CSS Code Focused Quantum Error Correction

Rohan Wassink and Noah Silverberg

1 Description of problem/fundamental claim

1.1 Introduction

Noise is a prevalent issue in both classical and quantum information systems. When designing such systems, one attempts to minimize noise sources. However, despite best efforts, noise is an ineluctable part of any information system, so it is our job to minimize its effects. With classical noise, the only possible error that can occur is a bit flip error, which flips $0 \to 1$ or $1 \to 0$. However, with quantum noise, a continuous range of errors can occur. For example, we can have the bit flip error (e.g., $|0\rangle \to |1\rangle$), or we can have an arbitrary X rotation (e.g., $|0\rangle \to X_\theta |0\rangle$ for any $\theta \in \mathbb{R}$).

One important distinction between classical and quantum error correction, of course, is that measuring to see if an error has occurred can cause state collapse. Therefore, we need to take care when designing quantum error correction algorithms to ensure that we do not alter the state of a qubit when diagnosing it for errors. Similarly, we need to ensure that we do not cause state collapse when correcting errors.

1.2 Classical Error Correction

One underlying principle of error correction is redundancy. To gain intuition for this, we can consider a simple example of classical error correction. Let us assume we are given 1 classical bit of information (either 1 or 0). We then send that bit through a potentially noisy channel with a probability p of a bit flip error. Whichever message we receive, we assume that was the original message sent (we assume no error has occurred when decoding the message). Then, there is a probability p that we incorrectly decode the message.

Now let us consider what happens if we use a very basic error correction technique. We can encode the bit into 3 bits as follows:

$$0 \to 000$$

$$1 \to 111$$

If we receive 000, we know the original message was 0, and if we receive 111, we know the original message was 1. Now, let us consider what happens if we have a probability p of a bit flip error occurring in our encoding. There are 3

possible single-bit errors, so there is a probability $3p(1-p)^2$ of a single-bit error. There are 3 possible double-bit errors, so there is a $3p^2(1-p)$ probability of a double-bit error. Then there is a p^3 probability of all three bits flipping.

Let us decode the message as follows: if there are more 1s than 0s in our received message, we interpret it as a 1; if there are more 0s than 1s, we interpret it as a 0. Therefore, we would only incorrectly interpret the message if there are two or three bit flip errors, which has a probability of $3p^2(1-p)+p^3 = 3p^2-2p^3$. If we compare that to our probability p of incorrectly interpreting the message without any error correction, we find that we improve our odds of correctly interpreting the message when $3p^2 - 2p^3 < p$, or when $0 < p < \frac{1}{2}$. Therefore, for $p \ll 1$, we find that classical error correction can improve the accuracy of our communication channels.

1.3 Quantum Error Correction

When we try to do the same repetitive encoding with quantum information, however, we run into a few issues:

1. No-cloning theorem: we cannot duplicate quantum states as we did above, because the no-cloning theorem dictates that we cannot use a unitary gate to duplicate a quantum state.

2. Continuous range of errors: as mentioned above, the errors in quantum states are not only bit flip errors. There are infinitely many possible errors that can occur, which means that our quantum error correction techniques must be more clever.

3. State collapse after measurement: as mentioned above, diagnosing errors in quantum states can cause state collapse, so we need to find ways to find out when an error has occurred without destroying any information via state collapse.

However, we can find ways to circumvent these issues. Let us explore an equivalent quantum method of error correction to the classical one given above. Let us begin with an arbitrary state $|\varphi\rangle$. First, we can encode the state by starting with two ancilla bits prepared in the state $|0\rangle$ then applying CNOT to both of them, with $|\varphi\rangle$ being the control bit. So we have the state

$$\text{CNOT}\,|0\,0\,\varphi\rangle$$

where

$$\begin{aligned}\text{CNOT} = {} & |000\rangle\langle 000| + |111\rangle\langle 001| + |010\rangle\langle 010| + |101\rangle\langle 011| \\ & + |100\rangle\langle 100| + |011\rangle\langle 101| + |110\rangle\langle 110| + |001\rangle\langle 111|\end{aligned}$$

is unitary. And if we write $|\varphi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, we have

$$\text{CNOT}\,[|00\rangle \otimes (\alpha\,|0\rangle + \beta\,|1\rangle)] = \alpha\,|000\rangle + \beta\,|111\rangle$$

So, essentially we map $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle \rightarrow \alpha|000\rangle + \beta|111\rangle$ by using two ancilla bits prepared in the state $|0\rangle$ and a CNOT gate. So, we have gotten around the issue of the no-cloning theorem - we added redundancy without technically cloning the original state. The states $|000\rangle$ and $|111\rangle$ are referred to as the logical $|0\rangle$ and $|1\rangle$ states, respectively. And the states $|0\rangle$ and $|1\rangle$ are referred to as the physical states.

Let us only look at bit flip errors again. Of course, as noted above, errors in quantum states can be continuous, but let's only look at bit fip errors so we have a direct comparison to the classical error correction procedure described above.

There are two things we must accomplish: (1) figuring out which qubit has been flipped, and (2) fixing the qubit by flipping it back.

For (1), we have 4 possible results assuming only a single qubit or no qubits have been flipped:

1. There are no bit flips
2. There is a bit flip on qubit 0
3. There is a bit flip on qubit 1
4. There is a bit flip on qubit 2

where we have number the qubits from right to left as 0, 1, and 2 (so, for example, the qubit in state $|1\rangle$ of the three-qubit state $|001\rangle$ would be qubit 0). To figure out which of the four results we have, we need to measure 2 classical bits of information. There are many ways to diagnose which error syndrome has occurred, but here is one such way.

1. Prepare two ancilla bits in the state $|0\rangle$, so you have the state $|00\rangle$
2. Apply a CNOT gate to ancilla qubit 0, with qubit 0 of the encoded state being the control bit
3. Apply a CNOT gate to ancilla qubit 0, with qubit 1 of the encoded state being the control bit
4. Apply a CNOT gate to ancilla qubit 1, with qubit 1 of the encoded state being the control bit
5. Apply a CNOT gate to ancilla qubit 1, with qubit 2 of the encoded state being the control bit

Then measure the state of each of the two ancilla qubits using the Z operator (see Fig. 1 for the IBM quantum circuit corresponding to the encoding and error diagnosis for a single bit flip). Then we interpret the measurements as follows, under the assumption that either no error has occurred or only a single bit flip:

1. If we measure that the ancilla qubits are in the state $|00\rangle$, then all 3 qubits of our encoded state are in the same state, so no bit flip error has occurred

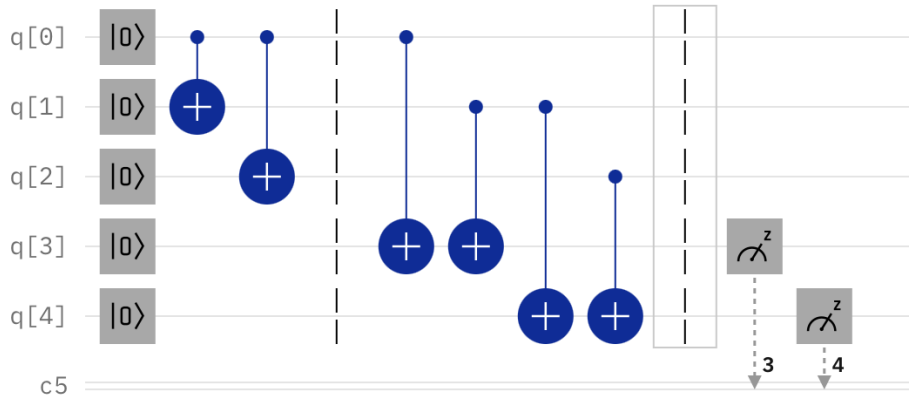

Figure 1: IBM Quantum Composer circuit for the encoding and error syndrome diagnosis of a single bit flip error.

2. If we measure that the ancilla qubits are in the state $|01\rangle$, then the qubit 0 of our encoded state has been flipped

3. If we measure that the ancilla qubits are in the state $|10\rangle$, then the qubit 2 of our encoded state has been flipped

4. If we measure that the ancilla qubits are in the state $|11\rangle$, then the qubit 1 of our encoded state has been flipped

So our error diagnosis is complete. Now, we need to correct the error, which involves simply applying an X gate (or NOT) to whichever qubit has been flipped (if any). If no qubit has been flipped, we don't have to do anything.

Note that our measurements for the error diagnosis did not cause state collapse since we only measured the parity between pairs of qubits instead of measuring the states of single qubits, so no information was lost. Note also that another way to do error diagnosis would be to measure Z_0Z_1 and Z_1Z_2 and compare the results - we will use this idea later for the Shor code.

Our error correction procedure described above only works if no qubits or a single qubit have/has been flipped. If, for example, qubits 0 and 1 of our encoded state were flipped, we would mesure the state $|10\rangle$ for our ancilla bits, which would cause us to incorrectly believe that qubit 2 our our encoded state was flipped. Then we would flip qubit 2 to correct our ostensible error. This would mean that we would end with the state $\alpha|111\rangle + \beta|000\rangle$, which is equivalent then to a bit flip error on all 3 qubits.

Let the probability of a bit flip error be p. Then the probability of 2 bit flip errors occurring is $3p^2(1-p)$, and the probability of 3 bit flips occurring is p^3, so the probability of our message being misinterpreted is $3p^2(1-p)+p^3 = 3p^2-2p^3$, which is identical to that of the classical error correction procedure above, so

again our odds of correctly interpreting the state are incresed by our error correction procedure if $0 < p < \frac{1}{2}$.

We were able to circumvent two of the three problems we identified that are associated with quantum error correction that are not present in classical error correction. We still haven't discussed how to deal with the continuous range of possible errors, however.

1.4 Continuous Errors on a Qubit

For example, instead of having a bit flip error on $|0\rangle$ or $|1\rangle$, which is equivalent to applying X, we could apply Z, which would have no effect on $|0\rangle$, but would turn $|1\rangle$ into $-|1\rangle$. This may not seem to be a big deal, but consider what happens if you apply Z to $|+\rangle$ and $|-\rangle$. We know that $Z|+\rangle = |-\rangle$ and $Z|-\rangle = |+\rangle$, so Z acts as a bit flip operator for the sates $|+\rangle$ and $|-\rangle$, whereas X has no effect on $|+\rangle$ and turns $|-\rangle$ into $-|-\rangle$.

Let us look at an example of a quantum error correction algorithm that can deal with arbitrary errors. Let us assume that a single qubit of an encoded state has an arbitrary error, which is of the form

$$U = c_0 I + c_1 X + c_2 Y + c_3 Z$$

for arbitrary complex amplitudes c_0, c_1, c_2, c_3. Note that U can map any state on the Bloch sphere to any other state, so U can represent any arbitrary error on a single qubit. Note also that

$$iXZ = i \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = Y$$

so we can re-write U as

$$U = c_0 I + c_1 X + c_2 XZ + c_3 Z$$

for arbitrary c_0, c_1, c_2, c_3. In other words, any error on a single qubit can be written as a combination of I, bit flip errors (X), phase flip errors (Z), and both bit and phase flip errors (XZ). So we want to make a quantum error correction algorithm that can solve X, Z, and XZ errors. That is what the 5-qubit code can do, which is described below.

1.5 5-Qubit Code

The 5-qubit code is an algorithm that encodes a single qubit message using 5 physical qubits. It can correct any arbitrary error on a single qubit - which, again, are bit flip (X) errors, phase flip (Z) errors, and combinations of the two (XZ). It can be shown that 5 qubits is the smallest number that can correct against an arbitrary single qubit error, so this is optimal, in the sense that it doesn't require redundant information. However, there are other codes/algorithms that are more resilient, in a sense. Redundancy is the most

important part of error correction and, in fact, the only reason it is possible, so more redundancy allows for more reliable codes. Nonetheless, the 5-qubit code is a useful case study since it is the lest redundant single qubit error detection/correction code.

Let us again start with a qubit in the state $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$. We encode the qubit as follows

$$\begin{aligned}
|0\rangle \rightarrow \frac{1}{4}[&|00000\rangle + |10010\rangle + |01001\rangle + |10100\rangle + |00101\rangle + |01010\rangle \\
&- |11011\rangle - |00110\rangle - |11000\rangle - |11101\rangle - |00011\rangle - |11110\rangle \\
&- |01111\rangle - |10001\rangle - |01100\rangle - |10111\rangle] \\
|1\rangle \rightarrow \frac{1}{4}[&|11111\rangle + |01101\rangle + |10110\rangle + |01011\rangle + |11010\rangle + |10101\rangle \\
&- |00100\rangle - |11001\rangle - |00111\rangle - |00010\rangle - |11100\rangle - |00001\rangle \\
&- |10000\rangle - |01110\rangle - |10011\rangle - |01000\rangle]
\end{aligned}$$

And we measure the following operators: $XZZXI, IXZZX, XIXZZ, ZXIXZ$. The error diagnosis is not as obvious for this algorithm. These operators are not as simple as the ones for the bit flip code or the Shor code (to be described later), so it's helpful to use the following method to calculate the measurement results. Suppose you have an error U and an operator σ. Then if $U\sigma = \sigma U$, the measurement result of σ after error U is applied to the encoded state will be 1. If $U\sigma \neq \sigma U$, then the measurement result will be -1. An example calculation is shown for the error X_0 with the four operators. We check for equality by seeing if $U\sigma - \sigma U = 0$.

$$\begin{aligned}
(IIIIX)(XZZXI) - (XZZXI)(IIIIX) &= XZZXX - XZZXX = 0 \\
(IIIIX)(IXZZX) - (IXZZX)(IIIIX) &= IXZZI - IXZZI = 0 \\
(IIIIX)(XIXZZ) - (XIXZZ)(IIIIX) &= -iXIXZY - iXIXZY \neq 0 \\
(IIIIX)(ZXIXZ) - (ZXIXZ)(IIIIX) &= -iZXIXY - iZXIXY \neq 0
\end{aligned}$$

So for X_0, we get the measurement results $1, 1, -1, -1$.
All of the errors and their measurement results are shown in the table below

Error	$XZZXI$	$IXZZX$	$XIXZZ$	$ZXIXZ$
X_0	1	1	-1	-1
X_1	1	-1	-1	1
X_2	-1	-1	1	1
X_3	-1	1	1	1
X_4	1	1	1	-1
Z_0	1	-1	1	1
Z_1	-1	1	1	-1
Z_2	1	1	-1	1
Z_3	1	-1	1	-1
Z_4	-1	1	-1	1
X_0Z_0	1	-1	-1	-1
X_1Z_1	-1	-1	-1	-1
X_2Z_2	-1	-1	-1	1
X_3Z_3	-1	-1	1	-1
X_4Z_4	-1	1	-1	-1

One can see that there are no repetitions in the table, so each error maps to a unique error syndrome measurement, which is what we want. Then, as always, we correct the error by applying the appropriate unitary gate (X_0, for example, to correct an X_0 error).

Notice that the 5-qubit code isn't simply a bit flip code and a phase flip code combined. However, it still is able to identify bit flips, phase flips, and combinations of the two, so it can detect and correct any arbitrary single qubit error, as desired. But it doesn't separate the detection of bit flip errors from phase flip errors.

2 Motivation

There is a class of codes, unlike the 5-qubit code, that are combinations of a bit flip code and a phase flip code, however. These are known as the Calderbank-Shor-Steane (CSS) codes.

One major benefit of these CSS codes is that of code concatenation. This term refers to the ability of CSS code architectures to combine correction mechanisms for different types of errors. For instance, the nine qubit Shor code (described later) is a CSS code that concatenates an encoding for bit flip errors and an encoding for phase flip errors. The concatenated structure of this code allows it to be simple and intuitive, as the division between the bit flip and phase flip portions of the circuit allow the user to immediately determine which function is performed where.

In general, the relative simplicity of CSS codes serves as a second major benefit after concatenation. For instance, knowledge of topology is not required to understand the nine qubit Shor code, although it is indeed integral to the toric code architecture. Additionally, non-CSS error correction codes may not have the property in which different types of errors can be corrected in separate steps.

Again, another benefit of CSS codes is that they can correct X and Z errors independently, which allows them to also correct XZ errors, so they can correct for any arbitrary error.

CSS codes are also robust since if the identification/correction of an X error, for example, will not affect the detection/correction of a Z error. In other words, CSS codes are fault-tolerant.

They are also generally intuitive since they can be decomposed into the bit flip code and the phase flip code.

2.1 The Nine Qubit Shor Code

Before we formalize what exactly a CSS code is, an example is the nine qubit Shor code, which encodes a single qubit into a nine qubit state. Notice that the Shor code, as said earlier, combines a bit flip code with a phase flip code.

Let us again start with a qubit in the state $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$. There are two encoding steps. First, we encode for phase flips (aka Z) using the encoding

$$|0\rangle \to |+++\rangle$$

$$|1\rangle \to |---\rangle$$

Then we encode for bit flips (aka X) using the encoding

$$|0\rangle \to |000\rangle$$

$$|1\rangle \to |111\rangle$$

from earlier. So our encoding of the state $|\varphi\rangle$ is shown below:

$$\begin{aligned}
|\varphi\rangle &= \alpha|0\rangle + \beta|1\rangle \\
&\to \alpha|+++\rangle + \beta|---\rangle \quad \text{(phase flip encoding)} \\
&= \alpha\left[\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right]^{\otimes 3} + \beta\left[\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right]^{\otimes 3} \\
&\to \alpha\left[\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)\right]^{\otimes 3} + \beta\left[\frac{1}{\sqrt{2}}(|000\rangle - |111\rangle)\right]^{\otimes 3} \quad \text{(bit flip encoding)} \\
&= \frac{1}{2\sqrt{2}}\left[\alpha(|000\rangle + |111\rangle)^{\otimes 3} + \beta(|000\rangle + |111\rangle)^{\otimes 3}\right].
\end{aligned}$$

We can accomplish this as shown in Fig. 2 on the next page.

We can do the phase flip encoding by doing the bit flip encoding described earlier using the CNOT gate, then applying the Hadamarad gate (H) to all three qubits. Then the bit flip encoding can be done in the same way as before, but this time with the three phase flip encoded qubits as the control bits and six ancilla bits in $|0\rangle$ instead of a single control bit and two ancilla bits. So our encoded state has nine quibts instead of three like earlier. Let us write out our

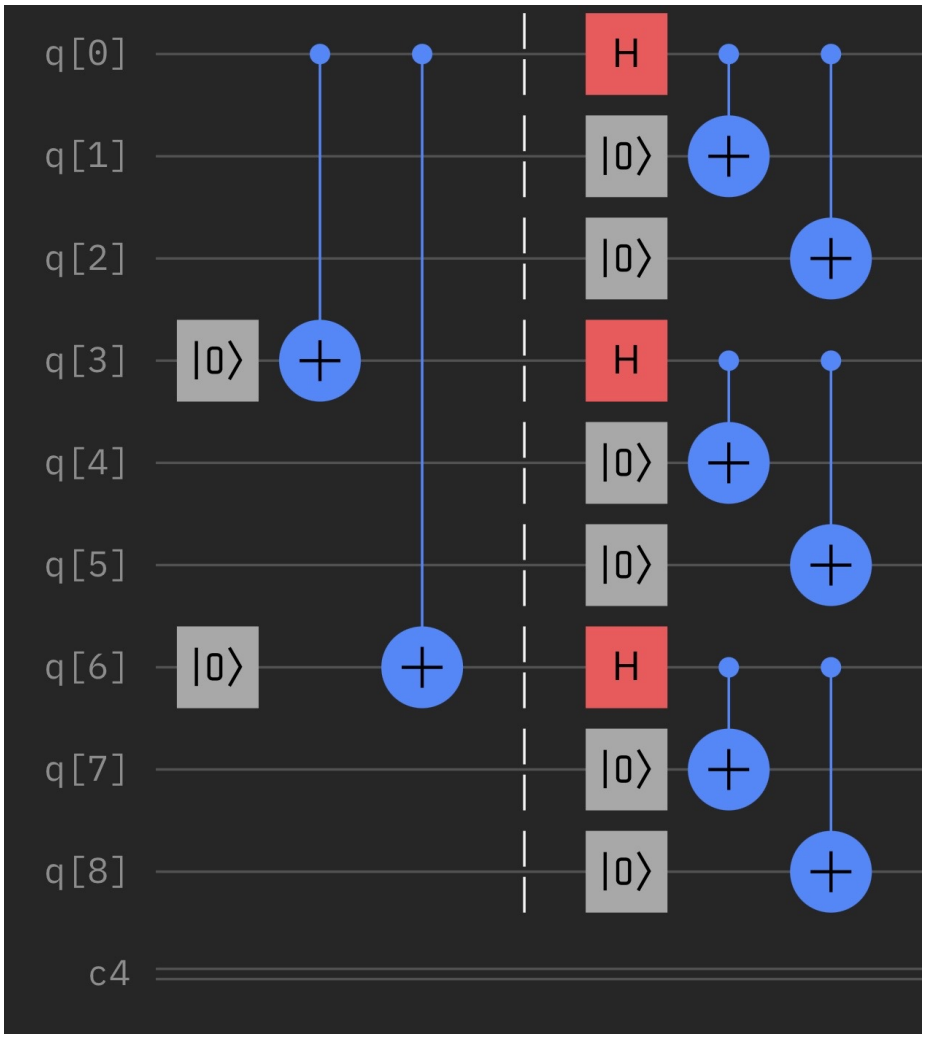

Figure 2: Encoding of the nine-qubit Shor code in a quantum circuit. The portion to the left of the dotted barrier encodes for bit flip errors, while the portion to the right of the dotted barrier encodes each of the bit flip qubits to account for phase flip errors on each one.

encoded state using all nine qubits so we can more easily see what exactly our state is and so we can more easily reference each qubit.

$$|\varphi\rangle \rightarrow \frac{1}{2\sqrt{2}}\left[\alpha(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)\right.$$
$$\left. + \beta(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)\right]$$

Let us number the qubits as follows: the rightmost qubit is 0, and the leftmost

quit is 8. So, for example, the rightmost qubit of the leftmost $(|000\rangle - |111\rangle)$ would be numbered 6.

Now let us assume that an arbitrary error U (described above) has been applied to a single qubit. Let's first check for a phase flip. Notice that, for example

$$Z_i \left[\frac{1}{2\sqrt{2}}(|000\rangle + |111\rangle) \right] = \frac{1}{2\sqrt{2}}(|000\rangle - |111\rangle)$$

for $i \in \{0, 1, 2\}$. In other words, a phase flip applied to any of the qubits in the three-qubit state is equivalent to a phase flip on any of the other qubits in the state. So, in our encoded state for $|\varphi\rangle$, we treat every three qubits (0,1,2 and 3,4,5 and 6,7,8) as a group which we can diagnose and correct together. So if we measure the phase of the three groups using the XXX operator, we can diagnose which group has a phase flip and correct it. So, let us measure the operators $X_0X_1X_2X_3X_4X_5$ and $X_3X_4X_5X_6X_7X_8$. Then we interpret the results as follows

1. If $X_0X_1X_2X_3X_4X_5 = X_3X_4X_5X_6X_7X_8 = 1$, then no phase flip error has occurred
2. If $X_0X_1X_2X_3X_4X_5 = 1$ and $X_3X_4X_5X_6X_7X_8 = -1$, then a phase flip error has occurred in the third group (qubits 6,7,8)
3. If $X_0X_1X_2X_3X_4X_5 = -1$ and $X_3X_4X_5X_6X_7X_8 = 1$, then a phase flip error has occurred in the first group (qubits 0,1,2)
4. If $X_0X_1X_2X_3X_4X_5 = X_3X_4X_5X_6X_7X_8 = -1$, then a phase flip error has occurred in the second group (qubits 3,4,5)

If no phase flip error has occurred, obviously no error correction is necessary. If a group has a phase flip, then we apply the operator ZZZ to that group of qubits to rectify the error. Again, note that the error syndrome measurements do not cause state collapse, so we do not lose any information.

Now we want to see if a bit flip error has occurred. To do so, we can measure ZZ on the first two qubits in each group and ZZ on the second two qubits in each group. In other words, we measure Z_0Z_1, Z_1Z_2 and Z_3Z_4, Z_4Z_5 and Z_6Z_7, Z_7Z_8. Then we interpret the results as follows (we will only look at the first group - the analysis is the same for the other ones)

1. If $Z_0Z_1 = Z_1Z_2 = 1$, then no bit flip error has occurred
2. If $Z_0Z_1 = 1$ and $Z_1Z_2 = -1$, then a bit flip error has occurred on qubit 2
3. If $Z_0Z_1 = -1$ and $Z_1Z_2 = 1$, then a bit flip error has occurred on qubit 0
4. If $Z_0Z_1 = Z_1Z_2 = -1$, then a bit flip error has occurred on qubit 1

(Note that this an equivalent method of measuring for a bit flip error as described earlier using 2 ancilla qubits and CNOT gates.) Then, we apply X to whichever bit (if any) has a bit flip error to correct it.

One important part of our procedure is that the detection/correction of phase flip errors has no effect on our detection/correction of bit flip errors. Let us look at, for example

$$|\varphi_0\rangle = \frac{1}{\sqrt{2}}|000\rangle + |111\rangle \quad \text{and} \quad |\varphi_1\rangle = Z|\varphi_0\rangle = \frac{1}{\sqrt{2}}|000\rangle - |111\rangle$$

So $|\varphi_1\rangle$ is a phase flipped version of $|\varphi_0\rangle$. If we measure Z_0Z_1 and Z_1Z_2 on both states, we get the same results (and any other pairs of 2-qubit states where one is simply a phase flipped version of the other would also have the same property).

The fact that our bit flip section of the Shor code is unaffected by the phase flip section means that we can correct also for XZ (or both phase and bit flip) errors. As explained above, since any U can be expressed as a combination of I, X, Z, XZ operators, this means that the Shor code can protect us against any arbitary error U.

The Shor code requires 2 bits of information for phase flip error detection and correction, and 6 bits of information for bit flip error detection and correction. So overall we used 8 bits of information for error detection and correction. One consideration is that the Shor code protects against single qubit errors - specifically, there are three types of errors (X, Z, XZ), which can occur on any of the nine qubits, so there are $3 \times 9 = 27$ different states post-error. And there is 1 state that is due to no error, so there are 28 states that can be detected by the Shor code. Although, 6 of these have identical error syndrome detection results and correction (since within any group, a phase flip on any of the three qubits is indistinguishable from a phase flip on any of the other qubits within the group), so the Shore code only essentially looks at $28 - 6 = 22$ different states, which should require $\log_2 22 \approx 4.459$ bits of information, so we should be able to use 5 bits to diagnose errors using the Shor code.

Note that the Shor code is a CSS code because it combines the completely independent bit flip and phase flip error correction processes. This can be realized mathematically by the fact that the measurement operators all commute. Now that we have seen an example of a CSS code and seen how the combination of bit flip code and a phase flip code allows for us to intuitively solve both, we are ready to formalize our definition of a CSS code.

3 Background and Derivations (Formalism)

3.1 Code Spaces

We will now introduce the concept of code spaces, which allow messages to be encoded so that different types of errors can be corrected. Suppose the goal is to construct a system that encodes length k bit string messages represented as

$$\vec{m} = \begin{pmatrix} m_1 & m_2 & \dots & m_k \end{pmatrix}$$

in vectors of length n represented as

$$\vec{c} = \begin{pmatrix} c_1 & c_2 & \dots & c_n \end{pmatrix},$$

where $n > k$.

Now there exists a generator matrix operator G that encodes a given message $\vec{m}$ as a length n vector $\vec{c}$. We can describe the mapping $G : M \to C$ from the message space M to the code space C as follows:

$$G = \begin{pmatrix} g_{1,1} & g_{1,2} & \cdots & g_{1,n} \\ g_{2,1} & g_{2,2} & \cdots & g_{2,n} \\ \vdots & & & \vdots \\ g_{k,1} & g_{k,2} & \cdots & g_{k,n} \end{pmatrix}$$

where the rows of G are a set of k linearly independent orthogonal (with respect to the mod 2 dot product) basis vectors of $\{0,1\}^n$, $M = \{\vec{m} : \vec{m} \in \{0,1\}^k\}$ is the space of messages, and $C = \{\vec{c} : \vec{c} \in \{0,1\}^n, \vec{c} = \vec{m}G \text{ for some } \vec{m} \in M\}$ is the code space. Now, if the set of orthogonal basis vectors for $\{0,1\}^n$ being used is $\beta_G = \{\vec{g_i} : i \in \{1,2,\ldots,n\}\}$, G can be rewritten as

$$G = \begin{pmatrix} \vec{g_1} \\ \vec{g_2} \\ \vdots \\ \vec{g_k} \end{pmatrix}.$$

Thus, for a given encoding $\vec{c}$,

$$\begin{aligned} \vec{c} &= \vec{m}G \\ &= \sum_{j=1}^{n} \left(\sum_{i=1}^{k} m_i g_{i,j} \right) e_j \\ &= \sum_{i=1}^{k} \left(\sum_{j=1}^{n} m_i g_{i,j} e_j \right) \\ &= \sum_{i=1}^{k} m_i \left(\sum_{j=1}^{n} g_{i,j} e_j \right) \\ &= \sum_{i=1}^{k} m_i \vec{g_i}, \end{aligned}$$

where e_i represents the ith standard basis vector of $\mathbb{R}^n$.

Using this same notation, if we define

$$H = \begin{pmatrix} \vec{g}_{k+1} \\ \vec{g}_{k+2} \\ \vdots \\ \vec{g}_n \end{pmatrix},$$

we observe that if $(\vec{c})' = (\vec{c})^T$ is the vertical representation of $\vec{c}$, then

$$H(\vec{c})' = \begin{pmatrix} \left(\sum_{i=1}^{k} m_i \vec{g_i}\right) \cdot \vec{g}_{k+1} \\ \left(\sum_{i=1}^{k} m_i \vec{g_i}\right) \cdot \vec{g}_{k+2} \\ \vdots \\ \left(\sum_{i=1}^{k} m_i \vec{g_i}\right) \cdot \vec{g}_n \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{k} (m_i \vec{g_i} \cdot \vec{g}_{k+1}) \\ \sum_{i=1}^{k} (m_i \vec{g_i} \cdot \vec{g}_{k+2}) \\ \vdots \\ \sum_{i=1}^{k} (m_i \vec{g_i} \cdot \vec{g}_n) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \vec{0}, \quad (1)$$

since all $\vec{g}_i$ are orthogonal basis vectors for $\{0,1\}^n$. Note that $H(\vec{v})' = \vec{0}$ will be true if and only if $\vec{v} \in C$. The matrix H is referred to as a parity check matrix, since it may be used to check whether a message-encoding vector has been transformed to a vector outside of code space C by some error (this transformation is indicated by the result $H(\vec{v})' \neq \vec{0}$). The vector $H(\vec{v})'$ is the syndrome for vector v.

Now define the weight of a binary bit string vector $\vec{v}$ to be the number of 1s in the string, and let the distance d of code space C be the minimum of the weights of all vector encodings of messages $\vec{m} \in \{0,1\}^k$. We show that if $d = 2p + 1$, then the number of bit flip errors in a vector $\vec{v} \in \{0,1\}^n$ that the coding space C can distinguish is p.

Proof. Consider two bit flip errors $e_1, e_2 \in \{0,1\}^n$ with weights $w_1, w_2 \leq p$, where applying bit flip error e to encoding $\vec{c}$ results in $e \oplus \vec{v}$. The code fails to distinguish between e_1 and e_2 if the two errors have the same syndrome (i.e. $He_1 = He_2$), meaning $\vec{0} = He_1 \oplus He_2 = H(e_1 \oplus e_2)$. Since $H(e_1 \oplus e_2) = 0$, from (1) we conclude that $e_1 \oplus e_2 \in C$. However, this cannot be true, since $w_{e_1 \oplus e_2} \leq 2p$, which is below the minimum weight of all vectors in C, namely $d = 2p + 1$. Therefore, if $w_1, w_2 \leq p$, then $e_1 \oplus e_2 \notin C$ and therefore C distinguishes between e_1 and e_2 and the errors are correctable. □

The intuition behind this is as follows. The distance d of a code space C can also be interpreted as the minimum number of bit flips required to convert between two vectors in C (one can see this since $\vec{0}$ must be in C since it is a vector space, and the weight of a vector is also equal to its distance from $\vec{0}$). So if we have $d = 2p + 1$ and have $p + 1$ or more bit flips from a given code in C, the erroneous code can be closer to a different code than the original, so we wouldn't be able to reliably tell which code the message originally was. Take, for example, a code with weight $2p + 1$. If we flip $p + 1$ of the 1s in this code, we end with a vector of weight $2p + 1 - (p + 1) = p$. Then the distance between our code and $\vec{0}$ is now p, and the distance to our original code is now $p + 1$. So we are closer to $\vec{0}$, so there is no way to reliably determine which code we originally had. Therefore, we can only distinguish between p or fewer bit flips.

In a similar manner to (1), we also see that

$$HG^T = \begin{pmatrix} \vec{g}_{k+1} \\ \vec{g}_{k+2} \\ \vdots \\ \vec{g}_n \end{pmatrix} \begin{pmatrix} \vec{g_1} & \vec{g_2} & \dots & \vec{g_k} \end{pmatrix} = [0]_{n\times k}\,.$$

One may demonstrate from this that $GH^T = \{[0]_{n\times k}\}^T = [0]_{k\times n}$ is also true. The fact that HG^T and GH^T are both zero matrices is just essentially just a restatement of the condition that any vector $\vec{v} \in C$ must satisfy $H(\vec{v})' = 0$ since we know that any vector $\vec{v} \in C$ is a linear combination of $\vec{g_1}, ..., \vec{g_k}$ and each row of H is a vector g_i for $i \in \{k+1, ..., n\}$ that is orthogonal to every g_i for $i \in \{1, ..., k\}$.

The consequence of this is that the $(n-k) \times n$ matrix H can now function as a generator for a new code space $C^\perp$ of dimension $n-k$, where $C^\perp$ is the orthogonal complement of C because the row vectors in H and G are orthogonal and linearly independent. $C^\perp$ is referred to as the dual code of C.

Now we show that

$$\sum_{\vec{c}\in C}(-1)^{\vec{c}\cdot\vec{x}} = \begin{cases} 2^k, \vec{x} \in C^\perp \\ 0, \vec{x} \notin C^\perp. \end{cases} \tag{2}$$

Proof. If $\vec{x} \in C^\perp$, $\vec{x}$ is orthogonal to all $\vec{c} \in C$, then $\vec{c} \cdot \vec{x} = 0$, so $(-1)^{\vec{c}\cdot\vec{x}} = 1$. It follows that the sum is 2^k because $|C| = 2^k$. On the other hand, if $\vec{x} \notin C^\perp$, then we can use the fact that

$$\sum_{\vec{y}\in\{0,1\}^k} (-1)^{\vec{y}\cdot\vec{x}} = 0 \tag{3}$$

whenever $\vec{x} \neq \vec{0}$. Equation (3) can be seen from the fact that $\vec{x}$ must have an entry x_i such that $x_i = 1$. Thus, for any $\vec{y} \in \{0,1\}^k$, we can find a corresponding $(\vec{y})'$ (note: the notation $(\vec{y})'$ does not refer to $(\vec{y})^T$, here) by letting $(y_i)' = y_i \oplus 1$ and keeping all other entries the same. Thus, $\vec{y} \cdot \vec{x} = \left[(\vec{y})' \cdot \vec{x}\right] \oplus 1$, so $(-1)^{\vec{y}\cdot\vec{x}} = -(-1)^{(\vec{y})'\cdot\vec{x}}$, meaning $(-1)^{\vec{y}\cdot\vec{x}} + (-1)^{(\vec{y})'\cdot\vec{x}} = 0$. Since the sets $S_1 = \{\vec{y} \in \{0,1\}^k : (\vec{y})' \notin S_1\}$ and $S_2 = \{(\vec{y})' \in \{0,1\}^k : \vec{y} \in S_1\}$ split $\{0,1\}^k$ into equal pieces (i.e. $S_1 \cup S_2 = \{0,1\}^k$ and $|S_1| = |S_2|$), this implies equation (3). Using this result, we get for $\vec{x} \neq \vec{0}$ that

$$\begin{aligned} \sum_{\vec{c}\in C}(-1)^{\vec{c}\cdot\vec{x}} &= \sum_{\vec{m}\in\{0,1\}^k} (-1)^{\vec{m}\cdot G\vec{x}} \\ &= 0, \end{aligned}$$

using (3) and the fact that all $\vec{c_i}$ can be written as $\vec{m_i}G$ for some $\vec{m} \in \{0,1\}^k$. We know that $G\vec{x} \neq 0$ (which is a condition for (3)) because $\vec{x} \notin C^\perp$ means $\vec{x} \cdot \vec{g}_i \neq 0$ for some $1 \leq i \leq k$ (otherwise $\vec{x} \in C^\perp$ would be true). □

The intuition behind this statement is that if a vector $\vec{x}$ is in $C^{\perp}$, then $\vec{c} \cdot \vec{x} = 0$ for all $\vec{c} \in C$, so $(-1)^{\vec{c} \cdot \vec{x}} = (-1)^0 = 1$ for all $\vec{c} \in C$, and $|C| = 2^k$, so our sum is simply $2^k \times 1 = 2^k$. If $\vec{x} \notin C^{\perp}$, we find that half of the code words in $\vec{c} \in C$ have a dot product $\vec{c} \cdot \vec{x} = 0$ and the other half have a dot product of 1, so our sum becomes 0.

So far, the framework of encoding messages of length k in an n-dimensional code space using a generator matrix and identifying errors with the parity check matrix is a common procedure used for classical codes. We now extend this to the case of CSS codes.

3.2 CSS Code Formalism

Consider two code spaces C_1 and C_2 of dimension k_1 and k_2 with bases β_1 and β_2 respectively, where $k_1 < k_2$, $\beta_2 \subset \beta_1$, and $C_2 \subset C_1$. Recall that C_1 and C_2 are subspaces of the computational space $\{0,1\}^n$. We can now translate the two code spaces into a quantum CSS code space as follows. Each classical string $\vec{m}$ (where either $\vec{m} \in C_1$ or $\vec{m} \in C_2$) can be converted into a quantum state $|m\rangle$, where for instance 01 would map to $|01\rangle$. Thus, C_1 is converted into a 2^{k_1} dimensional Hilbert space Q_1 while C_2 is converted into a 2^{k_2} dimensional Hilbert space Q_2 ($Q_2 \leqslant Q_1$). We can create a $2^{k_1-k_2}$ dimensional subspace S of Q_1 by using each $\vec{w_i} \in \text{span}\{\beta_1 \backslash \beta_2\}$ to define $2^{k_1-k_2}$ linearly independent vectors of the form

$$|v_i\rangle = \frac{1}{\sqrt{2^{k_2}}} \sum_{\vec{c_j} \in C_2} |\vec{c_j} \oplus \vec{w_i}\rangle \,. \tag{4}$$

These $2^{k_1-k_2}$ linearly independent vectors will serve as the code words of the CSS code, meaning that a $k_1 - k_2$ bit classical message can be represented as a single $|v_i\rangle$. Note that for any $\vec{c_j}, \vec{c_l} \in C_2$, we have $|\vec{c_j} \oplus \vec{w_{i_1}}\rangle \neq |\vec{c_l} \oplus \vec{w_{i_2}}\rangle$ if $i_1 \neq i_2$, since $\vec{w_{i_1}}$ and $\vec{w_{i_2}}$ are linearly independent, meaning $(w_{i_1} \oplus C_2) \cap (w_{i_2} \oplus C_2) = \varnothing$.

We can use this space S to analyze bit flip errors. We can create a space T_i, where

$$T_i = \left\{ E_l \,|v_i\rangle = \frac{1}{\sqrt{2^{k_2}}} \sum_{\vec{c_j} \in C_2} |\vec{c_j} \oplus \vec{w_i} \oplus \vec{e_l}\rangle : \vec{e_l} \in \{0,1\}^n \text{ and } \vec{e_l} \text{ correctable} \right\} \tag{5}$$

is the set containing the images of $|v_i\rangle$ after each correctable bit flip error E_l was applied. We define bit flip error E_l acting on $|v_i\rangle$ as

$$E_l \,|v_i\rangle = \frac{1}{\sqrt{2^{k_2}}} \sum_{\vec{c_j} \in C_2} |\vec{c_j} \oplus \vec{w_i} \oplus \vec{e_l}\rangle \,, \tag{6}$$

where $\vec{e_l}$ is a length n bit string and at every index of $\vec{e_l}$ equal to 1, the corresponding index of $|v_i\rangle$ is flipped. Now if there are L correctable bit flip errors $\vec{e_l}$, we can correspond to each a length $h = \lceil \log(L) \rceil$ bit string $\vec{l}$, where $\vec{l}$ is the binary representation of the index l. Now suppose a superposition of bit flip

errors acted on $|v_i\rangle$ as follows:

$$E\,|v_i\rangle = \sum_{l=1}^{L} b_l(E_l\,|v_i\rangle), \tag{7}$$

where b_l is the amplitude of each individual error E_l. Then we can define a unitary

$$U = \sum_{l=1}^{L}(|E_l v_i\rangle \otimes |\vec{l}\rangle)(\langle E_l v_i| \otimes \langle\vec{0}|), \tag{8}$$

so that

$$U(E\,|v_i\rangle \otimes |\vec{0}\rangle) = \sum_{l=1}^{L} b_l(E_l\,|v_i\rangle \otimes |\vec{l}\rangle), \tag{9}$$

where $|\vec{0}\rangle$ and $|\vec{l}\rangle$ have length $h = \lceil \log(L) \rceil$. Now, if we measure the operator

$$\hat{M} = \sum_{l=1}^{L} l|\vec{l}\rangle\langle\vec{l}| \tag{10}$$

on the last h qubits, we will measure l for some $1 \leq l \leq L$ and the state $U(E\,|v_i\rangle \otimes |\vec{0}\rangle)$ will collapse to $E_l|v_i\rangle \otimes |\vec{l}\rangle$. Now we can restore $E_l|v_i\rangle$ to $|v_i\rangle$ by adding $\vec{e}_l$ to the bit string of each term comprising $|v_i\rangle$.

We have successfully addressed the case of bit flip errors acting on an encoding $|v_i\rangle$, but in order to deal with phase flip errors we will need the Hadamard transformation. Recall that

$$H^{\otimes n}\,|\vec{x}\rangle = \frac{1}{\sqrt{2^n}} \sum_{\vec{z}\in\{0,1\}^n} (-1)^{\vec{x}\cdot\vec{z}}\,|\vec{z}\rangle\,. \tag{11}$$

Using definition (7), we can establish that

$$H^{\otimes n}\,|\vec{v_i}\rangle = \frac{1}{\sqrt{2^{k_2}}} \sum_{\vec{c_j}\in C_2} H^{\otimes n}\,|\vec{c_j} \oplus \vec{w_i}\rangle \tag{12}$$

$$= \frac{1}{\sqrt{2^{k_2}}} \sum_{\vec{c_j}\in C_2} \frac{1}{\sqrt{2^n}} \sum_{\vec{z}\in\{0,1\}^n} (-1)^{(\vec{c_j}\oplus\vec{w_i})\cdot\vec{z}}\,|\vec{z}\rangle \tag{13}$$

$$= \frac{1}{\sqrt{2^{k_2+n}}} \sum_{\vec{c_j}\in C_2} \sum_{\vec{z}\in\{0,1\}^n} (-1)^{\vec{c_j}\cdot\vec{z}}(-1)^{\vec{w_i}\cdot\vec{z}}\,|\vec{z}\rangle \tag{14}$$

$$= \frac{1}{\sqrt{2^{k_2+n}}} \sum_{\vec{z}\in\{0,1\}^n} (-1)^{\vec{w_i}\cdot\vec{z}} \sum_{\vec{c_j}\in C_2} (-1)^{\vec{c_j}\cdot\vec{z}}\,|\vec{z}\rangle \tag{15}$$

$$= \frac{1}{\sqrt{2^{k_2+n}}} \sum_{\vec{z}\in C_2^{\perp}} (-1)^{\vec{w_i}\cdot\vec{z}} 2^{k_2}\,|\vec{z}\rangle \tag{16}$$

$$= \frac{1}{\sqrt{2^{n-k_2}}} \sum_{\vec{z}\in C_2^{\perp}} (-1)^{\vec{w_i}\cdot\vec{z}}\,|\vec{z}\rangle\,. \tag{17}$$

Similarly to bit flip errors, we can analyze phase flip errors by creating a space V_i, where

$$V_i = \left\{ P_l \left|v_i\right\rangle = \frac{1}{\sqrt{2^{k_2}}} \sum_{\vec{c_j} \in C_2} (-1)^{(\vec{c_j} \oplus \vec{w_i}) \cdot \vec{p_l}} \left|\vec{c_j} \oplus \vec{w_i}\right\rangle : \vec{p_l} \in \{0,1\}^n \text{ and } \vec{p_l} \text{ correctable} \right\} \tag{18}$$

is the set containing the images of $|v_i\rangle$ after each correctable phase flip error P_l was applied. We define phase flip error P_l acting on $|v_i\rangle$ as

$$P_l \left|v_i\right\rangle = \frac{1}{\sqrt{2^{k_2}}} \sum_{\vec{c_j} \in C_2} (-1)^{(\vec{c_j} \oplus \vec{w_i}) \cdot \vec{p_l}} \left|\vec{c_j} \oplus \vec{w_i}\right\rangle, \tag{19}$$

where $\vec{e_l}$ is a length n bit string and at every index of $\vec{e_l}$ equal to 1, the sign of the corresponding term of $|v_i\rangle$ is flipped. One may show through a similar process as steps (12) through (17) that

$$H^{\otimes n} P_l \left|\vec{v_i}\right\rangle = \frac{1}{\sqrt{2^{k_2}}} \sum_{\vec{c_j} \in C_2} H^{\otimes n} (-1)^{(\vec{c_j} \oplus \vec{w_i}) \cdot \vec{p_l}} \left|\vec{c_j} \oplus \vec{w_i}\right\rangle \tag{20}$$

$$= \frac{1}{\sqrt{2^{k_2}}} \sum_{\vec{c_j} \in C_2} \frac{1}{\sqrt{2^n}} (-1)^{(\vec{c_j} \oplus \vec{w_i}) \cdot \vec{p_l}} \sum_{\vec{z} \in \{0,1\}^n} (-1)^{(\vec{c_j} \oplus \vec{w_i}) \cdot \vec{z}} \left|\vec{z}\right\rangle \tag{21}$$

$$= \frac{1}{\sqrt{2^{k_2+n}}} \sum_{\vec{c_j} \in C_2} (-1)^{\vec{c_j} \cdot \vec{p_l}} (-1)^{\vec{w_i} \cdot \vec{p_l}} \sum_{\vec{z} \in \{0,1\}^n} (-1)^{\vec{c_j} \cdot \vec{z}} (-1)^{\vec{w_i} \cdot \vec{z}} \left|\vec{z}\right\rangle \tag{22}$$

$$= \frac{1}{\sqrt{2^{k_2+n}}} \sum_{\vec{z} \in \{0,1\}^n} (-1)^{\vec{w_i} \cdot \vec{p_l}} (-1)^{\vec{w_i} \cdot \vec{z}} \sum_{\vec{c_j} \in C_2} (-1)^{\vec{c_j} \cdot (\vec{p_l} \oplus \vec{z})} \left|\vec{z}\right\rangle \tag{23}$$

$$= \frac{1}{\sqrt{2^{k_2+n}}} \sum_{\vec{z} \oplus \vec{p_l} \in C_2^{\perp}} (-1)^{\vec{w_i} \cdot (\vec{z} \oplus \vec{p_l})} \cdot 2^{k_2} \left|\vec{z}\right\rangle \tag{24}$$

$$= \frac{1}{\sqrt{2^{n-k_2}}} \sum_{\vec{y} \in C_2^{\perp}} (-1)^{\vec{w_i} \cdot \vec{y}} \left|\vec{y} \oplus \vec{p_l}\right\rangle. \tag{25}$$

It is now clear that applying the Hadamard gate to the phase flipped state turns the phase flip errors into bit flip errors. These errors can now be corrected using the bit flip error correction procedure described above. Note that in the bit flip error case, the state to be corrected in (6) summed over $\vec{c_j} \in C_2$, while for the phase flip error correction the state to be corrected in (25) summed over $\vec{y} \in C_2^{\perp}$. This indicates that the codes for correcting the bit flip and phase flip errors work in an orthogonal manner.

3.3 CSS Code Intuition

Essentially, a CSS code combines two error correcting codes that are completely independent of one another (i.e. they are orthogonal). All of the operators within a CSS code commute, so the measurements for the error syndrome diagnosis don't cause state collapse that would in any way affect the resulting

measurements of the other operators. One simple way to tell if a code is a CSS code is if the operators only have X or only have Z in them (and I). If there are both X and Z within a single operator, a code is not CSS (such as the 5-qubit code). Another thing that distinguishes CSS codes is that one can use Hadamard gates to convert between the X and Z errors, using the procedure from one to correct the transformed version of the other before converting back again.

4 Example - Steane Code

The Steane code is a 7 qubit CSS code. The encoding of a qubit $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$ is given by

$$\begin{aligned}|0\rangle \to \frac{1}{2\sqrt{2}}[&|0000000\rangle + |1010101\rangle + |0110011\rangle + |1100110\rangle \\ &+ |0001111\rangle + |1011010\rangle + |0111100\rangle + |1101001\rangle] \\ |1\rangle \to \frac{1}{2\sqrt{2}}[&|1111111\rangle + |0101010\rangle + |1001100\rangle + |0011001\rangle \\ &+ |1110000\rangle + |0100101\rangle + |1000011\rangle + |0010110\rangle]\end{aligned}$$

And the operators $IIIXXXX, IXXIIXX, XIXIXIX, IIIZZZZ, IZZIIZZ, ZIZIZIZ$ are measured. It can be shown that the Steane code can detect and correct bit flip and phase flip erorrs and combinations of the two, so it can detect and correct any arbitrary error on a single qubit.

However, what is of more importance to us now that we have defined and formalized CSS codes, is demonstrating that the Steane code is, in fact, a CSS code.

There are two easy ways to tell that the Steane code is a CSS code. The first is that all of the operators commute - that is, two operators σ_0 and σ_1 obey $\sigma_0\sigma_1 = \sigma_1\sigma_0$. The second way is that the operators either only have X or only have Z - none of them have both X and Z.

Now let us show that the Steane code is a CSS code more rigorously. We want to show that $C_2 \subset C_1$, where C_1 is the Steane code defined by the parity check matrix

$$H_1 = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

and C_2 is defined by the parity check matrix

$$H_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Remember, a parity check matrix H for a code space C should obey $\vec{c}H = 0$ for all $c \in C$. So, to show that the Steane code is a CSS code, we want to show

that $C_2 \subset C_1$, which requires that $\vec{c}H_2 = 0 \implies \vec{c}H_1 = 0$ for all $\vec{c} \in C_2$, or in other words, the null space of H_2 should be contained within the null space of H_1.

Note that the first row of H_1 and the last row of H_2 are equal. And the second row of H_1 is the direct sum of the second and third rows of H_2. And the third row of H_1 is the direct sum of the first and third rows of H_2. So the span of the rows of H_1 is a subset of the span of the rows of H_2. So the null space of H_2 is contained within the null space of H_1, as desired. Therefore the Steane code is a CSS code.

5 Outlook and Conclusion

CSS codes are an important development in quantum error correction because of their simplicity and efficiency. In terms of simplicity, their layout is straightforward, as one simply tailors a code to a specific error type and uses a transform such as the Hadamard to apply the same code a different type of error. In terms of efficiency, CSS codes are beneficial because there is a positive bound on the ratio $\frac{k}{n}$, where k is the number of bits in a message and n is the number of bits in the computational space. That means that one will never need infinitely more auxiliary bits than the number message of bits to send CSS-encoded information. This is significant because it suggests that one can scale a CSS code to accommodate arbitrarily large input messages.

As we continue to explore the potential of QEC in building robust and fault-tolerant quantum computing systems, the development and implementation of CSS codes are promising. The future of CSS codes will likely involve further research into optimization, as well as the discovery of new codes that offer improved error correction capabilities. As QEC algorithms like CSS codes become more efficient and robust, they will help overcome the challenges associated with noise/errors in quantum computing systems, allowing quantum computation to become a reliable method of computation.

In conclusion, CSS codes are an essential aspect of quantum error correction, providing a robust framework for the detection and correction of errors that occur during quantum computation. CSS codes allow for more resilient quantum computing systems that can withstand the impact of noise and other error sources. The development of CSS codes and their successful implementation in quantum computers/quantum information systems will help to enable a quantum computers to reach their full potential.

6 References

References

[1] Michael A. Nielsen & Isaac L. Chuang *Quantum Computation and Quantum Information (10th Anniversary Edition)*

[2] Dave Bacon *CSE 599d - Quantum Computing Stabilizer Quantum Error Correcting Codes (Lecture 18)* https://courses.cs.washington.edu/courses/cse599d/06wi/lecturenotes18.pdf

[3] Eleanor Rieffel & Wolfgang Polak. (2011). *Quantum computing: a gentle introduction.* Cambridge, Mass.: MIT Press

[4] https://intra.ece.ucr.edu/korotkov/courses/EE214-QC/QC-7-error-correction.pdf

[5] John Preskill *Physics 219 - Quantum Error Correction (Chapter 7)*

[6] https://viterbi-web.usc.edu/ tbrun/Course/lecture21.pdf

[7] Andrew Steane. (2006). *A tutorial on quantum error correction.* Paper presented at the International School of Physics "Enrico Fermi," 162 1-32. https://doi.org/10.3254/1-58603-660-2-1