QUANTUM ERROR CORRECTION

For C7.4 Introduction to Quantum Information
updated February 28, 2024

ZHENYU CAI

1. Preliminary

1.1. **General Idea.** In quantum error correction (QEC), we have n *physical qubits* which lead to a Hilbert space of dimension 2^n. Within this full Hilbert space $\mathcal{H}$, we will define a subspace of dimension 2^k (with $k \leq n$) as our *code space* $\mathcal{C}$, which can be used to encode k *logical qubits*. By storing our quantum information in these logical qubits and thus in the code space $\mathcal{C}$, whenever an error takes our quantum state outside of the code space $\mathcal{C}$, we can detect such an error and attempt to correct it by projecting the erroneous state back into the code space $\mathcal{C}$.

1.2. **Example: Bit-flip Repetition Code.** Given $n = 3$ physical qubits, we define our code space $\mathcal{C}$ to be the subspace spanned by $\{|000\rangle, |111\rangle\}$, which encodes $k = 1$ logical qubits with the following mapping:

$$|\overline{0}\rangle \mapsto |000\rangle$$
$$|\overline{1}\rangle \mapsto |111\rangle$$

where we use top-bar $^{\overline{}}$ to denote the logical information encoded.

For an arbitrary state $|\overline{\psi}\rangle$ within the code space $\mathcal{C}$:

$$|\overline{\psi}\rangle = \alpha |000\rangle + \beta |111\rangle,$$

if a bit-flip error (X error) occurs to the first qubit, we then have the erroneous state:

$$|\psi_\epsilon\rangle = X_1 |\overline{\psi}\rangle = \alpha |100\rangle + \beta |011\rangle,$$

which is now *outside* the code space $\mathcal{C}$. We can then detect such an error by checking whether our state is still within the code space $\mathcal{C}$.

In fact for the bit-flip repetition code, if a single-qubit bit flip happens on *any* of the qubits, the process of checking whether our state is still within the code space $\mathcal{C}$ will also tell us about which qubit is flipped, thus enable us to correct any single-qubit bit-flip (assuming no two-qubit bit flips).

2. Stabiliser Formalism

2.1. **Code Space.** We will use $\mathbb{G}$ to denote the Pauli group for n qubits:

$$\mathbb{G} = \{\pm 1, \pm i\} \times \{\mathbb{1}, X, Y, Z\}^{\otimes n}.$$

The code space $\mathcal{C}$ is defined by a subgroup of the Pauli group $\mathbb{S} \subset \mathbb{G}$ called the *stabiliser group*, such that for any states $|\psi\rangle$ within the code space $\mathcal{C}$, it is invariant under the action of any elements S in the stabiliser group $\mathbb{S}$:

$$\mathcal{C} = \{|\psi\rangle \in \mathcal{H} \quad | \quad S|\psi\rangle = |\psi\rangle \ \ \forall S \in \mathbb{S}\}. \tag{1}$$

Note that to have a non-trivial (non-zero dimension) code space, $\mathbb{S}$ must not contain $-\mathbb{1}$ (in fact, this is the only requirement for a Pauli subgroup to be a valid stabiliser group). This also implies that $\mathbb{S}$ is *Abelian*.

Why must the stabiliser group S satisfy these properties?

The generators of the stabiliser group (*stabiliser generators* or *stabiliser checks*) will be denoted as $\widetilde{\mathbb{S}}$. It is easy to see that if a state $|\psi\rangle$ satisfy $\widetilde{S}|\psi\rangle = |\psi\rangle$ for all $\widetilde{S} \in \widetilde{\mathbb{S}}$,

then $|\psi\rangle$ also satisfy $S|\psi\rangle = |\psi\rangle$ for all $S \in \mathbb{S}$. Hence, we can rewrite our definition of the code space in terms of the stabiliser checks instead:

$$\mathcal{C} = \{|\psi\rangle \in \mathcal{H} \quad | \quad \widetilde{S}|\psi\rangle = |\psi\rangle \ \ \forall \widetilde{S} \in \widetilde{\mathbb{S}}\}. \tag{2}$$

Since the stabiliser group $\mathbb{S}$ is Abelian and it contains only Pauli operators which square to $\mathbb{1}$, we have:

The absolute sign $|\ |$ is used to denote the size of a set.

$$|\mathbb{S}| = 2^{|\widetilde{\mathbb{S}}|}. \tag{3}$$

Hence, the set of stabiliser checks is *exponentially smaller* than the full set of stabilisers, checking only the generators is so much easier!

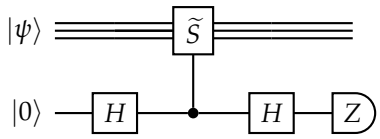
Why is Eq. (3) true? Hint: since the stabiliser group $\mathbb{S}$ is Abelian, to construct a given stabiliser, the order of composition of the stabiliser checks is not important, we only need to care about which stabiliser checks are used.

Each stabiliser check can be viewed as an independent constraint on the code space that freezes out one qubit degree of freedom. Hence, the number of logical qubits k encoded in the code space, which is related to the degrees of freedom left after applying the stabiliser generator constraints, is given by:

$$k = n - |\widetilde{\mathbb{S}}|\,. \tag{4}$$

where n is the total number of physical qubits.

2.2. **Performing Stabiliser Checks.** Following the definition of the code space $\mathcal{C}$ in Eq. (2), given an arbitrary state $|\psi\rangle \in \mathcal{H}$, to check whether it lies within the code space $\mathcal{C}$, we only need to perform measurements of all the stabiliser checks $\widetilde{S} \in \widetilde{\mathbb{S}}$ on the state and it lies in the code space iff all the measurements returns $+1$. The circuit for performing the stabiliser check $\widetilde{S}$ is given by:



The measurement of the ith stabiliser check $\widetilde{S}_i \in \widetilde{\mathbb{S}}$ will return a measurement result of $s_i = \pm 1$. The set of stabiliser check measurement results $\vec{s} = \{s_i\}$ is called the *error syndrome*, which would inform us about the errors that occurred and thus enable us to correct them. For the syndrome in which all entries are $+1$: $\vec{s} = \vec{1}$, it simply means the state is in the code space since it passes all the stabiliser checks.

Note that in the usual convention, the stabiliser measurement result is often denoted using the corresponding eigenstate $|m_i\rangle$ instead: $s_i = +1 \Rightarrow m_i = 0$, $s_i = -1 \Rightarrow m_i = 1$ (i.e. $s_i = (-1)^{m_i}$), and we call $\vec{m}$ the error syndrome instead of $\vec{s}$. In this note, for simplicity we stick with using $\vec{s}$ instead of $\vec{m}$ as our error syndrome, this distinction does not change any of the arguments we are going to make.

2.3. **Error Syndrome.** Recall that two Pauli operators G_1, G_2 will either commute or anti-commute. We can define their commutator $\eta(G_1, G_2)$ as:

Does the syndrome $\vec{s} = \vec{1}$ means that no errors have happened?

$$G_1 G_2 = \eta(G_1, G_2) G_2 G_1$$

for which

$$\eta(G_1, G_2) = \begin{cases} +1 & G_1 \text{ and } G_2 \text{ commute} \\ -1 & G_1 \text{ and } G_2 \text{ anti-commute.} \end{cases} \tag{5}$$

When a Pauli error E occurs on a state within the code space $|\overline{\psi}\rangle \in \mathcal{C}$, measuring $\widetilde{S}_i$ on this erroneous state $E|\overline{\psi}\rangle$ would return:

$$\widetilde{S}_i E|\overline{\psi}\rangle = \eta(\widetilde{S}_i, E) E \widetilde{S}_i |\overline{\psi}\rangle = \eta(\widetilde{S}_i, E) E|\overline{\psi}\rangle$$

i.e. the stabiliser measurement of $\widetilde{S}_i$ on the erroneous state $E|\overline{\psi}\rangle$ would return the measurement result of $s_i(E) = \eta(\widetilde{S}_i, E)$, which means the full error syndrome for the Pauli error E is:

$$\vec{s}(E) = \{\eta(\widetilde{S}_i, E) \quad \forall \widetilde{S}_i \in \widetilde{\mathbb{S}}\}. \tag{6}$$

2.4. **Applying Correction.** The process of translating a given error syndrome $\vec{s}$ into the recovery operation (correction) is called *decoding*. In the simplest decoding scheme called *minimum-weight decoding*, we look at all errors that can give rise to the same error syndrome and pick the one with the lowest weight as our guess for the error that occurred. This is because assuming local error events, the probability of a given error occurring will decay exponentially with its weight. Hence, the error with the lowest weight will have the highest probability of occurring for the given syndrome. If we guess the error to be some Pauli operator E, then the correction we need to apply is simply an additional E gate since $E^2 = \mathbb{1}$ for Pauli operators.

The *weight* of a Pauli operator G is the number of qubits that it acts non-trivially on, denoted as $\mathrm{wt}(G)$. E.g. $\mathrm{wt}(Z_1Z_2) = 2$, $\mathrm{wt}(X_2X_5X_8X_9) = 4$.

If we assume each single-qubit error occurs with probability p, then a weight-w error will occur with the probability $\mathcal{O}(p^w)$.

Is minimum-weight decoding the optimal decoding scheme? This can be better answered after you learn about logical gates later.

2.5. **Example Revisited: Bit-Flip Repetition Code.** In the stabiliser formalism, the code space spanned by $\{|000\rangle, |111\rangle\}$ is defined by the set of stabilisers

$$\mathbb{S} = \{\mathbb{1},\ Z_1Z_2,\ Z_2Z_3,\ Z_1Z_3\},$$

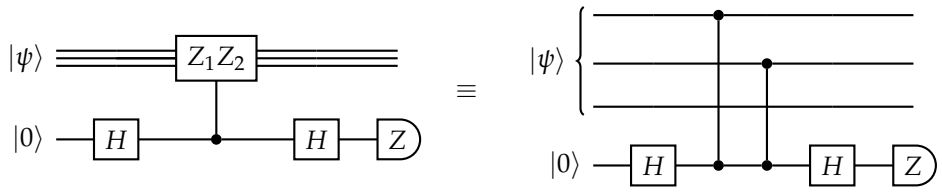
which can be generated from the set of stabiliser checks

$$\widetilde{\mathbb{S}} = \{Z_1Z_2,\ Z_2Z_3\}.$$

Following Eq. (4), the number of logical qubits encoded is $k = n - |\widetilde{\mathbb{S}}| = 3 - 2 = 1$ as expected.

We often use subscripts to denote which qubits the Pauli operator acts on. For example, when there are 3 qubits, we have $Z_1Z_2 \equiv Z \otimes Z \otimes \mathbb{1}$ and $Z_1Z_3 \equiv Z \otimes \mathbb{1} \otimes Z$.

The Z_1Z_2 stabiliser check on the state $|\psi\rangle$ can be performed using:



And similarly for the stabiliser check for Z_2Z_3.

As discussed before, the error syndrome for a given error can be obtained through its commutation relationship with the stabiliser checks $\{Z_1Z_2,\ Z_2Z_3\}$ using Eqs. (5) and (6). Hence, the error syndrome for a bit-flip error in the first qubit (X_1) is:

$$\vec{s}(X_1) = \{\eta(Z_1Z_2, X_1), \eta(Z_2Z_3, X_1)\} = \{-1, +1\}.$$

Following the same arguments, we can obtain the error syndrome for all the bit-flip errors:

Syndrome	Possible Errors	Correction
$\{+1, +1\}$	$\mathbb{1}$ or $X_1X_2X_3$	$\mathbb{1}$
$\{+1, -1\}$	X_3 or X_1X_2	X_3
$\{-1, +1\}$	X_1 or X_2X_3	X_1
$\{-1, -1\}$	X_2 or X_1X_3	X_2

We notice that multiple errors can give rise to the same syndrome. As mentioned in Section 2.4, we will choose the lowest-weight error as the correction since they are more likely to occur. In such a way, when any weight-2 bit flip happens, a wrong correction can be applied, causing our code to fail. If the probability of a single-qubit bit flip is p, then the probability of a weight-2 bit flip happening, leading to the failure of our code is $\mathcal{O}(p^2)$. This is a quadratic reduction compared to the bit-flip error probability of a single unprotected qubit (which is simply p).

3. Beyond Bit-flip Errors

3.1. **Syndrome Subspace.** The projector of the ± 1 eigenspace of a Pauli operator G is given as:

$$\Pi_{\pm} = \frac{\mathbb{1} \pm G}{2}.$$

For a ± 1 eigenstates of G: $|G_{\pm}\rangle$, we can easily verify that: $\Pi_{\pm} |G_{\pm}\rangle = |G_{\pm}\rangle$, $\Pi_{\pm} |G_{\mp}\rangle = 0$ as expected. We can also verify that $\Pi_{\pm}$ are idempotent (since they are projectors).

Hence, when measuring the ith stabiliser check $\widetilde{S}_i$, if the result is s_i, the incoming state is projected into the subspace defined by $\Pi_{s_i} = \frac{\mathbb{1}+s_i\widetilde{S}_i}{2}$. After measuring all the stabiliser checks $\widetilde{S}_i \in \widetilde{\mathbb{S}}$ and obtain the error syndrome $\vec{s}$, the incoming state is projected into the $\vec{s}$-syndrome subspace defined by the projector:

$$\Pi_{\vec{s}} = \prod_{i=1}^{|\widetilde{\mathbb{S}}|} \frac{\mathbb{1} + s_i \widetilde{S}_i}{2}.$$

More explicitly, let us again look at the simple example of bit-flip repetition code, for which the full Hilbert space can be divided into four different syndrome subspaces:

Error Syndrome	Projector	Subspace
$\{+1,+1\}$	$\frac{\mathbb{1}+Z_1Z_2}{2}\frac{\mathbb{1}+Z_2Z_3}{2}$	$\text{span}\{\|000\rangle, \|111\rangle\}$
$\{+1,-1\}$	$\frac{\mathbb{1}+Z_1Z_2}{2}\frac{\mathbb{1}-Z_2Z_3}{2}$	$\text{span}\{\|001\rangle, \|110\rangle\}$
$\{-1,+1\}$	$\frac{\mathbb{1}-Z_1Z_2}{2}\frac{\mathbb{1}+Z_2Z_3}{2}$	$\text{span}\{\|100\rangle, \|011\rangle\}$
$\{-1,-1\}$	$\frac{\mathbb{1}-Z_1Z_2}{2}\frac{\mathbb{1}-Z_2Z_3}{2}$	$\text{span}\{\|010\rangle, \|101\rangle\}$

Equipped with the idea of syndrome subspaces, now let us see how can we deal with errors beyond simple bit flips.

3.2. **Linear Combination of Errors.** So far we have only considered the case in which discrete X errors have occurred. However, in a quantum system, we can also have a superposition of errors. Let us suppose an error of the form $\alpha_0\mathbb{1} + \alpha_1 X_1 + \alpha_2 X_2 + \alpha_3 X_3$ has occurred on the code state $|000\rangle$ in the bit-flip repetition code, results in a noisy state

$$(\alpha_0\mathbb{1} + \alpha_1 X_1 + \alpha_2 X_2 + \alpha_3 X_3) |000\rangle = \alpha_0 |000\rangle + \alpha_1 |100\rangle + \alpha_2 |010\rangle + \alpha_3 |001\rangle$$

By performing stabiliser checks to measure the error syndrome on the incoming noisy state, we have collapsed the incoming noisy state into one of the syndrome subspaces listed above. The resulting output states and the corresponding probabilities are given by:

Error Syndrome	Output State	Probability	Correction needed
$\{+1,+1\}$	$\|000\rangle$	$\|\alpha_0\|^2$	$\mathbb{1}$
$\{+1,-1\}$	$\|001\rangle$	$\|\alpha_3\|^2$	X_3
$\{-1,+1\}$	$\|100\rangle$	$\|\alpha_1\|^2$	X_1
$\{-1,-1\}$	$\|010\rangle$	$\|\alpha_2\|^2$	X_2

Hence, the act of trying to detect the X errors using stabiliser checks has collapsed the incoming errors into one of the X errors that we can correct. More generally if the incoming error is a linear sum of correctable errors, then the act of syndrome measurement will collapse the incoming error into one of the correctable error components, i.e. *if a QEC code can correct a given set of errors, then it can also correct any linear combinations of these errors.*

Up till now we are still focusing on bit-flip (X) errors. If we have a code that can correct both single-qubit X errors and single-qubit Z errors, then it can also correct single-qubit Y errors which is equivalent to both X and Z errors happening ($Y = iXZ$). Following our arguments above, this code is then able to correct any linear combination of single-qubit Pauli errors. Since the Pauli operators form a complete basis for single-qubit operators, we can correct any single-qubit errors using this code. This can be extended to the multi-qubit case. Hence, *it is sufficient to only focus on the correction of X and Z errors up to a certain weight*, which implies the ability to correct any errors up to that weight. Now let us look at an explicit example of a code that can correct both X and Z errors, Shor's 9-qubit code.

3.3. **Shor's 9-qubit Code.** We have shown that by encoding our logical information with the stabiliser checks $\{Z_1Z_2, Z_2Z_3\}$ in the following way:

$$|\widetilde{0}\rangle \mapsto |000\rangle \qquad |\widetilde{1}\rangle \mapsto |111\rangle\,, \tag{7}$$

we can correct any single-qubit X (bit-flip) errors.

A simple switch between the X and Z basis, we can devise a code that can correct any single-qubit Z (phase-flip) errors instead:

$$|\widetilde{+}\rangle \mapsto |+++\rangle \qquad |\widetilde{-}\rangle \mapsto |---\rangle \tag{8}$$

with the stabiliser checks $\{X_1X_2, X_2X_3\}$

To construct a code that can correct both X and Z errors, we can further encode the $|+\rangle$ and $|-\rangle$ in the phase-flip code in Eq. (8) using the bit-flip code in Eq. (7):

$$\begin{aligned}
|\widetilde{\widetilde{+}}\rangle &\xrightarrow{Eq.\ (8)} |\widetilde{+}\widetilde{+}\widetilde{+}\rangle = \frac{1}{2\sqrt{2}}\left(|\widetilde{0}\rangle + |\widetilde{1}\rangle\right)^{\otimes 3} \xrightarrow{Eq.\ (7)} \frac{1}{2\sqrt{2}}\left(|000\rangle + |111\rangle\right)^{\otimes 3} \\
|\widetilde{\widetilde{-}}\rangle &\xrightarrow{Eq.\ (8)} |\widetilde{-}\widetilde{-}\widetilde{-}\rangle = \frac{1}{2\sqrt{2}}\left(|\widetilde{0}\rangle - |\widetilde{1}\rangle\right)^{\otimes 3} \xrightarrow{Eq.\ (7)} \frac{1}{2\sqrt{2}}\left(|000\rangle - |111\rangle\right)^{\otimes 3}
\end{aligned} \tag{9}$$

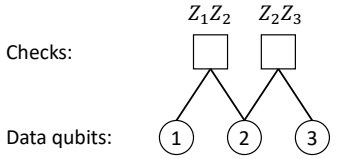
This is called Shor's 9-qubit code and has the set of stabiliser checks:

$$\{Z_1Z_2,\ Z_2Z_3,\ Z_4Z_5,\ Z_5Z_6,\ Z_7Z_8,\ Z_8Z_9,\ X_1X_2X_3X_4X_5X_6,\ X_4X_5X_6X_7X_8X_9\}.$$

One can verify that for different single-qubit X and Z errors, measuring the stabiliser checks above will obtain different non-trivial (i.e. not all $+1$) error syndrome, which can then be used to uniquely identify these single-qubit errors and make the appropriate correction. In fact, Shor's code can correct certain errors beyond single-qubit errors (e.g. X_1X_4). Note that since there are more Z-checks (which can detect X errors) than X-checks (which can detect Z errors), this code is more robust against X (bit-flip) errors.

There is also another version of Shor's code with a phase-flip code underneath a bit-flip code instead, which is more robust against Z (phase-flip) errors.

3.4. **Tanner Graph and CSS Codes.** A natural question is how one would construct a quantum error correction code beyond those we have discussed. Possible insights can be gained from looking at a graphical representation of the QEC code called the Tanner graph. The Tanner graph is a bipartite graph with the two disjoint subsets of vertices representing the data qubits and the checks (stabiliser generators), respectively. A check vertex is connected to a data vertex if and only if the given check acts non-trivially (i.e. a non-identity action) on the given data qubit.



Example Tanner graph for a repetition code. Tanner graph plays a critical role in classical error correction. The quantum version is a generalisation of that with multiple types of edges as we will see later.

A particularly important class of QEC codes that can be studied using the Tanner graph is the Calderbank-Shor-Steane (CSS) codes. These are codes that have only two types of stabiliser checks: checks that are purely tensor products of X and checks that are purely tensor products of Z. Correspondingly, the Tanner graphs for CSS codes also have two types of check vertices, emitting edges corresponding to X parity checks and Z parity checks, respectively. An example Tanner graph for the nine-qubit Shor code is shown in Fig. 1.

A CSS code is often constructed from two classical codes, one for the X checks and the other for the Z checks.

Not every arbitrary Tanner graph we draw will correspond to a valid QEC code. All the stabiliser checks must commute with each other as mentioned before. In the

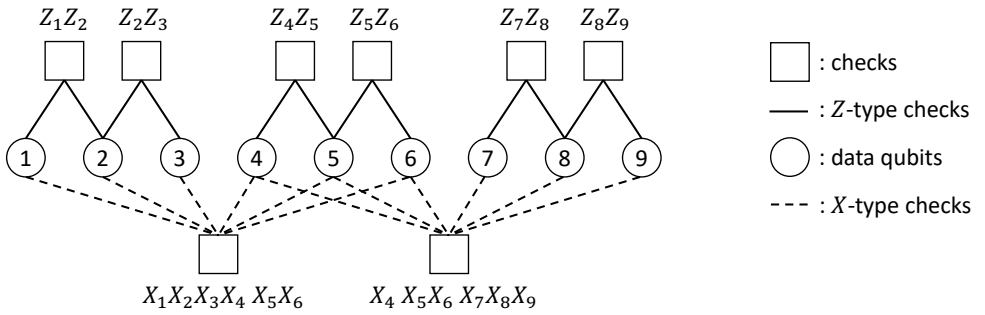


FIGURE 1. The Tanner graph for the nine-qubit Shor code

case of CSS codes, this means all the X checks must commute with all the Z checks. Translating into the language of the Tanner graph, this means *for any pair of X and Z checks, they must intersect at an even number of data qubits*. In this way, our Tanner graph will represent a valid QEC code.

Note that all stabiliser checks commute is not a sufficient condition for a valid QEC code, e.g. think of the set of checks $\{X_1X_2, Y_1Y_2, Z_1Z_2\}$. However, for the case of CSS code, this is sufficient (as long as no phase factors in the checks).

4. Logical Gates

4.1. **Logical Pauli Gates.** Besides protecting the logical information contained in a logical qubit, we would also want to perform logical gates on it in order to carry out computations. But how do we identify the physical operations that correspond to the target logical gates?

Recall that by definition, the set of stabilisers will act trivially on all code state $|\overline{\psi}\rangle$:

$$S\,|\overline{\psi}\rangle = |\overline{\psi}\rangle \quad \forall S \in \mathbb{S},\ |\overline{\psi}\rangle \in \mathcal{C}.$$

This simply implies that the stabilisers are the logical identity:

$$\mathbb{S} \equiv \overline{I}.$$

All the Pauli logical operations should commute with the logical identity. Hence, the set of logical Pauli gate $\overline{\mathbb{G}}$ is simply the set of Pauli operators that commute with the stabiliser group:

$$\overline{\mathbb{G}} = \{G \in \mathbb{G} \quad | \quad \mathbb{S}G = G\mathbb{S}\}.$$

In another word, $\overline{\mathbb{G}}$ is the *normaliser* of the stabiliser subgroup $\mathbb{S}$ in the Pauli group $\mathbb{G}$. Since $\mathbb{S} \equiv \overline{I}$, the logical action of a given logical operator $\overline{G} \in \overline{\mathbb{G}}$ is equivalent to all elements in $\overline{G}\mathbb{S} = \{\overline{G}S_1, \overline{G}S_2, \cdots\}$. E.g. all elements in $\overline{X}\mathbb{S}$ will perform the same logical action as $\overline{X}$.

Look up the difference between *centraliser* and *normaliser*, are they the same in the context of the Pauli group?

In group theoretic language, $\mathbb{S}$ is a normal subgroup of $\overline{\mathbb{G}}$. The set of logical operators $\overline{\mathbb{G}}$ is partitioned by $\mathbb{S}$ into different cosets, with operators in the same coset being logical equivalent, while operators in different cosets perform different logical actions.

But how do we know what logical action each element in $\overline{\mathbb{G}}$ correspond to? If we have identified a specific set of the logical basis states and we know their explicit forms (e.g. in bit-flip code, we know that $|\overline{0}\rangle = |000\rangle$, $|\overline{1}\rangle = |111\rangle$), we can just apply gates in $\overline{\mathbb{G}}$ on the basis states and see what logical action they perform.

If not, then within the set of logical Pauli gates $\overline{\mathbb{G}}$ we will try to identify the set of logical X and Z operators: $\{\overline{X}_1, \overline{Z}_1, \overline{X}_2, \overline{Z}_2, \cdots\}$ where the subscript i denotes action on the ith logical qubits, such that they satisfy the right commutation relationship. Once we have identified $\{\overline{X}_1, \overline{Z}_1, \overline{X}_2, \overline{Z}_2, \cdots\}$, we can use them to generate all the other logical Pauli gates. Note that such a choice is not unique, and by choosing the set of logical X and Z operators, we have also implicitly chosen the mapping between the logical basis states and physical states. Nevertheless in practice, there are usually some conventions to adhere to, e.g. for CSS codes, whenever possible, we will choose all logical X to be tensor products of only physical X, and all logical Z to be tensor products of only physical Z.

The right commutation relationship means logical X and Z acting on the same logical qubit will anti-commute, and all other pairings of logical X and Z commute.

4.2. **Code Distance and Correctable Errors.** Let us briefly look back at the process of quantum error correction. Suppose an error E occur on the code state $|\overline{\psi}\rangle$, which will give rise to the error syndrome $\vec{s}(E)$. Using minimum-weight decoding, we will pick a recovery operator (correction) R that is the minimum weight operator that gives rise to the same syndrome: $\vec{s}(E) = \vec{s}(R)$. The resultant state $RE|\overline{\psi}\rangle$ will have the trivial syndrome and thus is recovered back into the code space, which implies that RE must be a logical operator: $RE \in \overline{\mathrm{G}}$. This gives rise to two possible outcomes:

- $RE \in \mathrm{S}$, i.e. RE is a stabiliser. This implies $RE|\overline{\psi}\rangle = |\overline{\psi}\rangle$, and thus *successful error correction*.
- $RE \in \overline{\mathrm{G}} - \mathrm{S}$, i.e. RE is a logical operator beyond the stabilisers. This implies $RE|\overline{\psi}\rangle \neq |\overline{\psi}\rangle$, the logical information has been altered. We say that a *logical error* has happened to the state, which means that the error correction has failed.

The fact that we *do not require* need $RE = I$ for successful error correction, we only require $RE \in \mathrm{S}$, is a unique property of quantum codes. There are no equivalent statements in classical error correction. Quantum codes with such a property are called *degenerate* quantum codes.

The distance of the code is the weight of the smallest Pauli logical errors, usually denoted as d:

$$d = \min_{\overline{G} \in \overline{\mathrm{G}} - \mathrm{S}} \mathrm{wt}(\overline{G})$$

A code that encodes n physical qubits into k logical qubits with distance d is called an $[\![n, k, d]\!]$ code.

A code with distance d can correct any errors up to weight $\lfloor (d-1)/2 \rfloor$. To see why this is the case, let us suppose the error E has weight $\mathrm{wt}(E) \leq \lfloor (d-1)/2 \rfloor$. Using minimum weight decoding, the weight of the recovery operator R is by definition smaller than E, which implies $\mathrm{wt}(R) \leq \lfloor (d-1)/2 \rfloor$. Hence, the weight of the effective logical operation after recovery RE must satisfy

$$\mathrm{wt}(RE) \leq \mathrm{wt}(R) + \mathrm{wt}(E) \leq d - 1 < d.$$

Since by definition the smallest possible weight of a logical error is d, we know that RE cannot be a logical error, it can only be a stabiliser. Hence, we have successfully recovered our quantum state given $\mathrm{wt}(E) \leq \lfloor (d-1)/2 \rfloor$.

For CSS code, we can search for the logical X operators by looking at only tensor products of physical X operators and pick the ones that commute with all the Z checks (while not being a stabiliser). Similarly for the logical Z operators. Finding the smallest weight operator among these logical X and Z operators will give us the code distance.

4.3. **Examples.**

4.3.1. *Bit-flip Repetition Code.* Recalled that the stabiliser checks of the bit-flip repetition code are $\mathrm{S} = \{Z_1Z_2, Z_2Z_3\}$. It is a CSS code (that has no X checks), thus we can search for the logical $\overline{X}$ operators by looking for the tensor product of X that commutes with all the stabiliser checks, which is simply $\overline{X} = X_1X_2X_3$. We have $\overline{X}|000\rangle = |111\rangle$ as expected. Similarly, the logical $\overline{Z}$ operator can be found by looking at the tensor product of Z that commutes with the stabiliser checks (and not part of the stabilisers), and the one with the smallest weight is $\overline{Z} = Z_1$. Note that we can construct other logically equivalent operators by multiplying with the stabilisers, e.g. $Z_2 = Z_1 \cdot Z_1Z_2 \equiv \overline{Z}$. The minimum-weight logical error is simply $\overline{Z} \equiv Z_1$, which means that the code distance is $d = 1$. We have $\lfloor (d-1)/2 \rfloor = 0$ since it cannot correct any phase-flip (Z) errors.

4.3.2. *Shor's 9-qubit Code.* This is also a CSS code with the stabiliser checks $\{Z_1Z_2$, Z_2Z_3, Z_4Z_5, Z_5Z_6, Z_7Z_8, Z_8Z_9, $X_1X_2X_3X_4X_5X_6$, $X_4X_5X_6X_7X_8X_9\}$. After searching through all X and Z Pauli strings that commute with the stabiliser checks, we find that the smallest weight logical X operator is $\overline{X} = X_1X_2X_3$ and the smallest weight logical Z operator is $\overline{Z} = Z_1Z_4Z_7$. Hence, the code distance is $d = 3$ and it can correct all errors up to weight $\lfloor (d-1)/2 \rfloor = 1$.

5. Conclusion

In this note, we have introduced some of the most fundamental concepts in quantum error correction (QEC). There are still many interesting topics in QEC that we have not explored, e.g. threshold theorem, fault tolerance, state-of-the-art QEC codes like the surface code, codes beyond stabiliser codes like subsystem codes, etc. QEC is still a very dynamic research field with many of the open problems being keys to the practical realisation of quantum computers. Hopefully, this introductory note can act as a helpful guide for you to venture into various interesting topics in QEC!