Chapter 2

Introduction to quantum error correction

Dave Bacon

In Chapter 1 we saw that open quantum systems could interact with an environment and that this coupling could turn pure states into mixed states. This process will detrimentally impact any quantum computation, because it can lessen or destroy the interference effects that are vital to distinguishing a quantum computer from a classical computer. The problem of overcoming this effect is called the *decoherence problem*. Historically, the problem of overcoming decoherence was thought to be a major obstacle towards building a quantum computer. However, it was discovered that, under suitable conditions, the decoherence problem could be overcome. The main idea behind how this can be achieved is through the theory of quantum error correction (QEC). In this chapter we give an introduction into the way in which the decoherence problem can be overcome via the method of QEC. It is important to note that the scope of the introduction is not comprehensive, and focuses only on the basics of QEC without reference to the notion of fault-tolerant quantum computation, which is covered in Chapter 5. Quantum error correction should be thought of as a (major) tool in this larger theory of fault-tolerant quantum computing.

2.1 Error correction

When we are stumped about what to do in the quantum world, it is often useful to look to the classical world to see if there is an equivalent problem, and if so, how that problem is dealt with. Thinking this way, one realizes that the decoherence problem has an analogy in the classical world: classical noise. While this statement is not at all obvious, we will see that actually the analogy holds by thinking about decoherence and classical noise from a particular perspective.

Consider the following situation. We have a bit that we send from Seattle to New York over a phone line. This phone line, however, is noisy. With probability $1-p$ nothing happens to our bit, but with probability p the bit is flipped: 0 is turned into 1 and 1 is turned into 0. This setup is an example of classical communication over a communication *channel*. This particular channel is called a *binary symmetric channel*. We call the operation of flipping the bit an *error*. Later the

Quantum Error Correction, ed. Daniel A. Lidar and Todd A. Brun. Published by Cambridge University Press.
© Cambridge University Press 2013.

term *error* will take on a more specific meaning in the theory of fault tolerance. If we use this channel once, then the probability that we will receive a wrong bit is p. If p is sufficiently close to $\frac{1}{2}$ this poses quite a problem as we want the information encoded in our bit to arrive safely at its destination. Thus we are naturally led to the question: is there a way to use this channel in such a way that we can decrease this probability? The answer to this question is yes, and the way to do this is actually rather simple.

The solution is to just use the channel multiple times. In other words, we use *redundancy*. Thus, if we want to send a 0, we use the *encoding* $0 \to 000$ and $1 \to 111$ and send each of these bits through the channel. Of course there will still be errors on the information. Assuming that the channel's noise acts independently (note this assumption!), with probability $(1-p)^3$ no errors occur on the bits, with probability $3(1-p)^2p$ one error occurs on the bits, with probability $3(1-p)p^2$ two errors occur on the bits, and with probability p^3 three errors occur on the bits. Now assume that p is small for intuition's sake (we will calculate what small means in a second). Notice that the three probabilities we have listed above will then be in decreasing order. In particular, the probability of no or one error will be greater than there being two or three errors. But if a single error occurs on our bit, we can detect this and correct it. In particular if, on the other end of the channel, we *decode* the bit strings by $\{000, 001, 010, 100\} \to 0$ and $\{111, 110, 101, 011\} \to 1$, then in the case of no error or one bit-flip we will have correctly transmitted the bit. We can thus calculate the probability that this procedure – encoding, sending the bits individually through the channel, and decoding – fails. It is given by $3(1-p)p^2 + p^3 = 3p^2 - 2p^3$. Now if this is less than p, the failure probability with no encoding, we have decreased the probability of failing to transmit our bit. Indeed, this occurs when $3p^2 - 2p^3 \leq p$ and hence when $p < \frac{1}{2}$. Thus, if the probability of flipping our bit is less than $\frac{1}{2}$, then we will have decreased our failing (from p to $3p^2 - 2p^3$). This method of encoding information is known as a redundancy error-correcting code. The classical theory of error-correcting codes is devoted to expanding on this basic observation, that redundancy can be used to protect classical information. We will delve into some of the details of classical error-correcting codes in Section 2.7. The main take-home point at this juncture is that it is possible to use classical error correction to lessen the chance of errors destroying classical information.

2.1.1 Obstacles to quantum error correction

If classical error correction can be used to protect against noise in a noisy classical channel, a natural question to ask is whether a similar tactic can be used to protect quantum information. When we first encounter classical error correction and think about porting it over to the quantum world, there are some interesting reasons to believe that it will be impossible to make this transition. We list these here, since they are rather interesting (although claims that these were major blockades to discovering quantum error correcting are probably a bit exaggerated).

No cloning The no-cloning theorem [WZ82] states that there is no machine that can perform the operation $|\psi\rangle \to |\psi\rangle \otimes |\psi\rangle$ for all $|\psi\rangle$. Thus, a naive attempt to simply clone quantum information in the same way that we copy information in a redundancy code fails.

Measurement When we measure a quantum system, our description of the quantum system changes. Another way this is stated is that measurement disturbs the state of a quantum system. In error correction, we read out classical information in order to correctly recover our classical information. How do we perform measurements on quantum systems that do not destroy the quantum information we are trying to protect?

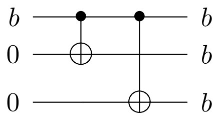
Quantum noise Quantum noise has a continuous set of parameters to describe it. So we might think that this will cause a problem, since in classical theory we could interpret the noise as probabilities of deterministic evolutions occurring, but in quantum theory we do not have such an interpretation (at least not yet.) Of course this feels like a little bit of a red herring, since classical noise also has continuous parameters (say the probabilities of the erring procedures) to describe it.

For these reasons we might expect that an equivalent to classical error correction in the quantum world does not exist. One of the surprising discoveries of the mid-nineties was that this is not true: protection of quantum information using QEC is possible.

2.2 From reversible classical error correction to simple quantum error correction

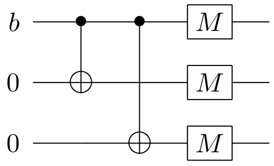
So where to begin in figuring out how to protect quantum information? Well, one place to begin is to try to understand how to perform the classical error correction we described above using classical reversible circuits, since reversible classical computation is the closest classical theory to quantum theory.

To this end let us work through using the classical three-bit redundancy code to protect classical information. The first part of our procedure is to encode our classical bit. Suppose that we represent our three bits by three wires. Then it is easy to check that an encoding procedure for taking a bit in the first wire to the encoded 000 and 111 configurations is



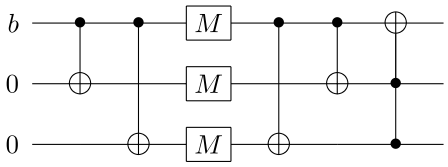
(2.1)

where the gates diagrammed are CNOT gates, which act in the computational basis as $CX|x\rangle|y\rangle = |x\rangle|y+x\rangle$, where the addition is done modulo 2, and where time flows from left to right. Next we send each of these bits through the bit-flip channel. Each of these bits is sent independently. We denote this by the gate M on these bits:



(2.2)

Now we need to describe the procedure for diagnosing an error and fixing this error. Consider what the two CNOTs do in our encoding circuit. They take $000 \to 000$, $001 \to 001$, $010 \to 010$, $011 \to 011$, $100 \to 111$, $101 \to 110$, $110 \to 101$, and $111 \to 100$. Notice that, except in the case where the last two bits are 11, after this procedure the first bit has been restored to its proper value, given that only zero or one bit-flip has occurred on our three bits. And when the last two bits are 11, then we need to flip the first bit to perform the proper correction. This implies that the decoding and fixing procedure can be done by the two CNOTs, followed by a Toffoli gate (which acts as $C^2(X)|x\rangle|y\rangle|z\rangle = |x\rangle|y\rangle|z+xy\rangle$). In other words,



(2.3)

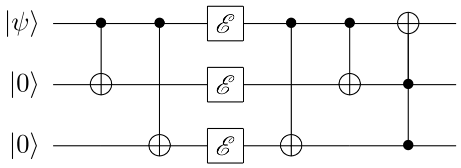
It is easy to check that if only one or no error occurs where the M gates are, then the output of the first bit will be b (and in the other cases the bit will be flipped). This is exactly the procedure we described in the previous section and we have done it using completely reversible circuit elements (except for the Ms, of course.)

2.2.1 *When in Rome do as the classical coders would do*

Now that we have a classical reversible circuit for our simple error-correcting procedure, we can see what happens when we use this circuit on quantum information instead of classical information. One thing we need to do is to choose the appropriate noise channel for our code (we will come back to more general noise eventually). A natural choice is the bit-flip channel that had the Kraus operator

$$E_0 = \sqrt{1-p}I, \quad E_1 = \sqrt{p}X. \tag{2.4}$$

The circuit we want to evaluate is now



(2.5)

where $|\psi\rangle$ is now an arbitrary quantum state $\alpha|0\rangle + \beta|1\rangle$, which we wish to protect. So what does this circuit do to our quantum data? Well, after the first two CNOTs, we can see that the state is given by

$$\alpha|000\rangle + \beta|111\rangle. \tag{2.6}$$

Notice that we have done something like redundancy here: but we have not copied the state, we have just "copied" it in the computational basis.

Next what happens? Recall that we can interpret $\mathscr{E}$ as a bit-flip error X happening with probability p and nothing happening with probability $1-p$. It is useful to use this interpretation to say that with probability $(1-p)^3$ no error occurred on all three qubits, with probability $(1-p)^2p$ a single error occurred on the first qubit, etc. So what happens if no error occurs on our system (the Kraus operator $I \otimes I \otimes I$ happens on our system)? Then we just need to run those two CNOTs on our state

$$(CX)_{13}(CX)_{12}(\alpha|000\rangle + \beta|111\rangle) = \alpha|000\rangle + \beta|100\rangle = (\alpha|0\rangle + \beta|1\rangle)|00\rangle. \tag{2.7}$$

The Toffoli then does nothing to this state and we see that our quantum information has survived. But this is not too surprising since no error occurred. What about when a single bit-flip error occurs? Let us say an error occurs on the second qubit. Then our state is $\alpha|010\rangle + \beta|101\rangle$. The effect of the CNOTs is then

$$(CX)_{13}(CX)_{12}(\alpha|010\rangle + \beta|101\rangle) = \alpha|010\rangle + \beta|110\rangle = (\alpha|0\rangle + \beta|1\rangle)|10\rangle. \tag{2.8}$$

Again, the Toffoli will do nothing to this state. And we see that our quantum information has survived its encounter with the bit-flip error! One can go through the other cases of a single bit-flip error. In the case where the bit-flip error is on the first qubit, the Toffoli is essential in correcting the error, but in the case where it is on the third qubit, then the Toffoli does nothing. But in all three cases the quantum information is restored!

One can go through and check what happens for the cases where two or three bit-flip errors occur. What one finds out is that in these cases the resulting state in the first qubit is $\beta|0\rangle + \alpha|1\rangle$. Thus, if we look at the effect of this full circuit, it will perform the evolution

$$\begin{aligned}\rho \otimes |00\rangle\langle 00| \rightarrow\ & (1-p)^3 \rho \otimes |00\rangle\langle 00| + (1-p)^2 p\rho \otimes |01\rangle\langle 01| \\ +\ & (1-p)^2 p\rho \otimes |10\rangle\langle 10| + (1-p)^2 p\rho \otimes |11\rangle\langle 11| \\ +\ & (1-p)p^2 X\rho X \otimes |01\rangle\langle 01| + (1-p)p^2 X\rho X \otimes |10\rangle\langle 10| \\ +\ & (1-p)p^2 X\rho X \otimes |11\rangle\langle 11| + p^3 X\rho X \otimes |00\rangle\langle 00|.\end{aligned} \tag{2.9}$$

Tracing over the second and third qubits, this amounts to the evolution

$$\rho \rightarrow [(1-p)^3 + 3p(1-p)^2]\rho + [3p^2(1-p) + p^3]X\rho X. \tag{2.10}$$

If we compare this with the evolution that would have occurred given no encoding,

$$\rho \rightarrow (1-p)\rho + pX\rho X, \tag{2.11}$$

we see that if $p < \frac{1}{2}$, then our encoding acts to preserve the state better than if we had not encoded the state.

Actually, how do we know that we have preserved the state better? What measure should we use to deduce this and why would this be a good measure? In particular, we might note that quantum errors will affect different states in different ways. The answer is given in terms of the fidelity discussed in Section 1.2.6.

2.2.2 Fidelity of classical error correction in the quantum model

What happens to the fidelity of the two cases we had above, one in which no error correction was performed and one in which error correction was performed? In the first case the fidelity, assuming we start in a pure state, is

$$F_1 = \left[\langle\psi|\left[(1-p)|\psi\rangle\langle\psi| + pX|\psi\rangle\langle\psi|X\right]|\psi\rangle\right]^{\frac{1}{2}} = \left[(1-p) + p|\langle\psi|X|\psi\rangle|^2\right]^{\frac{1}{2}}. \tag{2.12}$$

This is minimized (remember we want high fidelity) when $\langle\psi|X|\psi\rangle = 0$ and so

$$F_1 \geq \sqrt{1-p}\,. \tag{2.13}$$

Similarly, if we perform error correction using the three qubit scheme we obtain

$$\begin{aligned}F_3 &= \left[\langle\psi|\left[((1-p)^3 + 3p(1-p)^2)|\psi\rangle\langle\psi| + (3p^2(1-p) + p^3)X|\psi\rangle\langle\psi|X\right]|\psi\rangle\right]^{\frac{1}{2}} \\ &= \left[(1-p)^3 + 3p(1-p)^2 + (3p^2(1-p) + p^3)|\langle\psi|X|\psi\rangle|^2\right]^{\frac{1}{2}},\end{aligned} \tag{2.14}$$

which is again bounded by

$$F_3 \geq \sqrt{(1-p)^3 + 3p(1-p)^2}\,. \tag{2.15}$$

The fidelity is greater using error correction when $p < \frac{1}{2}$. So our naive analysis was not so much the dunce after all.

2.2.3 *Morals*

What lessons should we draw from our first success in QEC? One observation is that instead of *copying* the quantum information we *encoded* the quantum information into a *subspace*. In particular, we have encoded into the subspace spanned by $\{|000\rangle, |111\rangle\}$, i.e., we have encoded our quantum information as $\alpha|000\rangle + \beta|111\rangle$. This is our way of getting around the no-cloning theorem.

The second problem we brought up was measurement. Somehow we have made a measurement such that we could fix our quantum data (if needed) using the Toffoli gate. Let us examine what happens to our subspace basis elements, $|000\rangle$ and $|111\rangle$, under the errors that we could correct. Notice that they enact the evolution

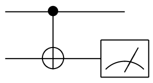
$$\begin{array}{ccc}
\begin{array}{c}|000\rangle\\|111\rangle\end{array} & \overbrace{\longrightarrow}^{I\otimes I\otimes I} & \begin{array}{c}|000\rangle\\|111\rangle\end{array}\\[1ex]
\begin{array}{c}|000\rangle\\|111\rangle\end{array} & \overbrace{\longrightarrow}^{X\otimes I\otimes I} & \begin{array}{c}|100\rangle\\|011\rangle\end{array}\\[1ex]
\begin{array}{c}|000\rangle\\|111\rangle\end{array} & \overbrace{\longrightarrow}^{I\otimes X\otimes I} & \begin{array}{c}|010\rangle\\|101\rangle\end{array}\\[1ex]
\begin{array}{c}|000\rangle\\|111\rangle\end{array} & \overbrace{\longrightarrow}^{I\otimes I\otimes X} & \begin{array}{c}|001\rangle\\|110\rangle\end{array}\,.
\end{array} \tag{2.16}$$

Now think about what this is doing. These error processes are mapping the subspace where we encoded the information into different *orthogonal* subspaces for each of the different errors. Further, when this map is performed, the orthogonality between the basis elements is not changed (i.e., $|000\rangle$ and $|111\rangle$ are orthogonal before and after the error occurs). Now this second fact is nice, because it means that the quantum information has not been *distorted* in an irreversible fashion. And the first fact is nice because, if we can measure which subspace our error has taken us to, then we will be able to fix the error by applying the appropriate operation to reverse the error operation. In particular, consider the operators $S_1 = Z \otimes Z \otimes I$ and $S_2 = Z \otimes I \otimes Z$. These operators square to identity and so have eigenvalues $+1$ and -1. In fact, we can see that these eigenvalues do not distinguish between states within a subspace, but do distinguish which of the four subspaces our state is in. That is to say, for example, that $|000\rangle$ and $|111\rangle$ have eigenvalues $+1$ for both S_1 and S_2. Further, the subspace that occurs if a single bit-flip occurs on our first qubit, $|100\rangle$ and $|011\rangle$, has eigenvalues -1 for S_1 and -1 for S_2. We can similarly calculate the other cases:

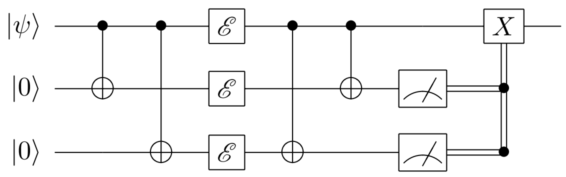
Basis states of subspace	S_1	S_2	Error
$\{\|000\rangle, \|111\rangle\}$	$+1$	$+1$	$I \otimes I \otimes I$
$\{\|100\rangle, \|011\rangle\}$	-1	-1	$X \otimes I \otimes I$
$\{\|010\rangle, \|101\rangle\}$	-1	$+1$	$I \otimes X \otimes I$
$\{\|001\rangle, \|110\rangle\}$	$+1$	-1	$I \otimes I \otimes X$

Thus we see that, if we could perform a measurement that projects onto the $+1$ and -1 eigenstates of S_1 and S_2, then we could use the results of this measurement to diagnose which subspace the error has taken us to and apply the appropriate X operator to recover the original subspace. So is it possible to measure S_1 and S_2? Well, we have already done it, but in a destructive way, in our circuit.

Consider the following circuit:


(2.17)

What does this circuit do? Well, if the input to this circuit is $\alpha|00\rangle + \beta|11\rangle$, then the measurement outcome will be $|0\rangle$ and if the circuit is $\alpha|01\rangle + \beta|10\rangle$, then the measurement outcome is $|1\rangle$. Associating $|0\rangle$ with $+1$ and $|1\rangle$ with -1, we thus see that this is equivalent to measuring the eigenvalue of the operator $Z \otimes Z$. Notice, however, that this is a destructive measurement, i.e., it does not leave the subspace intact after the measurement. In the circuit we have constructed above, then, the CNOTs after the errors have occurred would have measured the operators S_1 and S_2. This is enough to diagnose which error has occurred. Since this also does decoding of our encoded quantum information, only in the case where the error occurred on the first qubit do we need to do anything; and this is the case where the measurement outcomes are both $|1\rangle$ and so we use the Toffoli to correct this error. This suggests that a different way to implement this error-correcting circuit is to measure the second and third qubits. Since measurements commute through control gates turning them into classical control operations, we could thus have performed the following circuit:



(2.18)

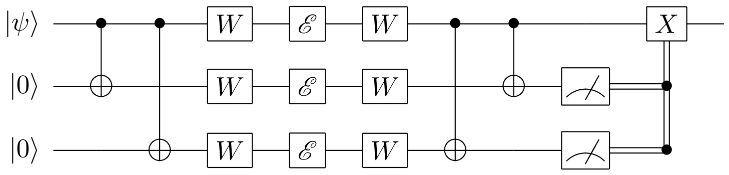
What do we learn from the above analysis? We learn that QEC avoids the fact that measuring disturbs a quantum system by performing measurements that project onto subspaces. These measurements do not disturb the information encoded into the subspaces since the measurements yield degenerate values outcomes for any state in this subspace. This technique, of performing measurements that do not fully project onto a basis, is essential to being able to perform quantum error correction.

2.2.4 Dealing with phase-flips

The third problem we brought up was the fact that quantum errors form a continuous set. For now, however, let us just move on to a different error model. In particular, let us consider, instead of a bit-flip model, a phase-flip model. In this model, the Kraus operators are given by

$$A_0 = \sqrt{1-p}I, \quad A_1 = \sqrt{p}Z. \tag{2.19}$$

The effect of Z on a quantum state is to change the phase between $|0\rangle$ and $|1\rangle$. So how are we going to correct this error? It changes a phase, not an amplitude! Well, we use the fact that phase changes are amplitude changes in a different basis. The basis where this occurs is the so-called plus minus basis: $|\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$. In particular, the gate that transforms this basis into the computational basis is the Hadamard gate, W; and note that $WZW^\dagger = X$. This suggests that just prior to sending our information through the quantum channel that causes phase errors and just after receiving the quantum information we should apply Hadamard gates. Hence we are led to the circuit:



(2.20)

Now, will this work? Certainly it will work, this is just a basis change and the fidelities will be identical to the bit-flip analysis. What does this circuit do? Instead of encoding into the subspace spanned by $|000\rangle$ and $|111\rangle$, this code encodes into a subspace spanned by $|{+}{+}{+}\rangle$ and $|{-}{-}{-}\rangle$, where $|\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$. Thus we see that, by expanding our notion of encoding beyond encoding into something simple like the repeated computational basis states, we can deal with a totally different type of error, one that does not really have a classical analogy in the computational basis.

We are just putting off the question of what happens when we have arbitrary errors, of course; but notice something here. The phase-flip error model is equivalent, via a change in Kraus operators, to the phase damping model that has Kraus operators:

$$B_0 = \begin{bmatrix} \sqrt{1-q} & 0 \\ 0 & \sqrt{1-q} \end{bmatrix}, \quad B_1 = \begin{bmatrix} \sqrt{q} & 0 \\ 0 & 0 \end{bmatrix}, \quad B_2 = \begin{bmatrix} 0 & 0 \\ 0 & \sqrt{q} \end{bmatrix}. \tag{2.21}$$

When $\frac{p}{2} = q$, these were the same superoperator. We can express the above Kraus operators as

$$B_0 = \sqrt{1-q}I, \quad B_1 = \frac{\sqrt{q}}{2}(I+Z), \quad B_2 = \frac{\sqrt{q}}{2}(I-Z). \tag{2.22}$$

Surely, since this is the same superoperator, our analysis of the error-correcting procedure will be identical and we will be able to increase the probability of successfully correcting this model given $q \leq \frac{1}{4}$. But the B_i operators are sums of I and Z errors. Thus, a code designed to correct single Z errors seems to be working on superoperators that have Kraus operators which are sums of identity and Z errors. Why is this so? Well, we have some encoded information $|\psi\rangle$. Then a Kraus operator that is the sum of terms which we can correct occurs (plus terms we cannot correct). Then we perform the measurement to distinguish which subspace our error has taken us to. But, at this point, the Kraus operators that are sums of errors get projected onto one of the error subspaces. Thus, in effect, the fact that the error is a sum of errors gets erased when we do this projection. We will return to this fact later, but this is an important point. While quantum errors may be a continuous set, the error-correcting procedure can, in effect, digitize the errors and deal with them as if they formed a discrete set.

2.2.5 The Shor code

So far we have dealt with two models of errors, bit-flip errors and phase-flip errors. We have also seen that if a code corrects an error then it will be able to correct Kraus operator errors that have a sum of these errors. Thus, we expect that if we can design a quantum error-correcting code (QECC) that can correct a single X, Y, or Z error, then this can correct an arbitrary error on a single qubit. Indeed, Peter Shor designed [S95] just such a code (Steane independently arrived at the idea of QECC in [S96c]). How does this code work? Well we have already seen that if we encode into the subspace spanned by $|000\rangle$ and $|111\rangle$, then we can correct a single bit-flip error. What do single phase-flip errors do to this code? Well, notice that $ZII|000\rangle = IZI|000\rangle = IIZ|000\rangle = |000\rangle$ but $ZII|111\rangle = IZI|111\rangle = IIZ|111\rangle = -|111\rangle$. Thus we see that, unlike bit-flip errors, single phase-flip errors on this code act to *distort* the information encoded into the subspace. But also notice that these single phase-flip errors act like phase-flip errors on the encoded basis $|000\rangle$, $|111\rangle$. But we have seen how to deal with phase-flip errors. Suppose we define the states

$$|p\rangle = \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle), \tag{2.23a}$$

$$|m\rangle = \frac{1}{\sqrt{2}}(|000\rangle - |111\rangle). \tag{2.23b}$$

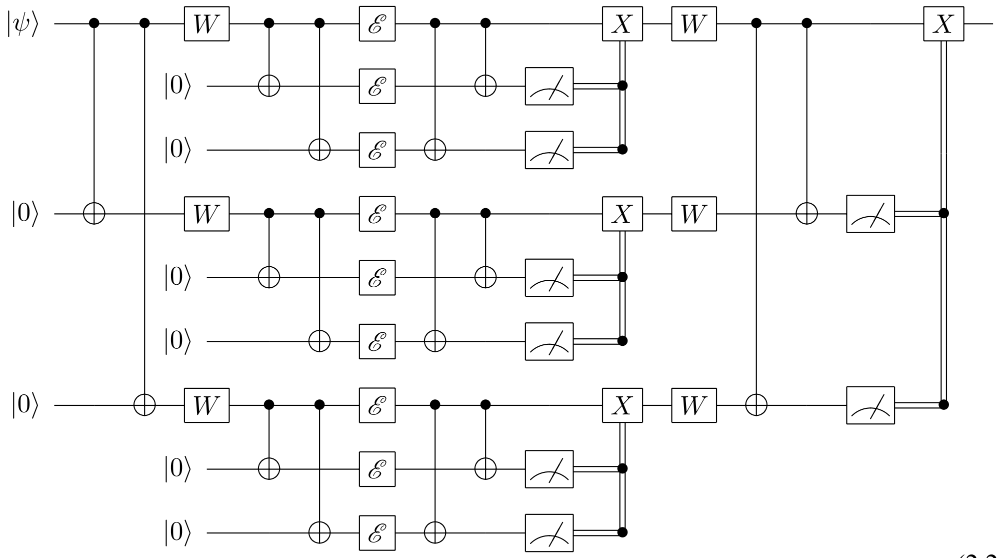
Then a single phase-flip error on these two states sends $|p\rangle$ to $|m\rangle$ and vice versa, i.e., it looks like a bit-flip on these qubits. We can thus deal with these single phase-flip errors by using a bit-flip code. In particular, we define the two nine-qubit states:

$$|0_L\rangle = |ppp\rangle = \frac{1}{2\sqrt{2}}(|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle), \tag{2.24a}$$

$$|1_L\rangle = |mmm\rangle = \frac{1}{2\sqrt{2}}(|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle). \tag{2.24b}$$

Suppose we encode a qubit of quantum information into the subspace spanned by these two states. Now single phase-flip errors can be fixed by diagnosing what subspace the $|ppp\rangle$ and $|mmm\rangle$ subspace has been sent to. Further single bit-flip errors can be dealt with from within each $|p\rangle$ and $|m\rangle$ state: these states are encoded states in our original bit-flip code. Putting this together, we see that we should be able to use this error-correcting code to correct bit-flips and phase-flips. Actually it can do more, and indeed it can handle a single Y error as well.

To see this let us construct the circuit for encoding into this code and then performing the decoding and correction.



(2.25)

Now, first of all, isn't this beautiful! Notice the three blocks of three in this code. Also notice how these blocks, when there is either no error or a single X error in the superoperators,[1] produce a channel on the outer three wires (just after the first triple of Ws and before the second triple of Ws) which is the identity (because these are designed to correct just that error). But what about if a single Z error occurs. As we noted above, this means that on the encoded quantum information this acts like a Z error on the $|p\rangle$, $|m\rangle$ states. Since only a Z error occurs, we are not taken out of the $|p\rangle$ and $|m\rangle$ subspace by this error, and so the effect of the error correction in an inner block will be to produce a state that has a single-qubit Z error on the outer wire. Now we see how the code corrects this single Z error: this is just the phase-flip error-correcting code!

But what happens if, say, a single $Y = iXZ$ error occurs? First notice that the global phase i doesn't really matter. So we can assume that the error is XZ. Now the Z error acting on the encoded state acts within the encoded space as a Z error. Thus, imagine performing this error, and then running the bit-flip code. The bit-flip code will correct the X error, but the end result will be that a Z error has occurred on the encoded information. But then the Z error will be corrected by the outer code. Thus we see that a single Y error will be corrected by this code.

So what have we in Shor's code? We have a code that can correct any single-qubit error from the set $\{X, Y, Z\}$. But, as we have argued above, this means that any single-qubit error that is a sum of these errors will also be corrected (we will make this argument rigorous in Section 2.6). So if a single arbitrary error occurs on our qubit, Shor's code will correct it. We call this a QECC that can correct a single-qubit error.

What have we learned? We have learned that it is possible to circumvent the objections we raised early: cloning, measurement, and continuous sets of errors, and to enact error correction

[1] Of course, in general our superoperators will each contain error terms, but we will simply talk about the case where there is one such error since this is, to first order, the most important error, assuming that the errors are "weak enough." Yes, this is all rather vague right now, but we need to make progress without getting bogged down in the details.

on quantum information. What does this mean for practical implementations of a quantum computer? Well, it means that more sophisticated versions of QEC might be able to make a very robust computer out of many noisy components. There are many issues that we need to discuss in this theory, and the full story of how to build such a quantum computer is the subject of fault-tolerant quantum computation. A main aim in the next section will be to obtain an understanding of the theory of QEC.

2.3 The quantum error-correcting criterion

Above we have seen the simplest example of a QECC that is truely quantum in nature, Shor's code. The basic idea there was that we *encode* quantum information into a subspace of the Hilbert space of many quantum systems. This is the principle idea of a *quantum error correcting code*. In particular, this is an example of a subspace QECC where the information is encoded into a subspace of the larger Hilbert space of the system in question. Later we will encounter a variation on this idea known as subsystem QECCs. Give that a QECC is nothing less than a subspace, an important question to ask is under what conditions does a subspace act as a QECC.

Suppose that we have a quantum system that evolves according to some error process that we represent by the superoperator $\mathscr{E}$. We assume that this superoperator is given by some Krauss operator sum representation

$$\mathscr{E}[\cdot] = \sum_k E_k[\cdot]E_k^\dagger. \tag{2.26}$$

In general, our codes will not be able to reverse the effect of all errors on our system: the goal of QEC is to make the probability of error so small that it is effectively zero, not to eliminate the possibility of error completely (although philosophers will argue about the difference between the two). It is therefore useful to assume that the Kraus operators in the expansion for $\mathscr{E}$ are made up of some errors E_i, $i \in S$, which we wish to correct. This will be a good assumption because the real error process will contain these terms, which we will then be certain we have fixed, plus the errors that we might not fix. We will return to this point later. Thus, with this assumption, we may think about $\mathscr{E}$ as having Kraus operators, some of which are errors E_i that we wish to correct and some of which are not. Note that this assumption is not necessary, but is simply a convenience for the short term. Define $\mathscr{F}$ as the operator,

$$\mathscr{F}[\cdot] = \sum_{i \in S} E_i[\cdot]E_i^\dagger. \tag{2.27}$$

Notice that $\mathscr{F}$ will not necessarily preserve the trace of a density matrix. This will not stop us from considering reversing its operation.

So given $\mathscr{F}$ with some Kraus operators E_i we can ask, under what conditions is it possible to design a quantum code and recovery operations $\mathscr{R}$ such that

$$\mathscr{R} \circ \mathscr{F}[\rho_C] \propto \rho_C, \tag{2.28}$$

for ρ_C with support over the code subspace, $\mathscr{H}_C \subseteq \mathscr{H}$? Why do we use $\propto$ here instead of $=$? Well, because $\mathscr{E}$ is not trace-preserving. This means that there may be processes occurring in the full $\mathscr{D}$ that occur with some probability and we do not need to preserve ρ on these errors.

Let us call a basis for the code subspace $|\phi_i\rangle$. $\mathscr{H}_C = \text{span}\{|\phi_i\rangle\}$. We will show that a necessary and sufficient condition for the recovery operations to preserve the subspace is given by the Knill–Laflamme *quantum error-correcting code criterion* [KL97]:

$$\langle\phi_i|E_k^\dagger E_l|\phi_j\rangle = C_{kl}\delta_{ij}, \tag{2.29}$$

where C_{kl} is a Hermitian matrix, sometimes called the code matrix. It tells us when our encoding into a subspace can protect us from quantum errors E_k. As such, it is a very important criterion for the theory of QEC. Lets show that this is a necessary and sufficient condition.

2.3.1 Sufficiency

Let us begin by showing that if this criterion is satisfied, we can construct a recovery operation $\mathscr{R}$ with the desired properties.

The first thing to do is to change the error operators. Instead of discussing the error operators E_k, define a new set of error operators $F_m = \sum_k u_{mk}E_k$, where u_{lk} are the elements of a unitary matrix u. It is a simple exercise to show that this means that F_l represents the same superoperator. Now we see that, since the E_i satisfy the error-correcting criterion,

$$\langle\phi_i|F_m^\dagger F_n|\phi_j\rangle = \sum_{k,l}\langle\phi_i|u_{mk}^* E_k^\dagger u_{nl}E_l|\phi_j\rangle = \sum_{k,l} u_{mk}^* C_{kl}u_{nl}\delta_{ij}. \tag{2.30}$$

Since C_{kl} is Hermitian, it is always possible to choose $u = \{u_{ij}\}$ such that it diagonalizes this matrix,

$$\langle\phi_i|F_m^\dagger F_n|\phi_j\rangle = d_m\delta_{m,n}\delta_{i,j}, \tag{2.31}$$

with $d_m \in R$. Now define the following operators for $d_k \neq 0$:

$$R_k = \frac{1}{\sqrt{d_k}}\sum_i |\phi_i\rangle\langle\phi_i|F_k^\dagger, \tag{2.32}$$

and if $d_k = 0$ then let $R_k = 0$. Here $\sum_i |\phi_i\rangle\langle\phi_i|$ is the projector onto the code subspace. We want to show that a recovery superoperator with R_k as its Kraus operators will correctly recover our erred quantum information:

$$\sum_k R_k \sum_l \left(F_l\rho_C F_l^\dagger\right) R_k^\dagger = \sum_{k|d_k\neq 0}\frac{1}{\sqrt{d_k}}\sum_i |\phi_i\rangle\langle\phi_i|F_k^\dagger \sum_l \left(F_l\rho_C F_l^\dagger\right)\frac{1}{\sqrt{d_k}}\sum_j F_k|\phi_j\rangle\langle\phi_j|. \tag{2.33}$$

If we can show that for $\rho_C = |\phi_m\rangle\langle\phi_n|$ this produces something proportional to ρ_C, then we will have shown that the recovery correctly restores information in the subspace (since by linearity this will work for the entire density matrix). Substituting this ρ_C in, we obtain

$$\sum_{k|d_k\neq 0}\frac{1}{d_k}\sum_i |\phi_i\rangle\langle\phi_i|F_k^\dagger \sum_l \left(F_l|\phi_m\rangle\langle\phi_n|F_l^\dagger\right)\sum_j F_k|\phi_j\rangle\langle\phi_j|. \tag{2.34}$$

Using the quantum error-correcting criterion, this becomes

$$\sum_{k|d_k\neq 0}\frac{1}{d_k}\sum_{ilj}|\phi_i\rangle d_k\delta_{lk}\delta_{im}d_k\delta_{lk}\delta_{jn}\langle\phi_j| = \sum_k d_k|\phi_m\rangle\langle\phi_n| = \left(\sum_k d_k\right)\rho_C. \tag{2.35}$$

Thus we see that, indeed, the recovery produces a state proportional to ρ_C. Notice that if $\mathscr{E}$ is trace-preserving, then $\sum_k d_k = 1$ and then we recover exactly ρ_C, as desired.

We need to check that R_k forms a valid superoperator. Check,

$$\begin{aligned} R &= \sum_k R_k^\dagger R_k \\ &= \sum_{k|d_k\neq 0} \frac{1}{d_k} \sum_{i,j} F_k|\phi_i\rangle\langle\phi_i||\phi_j\rangle\langle\phi_j|F_k^\dagger = \sum_{k|d_k\neq 0} \frac{1}{d_k}\sum_i F_k|\phi_i\rangle\langle\phi_i|F_k^\dagger. \end{aligned} \tag{2.36}$$

Notice, using the quantum error-correcting criterion, that this operator is a projector:

$$\begin{aligned} R^2 &= \sum_{k|d_k\neq 0} \frac{1}{d_k}\sum_i F_k|\phi_i\rangle\langle\phi_i|F_k^\dagger \sum_{k'|d_{k'}\neq 0}\frac{1}{d_{k'}}\sum_{i'} F_{k'}|\phi_{i'}\rangle\langle\phi_{i'}|F_{k'}^\dagger \\ &= \sum_{k|d_k\neq 0} \frac{1}{d_k}\sum_i F_k|\phi_i\rangle \sum_{k'|d_{k'}\neq 0}\frac{1}{d_{k'}}\sum_{i'} d_k\delta_{k,k'}\delta_{i,i'}\langle\phi_{i'}|F_{k'}^\dagger \\ &= \sum_{k|d_k\neq 0} \frac{1}{d_k}\sum_i F_k|\phi_i\rangle\langle\phi_i|F_k^\dagger = R. \end{aligned} \tag{2.37}$$

Thus, if we add one extra (if necessary) projector to the R_ks that has support on the space orthogonal to this projector, $I - \sum_k R_k^\dagger R_k$, then we will obtain a complete set of Kraus operators that satisfy the proper normalization condition for the Kraus operators. Thus, we have seen that we have a valid recovery operator that does the proper recovery and that this valid recovery operator, with the addition of possibly one extra Kraus operator, is indeed a valid superoperator.

2.3.2 Necessity

Let us show the necessity of the quantum error-correcting criterion. Errors followed by recovery produces the following evolution on an encoded state:

$$\sum_k R_k \left(\sum_i E_i\rho_C E_i^\dagger\right) R_k^\dagger = c\rho_C. \tag{2.38}$$

We want to show that this implies the error-correcting criterion. Note that ρ_C by itself is equivalent to a superoperator in which no evolution has taken place. Express the above as

$$\sum_{k,l}(R_kE_i)\rho_C(E_i^\dagger R_k^\dagger) = cI\rho_C I. \tag{2.39}$$

Now let P_C be a projector onto the code subspace, $P_C = \sum_i |\phi_i\rangle\langle\phi_i|$. Then the above criterion is that, for all ρ,

$$\sum_{k,l}(R_kE_iP_C)\rho(P_CE_i^\dagger R_k^\dagger) = cP_C\rho P_C. \tag{2.40}$$

Invoke now the unitary degree of freedom of the Kraus operator sum representation. This says that there must exist a unitary transform on the superoperators on the left-hand side of the equation such that one obtains only the single superoperator on the right-hand side of the equation. This means that there must exist an orthogonal set of vectors with coefficients u_{ki} such that

$$R_kE_iP_C = u_{ki}cP_C. \tag{2.41}$$

Taking the conjugate transpose of this equation and setting $i = j$ yields

$$P_C E_j^\dagger R_k^\dagger = u_{kj}^* c^* P_C. \tag{2.42}$$

Multiplying this equation on the left of the original equation yields

$$P_C E_j^\dagger R_k^\dagger R_k^\dagger E_i P_C = u_{kj}^* u_{ki} |c|^2 P_C. \tag{2.43}$$

Summing this equation and using the fact that $\mathscr{R}$ must be a trace-preserving operator,

$$P_C E_i^\dagger E_j P_C = \sum_k u_{kj}^* u_{ki} |c|^2 P_C. \tag{2.44}$$

Defining $C_{ij} = \sum_k u_{kj}^* u_{ki} |c|^2$, this is just

$$P_C E_i^\dagger E_j P_C = C_{ij} P_C, \tag{2.45}$$

where we see that C_{ij} is Hermitian. Taking matrix elements of this equation and relabeling i and j as k and l then yields the quantum error-correcting criterion, Eq. (2.29). Thus we have established the necessity and sufficiency of the quantum error-correcting criterion.

2.4 The distance of a quantum error-correcting code

The general theory of QECCs is designed to deal with an arbitrary set of errors $\{E_i\}$ and an arbitrary size of the subspace into which we encode. However, in many cases we will specialize to the case where the information is encoded across qubits, and further where we consider $\{E_i\}$ to be made up of tensor products of Pauli operators of weight less than some fixed length t. Consider an operator P on n qubits that is made up of a tensor product of Pauli operators $\{I, X, Y, Z\}$. The number of nonidentity (i.e., not I) terms in this operator is called the weight of P. Thus, the situation we are concerned with is where the set of correctable errors is $\mathscr{E}_t = \{P|$ weight of P is less than $t\}$. In the case where we have a QECC on n qubits, where we encode k qubits into the code, and the maximum number of errors the code can correct is t, then we call this a $[[n, k, d]]$ code, where $d = 2t + 1$ is the distance of the code. The distance of a code is the smallest weight operator that can be used to enact a transformation on the information encoded into the code.

2.5 Content of the quantum error-correcting criterion and the quantum Hamming bound

What is the content of the quantum error-correcting criterion, $\langle \phi_i | E_k^\dagger E_l | \phi_j \rangle = C_{kl} \delta_{ij}$, Eq. (2.29)? First look at the δ_{ij}. This implies that codewords after being changed by the error E_l are orthogonal to the codewords after being changed by the error E_k. If $l = k$ this implies that information in codewords is not distorted by the effect of error E_k: they may be rotated, but the inner product between all codewords will be the same before as after (up to a full normalization factor). In our example of QECCs for the bit-flip code, we saw that each possible error could act to take the error to an orthogonal subspace. If every such error acts this way for a code, then the code is said to be nondegenerate. In this case, C_{kl} will be diagonal. Some codes, however, do not possess this property: there are multiple errors that can produce the same syndrome, but the recovery procedure works in spite of this.

For nondegenerate codes there is a nice bound on the size of the codes. Suppose that we wish to encode k qubits into n bare qubits in a QECC that corrects errors on t or fewer qubits, (i.e., a $[[n, k, 2t + 1]]$ code). In order for a nondegenerate QECC to correct all of these errors, for each error there must be an orthogonal subspace. There are $\binom{n}{j}$ places where j errors can occur. And in each of these places there are three different nontrivial Pauli errors. Thus, the total number of errors for such a code we have described is

$$\sum_{j=0}^{t} \binom{n}{j} 3^j. \tag{2.46}$$

For each of these errors, there must be a subspace as big as the size of the encoded space, 2^k, and these subspaces must be orthogonal. Thus, each subspace must fit into the full space of n qubits. We then obtain the bound

$$\sum_{j=0}^{t} \binom{n}{j} 3^j 2^k \leq 2^n. \tag{2.47}$$

This is called the *quantum Hamming bound*. Suppose that we want a code that corrects $t = 1$ error and encodes $k = 1$ qubit. Then we obtain the inequality $(1 + 3n)2 \leq 2^n$. This inequality cannot be satisfied for $n \leq 4$. Thus, for nondegenerate codes, the smallest code that can correct a single error and encodes a single qubit has $n = 5$. Indeed, we will find that just such a code exists (such codes that saturate this bound are called perfect codes).

2.6 Digitizing quantum noise

Suppose that we have an error-correcting code that corrects a set of errors $\{E_k\}$. What other errors will this code correct? It turns out that this code will correct any linear combination of these errors. To see this, work with the errors that satisfy the diagonal error-correcting criterion, as in the sufficiency construction above (the F_ls). Now suppose that the actual F_ls are written as a sum over the F_ls we can correct: $G_l = \sum_p f_{lp} F_p$. Then, using the recovery operation we defined in the sufficiency proof, we obtain that the action of recovery after the error is

$$\sum_k R_k \sum_l \left(G_l \rho_C G_l^\dagger \right) R_k^\dagger$$
$$= \sum_{k|d_k \neq 0} \frac{1}{\sqrt{d_k}} \sum_i |\phi_i\rangle\langle\phi_i| F_k^\dagger \sum_l \left(G_l \rho_C G_l^\dagger \right) \frac{1}{\sqrt{d_k}} \sum_j F_k |\phi_j\rangle\langle\phi_j|. \tag{2.48}$$

We wish to show that if we operate on $\rho_C = |\phi_m\rangle\langle\phi_n|$ we will again obtain something proportional to ρ_C. Thus we obtain

$$\sum_{k|d_k \neq 0} \frac{1}{d_k} \sum_i |\phi_i\rangle\langle\phi_i| F_k^\dagger \sum_l \left(G_l |\phi_m\rangle\langle\phi_n| G_l^\dagger \right) \sum_j F_k |\phi_j\rangle\langle\phi_j|. \tag{2.49}$$

Substituting our expression for G_l as a sum over F_k yields

$$\sum_{k|d_k\neq 0}\frac{1}{d_k}\sum_i|\phi_i\rangle\langle\phi_i|F_k^\dagger\sum_l\left(\sum_p f_{lp}F_p|\phi_m\rangle\langle\phi_n|\sum_q f_{lq}^*F_q^\dagger\right)\sum_j F_k|\phi_j\rangle\langle\phi_j|. \quad (2.50)$$

Using the quantum error-correcting criterion, we see that this becomes

$$\begin{aligned}\sum_{k|d_k\neq 0}\frac{1}{d_k}\sum_{iljpq}|\phi_i\rangle d_k\delta_{pk}\delta_{im}d_k\delta_{qk}f_{lp}f_{lq}^*\delta_{jn}\langle\phi_j| &= \sum_{kl}d_kf_{lk}f_{lk}^*|\phi_m\rangle\langle\phi_n|\\ &= \left(\sum_{kl}d_kf_{lk}f_{lk}^*\right)\rho_C. \quad (2.51)\end{aligned}$$

So, even for this linear sum of errors, we correctly restore the coded subspace.

What have we done? We have shown that even though we have designed a code to correct E_k operators, it can in fact correct any linear sum of these operators. This is great! Why? Because, for example, if we want to correct a superoperator that has one qubit that has been arbitrarily erred (and only one qubit), then we need only consider a code that corrects X, Y, and Z errors, since every single-qubit error operator can be written as a sum of these errors (plus identity, which we, by default almost always include in our possible errors). This is what is known as making the errors discrete or digital. This discovery, that a code that was designed to correct a discrete set of errors can also correct a continuous set of errors, is one of the most important discoveries in all of quantum computing. The reason for this property is that quantum theory is linear. This linearity has a lot to do with why we can treat amplitudes like fancy probabilities and indeed, when we view quantum theory this way, we are not quite as surprised as if we thought about the components of a wave function as being some parameters with a reality all their own.

2.7 Classical linear codes

Above we have seen one of the most basic QECCs, the Shor code. The Shor code is a [[9,1,3]] QECC. In order to discuss further QECCs, it is useful to understand a class of traditional classical codes known as classical linear codes (of course they were known just as linear codes until quantum computing came along). In this section we will discuss classical binary linear codes, a subset of classical linear codes.

A *classical binary code* is a method for encoding k bits into n bits. In other words, for every one of the 2^k combinations of the k bits, a different n-bit vector is used to encode these k bits (so, obviously $k \leq n$). A classical binary code is called *linear* if the set of binary strings used in the code forms a closed linear subspace C of F_2^n. Recall that F_2 is the finite field with two elements, which we can call 0 and 1. Since F_2 is a field we can add and multiply these numbers using the normal rules of arithmetic with the only modification that addition is done modulo 2. We can then create a linear vector space of dimension n, F_2^n, by considering ordered sets of n elements of F_2: these are just binary vectors. In the linear vector space we can now add two elements, which corresponds to the process of bit-wise addition modulo 2 of the elements of the two elements. To be clear, an example of the addition of two vectors over F_2^5 is $01100 + 01010 = 00110$. Using these simple definitions, a closed linear subspace C of F_2^n is a set of vectors such that if the codeword v_1 is in C and v_2 is in C, then $v_1 + v_2$ is in C.

Since a classical linear binary code forms a closed linear subspace, we may express any codeword in a complete basis, $\{w_i\}_{i=1}^k$, for this subspace:

$$v = \sum_{i=1}^{k} a_i w_i, \tag{2.52}$$

for $v \in C$, where $a_i \in \{0, 1\}$. From this expression we can explicitly see how to encode into such a code: if we let the a_i denote the k-bit binary string we wish to encode, then we see that, given a basis $\{w_i\}_{i=1}^k$, we can encode by simply taking the appropriate linear combination of the basis vectors. Of course there are different choices of basis vectors, and while these lead to different explicit schemes for encoding, the codes in different bases are essentially the same.

Given the basis expansion in Eq. (2.52) it is natural to define the *generator matrix*. This is a matrix with a set of basis vectors for the rows of the matrix:

$$G = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \end{pmatrix}. \tag{2.53}$$

Thus, the generator matrix for a k-bit linear code encoding into n bits is a k by n dimensional matrix. With the generator matrix one can immediately see how to encode a binary k-bit vector by simply taking the appropriate left multiplication. If a is the k-bit vector (a in F_2^k), then the encoding procedure is simply the matrix multiplication $a^T G$.

An example of a generator matrix is the matrix for the seven-bit Hamming code. In this case the generator matrix is

$$G = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{2.54}$$

To encode 0110 using this matrix we left multiply by the row vector $(0, 1, 1, 0)$ and obtain the result $(0, 1, 1, 1, 1, 0, 0)$. Thus 011 is encoded as 0111100 using this generator matrix.

The generator matrix is a nice way to specify a linear subspace, but what about the error-correcting properties of such codes? In order to discuss error correction on classical binary linear codes, it is useful to define the *parity check* matrix. The parity check matrix, H, is defined such that its null space is the code C. In other words, the check matrix is a $n - k$ set of constraints such that

$$Hv = 0, \tag{2.55}$$

for all $v \in C$. H is an $n - k$ by n matrix. Here 0 is a k-dimensional column vector made of zeros. Note that these dimensions arise because H has rows made up of the maximal number of linearly independent vectors that are orthogonal to the subspace C.

What use is the parity check matrix? Well, as one can guess from its name, it is used to check whether an error has occurred on information encoded into the binary code and to (hopefully) correct this error. Classical errors are simply bit-flips, so any error on a codeword v can be represented by addition over F_2^n. Thus, if e is an n-bit binary vector with 1s where the bit-flips occur, then the result of this error on the codeword is $v + e$. If we apply H to this resulting erred codeword, we obtain $H(v+e) = He$, since v is in C. He is an $n-k$-bit binary string which we call the *syndrome*.

Suppose that we wish to correct a set of errors $\mathscr{S} = \{e_i\}$. If for every error e_i there is a unique syndrome He_i, then we can correct this set of errors. Suppose that $w = v + e_i$ is the error e_i acting on codeword v. Error correction is then simply performed by calculating the syndrome $Hw = He_i$. Since He_i uniquely identifies the error e_i, we can apply to w the bit-flips corresponding to e_i. In other words, we can perform the error-correction procedure $w + e_i$, which since $x + x = 0$ over F_2^n, is equal to $w + e_1 = v + e_1 + e_1 = v$. Thus, we see that Hw denotes the syndrome of what error has occurred. Contrariwise, if there are two errors, $e_1 \neq e_2$, that have the same syndrome, then if the error was e_1 we might apply e_2 to fix the error, resulting in $w + e_1 + e_2$. Since $e_1 + e_2$ does equal 0, this will result in an error on the encoded information.

For the seven-bit code described in Eq. (2.54), the parity check matrix is

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}. \tag{2.56}$$

One can check that all single-bit errors give different syndromes using this parity check matrix. To see this, note that single-bit-flip errors correspond to errors e with one 1 and six 0s. When applying H to these vectors one obtains as an output a three-bit syndrome made up of one of the columns of H. Since all of these columns are different, we see that each such error yields a distinct syndrome.

Finally we introduce the equivalent notion of the parameters of classical linear binary code as we did for a QECC. In particular, we say a code is an $[n, k, d]$ code if it encodes k bits into n bits and the distance of the code is d. The distance of a code is the weight (number of 1s in the vector describing the error) of the smallest error that transforms one codeword into another. A code with a distance of d can correct $t = \lfloor \frac{d-1}{2} \rfloor$ errors. (The picture you should have in mind is that the codewords are separated by a distance d and thus in balls of radius t there must be unique codewords that allow one to uniquely diagnose the error.)

One further concept we will find useful in QECCs is the notion of the *dual* of a classical linear code. Recall that the generator matrix, G, for an $[n, k, t]$ code C is a $k \times n$ matrix and the parity check matrix, H, is an $(n-k) \times n$ matrix. One can then define the dual of C, denoted $C^\perp$, by exchanging the role of the parity check and generator matrices. In particular, if we define $G^\perp = H$ and $H^\perp = G$, then this defines an $[n, n-k, t']$ classical linear binary code. The requirement that the parity check matrix acting on a codeword yields 0 can be seen to be satisfied by noting that this condition for the original code C is $HG^T = 0$, and taking the transpose of this equation yields the similar condition for the dual code, $GH^T = 0$. The dual code is made up of all the codewords that are orthogonal to the original codewords. Note that codewords in F_2^n can be orthogonal to itself (if it is made up of an even number of 1s it will be orthogonal to itself). Thus it is possible that a code C may contain as a subspace its dual code $C^\perp$. Further, it may even be possible, if $n = 2k$, that $C = C^\perp$. Such a code is called a *self-dual* code and will be used in the constructions described in the next section.

2.8 Calderbank, Shor, and Steane codes

We previously saw how Shor's nine-qubit QECC could be used to protect against bit-flip errors as well as against phase-flip errors. Remarkably, this code also protected against a combined bit-flip and phase-flip error and hence, by linearity, any combination of these errors. Here we will describe a similar set of codes that independently treat bit-flip and phase-flip errors. These codes are constructed from classical linear binary codes and are known as Calderbank, Shor, and Steane (CSS for short) codes [CS96, S96d].

Suppose we have an $[n, k, d]$ classical binary linear code C with generator matrix G and parity check matrix H. One could naturally define a QECC that can correct for bit-flips on this code, but it is unclear how this could be used to construct a code that deals with phase-flips. Consider, however, a state that is an equal superposition over the codewords in C:

$$|\psi\rangle = \frac{1}{\sqrt{2^k}} \sum_{v \in C} |v\rangle. \tag{2.57}$$

To understand how phase-flips act on this state, we apply a Hadamard transform to every qubit (such that the role of phase-flips and bit-flips is exchanged):

$$W^{\otimes n}|\psi\rangle = \frac{1}{\sqrt{2^{k+n}}} \sum_{v \in C} \sum_{w \in F_2^n} (-1)^{vw} |w\rangle, \tag{2.58}$$

where $vw = v_1 w_1 + v_2 w_2 + \cdots + v_n w_n \bmod 2$. It is easy to verify that

$$\sum_{v \in C} (-1)^{vw} = \begin{cases} 2^k \text{ if } w \in C^\perp \\ 0 \text{ otherwise} \end{cases}, \tag{2.59}$$

and thus

$$W^{\otimes n}|\psi\rangle = \frac{1}{\sqrt{2^{n-k}}} \sum_{w \in C^\perp} |w\rangle. \tag{2.60}$$

This should give you an idea: since the Hadamard exchanged bit-flips and phase-flips, if the code $C^\perp$ corrects bit-flip errors, then we could protect $|\psi\rangle$ from phase-flip errors using this protection. Of course, at this point $|\psi\rangle$ is just a single state, not a QECC of any nontrivial dimension, but this is the basic idea of how CSS codes work.

Suppose that C_1 is an $[n, k_1, d_1]$ classical binary linear error-correcting code and C_2 is a subcode of C_1 that is a $[n, k_2, d_2]$ code (k_2 is less than k_1). Because C_2 is a subcode of C_1 we can define the different *cosets* of C_2 in C_1. In particular, consider the set $\mathscr{C}_w = \{v + w | v \in C_2\}$. These sets form a partition of C_2 into different cosets. In other words, it is possible to pick a set of $k_1 - k_2$ vectors such that each $\mathscr{C}_w$ is a unique set of vectors in C_1. We call these vectors *coset representatives*. Given this, we can now define the CSS codeword states. In particular, we define

$$|v\rangle = \frac{1}{\sqrt{2_2^k}} \sum_{w \in C_2} |w + v\rangle, \tag{2.61}$$

where v is a coset representative. Note that each of these states contains a superposition over codewords contained in C_1 and further, because the cosets do not overlap, these codewords are orthonormal.

The first idea here is that we will use the code C_1 to correct for bit-flip errors. Note that, as we have seen with the Shor code, the fact that we have a superposition of codewords is not an obstacle to protecting such states from bit-flip errors. The second idea is that we can use $C_2^\perp$ to protect for phase-flip errors. To see how this works, apply the n-qubit Hadamard transform to our codewords. A slight modification of our above calculation shows that

$$W^{\otimes n}|v\rangle = \frac{1}{\sqrt{2^{n-k_2}}} \sum_{w \in C_2^\perp} (-1)^{vw}|w\rangle. \tag{2.62}$$

Notice now that we have codewords that are superpositions over $C_2^\perp$. Thus, if $C_2^\perp$ can correct t_2 bit-flip errors, then we can use this correction (in the appropriately changed basis) to protect the quantum information in the above code space against t_2 phase-flip errors.

To summarize, if we have an $[n, k_1, d_1]$ code and an $[n, k_2, d_2]$ subcode whose dual is an $[n, n-k_2, d_3]$ code, the code described above is a QECC with parameters $[[n, k_1 - k_2, d]]$, where d is the minimum of d_1 and d_3. These codes are the *CSS codes*.

The most famous example of a CSS code is the [[7, 1, 3]] Steane code. This code arises from the Hamming code we have described in the previous section. In particular, let C_1 be the $[7, 4, 3]$ Hamming code described in Section 2.7. Examining the parity check matrix for this code one can easily see that the dual of C_1 is contained in C_1 (this is the code whose generator matrix is the parity check matrix of C_1). For the Steane code we choose C_1 as the Hamming code and $C_2 = C_1^\perp$. The dual of C_2 is therefore C_1 again. Thus, since C_1 has a distance of 3, the Steane code has a distance of 3. Further, since C_2 encodes one less bit than C_1, the Steane code can encode 1 qubit. Thus we see, as claimed, that the Steane code is a [[7, 1, 3]] QECC.

2.9 Stabilizer quantum error-correcting codes

CSS codes were among the first QECCs discovered. Following their discovery, a class of codes that are more general than the CSS codes was discovered, which had the property of also unifying other known QECCs [G96a, CRS+98]. These codes are known as stabilizer QECCs. The theory of stabilizer QECCs is a tool of great use in quantum computing today, both within the theory of QEC and elsewhere. It is, of course, important to remind oneself that this is not the only type of QECC out there. In this section we describe the basis of stabilizer QECCs.

2.9.1 Anticommuting

Suppose that we have a set of states $|\psi_i\rangle$ that are $+1$ eigenstates of a Hermitian operator S, $S|\psi_i\rangle = |\psi_i\rangle$. Further suppose that T is an operator that anticommutes with S, $ST = -TS$ (T is not zero). Then it is easy to see that $S(T|\psi_i\rangle) = -TS|\psi_i\rangle = -(T|\psi_i\rangle)$. Thus, the states $T|\psi\rangle$ are -1 eigenstates of S. Since the main idea of QEC is to detect when an error has occurred on a code space, such pairs of operators S and T can be used in such a manner: if we are in the $+1$ eigenvalue subspace of S then an error of T on these subspace vectors will move to a -1 eigenvalue subspace of S: we can detect that this error has occurred.

In fact, we have already seen an example of this in the bit-flip code. Recall that we noted that the code subspace for this code is spanned by $|000\rangle$ and $|111\rangle$ and that these two operators are $+1$ eigenstates of both $S_1 = Z \otimes Z \otimes I$ and $S_2 = Z \otimes I \otimes Z$. Further note that $(X \otimes I \otimes I)S_1 = -S_1(X \otimes I \otimes I)$ and $(X \otimes I \otimes I)S_2 = -S_2(X \otimes I \otimes I)$. Thus, if we start out in the $+1$ eigenvalue subspace of both S_1 and S_2 (like the bit-flip code), then if a single bit-flip occurs on the first qubit, we will now have a state that is in the -1 eigenvalue subspace of both S_1 and S_2. This at least fulfills our requirement that our errors should take us to orthogonal subspaces.

More generally, consider the following situation. Suppose that we have a set of operators S_i such that our code space is defined by $S_i|\psi\rangle = |\psi\rangle$ for $|\psi\rangle$ in the code subspace. Now suppose that we have errors E_i such that the products $E_k^\dagger E_l$ always anticommute with at least one S_i. Recall that the quantum error-correcting criterion was

$$\langle\phi_i|E_k^\dagger E_l|\phi_j\rangle = C_{kl}\delta_{ij}.$$

Since the codewords are $+1$ eigenvalue eigenstates of S_i, we find that

$$\langle\phi_i|E_k^\dagger E_l|\phi_j\rangle = \langle\phi_i|E_k^\dagger E_l S_i|\phi_j\rangle. \tag{2.63}$$

Suppose that S_i is one of the particular S_is that anticommute with $E_k^\dagger E_l$. Then this is equal to

$$\langle\phi_i|E_k^\dagger E_l|\phi_j\rangle = \langle\phi_i|E_k^\dagger E_l S_i|\phi_j\rangle = -\langle\phi_i|S_i E_k^\dagger E_l|\phi_j\rangle. \tag{2.64}$$

But, since S_i acts as $+1$ on the code space, this is just

$$\langle\phi_i|E_k^\dagger E_l|\phi_j\rangle = \langle\phi_i|E_k^\dagger E_l S_i|\phi_j\rangle = -\langle\phi_i|S_i E_k^\dagger E_l|\phi_j\rangle = -\langle\phi_i|E_k^\dagger E_l|\phi_j\rangle. \tag{2.65}$$

This implies that

$$\langle\phi_i|E_k^\dagger E_l|\phi_j\rangle = 0. \tag{2.66}$$

Thus, we have shown that, given the S_i and $E_k^\dagger E_l$ that properly anticommute with these S_i, the set of errors $\{E_k\}$ satisfies the quantum error-correcting criterion and therefore the code space is a valid QECC for these errors.

This trick, of defining the states as being the $+1$ eigenstates of some operators and then noting that if the product of error terms anticommutes then this is a valid quantum error-correcting code, is at the heart of the reason we use the stabilizer formalism. But what should we use for the S_is? Well, you might already be guessing what to use, because you recall that the Pauli group has nice commuting, anticommuting properties and eigenvalues that are ± 1 (or $\pm i$). Indeed, this is what we will use.

2.9.2 The Pauli and stabilizer groups

2.9.2.1 Pauli group First recall the definition of a group. A group is a set of objects $\mathscr{G}$ along with a binary operation of multiplication that satisfies (0) [closure] $g_1g_2 \in \mathscr{G}$ for all $g_1, g_2 \in \mathscr{G}$, (1) [associativity] $g_1(g_2g_3) = (g_1g_2)g_3$ for all $g_1, g_2, g_3 \in \mathscr{G}$, (2) [identity] there exists an element $e \in \mathscr{G}$ such that for all $g_1 \in \mathscr{G}$, $g_1 e = g_1$, (3) [inverse] for every $g_1 \in \mathscr{G}$ there exists an element $g_2 \in \mathscr{G}$ such that $g_1g_2 = e$, which we call the inverse of g_1, written $g_2 = g_1^{-1}$. The *Pauli group* is a particular group that satisfies these group axioms. Actually, people in quantum computing are very sloppy, and when they refer to the Pauli group they are usually refering to a particular representation of the Pauli group by unitary matrices. We will slip into this nomenclature soon enough; having learned a bad habit it is hard to go back.

What is this representation of the Pauli group, $\mathscr{P}_n$? Recall that the Pauli operators on a single qubit are $\{I, X, Y, Z\}$. The representation of the Pauli group we will deal with is the group formed by elements of the form $i^k P_1 \otimes P_2 \otimes \cdots \otimes P_n$, where each P_i is an element of $\{I, X, Y, Z\}$ and $k \in \{0, 1, 2, 3\}$. From this representation, we see that the Pauli group is a non-Abelian group, i.e., the elements of the group do not all commute with each other. Some important properties of elements of the Pauli group:

(i) Elements of the Pauli group square to $\pm I$: $P^2 = \pm I$.
(ii) Elements of a the Pauli group either commute $PQ = QP$ or anticommute $PQ = -QP$.
(iii) Elements of the Pauli group are unitary $PP^\dagger = I$.

2.9.2.2 Stabilizer group Define a *stabilizer group* $\mathscr{S}$ as a subgroup of $\mathscr{P}_n$ which has elements that all commute with each other and which does not contain the element $-I$. An example of a stabilizer group on three qubits is the group with elements $\mathscr{S} = \{III, ZZI, ZIZ, IZZ\}$. Notice that here we have dropped the tensor product between the elements, i.e., $ZZI = Z \otimes Z \otimes I$. We usually do not specify all of the elements of the stabilizer group. Instead we specify a minimal set of generators. A set of generators of a group is a set of elements of the group such that multiplication of these generators leads to the full group. A minimal set of such generators is a set of generators of minimal size that has this property. In the example of $\mathscr{S} = \{III, ZZI, ZIZ, IZZ\}$, this group is generated by ZZI and ZIZ: $(ZZI)(ZIZ) = IZZ$ and $(ZZI)^2 = III$. We write this fact as $\mathscr{S} = \langle ZZI, ZIZ \rangle$. For a stabilizer $\mathscr{S}$ we write a set of minimal generators as $S_1, S_2, \ldots, S_r$.

Now, since $-I$ is not in the stabilizer, all elements of our stabilizer must square to $+I$. Operators that square to $+I$ must have eigenvalues $+1$ or -1. Since $\mathscr{S}$ is Abelian, we can write a generic element of the stabilizer as $S_1^{a_1} S_2^{a_2} \cdots S_k^{a_r}$. Since $S_i^2 = I$, $a_i \in \{0, 1\}$. Further, for each $a \in Z_2^r$ the element of the stabilizer so specified is unique. Suppose that this was not true, that $S_1^{a_1} S_2^{a_2} \cdots S_k^{a_r} = S_1^{b_1} S_2^{b_2} \cdots S_k^{b_r}$ for $a \neq b$. Then $S_1^{a_1+b_1} S_2^{a_2+b_2} \cdots S_k^{a_r+b_r} = I$. But this is only true if $a_i = b_i$.

2.9.3 Stabilizer subspace and error correction

Now, given a stabilizer group $\mathscr{S}$, we can define a subspace on our n qubits. In particular we define this subspace as all states $|\psi\rangle$ that satisfy $S|\psi\rangle = |\psi\rangle$ for all stabilizer elements $S \in \mathscr{S}$. Actually, we do not need all of these equations to define the code space. All we need are the equations for the generators of the stabilizer: $S_i|\psi\rangle = |\psi\rangle$. Let us call the subspace defined by these equations $\mathscr{H}_S$. Such stabilizer subspaces are very nice. One reason that they are nice is that instead of specifying the states of the subspace we can just specify the generators of the stabilizer group. This is oftentimes much easier. Further, as we will see, it is easy to figure out the error-correcting properties of stabilizer subspaces.

In particular, suppose we have a stabilizer subspace for a code generated by $S_1, S_2, \ldots, S_r$. Then, suppose the Pauli operator P anticommutes with one of these generators S_i. Then, as above, for $\{|\psi_i\rangle\}$ a basis for $\mathscr{H}_S$,

$$\langle \psi_i | P | \psi_j \rangle = \langle \psi_i | P S_i | \psi_j \rangle = -\langle \psi_i | S_i P | \psi_j \rangle, \tag{2.67}$$

so

$$\langle \psi_i | P | \psi_j \rangle = 0. \tag{2.68}$$

Thus, as above, if $\{E_a\}$ is a set of Pauli group errors, if we consider the products $E_a^\dagger E_b$ and these anticommute with at least one of the generators of $\mathscr{S}$, then we have satisfied the error-correcting criterion for these errors:

$$\langle\psi_i|E_b^\dagger E_a|\psi_j\rangle = 0. \tag{2.69}$$

If these elements are themselves elements of the stabilizer, $E_a^\dagger E_b \in \mathscr{S}$, then

$$\langle\psi_i|E_b^\dagger E_a|\psi_j\rangle = \langle\psi_i|S|\psi_j\rangle = \delta_{ij}. \tag{2.70}$$

For an error set $\{E_a\}$, if all of the products $E_a^\dagger E_b$ either anticommute with generators of the stabilizer $S_1, S_2, \ldots, S_r$ or are elements of the stabilizer, then we see that the E_a satisfy the quantum error-correcting criterion.

In our example where $\mathscr{S} = \langle ZZI, ZIZ\rangle$, we can consider the set of errors comparising the identity operator and all single-qubit bit-flip errors, $\{III, XII, IXI, IIX\}$. Then the set of products of these errors is $\{III, XII, IXI, IIX, XXI, XIX, IXX\}$. Of these, the first III is in the stabilizer. All of the others, however, anticommute with either ZZI or ZIZ. For example, $(XXI)(ZIZ) = -(ZIZ)(XXI)$. How do we check whether two Pauli group elements commute or anticommute? Suppose these group elements are $P_1 \otimes P_2 \otimes \cdots \otimes P_n$ and $Q_1 \otimes Q_2 \otimes \cdots \otimes Q_n$. Then we count the locations where P_i and Q_i differ and neither P_i nor Q_i is I. If this number is even, then these two Pauli group elements commute, and if it is odd, then they anticommute.

If a stabilizer group has a minimal number of generators, which are $S_1, S_2, \ldots, S_r$, what is the dimension of the stabilizer subspace? Take the first stabilizer generator. This stabilizer generator squares to identity, so has ± 1 eigenvalues. Further, this stabilizer generator has trace zero. Why? All of the Pauli operators are trace 0 except I which has trace 2. Since the stabilizer generator cannot be identity (unless the stabilizer consists solely of the identity, a case we will disregard) it is a tensor product of terms, at least one of which must be a Pauli element not equal to I. Then, since $\mathrm{Tr}[A \otimes B] = \mathrm{Tr}[A]\mathrm{Tr}[B]$, this implies that $\mathrm{Tr}[P] = 0$, for all Pauli group elements except $\pm I$. Thus, if we take S_1 it must have 2^{n-1} eigenvalues $+1$ and 2^{n-1} eigenvalues -1. So $S_1|\psi\rangle = |\psi\rangle$ splits the Hilbert space of our n qubits in half. What happens when we impose $S_2|\psi\rangle = |\psi\rangle$? Note that $\frac{1}{2}(I + S_1)$ is the projector onto the $+1$ eigenvalue eigenspace of S_1. We can thus use $\frac{1}{2}(I + S_1)S_2$ to understand how much of this subspace has eigenvalues of S_2 that are $+1$. Note that $\mathrm{Tr}[\frac{1}{2}(I + S_1)S_2] = 0$. Thus, we see that for the 2^n-dimensional subspace that satisfies $S_1|\psi\rangle = |\psi\rangle$, a subspace of dimension 2^{n-2} satisfies $S_2|\psi\rangle = |\psi\rangle$. Continuing inductively, we see that each $S_i|\psi\rangle = |\psi\rangle$ cuts the space of the previous $S_1|\psi\rangle = |\psi\rangle, \ldots, S_{i-1}|\psi\rangle = |\psi\rangle$ in half.

What does this mean? This means that the dimension of a stabilizer subspace for a stabilizer with r generators is 2^{n-r}. It is also useful to notice that the dimension of the subspace with a fixed set of ± 1 eigenvalues of the S_i operators is also 2^{n-r}. For our example of $S_1 = ZZI$ and $S_2 = ZIZ$, this implies that the stabilizer subspace is two-dimensional, which is correct.

2.9.4 Logical operators for stabilizer codes

So, given a stabilizer group $\mathscr{S}$ generated by r elements, we now know that this specifies a subspace of dimension 2^{n-r}. This subspace can be used to encode $k = n - r$ qubits. Is there a nice way to talk about this subspace as k qubits? Indeed there is.

The centralizer of the stabilizer group $\mathscr{S}$ in $\mathscr{P}_n$ is the set of operators $P \in \mathscr{P}_n$ that satisfy $PS = SP$ for all $S \in \mathscr{S}$. Since our stabilizer group does not contain $-I$, it turns out that the centralizer is equal to the normalizer. The normalizer $\mathscr{N}$ of $\mathscr{S}$ in $\mathscr{P}_n$ is the set of operators $P \in \mathscr{P}_n$ such that $PSP^\dagger \in S$ for all $S \in \mathscr{S}$. Notice that the stabilizer is automatically in the normalizer, since all of the elements of the stabilizer commute with each other and are all unitary. An important set of operators are those that are in the normalizer $\mathscr{N}$ but not in the stabilizer, $\mathscr{N} - \mathscr{S}$. Why are these elements important? Because they represent *logical Pauli* operators on the k encoded qubits of our subsystem code.

Let us see this for an example and then move on to understanding the logical operators in general. Our example is the stabilizer group generated by $S_1 = ZZI$ and $S_2 = ZIZ$. Then, elements of $\mathscr{N} - \mathscr{S}$ are $i^k \times \{XXX, YYX, YXY, XYY, ZII, IZI, IIZ, ZZZ, YYY, XXY, XYX, YXX\}$. Notice that if we take the group generated by the two operators XXX and ZII, each of these elements is equal to such a group member times an element of the stabilizer group. What does XXX do to our code space? Recall that the stabilizer subspace in this example is spanned by $|0_L\rangle = |000\rangle$ and $|1\rangle_L = |111\rangle$. Thus we see that $XXX|0_L\rangle = |1_L\rangle$, $XXX|1_L\rangle = |0_L\rangle$ and $ZII|0_L\rangle = |0_L\rangle$, $ZII|1_L\rangle = -|1_L\rangle$. In other words, XXX acts like an encoded X operation on the subspace and ZII acts like an encoded Z operation on the subspace. Similarly, one can calculate that YXX acts as Y on the code subspace. Also notice that these operators preserve the code subspace. Now, since all the other elements of the normalizer are either stabilizer elements or products of stabilizer elements, they will also preserve the code subspace.

Following this example, we are motivated to define the group $\mathscr{N}/\mathscr{S}$. This is the normalizer group quotient the stabilizer. We can write every element of $\mathscr{N}$ as RS, where S is a stabilizer element and R is not. Then, multiplication of these elements is like $R_1S_1R_2S_2 = (R_1R_2)(S_1S_2)$. This defines a multiplication rule for the R_is. This group is the group $\mathscr{N}/\mathscr{S}$. It is possible to show that this group is equal to the Pauli group of size $k = n - r$. This means that it is possible to generate this group by a set of Pauli operators $\bar{X}_1, \bar{Z}_1, \ldots, \bar{X}_k, \bar{Z}_k$ (along with a phase $i^k I$). These operators are the encoded Pauli operators on the subspace. (One thing to note is that this choice of division into k different qubits is not unique. However, we will rarely deal with the case where there is more than one encoded qubit, so this will not matter much for us.)

Elements of $\mathscr{N}$ represent nontrivial operations on the encoded subspace. They are, then, exactly the type of operators whose action we cannot detect on our QECC. Using this fact, it is possible to give a very nice characterization of the quantum error-correcting criterion. Suppose that we have a stabilizer $\mathscr{S}$ with normalizer $\mathscr{N}$. Let E_a denote a set of Pauli errors on the qubits. If $E_a^\dagger E_b \notin \mathscr{N} - \mathscr{S}$ for all possible error pairs, then E_a is a set of correctable errors. Why is this so? Well, there are two cases. One is that $E_a^\dagger E_b$ is in $\mathscr{S}$. Then $\langle\phi_i|E_a^\dagger E_b|\phi_j\rangle = \langle\phi_i|\phi_j\rangle = \delta_{ij}$ since $E_a^\dagger E_b$ acts trivially on the stabilizer subspace. The other case is that $E_a^\dagger E_b$ is not in the stabilizer and it is also not in the normalizer. This implies that there must exist an element of the stabilizer S such that $E_a^\dagger E_b S \neq S E_a^\dagger E_b$ (recall the centralizer and normalizer are the same for stabilizers). But all elements of the Pauli group either commute or anticommute. This implies that $E_a^\dagger E_b S = -S E_a^\dagger E_b$. By our previous argument this implies that $\langle\phi_i|E_a^\dagger E_b|\phi_j\rangle = 0$.

2.9.5 Examples of stabilizer codes

Here we present some examples of stabilizer codes.

2.9.5.1 Three-qubit bit-flip and phase-flip codes The example we have been dealing with, which is generated by ZZI and ZIZ, is able to correct a single bit-flip. This code is called the three-qubit bit-flip code. Its stabilizer and logical operators are

Element	Operator
S_1	ZZI
S_2	ZIZ
$\bar{X}$	XXX
$\bar{Z}$	ZII

Similarly, we can construct the code that is designed to correct single phase flip errors. Recall that this code was related to the bit-flip code by a Hadamard change of basis. This implies that its stabilizer and logical operators are

Element	Operator
S_1	XXI
S_2	XIX
$\bar{X}$	XII
$\bar{Z}$	ZZZ

Let us get on to codes that can correct single-qubit errors. We have already seen an example of such a single-qubit error-correcting code, the Shor code. This is an example of a Shor [[9,1,3]] quantum code. It turns out, since this code is really a concatenation of the bit-flip and phase-flip codes, that the Shor code is also a stabilizer code. In fact, we see that in the Shor code we use three bit-flip codes plus a single phase-flip code on the encoded information. From this it is easy to deduce what the stabilizer of the Shor code is.

Element	Operator
S_1	$ZZIIIIIII$
S_2	$ZIZIIIIII$
S_3	$IIIZZIIII$
S_4	$IIIZIZIII$
S_5	$IIIIIIZZI$
S_6	$IIIIIIZIZ$
S_7	$XXXXXXIII$
S_8	$XXXIIIXXX$
$\bar{X}$	$XXXXXXXXX$
$\bar{Z}$	$ZZZZZZZZZ$

Notice that this is a degenerate code for single-qubit errors: operators like $ZZIIIIIII$ act as identity on the code space even though they are products of two single-qubit errors.

The nine-qubit code that Shor came up with is a rather large code (although it has a certain beauty to it in its simplicity). What is the smallest code that can correct a single qubit error? This code is the five-qubit QECC [BDS+96, LMP+96]. It is specified by

Element	Operator
S_1	$XZZXI$
S_2	$IXZZX$
S_3	$XIXZZ$
S_4	$ZXIXZ$
$\bar{X}$	$XXXXX$
$\bar{Z}$	$ZZZZZ$

Notice that this code has stabilizer generators that are related to each other by a cyclic shift of the qubits. This makes it rather easy to remember its stabilizer. The five-qubit code is notable in that it is not an example of a CSS code. CSS codes have stabilizers that can be written as operators that consist of all X or all Z (and I) operators in an appropriate chosen computational basis. This complicates things for this code when we look at the issue of fault tolerance; so while this is a nice code, it is not as nice as we would like. The five-qubit code is a [[5,1,3]] code. It is useful to test your ability to spot when elements of the Pauli group commute or anticommute on the five-qubit code, since it has nontrivial stabilizer elements.

The [[7,1,3]] Steane code that we discussed above can be written in stabilizer form as follows:

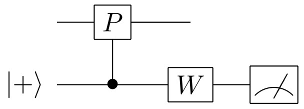
Element	Operator
S_1	$IIIXXXX$
S_2	$IXXIIXX$
S_3	$XIXIXIX$
S_4	$IIIZZZZ$
S_5	$IZZIIZZ$
S_6	$ZIZIZIZ$
$\bar{X}$	$XXXXXXX$
$\bar{Z}$	$ZZZZZZZ$

The Steane code is nice since it is a CSS code. We will use it in a lot of our examples since it has some very useful properties and is easy to discuss, but not so big as to prevent us writing down statements about it in explicit form.

2.9.6 *Measurement of Pauli group projectors*

Now that we have defined stabilizer codes, it is useful to discuss how to use them in practice. One essential task we will need to perform is to make a projective measurement onto the $+1$

and -1 eigenvalue eigenspaces of a general Pauli operator P that squares to identity, $P^2 = I$. How do we do this? Well, we have already basically seen a trick for doing this. Consider the following circuit:



(2.71)

What is the effect of this circuit? We can calculate that if our measurement obtains the result $|0\rangle$, then the measurement operator we are measuring on the first qubit is

$$M_0 = \langle 0|_B U|+\rangle_B = \frac{1}{2}(I+P), \tag{2.72}$$

and if the measurement outcome is $|1\rangle$, then the measurement operator is

$$M_1 = \langle 1|_B U|+\rangle_B = \frac{1}{2}(I-P). \tag{2.73}$$

Thus, we see that this circuit can be used to make a projective measurement onto the $+1$ and -1 eigenvalue eigenspaces of P. This is a very useful primitive for stabilizer codes. Notice that, unlike in our introduction to error correction codes, here in order to make a measurement we need to attach an ancilla qubit and will end up with the result of the measurement in this ancilla qubit. Further note that the measurement is not destructive on the original qubit, in the sense that while the measurement projects onto the subspaces, it leaves the resulting state in the appropriate subspace.

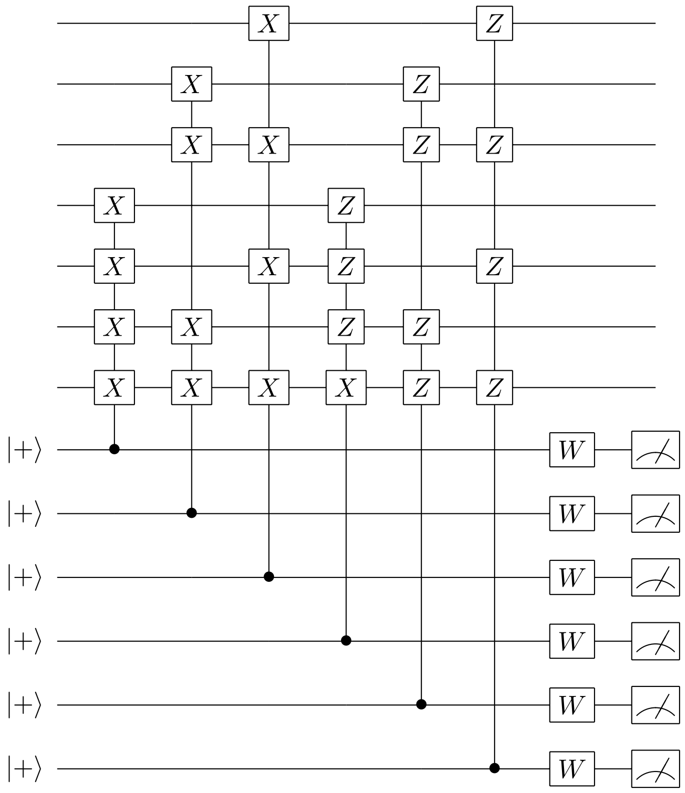
2.9.7 Error recovery routine for stabilizer codes

Given a stabilizer code, how do we perform a procedure for QEC?

Suppose that we have a set of Pauli errors E_a that are correctible via the error-correcting routine. Now we know that there must be a recovery routine for these errors. What is this recovery routine?

The first possibility for an error E_a is that it is in the stabilizer. In this case there is no need to perform QEC, since the effect of a stabilizer operator on our encoded quantum information is identity $E_a|\psi\rangle = |\psi\rangle$. So that case was easy! A second case is that E_a is not in the stabilizer. Now we know that E_a, since it satisfies the error-correcting criterion, must take us to an orthogonal subspace. What is this orthogonal subspace? Suppose that the stabilizer generators are $S_1, \ldots, S_r$. Then if E_a is a correctable error not in the stabilizer, it will anticommute with some of these generators S_i. For these generators, the state $E_a|\psi\rangle$ will no longer be in the $+1$ eigenvalue subspace, but will be in the -1 eigenvalue eigenspace. To see this, simply note that $S_i E_a|\psi\rangle = -E_a S_i|\psi\rangle = -E_a|\psi\rangle$ for $|\psi\rangle \in \mathscr{H}_S$. Further, if E_a commutes with S_i, then it does not change the eigenvalue of S_i. Thus, we see that the subspace the error sends us to is labeled by the ± 1 eigenvalues of the stabilizer operators. Thus, to perform QEC on a stabilizer code, it is enough to make measurements that project onto the ± 1 eigenvalue eigenspaces of the stabilizers. From this information, one can deduce what the error was and then apply an appropriate recovery operation. We call the values of the measured stabilizer generators the syndrome of the error. Now, having diagnosed what the error is by measuring the syndrome, one then applies the appropriate Pauli operator to reverse the error. One interesting fact is that, for degenerate codes, there are often multiple errors corresponding to a syndrome. In this case, however, one just reverses one of these errors and this guarantees that the net effect of this procedure is to either apply I or a stabilizer element, which is the same as applying I to the code space.

Now, how do we perform a measurement onto the ± 1 eigenvalue eigenspace of the S_i operators? We can measure them using the circuit described above for measuring Pauli operators. Another important fact to note is that, since the stabilizer generators all commute with each other, measurements that project onto their eigenstates can be simultaneously performed. Here, for example, is the syndrome measurement circuit for the [[7,1,3]] code:



(2.74)

The top seven qubits are the encoded qubits. The bottom six qubits are the ancilla registers that will hold the syndrome operators.

2.9.8 Preparation and measurement of stabilizer states

How do we prepare a stabilizer state? Well, this is rather easy. For example, suppose we just prepare the $|0^n\rangle$ state for our bare, unencoded qubits. Now, if we measure the syndrome and apply the error correcting recovery procedure, we are guaranteed to be in the code subspace. Now if, say, we have one encoded qubit, we can measure in the encoded $|0\rangle$, $|1\rangle$ basis by measuring the eigenvalues of the $\bar{Z}$ operator (using the tricks we learned above). If this outcome is $+1$ we have prepared the encoded $|0\rangle$ state. If this outcome is -1 then we have prepared the encoded $|1\rangle$ state and application of the $\bar{X}$ operator will then turn this into the $|0\rangle$ state. For multiple encoded qubits a similar procedure can be enacted. Thus, we have seen how to prepare encoded stabilizer states starting in, say, the encoded $+1$ eigenstates of the $\bar{Z}$ operator. Similar procedures apply for preparation of eigenstates of the other encoded Pauli operators. Thus, we have seen how to prepare these simple states.

What about more general states? Well, those will be harder. We will discuss some examples of these preparation procedures in Chapter 5, when we talk about fault-tolerant quantum computation.

And a further note. We could also ask how to take a generic unencoded qubit and then design a circuit for encoding into a stabilizer code. But this primitive – encoding into a code – is not a very useful one for our purposes (building a quantum computer). Why? Well, because we really never want to work with unencoded quantum computation. We always want to work with encoded quantum information. Every once in a while you will come across an argument against quantum computing that discusses a decoherence mechanism that always leads to errors on a single qubit. Such arguments miss the point, simply because we never work with quantum information in a single qubit: we always work with encoded quantum information. (The same is true for computing with classical faulty elements: that this is true was first pointed out by John von Neumann.)

Finally, we can ask how do we perform measurements on our stabilizer states. In particular, suppose we have k qubits and we want to measure in the encoded computational basis. Then we can do this by performing our Pauli measuring circuit for the encoded operators $\bar{Z}_i$. It is thus simple to see how to measure encoded Pauli operators on our qubits.

So we have seen how to prepare computational basis states, and measure computational basis states for our encoded information. The next obvious question is: how do we perform quantum computation, i.e., unitary gates, on our encoded quantum information? We will take this up in the next section.

2.9.9 Gates on stabilizer codes

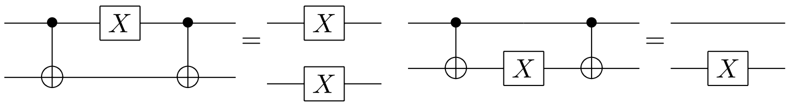
How do we perform unitary transforms on quantum information encoded into a stabilizer code? In some sense this question is trivial: just define unitary operations that carry out the desired unitary on the encoded subspace. On the other hand, this observation is not very useful for reasons that will become apparent when we talk about fault-tolerant quantum computation. Thus, here we will discuss a particularly beautiful and useful set of gates that we can implement on our stabilizer code.

Actually we have already seen some encoded operations on our code: the Pauli operators on our code. These operations were enacted on our encoded quantum information by the application of some Pauli group element. But in this section we will go beyond these simple gates.

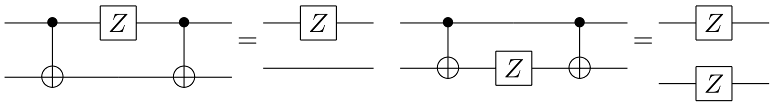
2.9.9.1 The Clifford group Consider our Pauli group on n qubits, $\mathscr{P}_n$. This is a subgroup of the unitary group on n qubits, $U(2^n)$. An important group, and we will see why in a second, is the Clifford group (in an abuse of mathematical nomenclature we will not even begin to try to undo). The Clifford group is the normalizer of the Pauli group in the unitary group on n qubits. In other words, it is the group of operators N that satisfy $NPN^\dagger \in \mathscr{P}_n$ for all $P \in \mathscr{P}_n$. What is an example of such an operation? Our good friend the Hadamard operator. In fact, we can easily show that

$$WIW^\dagger = I, \quad WXW^\dagger = Z, \quad WYW^\dagger = -Y, \quad WZW^\dagger = X. \tag{2.75}$$

Thus, a Hadamard on a single qubit is an operator in the normalizer of the Pauli group in $U(2^n)$. Indeed, multiple Hadamards are also in this normalizer. What are other examples? Well, elements of $\mathscr{P}_n$ themselves are obviously in the normalizer. Perhaps the first really interesting example beyond the Hadamard operator is the CNOT operation. The effect of the CNOT on Pauli operators is so useful we will write the action down here:



(2.76)



(2.77)

Note that we can deduce the action of conjugating a CNOT about Y operations by simply applying the X and Z conjugations in turn. Thus we see that the CNOT operation is indeed in the Clifford group.

A great question to ask is, what is needed from the Clifford group on n qubits to generate the entire group? In fact, the Clifford group on n qubits can be generated by W and the CNOT, C_X, and one extra gate,

$$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}. \tag{2.78}$$

Note that this gate acts as $SXS^\dagger = Y$, $SYS^\dagger = -X$, and $SZS^\dagger = Z$. A proof of this result can be found in [G96a].

2.9.9.2 Clifford group gates on stabilizer codes Suppose that we have a stabilizer code generated by the stabilizer operators $S_1, \ldots, S_r$. We saw that elements of the normalizer of the stabilizer group in the Pauli group were logical operators on our stabilizer code. What if we now examine the normalizer of the stabilizer group, not just in the Pauli group, but now in the full unitary group $U(2^n)$. These gates will certainly be in the Clifford group, since the stabilizer group is made up of Pauli group operators. But now these operations might perform a nontrivial operation on our encoded qubits.

Let us look at an example. Consider the seven qubit Steane code with generators specified above. Next consider the operator $W^{\otimes 7}$. This operator when conjugated about stabilizer elements will produce another stabilizer element for the Steane code. How do we see this? Recall that $WXW = Z$ and vice versa. Now notice that if we conjugate $W^{\otimes 7}$ about one of the generators that is made up completely of X operations, then we obtain a new operator that is made up completely of Z operations. But now notice that there is a symmetry of the Steane code that guarantees that, for the generators of the stabilizer made up completely of X operations, there is an equivalent generator formed by replacing the X operations with Z operations. Thus we see that, on the generators of the stabilizer code, conjugating by $W^{\otimes 7}$ produces a stabilizer generator. Since all stabilizer elements are made as products of stabilizer generators, this implies that the stabilizer group is preserved under conjugation by $W^{\otimes 7}$.

So now the question arises as to what $W^{\otimes 7}$ does on our logical encoded qubit? Recall that the logical operators on the Steane code are $\bar{X} = X^{\otimes 7}$ and $\bar{Z} = Z^{\otimes 7}$. We can then see that

$W^{\otimes 7}\bar{X}W^{\otimes 7} = \bar{Z}$ and $W^{\otimes 7}\bar{Z}W^{\otimes 7} = \bar{X}$. Thus, we see that $W^{\otimes 7}$ acts in the same way on the encoded quantum gates as a single W does on a single Pauli. In fact, this is enough to imply that $W^{\otimes 7}$ is exactly the Hadamard gate on our encoded quantum information.

More generally, then, we can consider elements of the Clifford group that, when conjugated about the stabilizer generators, produce elements of the stabilizer. Such elements act to preserve the stabilizer subspace, since $NS_iN^\dagger = S_j$, and therefore $N|\psi_C\rangle = NS_i|\psi_C\rangle = NS_iN^\dagger N|\psi_C\rangle = S_jN|\psi_C\rangle$. Thus, $S_j(N|\psi_C\rangle) = N|\psi_C\rangle$, so again $N|\psi_C\rangle$ is still stabilized (further note that $NS_iN^\dagger$ preserves the group multiplication rule of the stabilizer and thus $NS_iN^\dagger$ if N is in the normalizer are again generators of the stabilizer group). Further, these gates also act on the encoded qubits as elements of the Clifford group on these encoded qubits.

It is important to note that, when it comes to Clifford group elements, not all stabilizer codes were created equal. In particular, implementing different Clifford group elements is often much more difficult on some codes than on other codes. This fact is one reason for choosing different stabilizer codes.

We have seen in this section how to perform some Clifford group elements on stabilizer codes. Even if a code supports all Clifford group operations on our code this is not enough to perform universal quantum computation on the code. Universal quantum computation is discussed much further in Chapter 5 on fault-tolerant quantum computation.

2.10 Conclusions

In this chapter we have defined the basics of QECCs. We have seen how it is possible to protect quantum information by suitably encoding the quantum information across multiple independently erred quantum systems. Since the origin of the destruction of quantum information by decoherence is often the effect of a system becoming entangled with its environment, the QEC procedure has been dubbed by John Preskill as "fighting entanglement with entanglement." We have shown that QECCs exist and derived the necessary and sufficient condition for such codes to succeed in correcting a set of errors. A class of QECCs derived from self-dual classical codes, the CSS codes, was described. A more general set of codes was then derived using the stabilizer formalism. Stabilizer codes are the bread and butter of many ideas for QEC in the real world, and are distinguished by the relatively easy formalism with which one can discuss these codes. However, one should note that the theory of QEC contains many more important codes than just stabilizer codes, and the development of "good" codes is an important ongoing endeavor (where good depends on the color of the crystal through which you look, i.e., the problem you wish to solve). Many of these codes will described in later chapters in this book.

2.11 History and further reading

Classical error correction has a long and important history, with a seminal piece of work being Shannon's paper founding the study of information theory [S48]. A good introduction to information theory and error-correction is the book of Cover and Thomas [CT91]. Quantum error-correcting codes were first discovered by Shor [S95] and Steane [S96c] in 1995 and 1996. Knill and Laflamme were the first to derive the necessary and sufficient quantum error-correcting criterion [KL97]. The five-qubit error-correcting code was discovered by Bennett *et al.* [BDS+96] and Laflamme *et al.* [LMP+96] in 1996. Stabilizer codes and a closely related approach using codes over $GF(4)$ were introduced by Gottesman [G96a] and by Calderbank *et al.* [CRS+98] . Gottesman's Ph.D. thesis on stabilizer codes is very readable and a good introduction to these codes [G96a]. The quantum Hamming bound was introduced by Ekert and Macchiavello [EM96] and shown to be achievable by Gottesman [G96a]. Gentle introductions to quantum computing, which include the basics of QEC, include the classic textbook of Nielsen and Chuang [NC00] and the more introductory books by Mermin [M07] and Loepp and Wootters [LW06].