Quantum Coding Theory **(UC Berkeley CS294, Spring 2024)**

Lecture 5: More Knill-Laflamme; Linear Codes

February 7, 2024

Lecturer: John Wright *Scribe: Jack Spilecki*

1 Introduction

Last lecture we spent a lot of time introducing the Knill-Laflamme conditions, which characterize when a quantum error correcting code exactly corrects for a set of errors. Today, in the first half of the lecture, we continue and see several very interesting consequences of the Knill-Laflamme conditions.

2 More on the Knill-Laflamme Conditions

To begin, we restate two equivalent versions of the conditions from last time:

Theorem 2.1 (Knill-Laflamme conditions)**.** *A code $\mathcal{C}$ corrects a linear space of errors $\mathcal{E}$ if and only if either:*

1. *With $\{|\overline{x}\rangle\}$ an orthonormal basis of $\mathcal{C}$, and $E_1, \dots, E_m$ a linear (but not necessarily orthonormal) basis of $\mathcal{E}$, for all $|\overline{x}\rangle$ and $|\overline{y}\rangle$ and $1 \leq i, j \leq m$,*

$$\langle\overline{x}| E_i^\dagger E_j |\overline{y}\rangle = \delta_{x,y} \cdot O_{i,j},$$

   where $O_{i,j}$ is a number which depends on the indices i and j, but not on the codewords $|\overline{x}\rangle$ and $|\overline{y}\rangle$.

2. *$\langle\psi| E_i^\dagger E_j |\psi\rangle = O_{i,j}$ for all $|\psi\rangle \in \mathcal{C}$, for all $1 \leq i, j \leq m$.*

The first condition tells us that we can figure out if $\mathcal{E}$ is a correctable set of errors by looking at all inner products between vectors of the form $\{E_i |\overline{x}\rangle\}$. Recall from last time that these vectors, for different $|\overline{x}\rangle$ and $|\overline{y}\rangle$, must be orthogonal, since otherwise there is no way to map them back to orthogonal vectors. Moreover, the inner products should only depend on the errors E_i and E_j, and not on the codewords. Last time we saw several interpretations for this, but here are two of them as a reminder:

The first interpretation: if we have a recovery algorithm that takes $E_i |\overline{x}\rangle$ back to $|\overline{x}\rangle$, then that algorithm is going to produce the codeword as well as a syndrome. However, the syndrome should record information only about the error, and not the codeword, since otherwise we are losing information about the state in the syndrome. If, however, the syndrome depends does not depend on the state, then we obtain the Knill-Laflamme conditions.

The other interpretation we saw is that if the errors arise from the state interacting with the environment, then the environment's state should be independent of the codeword. If it isn't, it has taken away information about the codeword, making it impossible to recover the codeword perfectly. The environment's state being independent of the codeword again implies the Knill-Laflamme conditions.

We can use the Knill-Laflamme conditions to finally define the central object that we are going to care about, which is: what does it mean for a code to correct errors that only affect a small number of qubits?

Fact 2.2. *A code $\mathcal{C} \subset \mathcal{H}_{physical} = (\mathbb{C}^2)^{\otimes n}$ corrects all weight-t errors if and only if the Knill-Laflamme conditions hold for $\mathcal{E}$ the set of all weight-t errors.*

According to the second form of the Knill-Laflamme conditions above, all we need is that $\langle\psi| E_1^\dagger E_2 |\psi\rangle = O_{i,j}$ for all weight-t errors E_1, E_2 (above the conditions were for a basis of errors, but it is equivalent to have the conditions hold for arbitrary errors).

What is the right basis for the set of weight-t errors? Recall: $\{I, X, Y, Z\}$ forms a *linear basis* for all 2×2 matrices. Equivalently, they form a basis for all single-qubit matrices, so any single qubit error can always be decomposed into Pauli matrices. We used that to see that the weight-1 elements of $\{I, X, Y, Z\}^{\otimes n}$ form a basis for all weight-1 errors on n qubits. As a reminder, the weight-1 Paulis are those that have I for $n-1$ (all but one) of its tensor factors, and a single factor of a Pauli matrix (which may also be the identity). Similarly, weight-t Paulis form a basis for all weight-t errors.

We can now rewrite our Knill-Laflamme conditions as:

$$\langle\psi| P_1^\dagger P_2 |\psi\rangle = O_{P_1,P_2}, \text{ for all weight-}t\text{ Paulis } P_1, P_2.$$

The product of two weight-t Paulis is a Pauli with weight at most $2t$. As an example, if $n = 4$ and $P_1 = I \otimes X \otimes Z \otimes I$ and $P_2 = Y \otimes Z \otimes I \otimes I$ (both weight-2 Paulis), we get $Y \otimes Y \otimes Z \otimes I$ (up to a phase). Here, the product is 3, which is less than $2t$, but we can get up to $2t$ if P_1 and P_2 are supported on different factors. So, we can equivalently state the Knill-Laflamme conditions as

$$\langle\psi| P |\psi\rangle = O_P, \text{ for all weight-}2t\text{ Paulis } P.$$

Finally, we make a nice observation: the weight-$2t$ Paulis form a basis for the set of all weight-$2t$ errors. So equivalently,

$$\langle\psi| E |\psi\rangle = O_E, \text{ for all weight-}2t\text{ errors } E.$$

We now have four equivalent ways of viewing what it means to be a code that corrects all weight-t errors.

Such a code is said to have *distance* $2t+1$. That is, a code has distance $2t+1$ if it can correct errors of weight at most t.

For some classical intuition: if we have the five-bit repetition code, with codewords 00000 and 11111, we would define the distance of this code to be the number of bitflips needed to get from one codeword to the other (the Hamming distance between the codewords). As a result, we would say this code has distance 5, because to get from five 0s to five 1s, we need to flip five bits. If we have an error on the first codeword that flips the first two bits, so that we have 11000, we can always correct that error, because the majority of the bits are still 0, and so we know that if at most two bits have been flipped we must have come from 00000. So, classically, if we have a code of distance $2t+1$, we can always correct errors up to weight-t. We are defining distance for quantum error correcting codes in analogy to this.

We might hope that the distance d is also the minimal weight error taking one codeword to another codeword. Though this is true for codes we will see later in the course, Professor Wright believes this is not generally true.

2.1 Consequences of the Knill-Laflamme Conditions

2.1.1 Local Indistinguishability

Fact 2.3. *Suppose $\mathcal{C} \subset (\mathbb{C}^2)^{\otimes n}$ has distance $2t+1$. Let $|\psi_1\rangle, |\psi_2\rangle \in \mathcal{C}$, and let $S \subset [n]$ of size $2t$. Then*

$$\mathrm{tr}_{[n]\setminus S}\left(|\psi_1\rangle\langle\psi_1|\right) = \mathrm{tr}_{[n]\setminus S}\left(|\psi_2\rangle\langle\psi_2|\right).$$

That is, for any fixed subset of qubits S, so long as $|S|$ is one less than the distance, the reduced state of any codeword on S is identical. We cannot distinguish between any two states in our code by looking at subsets of size at most $2t$. We must look at at least $2t+1$ qubits.

Proof. What does it mean for these two states to be unequal? There would exist a measurement that distinguishes the states, that is, there's some measurement we can perform which has outcomes that occur with different probabilities. To show the two reduced states are equal, we will pick an arbitrary measurement, and prove it acts identically for any code states $|\psi_1\rangle, |\psi_2\rangle$.

Let Π be a $2t$-qubit projection matrix. Then the probability we get outcome Π is

$$\mathrm{tr}_S(\Pi \cdot \mathrm{tr}_{[n]\setminus S}(|\psi_1\rangle\langle\psi_1|) = \mathrm{tr}\left(\Pi_S \otimes I_{[n]\setminus S} \cdot |\psi_1\rangle\langle\psi_1|\right) = \langle\psi_1|\,\Pi_S \otimes I_{[n]\setminus S}\,|\psi_1\rangle.$$

Since $\Pi_S \otimes I_{[n]\setminus S}$ acts only on S, it is a weight-$2t$ error, and so the Knill-Laflamme conditions tell us that $\langle\psi_1|\,\Pi_S \otimes I_{[n]\setminus S}\,|\psi_1\rangle$ is independent of $|\psi_1\rangle$. In particular, we would get the same thing if we replaced $|\psi_1\rangle$ with $|\psi_2\rangle$. □

There is no classical analog of this result. For example, we can distinguish between the five-bit repetition code's codewords 00000 and 11111 by looking at only a single bit.

As an example of local indistinguishability, consider the Shor code. The basis states of the codespace are:

$$|0\rangle_L = (|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle),$$
$$|1\rangle_L = (|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle).$$

We saw that the Shor code has distance 3. For example, we can correct any single qubit error, but $Z_1 Z_4 Z_7$ takes $|0\rangle_L \to |1\rangle_L$, so we cannot correct this error. If we look at the subset $S = \{1,2\}$, and trace out $[n] \setminus S$, we get

$$\frac{1}{2}|00\rangle\langle 00| + \frac{1}{2}|11\rangle\langle 11|,$$

whether we start with $|0\rangle_L$ or $|1\rangle_L$. The same is true if we look at any two qubits in the same block of three. If we look instead at a subset with two qubits, each from different blocks, for example, the subset $S = \{1,4\}$, we get instead the state

$$\left(\frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1|\right)^{\otimes 2},$$

independent of starting codeword.

2.1.2 Located Errors

Let's consider a code state $|\psi\rangle$ on n qubits. In general, some error will occur on this state, but we won't know ahead of time which error, and in particular, we don't know where that error will act. In a *located error*, we are told exactly which qubits have been affected by noise, so we know where to focus our error recovery. Intuitively, this should make our job easier, since we know more about the error that has occurred, and the hope is that this will allow us to correct more errors. Indeed, there is the following fact:

Fact 2.4. *A distance $2t+1$ QECC can correct any weight-$2t$ located error.*

If we knew where the errors would occur before encoding, we could just not use those qubits. What makes this interesting is that we do not which qubits the errors will act on before encoding.

As an example, consider the classical five-bit repetition code. If, after an error has occured, we have the string 00001, but know the error has occured on the first four bits, then we know for sure the original codeword was 11111, since the last bit is correct. In this example, we see that we correct errors acting on up to four bits, which is one less than the distance. Fact 2.4 claims that the same holds for a quantum error correcting code.

Proof. How do we prove a QECC corrects a set of errors? By showing the Knill-Laflamme conditions are satisfied for those errors.

So, let $S \subset [n]$ be the location of the errors, with $|S| \leq 2t$. The set of errors we want to correct is $\mathcal{E} = \{\text{weight } 2t \text{ errors on } S\}$. We want to look at inner products of the form $\langle\psi| E_1^\dagger E_2 |\psi\rangle$ for $E_1, E_2 \in \mathcal{E}$. Since E_1 and E_2 act on the same $2t$ qubits S, their product still only acts on the qubits in S, and so $E = E_1^\dagger E_2$ has weight $2t$, so that the Knill-Laflamme conditions (for the original QECC, with non-located errors) tell us

$$\langle\psi| E_1^\dagger E_2 |\psi\rangle = \langle\psi| E |\psi\rangle = O_E.$$

Therefore, the Knill-Laflamme conditions (for $\mathcal{E}$) are satisfied. □

2.1.3 Degeneracy

Definition 2.5. A code $\mathcal{C}$ correcting errors $\mathcal{E}$ is *degenerate* if there exist distinct errors $E_1, E_2 \in \mathcal{E}$ such that $E_1 |\psi\rangle = E_2 |\psi\rangle$ for all $|\psi\rangle \in \mathcal{C}$.

For example, the Shor nine-qubit code is degenerate: $Z_1 |\psi\rangle = Z_2 |\psi\rangle$ for all states $|\psi\rangle$ in the code.

One thing we can look as is the difference $(E_1 - E_2) |\psi\rangle = 0$. Since $\mathcal{E}$ is a linear subspace, $E_1 - E_2$ is an error in $\mathcal{E}$ as well, and so we have a nonzero error $E \in \mathcal{E}$ so that $E |\psi\rangle = 0$ for all codewords $|\psi\rangle$. For example, in the Shor code, $Z_1 - Z_2$ maps all codewords to zero.

We can give an alternative characterization of when a code is degenerate in terms of the Knill-Laflamme conditions:

Fact 2.6. *Let $E_1, \ldots, E_m$ be a basis for $\mathcal{E}$. Set $O_{a,b} = \langle\psi| E_a^\dagger E_b |\psi\rangle$ for some $|\psi\rangle \in \mathcal{C}$ (it doesn't matter which $|\psi\rangle$ we use, by the Knill-Laflamme conditions). Then $\mathcal{C}$ is a non-degenerate code if and only if $O_{a,b}$ has full rank.*

This gives us a nice linear algebraic way to detect whether our code is degenerate or not.

Proof. First we prove that if $\mathcal{C}$ is degenerate, then $O_{a,b}$ does not have full rank. Let $E \in \mathcal{E}$ be a nonzero error so that $E |\psi\rangle = 0$. Expand E as $E = \sum_i \alpha_i E_i$, and extract the vector of coefficients $|\alpha\rangle = \sum_i \alpha_i |i\rangle$. Now we will look at the quantity $\langle i| O |\alpha\rangle$ for a generic basis state $|i\rangle$. Our goal is to show that this inner product is zero, which implies that $O |\alpha\rangle = 0$, and therefore O does not have full rank, since $|\alpha\rangle$ is an eigenvector with eigenvalue 0. We have

$$\begin{aligned}
\langle i| O |\alpha\rangle &= \langle i| O \left(\sum_j \alpha_j |j\rangle \right) \\
&= \sum_j \alpha_j O_{i,j} \\
&= \sum_j \alpha_j \langle\psi| E_i^\dagger E_j |\psi\rangle \\
&= \langle\psi| E_i^\dagger \left(\sum_j \alpha_j E_j \right) |\psi\rangle \\
&= \langle\psi| E_i^\dagger E |\psi\rangle \\
&= 0.
\end{aligned}$$

Now we show the converse. If $O_{a,b}$ does not have full rank, then there is a zero eigenvector $|\alpha\rangle = \sum_j \alpha_j |j\rangle$, and we claim $E = \sum_i \alpha_i E_i$ is a nonzero error so that $E |\psi\rangle = 0$. We have, using the above computation,

$$\langle\psi| E_i^\dagger E |\psi\rangle = \langle i| O |\alpha\rangle = 0.$$

Since this holds for all i, we can take a linear combination

$$0 = \sum_i \alpha_i^\dagger \langle\psi| E_i^\dagger E |\psi\rangle = \langle\psi| E^\dagger E |\psi\rangle = ||E|\psi\rangle||^2.$$

So $E|\psi\rangle = 0$. Since $O_{a,b} = \langle\psi| E_i^\dagger E_j |\psi\rangle$ does not depend on which code state $|\psi\rangle$ we use, we have $E|\psi\rangle = 0$ for all $|\psi\rangle \in \mathcal{C}$.

□

Later on we will see some bounds on quantum error correcting codes, and one of these will only apply to non-degenerate codes. However, many of the codes that we will see are degenerate (e.g. stabilizer codes). Degeneracy can be a positive though, since some combinations of errors will have no effect on code states, and therefore automatically resilient to some types of noise. Sometimes this can be used to the code's advantage.

3 Classical Linear Codes

Now we move on to several lectures on how to actually construct quantum error correcting codes. So far, we have been discussing QECCs abstractly, with only the Shor code as an example. We really need a systematic way to design QECCs, but this seems difficult because a QECC is just a subspace, and there are so many subspaces. Which ones should we try?

In the mid-90s, people figured out a systematic way to design QECCs, that makes it so we are able to use a lot of our intuition from classical error correction.

To motivate all of this, we need some definitions from classical error correction, specifically from linear error correcting codes. After a brief introduction to linear codes, we will use our intuition from this to build new quantum codes.

Definition 3.1. A *classical code* is a subset $\mathcal{C} \subset \{0,1\}^n$. It is *linear code* if $\mathcal{C}$ forms a subspace: for all $x, y \in \mathcal{C}$, $x + y \in \mathcal{C}$, where $x + y$ is bitwise addition modulo 2:

$$x + y = \big(x_1 + y_2 \pmod 2, \ldots, x_n + y_n \pmod 2\big) \in \mathcal{C}.$$

If $\mathcal{C}$ is a linear code, then we can take a basis for $\mathcal{C}$, $\{g_1, \ldots, g_k\}$, and write $\mathcal{C} = \text{span}\{g_1, \ldots, g_k\} = \{b_1 g_1 + \cdots + b_k g_k \,|\, b_1, \ldots, b_k \in \{0,1\}\}$. The dimension of the code is $\dim(\mathcal{C}) = k$, and we can think of the code as encoding k bits (the b_i's) into n bits.

As a running example, consider the three-bit repetition code: $\mathcal{C} = \{000, 111\}$. This is a linear code, and with $g_1 = 111$, $\{g_1\}$ is a basis.

Definition 3.2. The *generator matrix* is the matrix

$$G = \begin{bmatrix} — & g_1 & — \\ — & g_2 & — \\ & \vdots & \\ — & g_k & — \end{bmatrix},$$

where $g_1, \ldots, g_k$ is a basis of $\mathcal{C}$.

Let $b = (b_1, \ldots, b_k) \in \{0,1\}^k$, then $b \cdot G^T = b_1 \cdot g_1 + \cdots + b_k \cdot g_k \in \mathcal{C}$.

The standard way to check if some bitstring x is in the code is by using *parity checks*. For example, consider the three-bit repetition code again. To check if x is in the code, we check if pairs of adjacent bits are equal: if every pair is equal, then all the bits are equal, and the bitstring is in the code. We can accomplish this by measuring the parity of different subsets of bits of x.

For this example, we would check the parity of pairs of adjacent bits. We can describe our check with a bitstring $h_1 = 110$, which means that we check the parity of the sum of the bits where we have 1s. Alternatively, we take the dot product of the bitstring h_1 with x: $h_1 \cdot x_1 = x_1 + x_2$. This is zero if and only if x_1 and x_2 are the same, so for any bitstring x in the code, we must have $h_1 \cdot x = 0$. We also want to check if the second and third bits are the same, and $h_2 = 011$ is the corresponding parity check. We could also check if the first and last bit are equal with the parity check $h_3 = 101$, but this is redundant given h_1 and h_2.

Definition 3.3. A *parity check* is a string $h \in \{0,1\}^n$ such that $h \cdot c = 0 \pmod 2$ for all codewords $x \in \mathcal{C}$.

Note that if $h \cdot x = 0$ and $h \cdot y = 0$, then $h \cdot (x + y) = 0$, so that parity checks behave nicely with linear codes. This would not have been the case if we had asked for parity checks of 1.