A Methods Focused Guide to Quantum Error Correction

Abdullah Khalid

September 12, 2022

Abstract

These set of notes introduce quantum error correction to someone already familiar with the principles of quantum mechanics and quantum computing. A special emphasis in these notes is made on the methods of solving problems, which is frequently missing from other introductions of the subject. These include both mathematical methods and computational methods.

Warning

Currently, these notes are life-threatening if used to learn anything.

I am still figuring out the typographical formatting.

Many pictures are missing.

I am still creating exercises for the material already written.

Contents

1 Introduction

The field of error correction is mainly concerned with information processing with unreliable physical appratus. In such computing or communication units, the information is subjected to errors, which must be dealt with, if the results of the information processing

is to be trusted.

While error correction for classical computation and communication is a well developed and has been for many decades, only recently has it become central to the field of quantum computation and communication. This development is necessary, as without quantum error correction we are unlikely to construct useful quantum computers with current and forseeable levels of engineering technology.

Here, I aim to write a tutorial on quantum error correction (QEC) starting from the very basics. A particular focus in these tutorials will be made on the methods for solving problems, something remiss from other tutorials which only focus on the mathematical theory. The hope is that these tutorial will act as a self-contained introduction to self-motivated students.

The theory of QEC should be learned in tandem with classical error correction (CEC), as some of the most important quantum error correction codes (QECC), namely the Calderbank-Shor-Steane (CSS) codes, are built from classical error correction codes (CECC). Therefore, I review essential CEC theory before proceeding with QEC theory.

2 Prior work

There are number of other introductions to quantum error correction that one will find useful. These are listed here in reverse chronological order.

- J. Roffe, Quantum Error Correction: An Introductory Guide, ArXiv:1907.11157 (2019).
- S. J. Devitt, W. J. Munro, and K. Nemoto, Quantum Error Correction for Beginners, Rep. Prog. Phys. 76, 076001 (2013).
- D. Gottesman, An Introduction to Quantum Error Correction and Fault-Tolerant Quantum Computation, arXiv:0904.255 (2009).
- J. Preskill, Quantum Error Correction, `http://theory.caltech.edu/~preskill/`

ph229/notes/chap7.pdf.

- D. Gottesman, Stabilizer Codes and Quantum Error Correction, arXiv:9705052 (1997).

In the age of video, I would also point to some excellent online courses that a beginner can avail from.

- R. Raussendorf, Fault-tolerant quantum computation, https://phas.ubc.ca/~raussen/Phys523/Phys523.html (2021).
- D. Gottesman, B. Yoshida, Quantum Error Correction and Fault Tolerance, https://www2.perimeterinstitute.ca/personal/dgottesman/QECC2018/ (2018)

I have learned from all these resources. However, any mistakes in the following are all my own.

3 A model for error correction problems

Consider the scenario where Alice wants to send information to Bob, via a transmission line/channel. This channel is a noisy one, so any information (strings of bits for example) is subject to errors, for example bits are flipped or lost. Therefore, in order to reliably transmit information, Alice and Bob must agree on some method by which errors are detected and possibly corrected. The problem of finding such methods is one instance from the set of error correction problems.

It may seem that the above scenario is very specialized to a communication problem, but we can imagine Alice and Bob to be two points of time in a computational circuit. However, this situation is a bit more complicated because in such circuit there are computational gates that process the information. So their function must also be accounted for in the error correction method.

Finally, note that I have not made any mention of classical or quantum information in the above model, and this model is sufficient for the types of CECCs and QECCs I will

discuss below.

4 Classical Error Correction

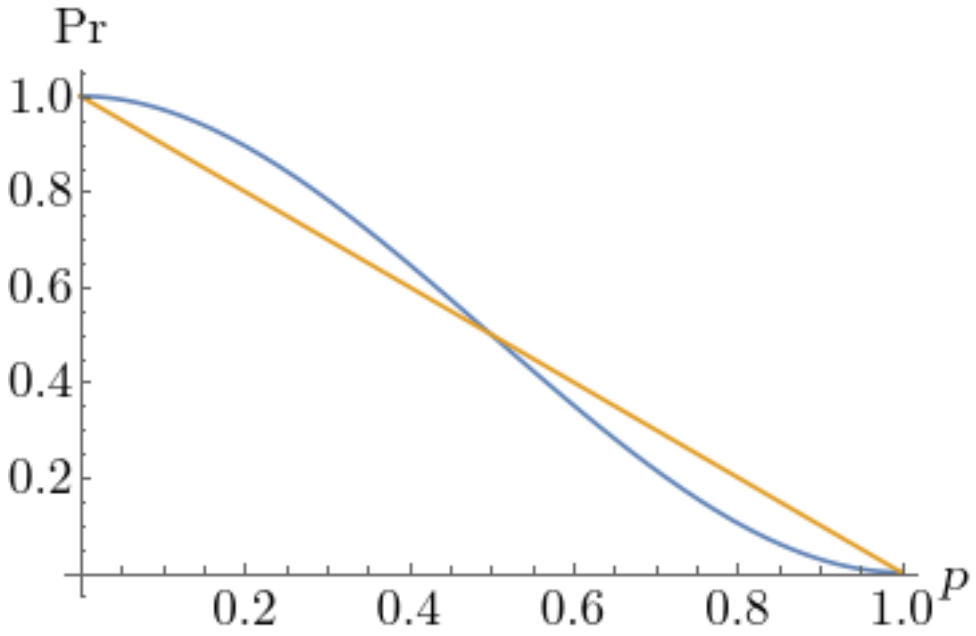
I begin by making a more concrete model of the Alice-Bob scenario above. Suppose that Alice wants to send Bob some information, which is in the form of a long stream of bits, called the data stream. The information has to be sent through a noisy transmission channel, as stated above. A simple such channel is one in which, with probability p, every bit sent via the channel is flipped. Otherwise, with probability $1 - p$, the bit is transmitted with no error.

Suppose, that Alice wants to send the data stream 101 to Bob. If she uses the naive strategy and sends the stream as is, the channel could corrupt the bits and, for instance, transform it into 100. Bob would not have any way to knowing that an error occured during transmission or not. This method only succeeds if there is no error, which has probability $(1 - p)^3$.

Instead, Alice and Bob can agree on some process that increases the probability of correct transmission. One such method is called the 'repetition code'. In this method, instead of sending 0, Alice sends 000, and instead of sending 1, Alice sends 111. This means that if the data stream is 101, then Alice sends the *code-string* 111000111. Suppose, that Bob receives the corrupted code-string 101100111, in which the second and fourth bits have been flipped by the noisy channel. To correct any errors, he will apply a *majority voting algorithm* to each block of three bits in the received code-string. The first block is 101, and since there are two 1s in it and one 0, he infers that the first sent bit is likely to be 1. Similarly, he infers from the second and third blocks of three, that the second and third bits are likely to be 01. Hence, he correctly infers that the message was 101.

Notice that this method only works if there is only one bit flip in each block of three. If there are two or more errors, then Bob will inferred bit string will be different from the one Alice intended to send. For example, if the corrupted code-string was 101110111, then using the same algorithm Bob infers that that message was 110.

Using the repetition code, what is the probability that Bob correctly infers the correct bit-string? Each block of three is treated separately, so we only need to estimate the probability of correct inference from each block. The probability of zero or one errors in a three-bit block is clearly $(1-p)^3 + 3p(1-p)^2$, while the probability of two or three errors is $3p^2(1-p)+p^3$. Bob makes the correct inference in the former case, so that is the probability of successful transmission. Now, some basic analysis shows that as long as $p < 0.5$, the repetition code is more successful at transmitting the correct message than the naive strategy. This is illustrated in the following graph.



The blue line shows the probability of success of the repetition code, and the orange the probability of success of the naive method.

The above discussion demonstrates a general principle of error correction: *With the right method, one can increase the probability of successfully transmitting messages correctly.*

We require one additonal construction to complete our example. During computation, gates are applied to the sets of bits. Once, the message has been transformed into three-bit blocks, how does one apply the gates? For the repetition code, this is quite simple. Suppose the three-bit block is $b_1b_2b_3$. A one-bit gate g, can be applied by applying it to each bit in the block So the output of the gate is $g(b_1)g(b_2)g(b_3)$. For two three-bit blocks $b_1b_2b_3$ and $c_1c_2c_3$, a two-bit gate simply

4.1 Linear classical codes

The previous subsection showed an example method, called a *code*, to protect information. Now we formalize the notions present above, but only to the level of linear classical codes, rather than nonlinear classical codes. This is mainly because only linear classical codes are relevant to quantum codes. We will also restrict ourselves to binary codes, because we are interested in qubit codes in the quantum case.

4.1.1 The definition of a code

We define the abstract notion of a code using the notions of linear algebra.

Definition 1. *The binary field is labelled* $\mathbb{F}_2$*. A* m*-dimensional vector space over the binary field is labelled as* $\mathbb{F}_2^m$*.*

As discussed above, the information to be sent exists in the form a some long data stream. However, linear codes usually break up the stream into blocks of fixed size, and treats each block independently. Therefore, for our analysis, we only need to consider the blocks. Therefore, we establish the following terminology.

Definition 2. *The information to be protected is a bit string of length* k*, and each such string is called a message.*

Definition 3. *The message space is the vector space with all* 2^k *possible messages. This vector space is clearly defined over* $\mathbb{F}_2$*. Hence, it is the space* $\mathbb{F}_2^k$*.*

Exercise 1. What is the dimension of the message space?

As in the repetition code, for linear codes each message is transformed into a longer bit-string before being transmitted. This longer bit-string is called a codeword, and the tranformation from message to codeword is called encoding the message.

Definition 4. *A message is transformed or encoded to a codeword, which is a bit string of length* n*.*

The codewords are vectors in a larger space, defined as follows.

Definition 5. *The codespace is a n-dimensional vector space over $\mathbb{F}_2$. We require that $n \geq k$.*

Finally, we can abstractly define the notion of a code.

Definition 6. *A binary linear code $\mathcal{C}$ of dimension k and length n is a k-dimensional subspace of $\mathbb{F}_2^n$. Compactly, we can write $\mathcal{C} \subset \mathbb{F}_2^n$. Such a code is labeled as a $[n, k]$ code.*

On it's own, the above definition allows us to identify if a particular construction is a valid code.

Exercise 2. Check that the repetition code described above is indeed a code. Identify the message space, the codewords, and the codespace.

Other codes that you will encounter can be defined as above. However, we will need more structure in order to understand and make use of codes.

4.1.2 Encoding process

The definition of a binary linear codes is abstract. To encode messages we need a more concrete definition that gives well-defined calculational procedures. This is done by given an explicit definition of the subspace at the heart of the linear code definition.

For any code, we can identify a basis set of the subspace. For instance, for the repetition code, the basis set is $\{000, 111\}$. In general, there will be k such vectors because the subspace is k-dimensional, and each vector will be of length n. We can collect these k vectors into the rows of a matrix, called the *generator matrix*.

Definition 7. *The generator matrix, G, is a $k \times n$ matrix whose rows are a basis of the code.*

Any message m can be encoded into a codeword c using the generator matrix. The codewords of the code are all in the row space of its generator matrix G, i.e. the set of codewords is

$$\left\{ c = mG : m \in F_2^k \right\}, \tag{1}$$

where m is the message in the form of a row vector.

Example: For the repetition code, a generator matrix is

$$G = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}. \tag{2}$$

Using this, we find the codewords to be

$$\begin{pmatrix} 0 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}, \tag{3}$$

$$\begin{pmatrix} 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}, \tag{4}$$

which is what we had before. In this way, we have an explicity encoding process for any code.

Definition 8. *A code may be defined using a generator matrix.*

Because basis for the code C is not unique, there can be many possible generator matrices for the same code C, and as we will see, some of matrices will have more calculational utility than others. We can convert between different codes by doing row operations on the generator matrix, which will not change the row space. Among the forms of the generator matrices, there is a standard form, which is when $G = [I_k|A]$, where I_k is the $k \times k$ identity matrix and A is a $k \times (n-k)$ matrix. The standard form can be obtained via row operation from any other form.

4.1.3 Hamming codes

One of the most interesting code is the Hamming codes. This is actually a family of codes, but we will discuss the simplest among them. This is a $[7, 4]$ code, in which the first 4 bits of the codeword are the message, and the last 3 bits are *parity bits*.

Exercise 3. Identify the message space and the codespace of the Hamming code.

Definition 9. *The parity of a bit string is the number of ones in it. If there are even (odd) number of ones, the parity of the string is said to be even (odd).*

Note that $4C3 = 3$. In the $[7, 4]$ Hamming code, for each combination of three of the four data bits, we calculate the parity. With this method, as we will see, the code can detect and correct single-bit errors. We now construct the generator matrix G for the Hamming code, using the description above.

The first four bits have to be copied as is, so the first four columns of G must be the identity matrix. The next three columns must be the various ways of summing up the data bits. Hence, we get

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}. \tag{5}$$

This is clearly in the standard form.

4.2 Identifying and correcting errors

The previous section provided us with a way of encoding any message m into a codeword c. Now, we discuss how to identify any errors that occur when the codeword is sent through the noisy channel.

Suppose that Bob receives the block $\tilde{c}$. This block could or could not be corrupted in some way. Every code has a certain tolerance for how many errors it can correctly identify and subsequently correct. The reason for this is that all Bob can do is try to determine the *closest* codeword c to the received $\tilde{c}$. This is the best guess under the assumption that the each bit has an independent probability of being flipped. If there are too many errors, then the closest codeword to $\tilde{c}$ is no longer the one sent by Alice, and error correction fails.

The notion of closeness of bit strings can be formalized via the Hamming distance.

Definition 10. *The Hamming distance between two bit strings s, t is the number of places where they have different symbols. This will be denoted $d_H(s, t)$.*

The Hamming distance determines the number of substitutions or errors required to transform s into t or vice versa. Hence, it is a very natural definition for our purposes. The Hamming distance can be calculated as

$$d_H(s,t) = \sum_i (s_i \oplus t_i) \mod 2 \tag{6}$$

The brute force method of checking if an error and which error has occured is calculating the Hamming distance between $\tilde{c}$ and every possible codeword $\{c^i\}$. However, there are 2^k codewords and we would like a more efficient process.

This is done by exploiting the fact that the code has some structure in built. For instance, in the Hamming code we have recorded the parity of various subsets of the message bits. Bob should be checking if in $\tilde{c}$ the parity bits are consistent with the information bits, and if not what is the smallest change that resolves consistency. Generically, we need a set of constraints that all codewords obey.

These contraints naturally emerge when we recall that the code is a subspace of a vector space, and therefore it is the kernel of some linear transformation. This transformation, which we will call H, maps the codewords to zero.

Definition 11. *The matrix $H \in \mathbb{F}_2^{(n-k)\times n}$ is a parity-check matrix for code C if $C = \{x \in \mathbb{F}_2^n : Hx^T = 0\}$.*

As there are $n-k$ rows to this matrix, the matrix places $n-k$ constraints on the elements of the codespace, leaving behind a $n-(n-k) = k$-dimensional space.

How does one obtain the parity-check matrix from the generator matrix? As the rows of the generator matrix form the basis of the code, we can use them to place sufficient constraints on the parity-check matrix.

Lemma 1. *For a code C with generator matrix G and parity-check matrix H, we have the constraint $HG^T = 0$.*

Proof. Recall that any codeword $c = mG$, so $c^T = G^T m$. Now, by the definition of H,

we have $Hc^T = 0$, or $HG^T m = 0$. But this last statement is true for all m, which is only possible if $HG^T = 0$. □

Using this theorem, we give a procedure when G is in standard form, using the following lemma.

Lemma 2. *If $G = [I_k|A]$, then $H = [-A^T|I_{n-k}]$, where I_k is $k \times k$, A is $k \times (n-k)$, A^T is $(n-k) \times k$ and I_{n-k} is $(n-k) \times (n-k)$.*

Proof. We use the fact that $HG^T = 0$. Then,

$$HG^T = \begin{pmatrix} -A^T & I_{n-k} \end{pmatrix} \begin{pmatrix} I_k \\ A^T \end{pmatrix} = -A^T I_k + I_{n-k} A^T = -A^T + A^T = 0. \tag{7}$$

Hence, proved. □

The above lemma gives a very straightforward procedure for computing the parity check matrix.

4.2.1 Error syndromes

We have learned that the parity-check matrix can be used to determine if $\tilde{c}$ is a codeword or not. But does it tell us more? Imagine Alice sends the codeword c and it gets distorted by error e to $\tilde{c} = c + e$. Now, the effect of the parity-check matrix is,

$$s = H\tilde{c}^T = Hc^T + He^T = 0 + He^T = He^T. \tag{8}$$

The quantity s informs us of what error has occured, and is called an error syndrome. It is a vector of length $n - k$. One can precompute a look-up table for s vs e and use this to determine the error at runtime.

4.2.2 Correcting the error

Once one knows the error e that has occured, to correct the codeword, one can simply note that $\tilde{c} + e = c + e + e = c$. So the codeword is straightforwardly obtained.

4.3 Recovering the message

Bob now has the correct codeword c with high probability, and wants to recover the corresponding message m. This can be done using the definition $c = mG$. Note that the rows of G are the basis of the code. So, there exists a matrix U, such that $\bar{G} = GU$ has orthogonal rows. Then $\bar{G}\bar{G}^T = I$. In other words, $GU(GU)^T = GUU^TG^T = I$. Hence, we can right multiply $c = mG$ by...

4.3.1 Distance of a code

A crucial quantity of interest when discussing codes, is their distance. The distance determines the number of errors that can be corrected. The distance is the minimum Hamming distance between any two codewords.

Definition 12. *For code C the distance d is*

$$d = \min_{c,c' \in C} d_H(c, c'). \tag{9}$$

This seems to be a tedious optimization problem. However, we can simplify the calculation in the following way. First, we define the weight of a bit string.

Definition 13. *The weight of a string s, denoted $wt(s)$, is the number of ones in the string.*

The weight can also be thought of as as the Hamming distance of s from the all 0 string, i.e. $wt(s) = d_H(s, 0^n)$.

Next note that the zero message, $m = 0^k$, maps to the zero codeword $c_0 = 0^k G = 0^n$. Then, using the properties of the Hamming distance and the fact that the codespace is a

vector space, and letting $c = Gx \neq c' = Gx'$, we have

$$\begin{aligned} d_H(c, c') &= d_H(Gx, Gx'), && (10)\\ &= d_H(Gx - Gx', 0^k), && (11)\\ &= d_H(G(x - x'), 0^k), && (12)\\ &= d_H(c'', 0^k), && (13)\\ &= wt(c''), && (14) \end{aligned}$$

where in the second last step we have defined $c'' = G(x - x')$, which must exist in C because $x - x'$ is part of the message space, and $c'' \neq 0$. So we have discovered that distance of a code,

$$d = \min_{c \in C} wt(c), \tag{15}$$

is just the minimum weight string.

Another theorem simplifies the calculation even more.

Theorem 1. *The distance d of a code C is equal to the minimum number of linearly dependent columns of the parity check matrix H of the code.*

The use of the distance of a code is given by the following theorem.

Theorem 2. *A code with distance d can*

- *Correct at most $d - 1$ erasure errors,*
- *Detect at most $d - 1$ bit flip errors,*
- *Correct at most $\left\lfloor \frac{d-1}{2} \right\rfloor$ bit flip errors.*

Therefore, a single quantity tells us the power of a code at detecting and correcting various types of errors.

Exercise 4. Compute the distance of the $[7, 4]$ Hamming code, using both methods outlined above. Show $d = 3$.

Definition 14. *If the distance d of a code C is known, then C is denoted as $[n, k, d]$.*

4.3.2 Dual codes

Recall that the generator matrix G is a matrix whose rows form the basis of the code C. In other words, the only defining feature of G is that it's rows are independent. Now, we also know that the rows of the parity-check matrix H are also independent; otherwise, the kernel of H will have dimension less than k and could not be equal to the code C. Therefore, H can also form a code, and such a code is called the dual code $C^\perp$ of C.

Definition 15. *Given code C with parity-check matrix H, the code with generator matrix H is called the dual code to C and denoted by $C^\perp$.*

Another equivalent and perhaps more intuitive definition of the dual code is given by the fact that $HG^T = 0$, which leads to the fact that

Definition 16. *The dual code $C^\perp = \{x \in \mathbb{F}_2^n : x \cdot c = 0, \forall c \in C\}$.*

Exercise 5. If code C is a $[n, k]$ code, then what is the type of $C^\perp$. You can use the dimensions of the matrix H to determine this.

An interesting category of codes are those which are self-dual. These are codes for which $C = C^\perp$, i.e. the generator matrix G of C and the parity-check matrix H of C are equal to each other. From the dimension of these matrices, this clearly implies that $n - k = k$ or $k = n/2$. Hence, n must be even.

Example 1. The extended Hamming code is a $[8, 4]$ code that is equal to the Hamming code with an additional column in the parity check matrix that computes the parity of all four message bits. Show that this code is self-dual.

4.3.3 Logic gates on encoded states

We complete the discussion of classical codes with how to implement logic gates on the codewords. Our problem can be stated as follows. Let x, y, z be messages, and let f be some k-bit operation such that $z = f(x, y)$. Then we are interested in finding an n-bit operation F such that $F(Gx, Gy) = Gz$.

4.4 What are good codes?

5 An example oriented introduction to quantum error-correction

We are now in a position to introduce quantum codes. Our goal is to understand how a set of qubits transmitted through a noisy channel can be protected from any errors they experience. The process is similar to classical codes in that we will take the state of k-qubits and encode it into the state n-qubits. However, our task will be complicated by the fact that the operations on qubits are not just bit-flips but any possible interaction with the environment. Fortunately, we will discover that this possibility of infinite types of errors will not be too large a hindrance and we can construct good codes.

The core theory for quantum error correction is fairly involved, so we begin our discussion with some motivating examples. These examples will illustrate the types of noise that qubits experience and simple methods of mitigating that noise. We will also discuss how by *concatenating* these codes, we can improve the threshold behavior. This will lead into a discussion of *fault-tolerant* quantum computation.

5.1 Quantum repetition code for bit-flips

Suppose that Alice has a quantum transmission channel to Bob that is noisy. Let's assume for starters that the noisy channel only applies the quantum bit-flip operator X to any qubits that pass through it, with probability p. We will dicuss channels with different errors later on.

Alice wants to transmit to Bob the quantum state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. If she sends the state as is, there is a probability $(1-p)$ that Bob receives the state $|\psi\rangle$ and probability p that Bob receives the state $\left|\tilde{\psi}\right\rangle = \beta|0\rangle + \alpha|1\rangle$. Moreover, Bob has no way of knowing whether an error occured or not.

To magnify the chances of successful communication, Alice and Bob can employ a code

very similar to the classical repetition code.

5.1.1 Encoding

In the quantum version of the repetition code, Alice takes the unencoded qubit and encodes it into the state of three qubits in a repetitive manner. This is best seen by first noting the transformation on the basis states,

$$|0\rangle \to \left|\bar{0}\right\rangle \stackrel{\text{def}}{=} |000\rangle\,, \tag{16}$$

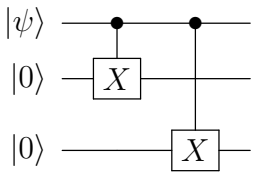
$$|1\rangle \to \left|\bar{1}\right\rangle \stackrel{\text{def}}{=} |111\rangle\,. \tag{17}$$

Then, we can realize that the qubit in state $|\psi\rangle$ is encoded as

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \to \left|\bar{\psi}\right\rangle = \alpha\,|000\rangle + \beta\,|111\rangle\,. \tag{18}$$

We will call the original qubit the unencoded logical qubit, the three qubits the physical qubits, and their combined state $\left|\bar{\psi}\right\rangle$ the encoded logical qubit.

The encoding transformation can be done using the quantum circuit,



which takes the unencoded logical qubit and two ancillas in state $|\psi\rangle\,|00\rangle$, to the encoded logical qubit in state $\left|\bar{\psi}\right\rangle$.

Exercise 6. Determine logical gate operations $\bar{X}$ and $\bar{Z}$ for the three-qubit repetition code. These are operations that act on the logical basis $\left\{\left|\bar{0}\right\rangle, \left|\bar{1}\right\rangle\right\}$, in the normal way, i.e.

$$\bar{X}\left|\bar{0}\right\rangle = \left|\bar{1}\right\rangle, \quad \bar{X}\left|\bar{1}\right\rangle = \left|\bar{0}\right\rangle, \tag{19}$$

$$\bar{Z}\left|\bar{0}\right\rangle = \left|\bar{0}\right\rangle, \quad \bar{Z}\left|\bar{1}\right\rangle = -\left|\bar{1}\right\rangle. \tag{20}$$

They can be constructed by some combination of operations on the three physical qubits. You will discover that there are possibly multiple ways of doing so.

5.1.2 Errors on the state

Alice sends the three physical qubits through the noisy channel. Each of them will have probability p of being flipped. If the second qubit is flipped, then Bob receives the state $X_2 \left|\bar{\psi}\right\rangle = \left|\tilde{\psi}\right\rangle = \alpha \left|010\right\rangle + \beta \left|101\right\rangle$. In total there, are eight possiblities for the error, with the probabilities similar to the classical case.

Error	Probability
I	$(1-p)^3$
X_1	$p(1-p)^2$
X_2	$p(1-p)^2$
X_3	$p(1-p)^2$
X_1X_2	$p^2(1-p)$
X_1X_3	$p^2(1-p)$
X_2X_3	$p^2(1-p)$
$X_1X_2X_3$	p^3

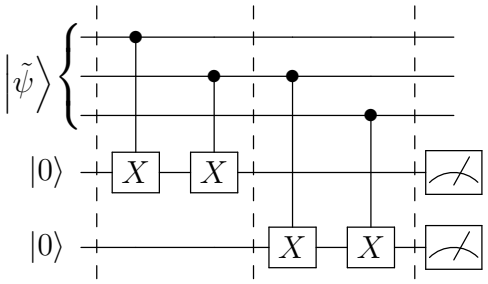
As in the classical case, the code can only correctly recover the state if only zero or one error occurs and fails otherwise. Simiarly, as in the classical case, if p is sufficiently small, then two or three errors are very unlikely to occur.

5.1.3 Decoding

Assuming that only errors from the set $\mathcal{E} = \{I, X_1, X_2, X_3\}$ occur, Bob has to (a) identify which of these errors has occurred and (b) then correct for it. His task is more difficult than in the classical case because he can't measure the received state as this would destroy the superposition, and he would gain very little information about α and β which is what Alice is really trying to transmit to Bob.

Instead the correct strategy for Bob to pursue is only do a partial measurement that only determines whether one of the qubits is different from the other two, which is analogous

to the majority voting algorithm in the classical case. This type of measurement does not probe the value of α or β and therefore the superposition is maintained. Specifically Bob compares the value of the first two qubits, and compares the value of the last two qubits. If there is a difference, then he will know which error occured. The following circuit accomplishes this task.



This circuit has three parts. In the first part, the values of the first and second qubits are added to the first ancilla qubit. This is done by CX gates, which if you recall, in the computational basis, add the value of the control qubit to the target qubit. If the first two qubits have the same value, then either neither CX gates will trigger or both will, and so the value of the ancilla will not change. This can be seen from the calculation,

$$CX_{14}CX_{24}(\alpha\,|000\rangle + \beta\,|111\rangle)\,|00\rangle \tag{21}$$

$$= \alpha CX_{14}CX_{24}\,|000\rangle\,|00\rangle + \beta CX_{14}CX_{24}\,|111\rangle\,|00\rangle\,, \tag{22}$$

$$= \alpha\,|000\rangle\,|00\rangle + \beta\,|111\rangle\,|0+1+1,0\rangle\,, \tag{23}$$

$$= \alpha\,|000\rangle\,|00\rangle + \beta\,|111\rangle\,|00\rangle\,. \tag{24}$$

Here, the CX gates don't trigger for the first term in the superposition, but do trigger for the second term, which we have made explicit. If the first two qubits are different, then only one of the CX gates will trigger. For instance,

$$CX_{14}CX_{24}(\alpha\,|010\rangle + \beta\,|101\rangle)\,|00\rangle \tag{25}$$

$$= \alpha CX_{14}CX_{24}\,|010\rangle\,|00\rangle + \beta CX_{14}CX_{24}\,|101\rangle\,|00\rangle\,, \tag{26}$$

$$= \alpha\,|010\rangle\,|0+1,0\rangle + \beta\,|101\rangle\,|0+1,0\rangle\,, \tag{27}$$

$$= \alpha \left|010\right\rangle \left|10\right\rangle + \beta \left|101\right\rangle \left|10\right\rangle . \tag{28}$$

Hence, the value of the ancilla has flipped.

The same story plays out for the second part of the circuit except for the second and third qubits. Subsequently, the two ancilla are measured which reveals a *syndrome* about which error has occured. This is summarized in the following table.

Syndrome	Inferred error
00	I
01	X_3
10	X_1
11	X_2

We say inferred error, because this is the error Bob assumes, but there is small chance of a two or three qubit error, so there is difference between the actual error and the inferred error.

Once, Bob has inferred an error, he can fix the corrupted codeword by applying the inverse of the error. In this case, the error operators are self-inverse, so he just has to apply them again to fix the corrupted codeword. Finally, Bob can undo the encoding operation to recover the one qubit state Alice meant to send. Since, the encoding operation was self-inverse, Bob just has to pass $\left|\bar{\psi}\right\rangle$ through it to recover $\left|\psi\right\rangle$.

5.1.4 A note on probability of success

We don't have to repeat our analysis of the probability of success as it is exactly the same as in the classical case.

5.1.5 A linear algebraic analysis of the quantum repetition code

One way of understanding why the code works is by understanding the vector spaces we have dealt with above. This formalism will be essential when we construct the formal theory of quantum error-correction later.

First, there was the 2-dimensional Hilbert space of the unencoded logical qubit. This was mapped to a 2-dimensional subspace of the 2^3-dimensional Hilbert space of the three physical qubits, the latter of which is called the *codespace*, and the former the *quantum code*. Errors move the encoded state into other 2-dimensional subspaces of the codespace. For instance, the quantum code had a basis $\{|000\rangle, |111\rangle\}$. The error X_2 moved the state to a subspace with basis $\{|010\rangle, |101\rangle\}$. We can tabulate all these movements as follows.

Error	Subspace basis
I	$\{\|000\rangle, \|111\rangle\}$
X_1	$\{\|100\rangle, \|011\rangle\}$
X_2	$\{\|010\rangle, \|101\rangle\}$
X_3	$\{\|001\rangle, \|110\rangle\}$

From this analysis we can see why the code works to correct bit-flip errors. Each error doesn't fundamentally destroy the state, it only moves it to a different subspace. The process of error-detection is to determine which subspace we are in, which does not distort the actual state. The process of error-correction moves us back to the original quantum code subspace. We will use this notion of subspaces again when we formally define quantum error-correction.

Exercise 7. Recall that a vector subspace is characterized by a projector $\Pi = \sum_i |\psi_i\rangle\langle\psi_i|$, where $\{|\psi_i\rangle\}_i$ is a basis for the subspace. Show that $\Pi^2 = \Pi$.

Exercise 8. Determine the projector onto the code for the three-qubit code.

Exercise 9. Consider the 5-qubit repetition code, in which the coding operates in a similar fashion as the 3-qubit one we have discussed. Can this larger code possibly correct errors in which two qubits are flipped? Mak a table of possible errors and the subspaces which they take the code to.

Exercise 10. The three-qubit repetition code encodes one logical qubit into the state of three physical qubits. Suppose, we wanted to encode two qubits, in distinct states $|\psi\rangle$ and $|\phi\rangle$. How woud we go about it?

5.1.6 Correcting X rotation errors

The code we have created can correct errors besides simple X errors. The unitary,

$$R_x^{(i)}(\theta) = \cos\theta I - i \sin\theta X^{(i)}, \tag{29}$$

is the rotation about the X axis on the Bloch sphere for the i-th qubit. Suppose, the encoded state $\left|\bar{\psi}\right\rangle$ is effected by this unitary during its passage through the communication channel between Alice and Bob. We now show that the error detection and correction procedures we have described above deal with this error, and Bob has to do nothing extra to correct for such an error.

To see this we calculate the corrupted codeword state,

$$\left|\tilde{\psi}\right\rangle = R_x^{(i)}(\theta)\left|\bar{\psi}\right\rangle = \cos\theta\left|\bar{\psi}\right\rangle - i\sin\theta X^{(i)}\left|\bar{\psi}\right\rangle. \tag{30}$$

When, we run this state through the error-detection circuit (in which we append two ancillas to the state), we obtain before the measurement step, the state

$$(\cos\theta\left|\bar{\psi}\right\rangle - i\sin\theta X^{(i)}\left|\bar{\psi}\right\rangle)\left|00\right\rangle \rightarrow \cos\theta\left|\bar{\psi}\right\rangle\left|00\right\rangle - i\sin\theta X^{(i)}\left|\bar{\psi}\right\rangle\left|s\right\rangle, \tag{31}$$

where $\left|s\right\rangle$ is the syndrome associated with $X^{(i)}$. When we measure the ancilla, either the measurement result is 00 (with probability $|\cos\theta|^2$) and the corrupted codeword state collapses to $\left|\bar{\psi}\right\rangle$, or the measurement result is s (with probability $|\sin\theta|^2$) and the corrupted codeword state collapses to $X^{(i)}\left|\bar{\psi}\right\rangle$. In either case, the subsequent error-correction procedure will correct the state.

What we see here is that linearity of quantum mechanics and the collapse mechanism on measurement allows us to correct errors which are a linear combination of errors in E. This is an incredibly powerful result, and we will prove it more generally later on.

5.1.7 Phase-flip errors

Quantum bit-flip errors are only one kind of errors. quantum systems in interacting with the environment can suffer from other types of error. One such possibility is phase-flip errors, i.e. where the qubits can be acted upon by the Z operator. This code fails to identify phase-flip errors and therefore cannot correct them either.

To see this note that the action of, say, Z_1 on the logical state is

$$Z_1 \left|\bar{\psi}\right\rangle = Z_1(\alpha \left|000\right\rangle + \beta \left|111\right\rangle) = \alpha \left|000\right\rangle - \beta \left|111\right\rangle . \tag{32}$$

This state is identical to a different state Alice might have sent, so there is no way that Bob can detemrine that an error has occured. In the subspace picture discussed above single-qubit Z errors keep the state within the code. This makes it impossible, both here and generally, for the error to be detected. A well-designed code ensures that every possible error moves the state out of the code and into some other subspace of the codespace.

5.2 Quantum repetition code for phase-flips

The code presented above could only correct for X error. Now, we present a code that can only correct for Z errors. In the next section, we will combine these codes together to correct for both X and Z errors and more.

Alice, again wants to send some one-qubit state $\left|\psi\right\rangle$ to Bob, but via a channel that applies the Z operator to qubits with probability p. This results in the phase flip,

$$Z \left|\psi\right\rangle = Z(\alpha \left|0\right\rangle + \beta 1) = \alpha \left|0\right\rangle - \beta \left|1\right\rangle \stackrel{\text{def}}{=} \left|\tilde{\psi}\right\rangle . \tag{33}$$

As before, Bob has no way of knowing if any errors occured.

5.2.1 Encoding

To protect against such type of errors, we will employ a repetition code, but with a different basis. Now,

$$|0\rangle \to \left|\bar{0}\right\rangle \stackrel{\text{def}}{=} |+++\rangle\,, \tag{34}$$

$$|1\rangle \to \left|\bar{1}\right\rangle \stackrel{\text{def}}{=} |---\rangle\,. \tag{35}$$

For this code basis, qubit in state $|\psi\rangle$ is encoded as

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \to \left|\bar{\psi}\right\rangle = \alpha\,|+++\rangle + \beta\,|---\rangle\,. \tag{36}$$

This is done via the circuit

$$X$$

5.2.2 Errors

The possible errors on the encoded state are given as follows.

Error	Probability
I	$(1-p)^3$
Z_1	$p(1-p)^2$
Z_2	$p(1-p)^2$
Z_3	$p(1-p)^2$
Z_1Z_2	$p^2(1-p)$
Z_1Z_3	$p^2(1-p)$
Z_2Z_3	$p^2(1-p)$
$Z_1Z_2Z_3$	p^3

As before, we will assume that p is small, so we will only attempt to correct the errors $\{I, Z_1, Z_2, Z_3\}$. The effect of the Z errors is to flip the state between plus and minus. So,

for instance,

$$Z_1 \left|\bar{\psi}\right\rangle = \alpha Z_1 \left|+++\right\rangle + \beta Z_1 \left|---\right\rangle = \alpha \left|-++\right\rangle + \beta \left|+--\right\rangle . \tag{37}$$

5.2.3 Decoding

The error detection strategy is the same as before. Bob employs a circuit that compares the value of two pairs of qubits, but in the plus/minus basis. This is accomplished using the circuit

$$X$$

The possible syndromes are as follows.

Syndrome	Inferred error
00	I
01	Z_1
10	Z_2
11	Z_3

In this case, if Bob infers that errors Z_i occurs, than he applies Z_i to the corrupted codeword to fix it.

5.2.4 Bit-flip errors

We won't belabor the point that this code cannot fix bit-flip errors.

5.3 Concatenation and Shor's code

It is now time to construct a code that correct more than one type of error. This will also give us the opportunity to show off a code construction technique that will be very useful in our discussion of fault-tolerant quantum computation. This technique is called code concatenation.

In code concatenation, we first encode the logical qubit using a particular code into n

physical qubits. Then, we encode each of then n physical qubits individually using the same or another code. To illustrate this, recall that, in the phase-flip code the logical basis states are encoded as

$$|0\rangle \to \left|\bar{0}\right\rangle \stackrel{\text{def}}{=} |+++\rangle\,, \tag{38}$$

$$|1\rangle \to \left|\bar{1}\right\rangle \stackrel{\text{def}}{=} |---\rangle\,. \tag{39}$$

Now, we are going to take each of the three physical qubits and encode them using the bit-flip repetition code. Recall that in the bit-flip repetition code, the plus and minus states encode as

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \to |\bar{+}\rangle \stackrel{\text{def}}{=} \frac{|000\rangle + |111\rangle}{\sqrt{2}}, \tag{40}$$

$$|-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \to |\bar{-}\rangle \stackrel{\text{def}}{=} \frac{|000\rangle - |111\rangle}{\sqrt{2}}. \tag{41}$$

When we apply this encoding to each of the three qubits in the phase-flip encoding, we obtain

$$|+++\rangle \to \left|\bar{\bar{0}}\right\rangle \stackrel{\text{def}}{=} \frac{(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)}{2\sqrt{2}}, \tag{42}$$

$$|---\rangle \to \left|\bar{\bar{1}}\right\rangle \stackrel{\text{def}}{=} \frac{(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)}{2\sqrt{2}}. \tag{43}$$

Visually, we can draw a tree diagram to show how one qubit is first encoded into three via the phase-flip code and then each of the three qubits is encoded into three more via the bit-flip code.

INSERT DIAGRAM

Here we refer to the phase-flip code as the *outer* code (level 1) and the bit-flip code as the *inner* code (level 2).

Exercise 11. Construct a nine-qubit code with the bit-flip code as the outer code and the phase-flip code as the inner code.

5.3.1 Encoding

This combined action that maps one logical qubit to the state of nine qubits as

$$|0\rangle \to \left|\bar{\bar{0}}\right\rangle = \frac{(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)}{2\sqrt{2}}, \tag{44}$$

$$|1\rangle \to \left|\bar{\bar{1}}\right\rangle = \frac{(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)}{2\sqrt{2}}, \tag{45}$$

defines the encoding for the Shor code.

Exercise 12. Construct a circuit that does the encoding for the Shor code.

5.3.2 Errors and decoding

We will now show that this code can correct every possible single-qubit error. Let's first show that it can correct bit-flip and phase-flip errors on any of the nine physical qubits.

Suppose a single-qubit X error occurs, on say the fourth qubit. How can one detect that this occur occured? It's not too hard to see that if we apply the error detection routine for the bit-flip code to each of the three blocks of three qubits, it will identify any qubit which has experienced a bit-flip error. In this example, comparing the value of the fourth qubit with the fifth, and the fifth with the sixth will show that the fourth qubit has a different value. The error can be fixed as before.

The case for a single-qubit Z errors is slightly more difficult to see. Note, for instance, that if a Z error occurs on any one of the first three qubits, the sign in the first block will change. For $i = 1, 2, 3$,

$$Z_i \left|\bar{\bar{0}}\right\rangle = \frac{(|000\rangle - |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)}{2\sqrt{2}}, \tag{46}$$

$$Z_i \left|\bar{\bar{1}}\right\rangle = \frac{(|000\rangle + |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)}{2\sqrt{2}}. \tag{47}$$

A different way of seeing it is at the outer code level, where the encoding is

$$\left|\bar{\bar{0}}\right\rangle = |\bar{+}\rangle\,|\bar{+}\rangle\,|\bar{+}\rangle\,, \tag{48}$$

$$\left|\bar{\bar{1}}\right\rangle = \left|\bar{-}\right\rangle \left|\bar{-}\right\rangle \left|\bar{-}\right\rangle . \tag{49}$$

Then the action of Z_i for $i = 1, 2, 3$ is

$$Z_i \left|\bar{\bar{0}}\right\rangle = \left|\bar{-}\right\rangle \left|\bar{+}\right\rangle \left|\bar{+}\right\rangle , \tag{50}$$

$$Z_i \left|\bar{\bar{1}}\right\rangle = \left|\bar{+}\right\rangle \left|\bar{-}\right\rangle \left|\bar{-}\right\rangle . \tag{51}$$

At this level, it is quite easy to see the error-detecting strategy. Apply the phase-flip code error-detection strategy to the three encoded qubits (at the outer level). We will discuss how one can do this later.

Right now, we want to show that the Shor code can detect and correct errors beyond just X and Z errors. One such error is the Y error, which is just $Y = iZX$, i.e. a combined bit-flip and phase-flip error. This error is detected at both the X detection stage, and the Z detection stage, and corrected at both as well. Hence, we have shown up till now that the Shor code can correct all errors in the set,

$$\mathcal{E} = I \cup \{X_i\}_i \cup \{Y_i\}_i \cup \{Z_i\}_i , \quad i = 1, \ldots, 9. \tag{52}$$

We showed before that the bit-flip code could also correct for X rotation errors. By the same arguments of linearity and collapse, the Shor code can correct any unitary error which is of the form,

$$E_i = e_0 I + e_1 X_i + e_2 Y_i + e_3 Z_i, \quad \sum_j |e_j|^2 = 1. \tag{53}$$

Exercise 13. Apply E_i to $\left|\bar{\psi}\right\rangle$ and then show that the error-detection circuit will collapse the state to just one of the possible errors.

The Shor code can, in fact, correct more than just unitary errors. It can correct any error that impacts a single qubit, up to, just throwing the qubit away (a type of error known as erasure error). However, we will defer this discussion to the next chapter where we

build up more theory to be able to characterize any error.

Exercise 14. Suppose the state $|\bar{\psi}\rangle$ encoded by the Shor code undergoes the two-qubit error $Z_1 Z_2$. What is the impact of this error? This is a phenomena not seen in classical codes.

Exercise 15. The Shor can correct some (but not all) two-qubit errors as well. Characterize all these errors.

Is there any three-qubit error that the Shor code can handle?

5.3.3 Probability analysis

5.4 Threshold behavior

6 Formal development of quantum error-correction

6.1 Errors on qubits

The errors on classical bit strings were quite easy to describe. On qubits, we require a bit more structure.

6.1.1 Pauli group

The simplest operations on a single qubit are the Pauli operators X, Y, Z. On multiple qubits, tensor products of these operators along with the identity are useful to study. These operations form a group.

Definition 17. *The Pauli group, $\mathcal{P}_n$ is the set of all n-fold tensor products of the I, X, Y, Z operators with an overall phase of $\pm 1, \pm i$, and the group operation is ordinary operator multiplication.*

The Pauli group has some useful properties.

- The size of the group $|\mathcal{P}_n| = 4^{n+1}$.
- If $P \in \mathcal{P}_n$ then $P^2 = \pm I$.

- If $P \in \mathcal{P}_n$ then P has eigenvalues ± 1 or $\pm i$.
- For every pair of elements $P, Q \in \mathcal{P}_n$, either P and Q commute ($PQ = QP$) or anticommute ($PQ = -QP$).

Definition 18. *Let $M \in \mathcal{P}_n$ be $M = M_1 \otimes M_2 \otimes \cdots \otimes M_n$. The weight of M, $wt(M)$ is the number of M_i which are not identity.*

Much of what we will do will involve the Pauli group.

6.1.2 Quantum operations

We will start with density matrices and CPTP maps.

Definition 19. *Given the Hilbert space $\mathcal{H}$, let $L(\mathcal{H})$ be the set of automorphisms on this space. Then a density operator is such a linear operator $\rho \in L(\mathcal{H})$, such that ρ is*

- *Hermitian:* $\rho = \rho^\dagger$,
- *Normalized:* $\mathrm{Tr}(\rho) = 1$,
- *Positive semidefinite* $\langle\psi| \rho |\psi\rangle \geq 0$ *(written* $\rho \geq 0$*).*

A density operator is the most complete description of a quantum system. The most general evolution of density operators, including unitary operations, transient interactions with other systems and measurements, is described by quantum operations, more formally referred to as Completely-Positive Trace-Preserving (CPTP) maps.

Definition 20. *A quantum operation is a CPTP map $\mathcal{S} : L(\mathcal{H}) \to L(\mathcal{H})$ that maps density operators to density operators such that $\mathcal{S}$ is*

- *Positive: if $\rho \geq 0$, then $\mathcal{S}(\rho) \geq 0$,*
- *Completely-positive: if $\sigma \in L(\mathcal{H}^{AB})$ such that $\sigma \geq 0$ and $\mathrm{Tr}_B(\sigma) = \rho$, then $(I \otimes S)(\sigma) \geq 0$.*
- *Trace-preserving:* $\mathrm{Tr}\big(S(\rho)\big) = \mathrm{Tr}(\rho)$.

Any such operation has a representation, called the Kraus representation.

Definition 21. *For any quantum operation $\mathcal{S}$, there exist a non-unique set of operators $\{A_k\}_k$, where $A_k \in L(\mathcal{H})$, such that*

$$S(\rho) = \sum_k A_k \rho A_k^\dagger, \tag{54}$$

and $\sum_k A_k^\dagger A_k = I$. This is called the Kraus representation.

Example 2. The dephasing channel

Example 3. The depolarizing channel

6.1.3 The noise channel model

We are now ready to define the noise that any qubits sent through the channel experience. Suppose, Alice sends n-qubits to Bob via the channel. Then, the noise model is the same single-qubit channel S applied to each qubit. So, to describe the noise the n-qubit quantum operation is

$$\mathcal{N} = \mathcal{S}^{\otimes n}. \tag{55}$$

Given the description of noise model an the n-qubit operator $\mathcal{N}$, we can identify a Krauss decomposition

$$\mathcal{N}(\rho) = \sum_i A_i \rho A_i^\dagger, \tag{56}$$

where there are some A_i with weight equal to n. However, recall that in the classical case if the probability of error p is small, then large errors are unlikely. There is a similar theorem for noise in the quantum systems.

Definition 22. *A CPTP map on n-qubits that can be written with Krauss operators each of which is a sum of operators with weight less than t is a t-qubit error.*

Theorem 3. *Let $\mathcal{S}$ be a single-qubit quantum channel close to the identity $||\mathcal{S} - I||_\diamond < \epsilon$ for some ϵ. Then there exists a t-qubit error channel $\tilde{\mathcal{E}}$ such that*

$$||\mathcal{S}^{\otimes n} - \tilde{\mathcal{E}}||_\diamond = \mathcal{O}\left(\binom{n}{t}\epsilon^t\right). \tag{57}$$

In simple language the above theorem states that if each of the n-qubits has only a very small chance of incurring an error, then one is very likely to observe errors on not very many qubits.

Suppose now that we have a t-qubit error channel $\mathcal{E}$, with a Krauss representation given by the operators $\{E_i\}_i$. From now on, we will refer to the E_i as errors, and $\{E_i\}_i$ as the set of errors. For instance, if S is the dephasing channel, and $n = 2$, then

$$\mathcal{E} = \{I, Z_1, Z_2, Z_1 Z_2\}\,, \tag{58}$$

where the I is included as the no-error possibility.

Our goal is to find quantum codes that are able to correct a given set of errors. Though often the research process proceeds in the opposite direction, and first a quantum code is defined and then its set of correctable errors is determined. We are now in the position to define a quantum code.

6.2 Defining a quantum code

We proceed as before by defining the message space, the codespace and finally the code. The quantum message space consists of states of qubits.

Definition 23. *For a quantum code, the message space is the state space of k-qubits, i.e. the 2^k-dimensional Hilbert space $\mathcal{H}_m$. Any $|\psi\rangle \in \mathcal{H}^m$ is a valid message.*

The quantum codespace is constructed from the state of n-qubits.

Definition 24. *For a quantum code, the codespace is the state space of n-qubits, i.e. the 2^n-dimensional Hilbert space $\mathcal{H}^c$.*

Finally, we can define the code.

Definition 25. *A qubit quantum code $\mathcal{C}$ is a 2^k-dimensional subspace of the codespace $\mathcal{H}^c$. Such a code is labelled as $[[n, k]]$.*

Elements of the code are codewords, and if we choses a basis for the code, then the basis

elements are also called basis codewords.

6.3 Quantum error-correction codes

Given a quantum code, it is not clear if it is useful. The above definition only talks about how to encode a message into the codespace. It does not speak at all about which errors can be detected or corrected with such an encoding. Why is this a problem, when this was not a significant problem for classical codes. There are three critical issues which make the construction of a quantum error-correction codes a more difficult task than their classical counterparts.

- The no-cloning theorem forbids us from copying quantum states. This limits the types of operations we can perform during the decoding process.
- Measuring quantum states in superposition destroys the superposition, so we can't simply measure the received blocks to determine what state they are in.
- Classical codes only have to deal with bit-flip errors, but quantum codes will have to deal with any sort of unitary or non-unitary noisy interactions. This includes phase errors, and it includes continuous rotations etc.

Fortunately, we will discover that all of these challenges can be surmounted. To build the theory to do so we start by explicitly defining what conditions make a quantum code an error-correcting one.

An encoding map is a transformation from the message space to the code, the subspace of the codespace, formally defined as follows.

Definition 26. *Given a quantum code $\mathcal{C}$, an encoding map or encoder is a unitary $U : \mathcal{H}^m \to \mathcal{H}^c$ that maps the message space to the code.*

Given any message $|\psi\rangle \in \mathcal{H}^m$, the state U
$ket\psi$ is an element of the code C. The encoding map is akin to the generator matrix for classical codes.

What we require is that every code C be paired with a set of errors $\mathcal{E}$ that it can correct,

and that there exist a process to actually do so.

Definition 27. *A quantum error-correction code $(C, \mathcal{E})$ is a quantum code along with a set of errors $\mathcal{E}$, such that given an encoding map U associated with the code, there exists a quantum channel $\mathcal{D}$, called the decoder, such that for all $E \in \mathcal{E}$ and for all $|\psi\rangle \in \mathcal{H}^m$,*

$$\mathcal{D}\left(EU\left|\psi\right\rangle\!\left\langle\psi\right|U^\dagger E^\dagger\right) = c(E, |\psi\rangle)\left|\psi\right\rangle\!\left\langle\psi\right|, \tag{59}$$

where $c(E, |\psi\rangle)$ is a constant.

This definition postulates that the existence of a decoder is necessary for a code to be an error-correcting one. When can such a decoder exist? Whenever, there is enough information extractable from every corrupted codeword so as to reverse the effects of the error. Let's build some basic results that will lead us to some precise mathematical conditions.

- First, note that given any error E_a and any state $|\psi\rangle \in \mathcal{C}$ the corrupted codeword $\left|\tilde{\psi}\right\rangle = E_a\left|\psi\right\rangle$ must be completely outside the code. If it is not, what would happen? We could write $|\psi\rangle = |\psi_c\rangle + |\psi_\perp\rangle$ such that $E_a\left|\psi_\perp\right\rangle \in \mathcal{C}$ and $E_a\left|\psi_\perp\right\rangle \in \mathcal{C}^\perp$. Now, $|\psi\rangle$ is in the code, then so is $|\psi_c\rangle$ (up to a normalization) and it is some other valid codeword. Hence, the effect of the error E_a on $|\psi_c\rangle$ is to keep it inside the code, i.e. map it onto a different codeword. This is clearly a bad situation because Bob would just assume that no error has occured and $E_a\left|\psi_c\right\rangle$ is the codeword Alice meant to send. Hence, by contradiction, we infer that our code must be designed so that all errors move every codeword out of the code.

- Next, suppose we have an orthogonal basis $\left\{|\psi_i\rangle\right\}_i$ for $\mathcal{C}$. Given any error E_a, what do we require of $E_a\left|\psi_i\right\rangle$ and $E_a\left|\psi_j\right\rangle$ for $i \neq j$, so the decoder can function. We will need these two corrupted codewords to remain orthogonal, so that the decoder has enough information to correctly decode. To see this, let's assume to the contrary that the corrupted states are not orthogonal, i.e. $(\langle\psi_i| E_a^\dagger)(E_a\left|\psi_j\right\rangle) \neq 0$. To tease

out the non-orthogonal part of this inner product, let's write

$$\left|\psi_j\right\rangle = \left|\psi_{\parallel}\right\rangle + \left|\psi_{\perp}\right\rangle, \tag{60}$$

such that $E_a \left|\psi_{\parallel}\right\rangle$ is parallel to $E_a \left|\psi_i\right\rangle$, while $E_a \left|\psi_{\perp}\right\rangle$ is perpendicular to $E_a \left|\psi_i\right\rangle$. As in the argument above, $\left|\psi_{\parallel}\right\rangle$ must also be a valid codeword (because it is part of $\left|\psi_j\right\rangle$). And because $\left|\psi_i\right\rangle$ and $\left|\psi_j\right\rangle$ are perpendicular, $\left|\psi_{\parallel}\right\rangle$ and $\left|\psi_i\right\rangle$ are distinct codewords. Hence, we discover that two distinct codewords, $\left|\psi_{\parallel}\right\rangle$ and $\left|\psi_i\right\rangle$, when acted upon by error E_a result in the same corrupted codeword. This is again a bad situation, because the decoder will be unable to decide how to fix the error on the corrupted codeword. Hence, in order to have an error-correcting code that works, we need each error to map orthogonal basis vectors of C to orthgonal states,

$$(\left\langle\psi_i\right| E_a^{\dagger})(E_a \left|\psi_j\right\rangle) = 0, \quad i \neq j. \tag{61}$$

- Let's now make an even stronger condition on our code. Consider the corrupted codewords $E_a \left|\psi_i\right\rangle$ and $E_b \left|\psi_j\right\rangle$, i.e. distinct errors on two distinct basis states. Do these need to be orthogonal? By exactly the same argument as above, yes. In words, if these two corrupted codewords are not orthogonal, then there is a part of $\left|\psi_j\right\rangle$, called $\left|\psi_{\parallel}\right\rangle$, that under the action of E_b becomes parallel to $E_a \left|\psi_i\right\rangle$. As above, $\left|\psi_{\parallel}\right\rangle$ is a valid codeword. If the Bob receives $E_a \left|\psi_i\right\rangle = E_b \left|\psi_{\parallel}\right\rangle$, then his decoder will be unable to tell if Alice sent $\left|\psi_i\right\rangle$ which was distorted by E_a or $\left|\psi_{\parallel}\right\rangle$ which was distorted by E_b. Hence, we can conclude that for a valid error-correcting code,

$$(\left\langle\psi_i\right| E_a^{\dagger})(E_b \left|\psi_j\right\rangle) = 0, \quad i \neq j. \tag{62}$$

- What about the case when $i = j$ with distinct errors? Meaning, is there any requirement on $E_a \left|\psi_i\right\rangle$ and $E_b \left|psi_i\right\rangle$? First of all, let's make it clear that these don't need to be orthogonal - though they can be. We saw the example of the phase-flip code where the action of Z on different qubits yielded the same corrupted codeword.

But this was not a problem, because in each case, the correction operation was the same. So up till now, our condition is

$$\langle\psi_i| E_a^\dagger E_b |\psi_j\rangle = \delta_{ij} A, \tag{63}$$

where the δ_{ij} ensures that the inner product is zero if we are dealing with two different basis states, and possibly non-zero if they are the same. This last part is determined by the unknown constant A. However, we can recognize that if $E_a |\psi_i\rangle$ and $E_b |\psi_i\rangle$ are indeed orthogonal for every pair of errors E_a and E_b, then our code will be an error-correcting one. We assume,

$$\langle\psi_i| E_a^\dagger E_b |\psi_j\rangle = \delta_{ij}\delta_{ab} A. \tag{64}$$

Now, recall that when we went from the noise channel to the set $\{E_a\}_a$ we choose from the one of the infinite such sets. Let's transform to a different set $\{F_c\}$ where

$$F_c = \sum_a \alpha_{ca} E_a. \tag{65}$$

Now, let's evalute the inner product

$$\langle\psi_i| F_c^\dagger F_d |\psi_j\rangle = \sum_{ab} \alpha_{ca}^* \alpha_{db} \langle\psi_i| E_a^\dagger E_b |\psi_j\rangle, \tag{66}$$

$$= \sum_{ab} \alpha_{ca}^* \alpha_{db} \delta_{ij}\delta_{ab} A, \tag{67}$$

$$= \left(\sum_a \alpha_{ca}^* \alpha_{da} A\right) \delta_{ij}, \tag{68}$$

$$= A'_{cd}\delta_{ij}. \tag{69}$$

What this calculation tells us that, unless we pick a very clever basis for the errors, the states $E_a |\psi_i\rangle$ and $E_b |\psi_i\rangle$ will not be orthogonal for every i and every pair of E_a and E_b.

In the above, we have taken care of every possible case of possible confusion of the

detector, and resolved them. We can now use (69) to write down the following theorem.

Theorem 4. *A code $\mathcal{C}$ is a quantum error-correcting code for the set of errors $\{E_a\}$ iff*

$$\langle\psi_i| E_a^\dagger E_b \left|\psi_j\right\rangle = A_{ab}\delta_{ij}, \tag{70}$$

for the orthogonal basis $\left\{|\psi_i\rangle\right\}_i$.

Algorithm 1 Method to verify that a given encoding is a valid quantum error correction code

Input: An encoding of the logical states into codewords.
Input: A set of errors
Check all possibilities in (70).

7 Stabilizer codes

Given the pauli group $\mathcal{P}_n$, the stabilizer code is defined as an abelian subgroup of $\mathcal{P}_n$, which does not include $-I$.

Then the space

$$\mathcal{H}^c = \left\{|\psi\rangle : M|\psi\rangle = |\psi\rangle \, \forall M \in S\right\}. \tag{71}$$

Stabilizer codes are usually specified by their generators $\{S_i\}$.

Fact 1. If there are r generators, then $k = n - r$.

Algorithm 2 Check that a given set of Paulis form a valid stabilizer code

Check that they all commute, and that $-I$ is not part of the group.

How to find the logical operators of a stabilizer code

The logical operators are the elements of $N(S) - S$. There are various methods of findings these efficiently. One is provided by Gottesman. Another by Wilde [https://arxiv.org/pdf/0903.5256.pdf

How to find the codewords of a stabilizer code

The naive way is as follows. The logical zero codeword can be

$$\left|0^k\right\rangle_L = \prod_{i=1}^{r}(I + S_i)\left|0^n\right\rangle . \tag{72}$$

If you have guessed the logical X_i operators, then you can simply apply them to the state above and find all the other codewords.

If not, apply the operator $\prod_{i=1}^{r}(I+S_i)$ (this is just the sum of all elements of S), to other states in $\mathcal{H}_c$. Some will be projected to zero, while others will yield the correct state. This process is easy for single-qubit codes and tedious for more qubits.

7.1 CSS codes

7.2 Examples of simple CSS codes

7.3 Surface codes

8 Fault-tolerant quantum computation

9 Decoding strategies