 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 and 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 ??). 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 $|\varphi\rangle \rightarrow |\varphi\rangle|\varphi\rangle|\varphi\rangle$.
- *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 build 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 level considers groups of three qubits in order to correct against phase errors.

The logical qubits of the Shor code are

$$|\bar{0}\rangle = \frac{1}{\sqrt{2^3}} (|000\rangle + |111\rangle) (|000\rangle + |111\rangle) (|000\rangle + |111\rangle), \quad (5)$$

$$|\bar{1}\rangle = \frac{1}{\sqrt{2^3}} (|000\rangle - |111\rangle) (|000\rangle - |111\rangle) (|000\rangle - |111\rangle). \quad (6)$$

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.,

$$|\pm\rangle = \frac{1}{\sqrt{2}} (|000\rangle \pm |111\rangle), \quad (7)$$

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

$$\begin{aligned} |\bar{0}\rangle &= \frac{1}{\sqrt{2^3}} (|000\rangle + |111\rangle)^{\otimes 3} = |\bar{+}_{(1)}\rangle^{\otimes 3} \\ |\bar{1}\rangle &= \frac{1}{\sqrt{2^3}} (|000\rangle - |111\rangle)^{\otimes 3} = |\bar{-}_{(1)}\rangle^{\otimes 3} \end{aligned}$$

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 flips⁹. Therefore, 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 quantum error correction decodes using parity measurements because they do not affect the logical information. Classically we have already seen the parity measures with the parity-check matrix (see Section ??) and the Tanner graph (see Section ??). 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 measurements, 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

⁹Recall that

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

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 $Z_1 Z_2$ 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, $Z_2 Z_3$. 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.,

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

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 $Z_1 Z_2$	Result of $Z_2 Z_3$	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 $X_1 X_2$ and $X_2 X_3$. The operator $X_1 X_2$ can be written as

$$\begin{aligned} X_1 X_2 &= (|++\rangle\langle++| + |--\rangle\langle--|) - (|+-\rangle\langle+-| + |-+\rangle\langle-+|) \\ &= Q_+ - Q_-, \end{aligned}$$

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 $X_1 X_2$ 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 correct it by 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 $|\pm\rangle$, 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

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

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 measure¹⁰ $Z_1 Z_2$, we obtain “even parity” with probability

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

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

¹⁰Here 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. (??)). 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 ??). 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 are¹¹

$$\begin{aligned} & Z_1 Z_2, Z_2 Z_3, \dots, Z_{n-1} Z_n, \\ & Z_{n+1} Z_{n+2}, \dots, Z_{2n-1} Z_{2n}, \\ & \quad \vdots \\ & Z_{n^2-n+1} Z_{n^2-n+2}, \dots, Z_{n^2-1} Z_{n^2}, \\ & X_1 \cdots X_n, \dots, X_n \cdots X_{2n}. \end{aligned}$$

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 space¹², \mathcal{H} , a quantum error correcting code 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

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

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

¹²For 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-Laflamme 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-Laflamme 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

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

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

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

The first term of Eq. (??) carries no error, and thus we do not need to correct it. The second term of Eq. (??) contains single-qubit errors, which we have seen in the previous chapter that the Shor code can correct. The rest of the terms Eq. (??) correspond to errors on more than one qubit and we do not know a general recovery map for them¹³. This means that the probability with which we can protect against errors on every single-qubit is given by the remaining terms ($\mathcal{O}((1 - p)^7 p^2)$) in Eq. (??). If we consider the Shor code on n^2 , the term $\mathcal{O}((1 - p)^7 p^2)$ 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

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

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

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

¹³The Shor code is able to correct against two-qubit errors such as $X_1 Z_2$, but it fails for errors of the form of $X_1 X_2$.

We can expand the error operator as

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

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. ??), 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

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

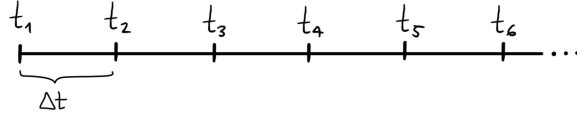


Figure 6: Discretisation of the time for continuous time errors.

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 these 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. (??).

Note that the Pauli group has order eight, $|\mathcal{P}_1| = 8$, which means that the group has eight elements. These elements are related by the commutation properties of the Pauli matrices, i.e.,

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

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, $\{G_{\vec{\alpha}}, G_{\vec{\beta}}\} = 0$. 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 abelian¹⁴ subgroup of \mathcal{P}_n . Then, a stabilizer code, \mathcal{C} , is defined as $\mathcal{C} \equiv \{|\psi\rangle \mid S|\psi\rangle = |\psi\rangle \ \forall S \in \mathcal{S}\}$.*

¹⁴A 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, $\mathcal{S} = \langle \{S_k\}_{k=1}^8 \rangle$, which can be written as

$$\begin{aligned} S_1 &= Z_1 Z_2, & S_2 &= Z_2 Z_3, & S_3 &= Z_4 Z_5, \\ S_4 &= Z_5 Z_6, & S_5 &= Z_7 Z_8, & S_6 &= Z_8 Z_9, \\ S_7 &= X_1 X_2 X_3 X_4 X_5 X_6, & S_8 &= X_4 X_5 X_6 X_7 X_8 X_9. \end{aligned}$$

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

$$\begin{aligned} S_1 S_2 |\varphi\rangle &= Z_1 Z_3 |\varphi\rangle \\ &= Z_1 Z_3 (\alpha |\bar{0}\rangle + \beta |\bar{1}\rangle) \\ &= Z_1 Z_3 (\alpha |+++ \rangle + \beta |-- \rangle) \\ &= Z_1 (\alpha |-++ \rangle + \beta |+- \rangle) \\ &= (\alpha |+++ \rangle + \beta |-- \rangle) \\ &= |\varphi\rangle, \end{aligned}$$

where $|\bar{0}\rangle$, $|\bar{1}\rangle$ and $|\pm\rangle$ are defined in Eq. (??), Eq. (??) and Eq. (??). Thus, we have $S_1 S_2 |\varphi\rangle = |\varphi\rangle$, which implies that $S_1 S_2 \in \mathcal{S}$.

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

$$\bar{X} = X_1 \cdot X_9 = \prod_{j=1}^9 X_j \quad \text{and} \quad \bar{Z} = Z_1 \cdot Z_9 = \prod_{j=1}^9 Z_j, \quad (9)$$

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

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

where $|\bar{0}\rangle$ and $|\bar{1}\rangle$ are defined in Eq. (??) and Eq. (??). 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. (??)) would be

$$\begin{aligned} \bar{X}S_8 &= X_1X_2X_3, \\ \bar{X}S_2 &= X_1Y_2Y_3X_4X_5X_6X_7X_8X_9, \\ \bar{X}S_8S_2 &= X_1Y_2Y_3. \end{aligned}$$

¹⁵Stabilizer 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 makes 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 ??).

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 eigenvalue $+1$ 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 ??).

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

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

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

$$\left(\begin{array}{c|c} H_z & 0 \\ \hline 0 & H_x \end{array} \right) \tag{10}$$

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

