6.443/8.371/18.436 Quantum Information Science II Lecture 6

Clifford group

Aram Harrow February 26, 2018

Topics:

- 5-qubit code
- Clifford Group

6.1 5 qubit code

We can define the stabilizer group for the 5-qubit code:

$$S = \langle ZXXZI, IZXXZ, ZIZXX, XZIZX \rangle \tag{6.1}$$

Later we will want to add a 5th "redundant" generator to the group:

$$S = \langle ZXXZI, IZXXZ, ZIZXX, XZIZX, XXZIZ \rangle \tag{6.2}$$

This last operator is technically not necessary to generate the group, since it is the product of the first four generators. But it adding it does give the set a nice cyclic symmetry whose properties we will find useful shortly.

We want to compute $N(S)$, which are the normalizers of S. These are the group of Paulis P_n which commute with all elements of the stabilizer group $N(S) = \{p \in P_n | pSp^\dagger = S\}$.

We can find that there are two Paulis which we can add to the generators of S to form $N(S)$:

$$S = \langle S, XXXXX, ZZZZZ \rangle \tag{6.3}$$

We call these last two Paulis the "logical operators", logical X and logical Z.

$$\bar{X} = XXXXX \tag{6.4}$$

$$\bar{Z} = ZZZZZ \tag{6.5}$$

A key property of these operators is that they act like "X" and "Z" within the logical space of the code words, that is they have the suitable $\bar{X}^2 = 1$ properties, and that $\bar{Z}\,|\bar{1}\rangle = |\bar{1}\rangle$ and $\bar{Z}\,|\bar{0}\rangle = -\,|\bar{0}\rangle$, and $\{\bar{X}, \bar{Z}\} = 0$, etc.

Claim 1. *Any single-qubit error leads to a distinct syndrome.*

Example 1. Let's assume we have the error $X_1 = X \otimes I \otimes I \otimes I \otimes I$. The syndrome is just a table indicating which stabilizer commute or anticommute with the given error. For this we find the syndrome is $[1, -1, 1, -1, -1]$ (the 5th column is for the 5th redundant stabilizer). If we instead use $\mathbb{F}_2$ notation the syndrome is $[1, 0, 1, 0, 0]$, where the 1 indicates anticommuting and the 0 indicates commuting.

Now let's say we want to find syndromes for X_2, X_3, etc. Do we have to recompute everything from scratch? The benefit of adding the 5th column and adding the cyclic symmetry can now becomes obvious: all we have to do is cyclically shift our syndrome:

X_1	1	0	1	0	0
X_2	0	1	0	1	0
X_3	0	0	1	0	1
X_4	1	0	0	1	0
X_5	0	1	0	0	1

We can also find for Z:

Z_1	0	0	0	1	1
Z_2	1	0	0	0	1
Z_3	1	1	0	0	0
Z_4	0	1	1	0	0
Z_5	0	0	1	1	0

To find syndrome for Y, we can just add these tables together:

Y_1	1	0	1	1	1
Y_2	1	1	0	1	1
Y_3	1	1	1	0	1
Y_4	1	1	1	1	0
Y_5	0	1	1	1	1

If we have a single qubit Pauli error, we can uniquely identify it using the syndrome and repair the state (by acting with $E_a^\dagger$). By linearity this extends to all single-qubit errors.

It turns out the error correcting condition is sufficient but not necessary. Actually we don't need all of this information to repair the code. For the Shor code, we can find that two different errors can cause the same syndrome. This doesn't matter as long as we know how to get back.

We can split things into two types of codes. Degenerate codes: some errors collapse onto the same error syndrome. Nondegenerate codes: all errors have a unique syndrome.

In practice we get a small advantage from using degenerate codes over nondegenerate codes. Still not obvious how to exploit these properties for something really useful.

In general for the normalizer group we have:

$$N(S) = \langle S, \bar{X}_1 ... \bar{X}_k, \bar{Z}_1 ... \bar{Z}_k \rangle \tag{6.6}$$

$\{\bar{X}_i, \bar{Z}_j\} = 0$ iff $i = j$ else $[\bar{X}_i, \bar{X}_j] = [\bar{Z}_i, \bar{Z}_j] = 0$

Example 2. Trivial Code

$V_S \in \{|0\rangle^{\otimes n-k} \otimes |\Psi\rangle : |\Psi\rangle \in \mathbb{C}^{2^k}\}$.

Our code space is very simple, we only care about the first $n-k$ qubits and we don't care about the last k. The generators for the stabilizer group are $S = \langle Z_1, Z_2 .... Z_{n-k}\rangle$ and $N(S) = \langle S, Z_{n-k+1} ... Z_n, X_{n-k+1} ... X_n\rangle$. Schematically this looks similar to the other non-trivial codes we discussed earlier.

Claim 2. *$\forall$ stabilizer code C (stabilized by S) , there exists unitary U such that UC (stabilized by $USU^\dagger$) is equal to the trivial code.*

To understand how to prove this we need to first prove how we can transform the codes at all under some unitary transformation U:

$$\begin{aligned} V_S \mapsto UV_S, UV_S = \{U|\Psi\rangle : |\Psi\rangle \in V_S\} \\ \forall g \in S, UgU^\dagger U|\Psi\rangle &= Ug|\Psi\rangle \\ &= U|\Psi\rangle \end{aligned}$$

so

$$UV_S = V_{USU^\dagger} \tag{6.7}$$

What we've shown is that we can perform transformations on the stabilizers and the code subspace into a new basis. We need the Clifford group to help complete this story.

6.2 Clifford Group

The key idea is that we want to find the group of unitary operators such that $UPU^\dagger \in P_n, \forall P \in P_n$. Formally: $Cl_n = \{U : UPU^\dagger \in P_n \, \forall P \in P_n\}$.

What are some things we know imediately about the Clifford group? First $P_n \subseteq Cl_n$, and that since $HXH^\dagger = Z$, and $HYH^\dagger = X$, the Hadamard gates $H_i \in Cl_n$.

Why don't we keep track of Y in all of this? We don't need it because if we know how X and Z behave, we get Y's behavior too, just like earlier. Formally $X_1, \ldots, X_n, Z_1, \ldots, X_n$ generate P_n as a group, meaning that multiplying them yields all of P_n. Similarly, they generate $L(\mathbb{C}^{2^n})$ as an algebra, meaning that taking linear combinations of products of them yields all of $L(\mathbb{C}^{2^n})$.

We can define the Phase Shift gate $S = \sqrt{Z} = \left(\begin{smallmatrix} 1 & 0 \\ 0 & i \end{smallmatrix}\right)$. Acting mechanically on this we can show $SZS^\dagger = Z$ and $SXS^\dagger = iY$.

One of the things we would like our group to have is entanglement. Otherwise we will just keep getting trivial behavior. We can add the SWAP gates and CNOT gates. $\text{SWAP}_{ij}\, X_i \,\text{SWAP}_{ij}^\dagger = X_j$. CNOT's action is more complicated:

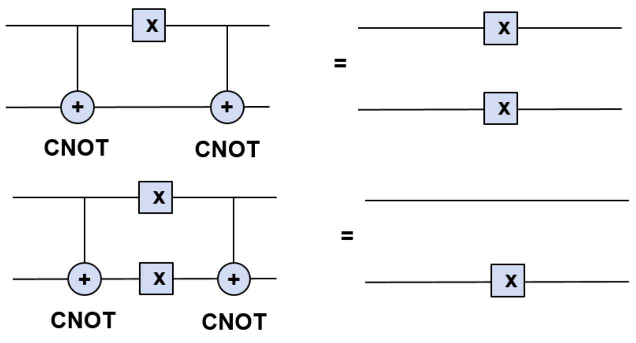

Figure 6.1: How we derive the action of CNOT X CNOT†.

P	$\text{CNOT}_{ij}\, P \,\text{CNOT}_{ij}^\dagger$
X_1	$X_1 X_2$
X_2	X_2
Z_1	Z_1
Z_2	$Z_1 Z_2$

Claim 3. $Cl_n = \langle H_i, S_j, \text{CNOT}_{ij} \rangle$

It turns out with the gates we have enumerated we can generate the entire Clifford group. We don't even need the Pauli's since they can be generated from S and H. It's not obvious that this is true but we will prove this on the PSET.

We want to study the action of the Clifford group from a linear algrebraic perspective. We start by writing:

$$X^a Z^b = \sigma^{\left(\begin{smallmatrix} a \\ b \end{smallmatrix}\right)}, \left(\begin{smallmatrix} a \\ b \end{smallmatrix}\right) \in F^{2n} \tag{6.8}$$

Taking $U \in Cl_n$, we write: $U\sigma^v U^\dagger = (-1)^{f(v)} \sigma^{g(v)}$, for some functions f, g. To find constraints on these functions, note that (from last lecture) $\sigma^{v_1}\sigma^{v_2} = (-1)^{v_1^T \begin{pmatrix} 0 & I \\ 0 & 0 \end{pmatrix} v_2} \sigma^{v_1+v_2}$, and so

$$\begin{aligned} U\sigma^{v_1+v_2}U^\dagger =& (-1)^{v_1^T \begin{pmatrix} 0 & I \\ 0 & 0 \end{pmatrix} v_2} U\sigma^{v_1}\sigma^{v_2}U^\dagger \\ =& (-1)^{v_1^T \begin{pmatrix} 0 & I \\ 0 & 0 \end{pmatrix} v_2} U\sigma^{v_1}U^\dagger U\sigma^{v_2}U^\dagger \\ =& (-1)^{f(v_1)+f(v_2)+v_1^T \begin{pmatrix} 0 & I \\ 0 & 0 \end{pmatrix} v_2} \sigma^{g(v_1)}\sigma^{g(v_2)} \\ =& (-1)^{f(v_1)+f(v_2)+v_1^T \begin{pmatrix} 0 & I \\ 0 & 0 \end{pmatrix} v_2 + g(v_1)^T \begin{pmatrix} 0 & I \\ 0 & 0 \end{pmatrix} g(v_2)} \sigma^{g(v_1+v_2)} \end{aligned}$$

On the other hand, this also equals $(-1)^{f(v_1+v_2)}\sigma^{g(v_1+v_2)}$. We conclude that g is linear, i.e. $g(v_1 + v_2) = g(v_1) + g(v_2)$. Since g is constrained to act this way, we can write $g = Mv$ where $M \in \mathbb{F}_2^{2n}$.

We also obtain a more complicated equation for f:

$$f(v_1 + v_2) = f(v_1) + f(v_2) + v_1^T \begin{pmatrix} 0 & I \\ 0 & 0 \end{pmatrix} v_2 + g(v_1)^T \begin{pmatrix} 0 & I \\ 0 & 0 \end{pmatrix} g(v_2). \tag{6.9}$$

This suggests that f is of the form

$$f(v) = \langle \alpha, v\rangle + v_1^T \begin{pmatrix} 0 & I \\ 0 & 0 \end{pmatrix} v_2 + g(v_1)^T \begin{pmatrix} 0 & I \\ 0 & 0 \end{pmatrix} g(v_2) \tag{6.10}$$

for some arbitrary $\alpha \in \mathbb{F}_2^{2n}$. We will not make use of this form, though.

Returning to g, we can ask whether all matrices M are possible. Clearly $M = 0$ is not, since it would imply that $U\sigma^v U^\dagger$ would always be the identity matrix, which is impossible. More generally, we know that conjugating by U shouldn't change whether two Paulis commute or anticommute.

To understand how this works, let's develop some notation for the commuting/anticommuting condition. The sympletic inner product is defined as:

$$(v, w) = v^\intercal \Lambda w, \tag{6.11}$$

where $\Lambda := \left(\begin{smallmatrix} 0 & I \\ I & 0 \end{smallmatrix}\right)$. When $(v, w) = 1$ then σ^v and σ^w anticommute, and when $(v, w) = 0$ then σ^v and σ^w commute. If we perform a unitary transform on σ^v and σ^w, then it should preserve commutation/anticommutation, and so the corresponding transformation M (from above) should preserve the symplectic inner product:

$$\begin{aligned} v^\intercal \Lambda w =& (Mv)^\intercal \Lambda M w \\ =& v^\intercal M^\intercal \Lambda M w \end{aligned}$$

. Since this holds for any v, w, we have

$$\Lambda = M^\intercal \Lambda M \tag{6.12}$$

When M obeys this condition we say that M is a symplectic matrix or a symplectic transformation. It is just another way of saying that it preserves the inner product defined by Λ. To relate this to something familiar, note that if we set $\Lambda = I$ in (6.12) then we are saying that M preserves the usual (real) inner product, i.e. that M is orthogonal.

To summarize, we can express elements of the Clifford group as symplectic matrices M, along with a vector α that affects only the phase of the output.

Example 3. Application: Any stabilizer code is equivalent to a trivial code

You will prove on the pset that any transformation that respects commutation relations (i.e. any symplectic matrix) is part of the Clifford group. One application is that we can transform any stabilizer code to a trivial code (discussed above). In general these transformations do not preserve distance, which is helpful because the trivial code has a bad distance. But still it can help us understand the structure of general stabilizer codes.

Example 4. Gottesman-Knill Theorem

Quantum computations are sometimes said to be hard because of entanglement. But the Gottesman-Knill theorem describes a large class of quantum circuits that can be simulated even though they can create large amounts of entanglement.

Suppose we start in a state $|0\rangle^{\otimes n}$. We can perform a sequence of Clifford gates onto this starting state to transform it. We can perform measurements in the Z-basis and even perform adaptive measurements, meaning that we perform measurements and then based on the results of those measurements perform different unitaries and measurements next.

Claim 4. *We can simulate such circuits in polynomial time classically.*

Proof. We track the stabilizers $Z_1 ... Z_n$:

1. Start with stabilizer state."stabilizer state" means stabilized by n independent generators.

2. To perform a unitary U, replace the stabilizers with $UZ_1U^\dagger, ...UZU_n^\dagger$. Or more generally if the stabilizers are $s_1, \ldots, s_n$, replace them with $Us_1U^\dagger, \ldots, Us_nU^\dagger$.

3. Do measurement. For example, say we measure Z_1.

   - If $Z_1 \in S$ then we get outcome $+1$. The state was an eigenstate, so measurement causes no change.
   - $Z_1 \in S$, then likewise we get outcome -1 and there is no change.
   - Say $\pm Z_1 \notin S$, which means we anticommute with one or more stabilizer generators, and the state will change.

     First we need to choose an appropriate set of stabilizer generators. Replace $s_1, \ldots, s_n$ with $s'_1, \ldots, s'_n$ such that $S = \langle s_1, \ldots, s_n \rangle = \langle s'_1, \ldots, s'_n \rangle$ and Z_1 anticommutes only with s'_1 and commutes with $s'_2, \ldots, s'_n$. Why is this possible? Because Z_1 commutes with at least one of the s_i. Relabel them so one of these is s_1. For each s_j that anticommutes with Z_1, replace it with $s_1 s_j$. This doesn't change S, but the new generator now commutes with Z_1. Call the resulting generators $s'_1, \ldots, s'_n$.

     Next, since s'_1 anticommutes with Z_1 it must be $X_1 g$ where g acts on qubits $2, \ldots, n$. Now we update the stabilizers by throwing out s'_1 and replacing it with $\pm Z_1$, depending on whether the outcome of the measurement was ± 1 (which we choose randomly). By doing this we can update the stabilizer to match the current code.

□