CMSC 657: Introduction to Quantum Information Processing Lecture 20

Instructor: Daniel Gottesman

Fall 2024

1 Example Stabilizer Codes

1.1 CSS Codes

Example 1 (Seven-qubit code). *A* $[[7,1,3]]$ *code.*

$$\begin{array}{ccccccc}
X & X & X & X & I & I & I \\
X & X & I & I & X & X & I \\
X & I & X & I & X & I & X \\
Z & Z & Z & Z & I & I & I \\
Z & Z & I & I & Z & Z & I \\
Z & I & Z & I & Z & I & Z
\end{array}$$

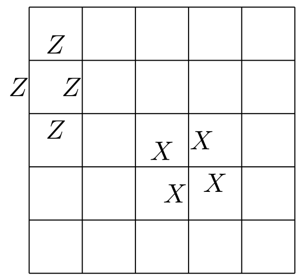
The first three generators and the last three generators are two classical codes (in this case both are the 7*-qubit Hamming code) converted to a quantum code by taking* 1*'s in the parity check matrices and converting them to* X*'s or* Z*'s.*

Once again, the logical Paulis $\overline{X}$ *and* $\overline{Z}$ *can be taken to be tensor products of* X *and* Z *on all qubits. Note, though, that this is not true for all codes.*

In general, we can define a classical linear code by a parity check matrix, which is basically a stabilizer with only Paulis I or Z. We can make a QECC out of two classical linear codes C_1 and C_2 if $C_2^{\perp} \subseteq C_1$. Codes formed in this way are known as CSS codes. For those already familiar with classical linear codes, each row of the parity check matrix corresponds to a generator of the stabilizer. We can convert the 1s in one of the two codes into Z's in the stabilizer (and convert 0s to Is); for the other code, we convert 1s into X's. The distance of the quantum code (related to the number of errors it can correct) is at least the distance of the smaller of C_1 and C_2.

1.2 Toric Code

Surface codes are another family of stabilizer codes that has emerged as one of the leading candidates for building a large quantum computer. The original surface code is called the *toric code*, because the qubits in it are naturally laid out on the surface of a torus (a donut).



In this picture, the left and right edges are actually the same edge, as are the top and bottom edges. Thus, the picture wraps up into a torus. The qubits sit on the *edges* of this graph. Each square (*plaquette* in the common terminology here) corresponds to a single stabilizer element on the four qubits on the edges of the square, and each vertex (sometimes called a *star* here) corresponds to a stablizer element on the four qubits whose edges are incident on the vertex. The plaquette operators are products of Zs:

$$A_P = Z \otimes Z \otimes Z \otimes Z \tag{1}$$

and the star operators are products of Xs:

$$B_s = X \otimes X \otimes X \otimes X. \tag{2}$$

Notice that each star and each plaquette either don't overlap at all or share exactly two edges. Thus, every A_P commutes with every B_s. Also, all the A_P's commute with each other because they are products of Zs and the B_s's commute with each other because they are products of Xs.

How many physical and logical qubits does this code have? If the grid is $(L+1) \times (L+1)$ vertices, there are a total of L^2 horizontal edges and L^2 vertical edges (because the top and bottom rows of the grid are actually the same and the left a right columns are the same), for a total of $2L^2$ qubits. There are a total of L^2 star operators (again recalling the wrapping) and L^2 plaquette operators, so it might appear we have $2L^2 - 2L^2 = 0$ logical qubits. However, some of the plaquette and star operators are redundant, so there are actually fewer generators of the stabilizer than this. If we take the product of all the A_P operators, each edge gets used twice, and so the product is I. Therefore, any one of the A_P's can be written as the product of all the others. Similarly, the product of all of the B_s operators uses each edge twice, so again one of the B_s's can be written as the product of the others. These turn out to be the only relationships between the operators. Therefore, there are actually $2L^2 - 2$ generators, leading to 2 logical qubits.

If you think about other products of A_P operators, you will see that they always form a set of closed loops: whenever two generators overlap, the shared qubit has the Z's cancel out, making a loop. Similarly for products of B_s operators, although to see the loops more clearly you should take perpendicular lines through the edges (giving the *dual lattice*). Importantly, these loops (for both X and Z) are always *contractible* because they are formed by putting together size one loops. If you have a loop that goes all the way around the torus (e.g., off the right end to the left end), it will automatically commute with all generators because it crosses two edges from each star (if it is in the primal, original, lattice) or plaquette (if it is in the dual lattice). Therefore, it is in $N(S)$, and is actually in $N(S) \setminus S$ since it is non-contractible. Indeed, all elements of $N(S) \setminus S$ are (products of) non-contractible loops. The logical $\overline{X}$ and $\overline{Z}$ on the two encoded qubits correspond to horizontal and vertical loops, $\overline{Z}$ on the primal lattice and $\overline{X}$ on the dual lattice.

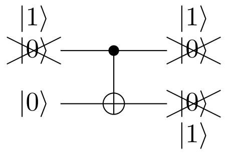
One consequence of this is that the smallest elements of $N(S) \setminus S$ are associated to the smallest loops that wrap all the way around the torus. The length of a minimal loop is L, so the distance of the code is L. The toric code is thus a $[[2L^2, 2, L]]$ code.

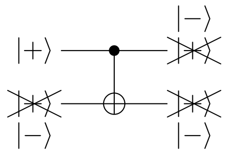
2 Fault Tolerant Gates

A *fault-tolerant protocol* is a set of actions that let you do quantum circuits with qubits that are encoded with a quantum error-correcting code while still retaining the protection against errors. For instance, you might imagine doing gates on encoded qubits by decoding, doing the gate, and then re-encoding, but this faces two problems: First, the encoding and decoding procedures themselves can be subject to error, and if we are not careful, an error in either of those steps could ruin the information we are trying to protect. Second, while you are doing the gate, the state is not protected and an error then will change the logical qubit.

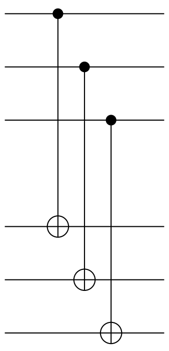
Typically, a fault-tolerant protocol consists of a set of *gadgets*. Each gadget represents one operation that you would want to do on logical qubits and tells you how to do that operation on encoded qubits in a way resistant to error. There will be a gadget for each kind of fault-tolerant gate, gadgets for creating encoded $|0\rangle$ states and maybe other specific states, and a gadget or gadgets for measuring the logical qubits. We will also need a gadget for performing fault-tolerant error correction, since any errors that occur during error correction itself can potentially cause a major problem if we are not careful.

A major concern for fault tolerant protocols is to prevent *error propagation*. For instance, if we perform a CNOT, errors can propagate from the control to the target, but they can also propagate backwards from the target to the control:





One error could become two errors in a single block of the code, which might be more than the code can correct. A solution is to use *transversal* gates — gates which only interact qubits with corresponding qubits in other blocks of the code. For instance, the transversal CNOT looks like this:

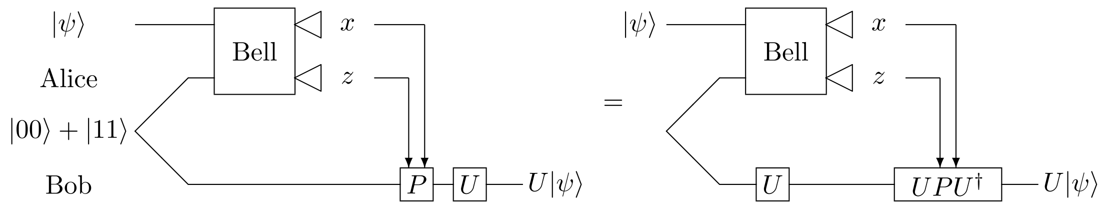


The 7-qubit code is particularly nice for fault tolerance because we can do a lot of gates transversally, including CNOT, Hadamard, $R_{\pi/4}$. The logical CNOT and Hadamard are both implemented as tensor products of 7 CNOTs or Hadamards. The logical $R_{\pi/4}$ is actually the tensor product of 7 $R_{-\pi/4}$ gates instead. Unfortunately, these gates are not universal. In fact, they generate a subgroup of the unitary group called the *Clifford group*, which consists of all unitaries that conjugate the Pauli group into itself:

$$\mathcal{C}_n = \{U \,|\, UPU^\dagger \in \mathcal{P}_n \, \forall P \in \mathcal{P}_n\}. \tag{3}$$

The Clifford group can be efficiently simulated on a classical computer, but is still very useful for dealing with quantum error-correcting codes. For instance, the encoding circuits for a stabilizer code use just Clifford group elements.

To get a universal set of gates, we need additional tricks (but it is possible). One trick is to use quantum teleportation.



Here, Alice and Bob are both parts of the same computer, but all of their qubits are encoded in a quantum error-correcting code. For certain gates U (in a set called C_3, which includes the $R_{\pi/8}$ gate), $UPU^\dagger$ is a Clifford group element, and the Bell measurement also uses just Clifford group operations. Therefore, if we have the capability to do fault-tolerant Clifford group operations and fault-tolerantly prepare the state $|\phi\rangle = (I \otimes U)(|00\rangle + |11\rangle)$ of two logical qubits, then we can do the gate U fault-tolerantly. This state $|\phi\rangle$ is known as a *magic state*, and preparing it is not easy. One common solution is to make many encoded $|\phi\rangle$ states in a non-fault-tolerant way and then compare them against each other to produce more reliable copies, a process known as *magic state distillation.* Since the Clifford group plus $R_{\pi/8}$ forms a universal set of gates, this gives us enough gates to do any quantum computation reliably.