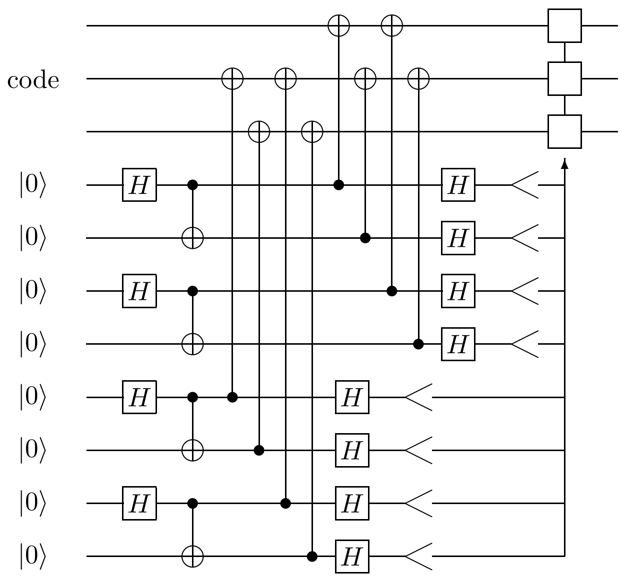
Solution Set #4

CO 639: Quantum Error Correction
Instructor: Daniel Gottesman

Problem 1. Phase Error Correction

a) Let us use Shor error correction, so the ancillas are two-qubit cat states. We don't actually need to do any ancilla verification for this circuit, for two reasons: First, only phase errors occur in this system, and they cannot propagate into the data. Second, even if bit flip errors could occur, flipping both qubits of a two-qubit cat state leaves the state invariant, so error propagation from a single ancilla state could never lead to more than one error in the data.



b) The code is a CSS code, so transversal CNOT is a valid operation. H and P are not, as they change the stabilizer generators into tensor products of Zs and Xs, respectively. The CNOT performs the logical CNOT between data encoded in the two blocks of the code.

Another valid transversal operation is $Q^{\otimes 3} = Q \otimes Q \otimes Q$, where under Q, $X \mapsto X$ and $Z \mapsto Y$; thus the stabilizer is preserved. It performs the logical $Q^\dagger$ operation, but is not fault-tolerant (see part d).

c) We choose a stabilizer code with generators $Y \otimes Y \otimes I$ and $I \otimes Y \otimes Y$. CNOT is longer a transversal operation (although a rotated version of it is valid). Transversal P and Q are not valid either, as both of them change Y. However, $H^{\otimes 3}$ is valid, taking $Y \otimes Y \otimes I$ to $(-1)^2 Y \otimes Y \otimes I$, and it performs the logical operation H (assuming that $\overline{X} = X \otimes X \otimes X$ and $\overline{Z} = Z \otimes Z \otimes Z$).

d) For instance, we could use the code from the previous part. However, it is not fault-tolerant: Hadamard converts phase errors to bit flip errors (Z to X), and while the code can correct Z errors, it cannot also correct bit flip errors. Thus, if one correctable error occurred before the Hadamard, it would become an uncorrectable error after the gate.

Problem 2. Transversal Operations For Any Stabilizer Code

a) The transformation defines a unitary operation if: All the values $A(X_i)$ and $A(Z_i)$ are independent of each other; $[A(X_i), A(X_j)] = [A(Z_i), A(Z_j)] = [A(X_i), A(Z_j)] = 0$ for $i \neq j$; and $\{A(X_i), A(Z_j)\} = 0$. The first condition (independence) is true whenever the vectors $\vec{a}_i = (A_{i1}, A_{i2}, \ldots, A_{in})$ for $i = 1, \ldots, n$ are linearly independent. We always have that $[A(X_i), A(X_j) = [A(Z_i), A(Z_j)] = 0$.

The remaining two conditions give us strong constraints on the values of A_{ij}: $[A(X_i), A(Z_j)] = 0$ iff $\vec{a}_i \cdot \vec{a}_j = 0$, and $\{A(X_i), A(Z_j)\} = 0$ iff $\vec{a}_i \cdot \vec{a}_i = 1$. These dot products are evaluated using binary arithmetic. They are equivalent to saying that the matrix A_{ij} is orthogonal (over the binary field). From this linear independence follows, so we conclude that the operation defines a valid unitary iff the matrix A_{ij} is orthogonal.

Since the mapping takes Pauli group operations to other Pauli group operations, it defines a Clifford group operation whenever it is a valid unitary, that is when A_{ij} is an orthogonal matrix.

b) Actually, this is not quite true: The operation is a valid transversal operation for any *real* stabilizer code. For other stabilizer codes, whether it is a valid transversal operation or not depends on the location of the Ys in the stabilizer and the weight of the vectors $\vec{a}_i$. We know these vectors have odd weight, but it is relevant whether their weights are congruent to 1 or 3 mod 4; if all are congruent to 1 mod 4, the operation really is transversal for all stabilizer codes.

To see this, note that

$$A(Y_i) = i^{1-\mathrm{wt}(\vec{a}_i)} \prod_{j=1}^{n-1} Y_j^{A_{ij}}. \tag{1}$$

If the code is real, all elements of the stabilizer are a product of Xs, Zs, and an even number of Ys. The transformation A therefore takes an arbitrary M of this form on the ith copy of the code to

$$A(M_i) = \prod_{j=1}^{n-1} M^{A_{ij}}. \tag{2}$$

This is clearly in the stabilizer $S \times S \times \cdots \times S$ of n copies of the same code.

If $\mathrm{wt}(\vec{a}_i) = 1 \bmod 4$ for all i, then $A(Y_i)$ always has sign $+1$, so (2) holds regardless of whether or not the code is real, and therefore, the operation is transversal for all stabilizer codes.

If the code is not real and the values of $\mathrm{wt}(\vec{a}_i)$ are not all 1 mod 4, there could be some generators M_i that are mapped to $-M_i$. There are two possible ways to deal with this: First, there is always some Pauli operator that will map $-M_i \mapsto M_i$ without changing any other generators of the stabilizer. Using an appropriate set of these Pauli operators, we can therefore return the code to its original stabilizer. The other option is to note, as per problem 4d of problem set 3, that any stabilizer code is equivalent to a real code. Either way, we get a transversal operation for an arbitrary stabilizer code.

Note that the transformation A performs the same transformation on the encoded state (possibly up to a logical Pauli operation).

c) Consider the code on 3 qubits with a single stabilizer generator $X \otimes Y \otimes Z$. Any element of the n-fold stabilizer $S \times \cdots \times S$ will have Xs or Is in the first qubit of each set of three. Therefore, any valid transversal operation must map X to a tensor product of Xs and Is. Similarly, it must map Y to a tensor product of Ys and Is, and must map Z to a tensor product of Zs and Is. Furthermore, in all cases, these must be the same tensor product, because all elements of the n-fold stabilizer keep X, Y, and Z together on a set of three qubits. This proves the claim.

d) By parts a) and b), we need to find a nontrivial orthogonal matrix over the binary field. This will give us a operation that acts transversally on any real stabilizer code. For instance, we could use the matrix:

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}. \tag{3}$$

There is in fact no orthogonal matrix over the binary field smaller than 4×4 except for permutations (including the identity). This is actually quite easy to see: Each row must have weight 1 or 3. For a 2×2 matrix, we clearly thus have just permutations. For a 3×3 matrix, we could only have one row of weight 3 (the rows must be linearly independent), so the other two rows would have weight 1 and would have odd dot product with the row of weight 3.

If we further wish all rows to have weight congruent to 1 mod 4, we must go to 6×6 so that we can have rows of weight 5. We can use the following matrix:

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}. \tag{4}$$

Problem 3. Repeating Syndrome Measurement

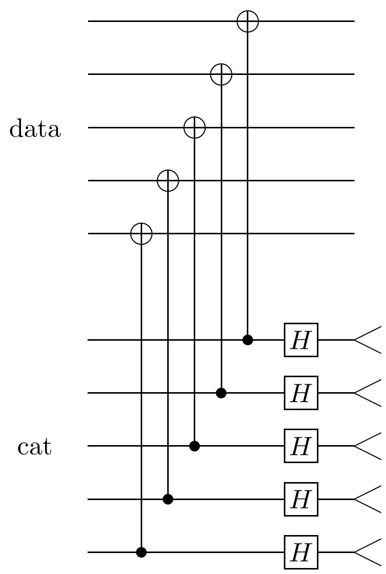
a) If we have an error halfway through syndrome measurement, we can get a syndrome which half reflects this error and half does not. For instance, imagine we are measuring the bit flip error syndrome with generators $Z \otimes Z \otimes Z \otimes Z \otimes I \otimes I \otimes I$, $Z \otimes Z \otimes I \otimes I \otimes Z \otimes Z \otimes I$, and $Z \otimes I \otimes Z \otimes I \otimes Z \otimes I \otimes Z$. Suppose a bit flip error X_1 on the first qubit occurs after measuring the eigenvalue of the first generator, but before measuring the eigenvalue of the second and third generators. Then the syndrome will be 011 instead of the correct 111. Indeed, 011 is the syndrome for X_5, bit flip on the fifth qubit, so we will think that that is the actual error. If we do not repeat the syndrome measurement, we will therefore try to correct it by flipping the fifth qubit. But the X_1 error has not disappeared in the meanwhile, so now we actually have the error $X_1 X_5$: two errors, even though only one occurred spontaneously.

   Another way this can happen is an error on one qubit of the ancilla which both propagates into the data and shows up in the syndrome, changing a bit of the syndrome, and causing us to perform an erroneous error correction.

b) The correct answer is "no." Actually, you don't really need to repeat the syndrome measurement for Steane error correction (although it does give you more confidence in the answer you get). The reason for this is that any error in the data or the ancilla will only look like an error on the same location, because it shows up as an error in the same location in the classical Hamming code. Contrast this with the situation for Shor error correction, in which changing one bit of the syndrome (as in part a) can make the error look like it is on a different qubit.

   To be more concrete, imagine that there is an error on the first qubit of one of the bit-flip correction ancillas for Steane error correction. It could propagate into the data, causing a phase error on the first qubit of the data block. It could cause an error in the final syndrome measurement, giving us a classical codeword that is erroneous on the first bit. We will interpret this as a bit flip error on the data (if we don't repeat the syndrome measurement), and try to correct it by flipping the first qubit of the data block. This gives us a net bit flip error on the first qubit of the data. It could even do both of these, giving us a Y error on the first qubit of the data. But in all cases, it only gives us one error.

c) Recall that we can let $\overline{X} = X \otimes X \otimes X \otimes X \otimes X$. We thus get the following circuit:



There are a lot of possible places where a single error in this circuit can give the wrong measurement outcome. One obvious example is a single-qubit phase error in the original cat state. This will switch the phase of the cat state $|00000\rangle + |11111\rangle$ to $|00000\rangle - |11111\rangle$ and will give us the opposite answer to the true eigenvalue of the codeword.

A less obvious example is a phase error in a single qubit of the data block. Z_i anticommutes with $\overline{X}$, so it switches the eigenvalue of $\overline{X}$. However, if that error Z_i were first corrected, the eigenvalue would return to its original value. The correct value is the value after an ideal error correction, so a phase error in the data causes the measurement to get the wrong answer.

d) The second example in the previous part, of a phase error in the data, shows that we can repeat the measurement and get the same wrong answer without an additional error occurring: The phase error remains in the data, so repeated measurements will give the same result.

A way to fix this is to do error correction in between measurements of $\overline{X}$. That way, if there is a single preexisting error for the first measurement, it will be corrected by the error correction and will not appear in the second measurement (provided no additional errors occur). If we repeat the measurement three times, performing error correction twice between the measurements (in the sequence Meas, EC, Meas, EC, Meas), we can then get a reliable measurement of $\overline{X}$. Any single error can only cause a single attempt at measurement to fail, so if we take the majority of the three outcomes, we will get the correct answer provided only one error occurred in the circuit as a whole.

Problem 4. Measurements and Stabilizers

a) The set of possible initial states has stabilizer $I \otimes Z$, so to steer the outcome of the $Y \otimes X$ measurement to the $+1$-eigenstate, we must perform $I \otimes Z$ (if the measurement gives a -1 outcome). Then the stabilizer becomes $Y \otimes X$, and to steer to the $+1$-eigenstate of $I \otimes Y$, we must perform $Y \otimes X$ (if measurement initially gives a -1 outcome).

b) Initially, we have (the first line is the stabilizer, and $\overline{X}$ and $\overline{Z}$ as usual represent the logical bit flip and phase flip operations):

$$\begin{array}{cc} & I \otimes Z \\ \overline{X} & X \otimes I \\ \overline{Z} & Z \otimes I \end{array} \tag{5}$$

After measuring $Y \otimes X$, and, if necessary, correcting with $I \otimes Z$, we have:

$$\begin{array}{cc} & Y \otimes X \\ \overline{X} & X \otimes Z \\ \overline{Z} & Z \otimes Z \end{array} \tag{6}$$

Finally, after measuring $I \otimes Y$, and correcting with $Y \otimes X$, if necessary, we get:

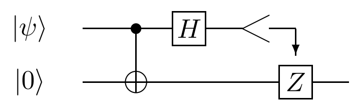
$$\begin{array}{cl} & I \otimes Y \\ \overline{X} & -Z \otimes Y = -Z \otimes I \\ \overline{Z} & X \otimes Y = X \otimes I \end{array} \tag{7}$$

The equalities come from the fact that $I \otimes Y$ is in the stabilizer. We can recognize this action as HZ on the first qubit, the phase flip Z followed by a Hadamard H. The second qubit is replaced by a $+1$-eigenstate of Y.

c) If we had started with the input state $|\psi\rangle \otimes (|0\rangle + |1\rangle)/\sqrt{2}$, the initial stabilizer would have been $I \otimes X$. Then $Y \otimes X$ would commute with the stabilizer, so measuring it would tell us information about the data. In particular, it would have measured Y on the first qubit. Measuring $I \otimes Y$ afterwards would give us no information, just a random result, but would not change the first qubit at all. The overall action would therefore be measurement of Y on the first qubit, plus a Y measurement on the second qubit, which provides a random bit.

Problem 5. Compressed Teleportation Constructions

a) We are looking at a two-qubit circuit, and Alice has one input qubit. We therefore can give Bob a initial qubit in whatever eigenstate is convenient. The CNOT is from Alice to Bob, after which we must make a measurement on Alice's qubit. We wish to have Alice's measurement anticommute with the stabilizer in order to get some kind of nontrivial action; we should therefore pick Bob's initial stabilizer to be Z so the CNOT will propagate it to Alice, and Alice's measurement to be X so it will anticommute. That is, we get the following circuit:



The final Z operation is performed only if Alice's measurement gets the -1 eigenstate (bit value 1); otherwise, Bob does nothing to the state.

We can show that this circuit does in fact perform teleportation by looking at the stabilizer transformations it induces. After the CNOT, but before the Hadamard, we have:

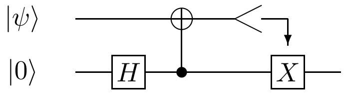
$$\begin{array}{cc} & Z \otimes Z \\ \overline{X} & X \otimes X \\ \overline{Z} & Z \otimes I \end{array} \tag{8}$$

The Hadamard and measurement combined act as a measurement of $X \otimes I$, and the conditional Z operation steers to the $+1$-eigenstate. We thus have:

$$\begin{array}{cl} & X \otimes I \\ \overline{X} & X \otimes X = I \otimes X \\ \overline{Z} & I \otimes Z \end{array} \tag{9}$$

We recognize that the first qubit has been moved to the second qubit — teleportation. One ebit and one bit of classical communication is used.

b) One way to get a circuit would be to note that by performing Hadamards on both qubits before and after a CNOT, we get a CNOT going the other direction. We could also use the same stabilizer insight, which suggests that Bob should start with an eigenstate of X, and that Alice should measure Z. Either way, we get:



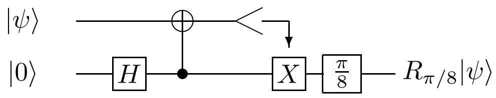
The Hadamard on the measured qubit cancels the one after the CNOT. The initial Hadamard for Alice can be done after teleportation, and it and Bob's Hadamard after the CNOT convert the conditional-Z into a conditional-X. Alternatively, we could follow the stabilizer, as in part a.

c) Recall that the $\pi/8$ gate is a C_3 gate, meaning it conjugates Pauli operators into Clifford group operators:

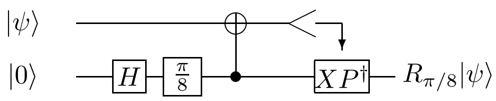
$$X \rightarrow e^{i\pi/4} X P^\dagger \tag{10}$$
$$Z \rightarrow Z, \tag{11}$$

where $P = \mathrm{diag}(1, i)$ is the usual $\pi/4$ rotation from the Clifford group.

We start with the circuit from part b, and imagine performing the $\pi/8$ rotation on the output qubit:



Commuting the $\pi/8$ gate back through the conditional-X converts it into a conditional-$(XP^\dagger)$. (The phase $\exp(i\pi/4)$ is conditioned only on a classical bit, and so has no importance.) The $\pi/8$ gate then commutes with the control qubit of the CNOT with no change, resulting in the circuit:



Given an encoded ancilla

$$R_{\pi/8} H|0\rangle = (|0\rangle + e^{i\pi/4}|1\rangle)/\sqrt{2}, \tag{12}$$

we can then perform the $\pi/8$ gate fault-tolerantly by doing a transversal CNOT from the ancilla to the data block, measuring (in a fault-tolerant way) the encoded qubit in the former data block, and performing encoded $P^\dagger$ followed by encoded X if the measurement result is $|1\rangle$.

The encoded ancilla is a +1-eigenstate of $XP^\dagger$, and can be created fault-tolerantly by measuring $XP^\dagger$, which can be done by performing controlled-(XP) operations from each qubit of a cat state to the corresponding qubits of the 7-qubit code (recall that the transversal P operation performs the encoded $P^\dagger$ for the 7-qubit code). We then perform a Hadamard transformation on every qubit of the cat state and measure it. Repeat the cat state interaction and measurement a few times for increased reliability; if we repeatedly get even parity, we conclude that we have the +1-eigenstate, and we can use it for the main circuit. If we get the -1-eigenstate, we can actually still use the state after performing a logical Z operation. (Z anticommutes with $XP^\dagger$ and therefore converts the -1-eigenstate of $XP^\dagger$ into the +1-eigenstate.)