Solution Set #7

Quantum Error Correction
Instructor: Daniel Gottesman

Problem #1. Concatenated Codes

a) First of all, each block of n_2 qubits of the concatenated code is itself a codeword of S_2, so that gives us part of the stabilizer.

Encoding a qubit in S_2 maps $X \mapsto \overline{X}$, $Y \mapsto \overline{Y}$, and $Z \mapsto \overline{Z}$, where $\overline{X}$, $\overline{Y}$, and $\overline{Z}$ are the logical Pauli operations for S_2. Since we start with a state which is a codeword of S_1, we must perform these replacements in the generators for the stabilizer of S_1 to get the remaining generators of the concatenated code.

Thus, the stabilizer of the concatenated code will have $n_1(n_2-1)+(n_1-k_1)$ generators altogether. Let A_i and B_i be the generators of S_1 and S_2, respectively. For the concatenated code, generator M_{an_2+b} with $a = 0, \ldots, n_1 - 1$ and $b = 1, \ldots, n_2 - 1$ will then have the form

$$M_{an_2+b} = I^{\otimes an_2} \otimes B_b \otimes I^{\otimes(n_1-a-1)n_2}. \tag{1}$$

The remaining generators $M_{n_1(n_2-1)+c}$, $c = 1, \ldots, n_1 - k_1$ are of the form

$$M_{n_1(n_2-1)+c} = \bigotimes_{j=1}^{n_1} \overline{A}_{cj}, \tag{2}$$

where $A_i = \bigotimes_{j=1}^{n_1} A_{ij}$, and $\overline{A}$ is the logical Pauli for S_2 ($\overline{X}$, $\overline{Y}$, or $\overline{Z}$) corresponding to the unencoded Pauli A.

As a concrete example, consider the stabilizer for the concatenated code formed when S_1 is the 4-qubit (distance 2) code and S_2 is the 5-qubit (distance 3) code:

$$\begin{array}{ccccccccccccccccccccc}
X & Z & Z & X & I & & & & & & & & & & & & & & & & \\
I & X & Z & Z & X & & & & & & & & & & & & & & & & \\
X & I & X & Z & Z & & & & & & & & & & & & & & & & \\
Z & X & I & X & Z & & & & & & & & & & & & & & & & \\
 & & & & & X & Z & Z & X & I & & & & & & & & & & & \\
 & & & & & I & X & Z & Z & X & & & & & & & & & & & \\
 & & & & & X & I & X & Z & Z & & & & & & & & & & & \\
 & & & & & Z & X & I & X & Z & & & & & & & & & & & \\
 & & & & & & & & & & X & Z & Z & X & I & & & & & & \\
 & & & & & & & & & & I & X & Z & Z & X & & & & & & \\
 & & & & & & & & & & X & I & X & Z & Z & & & & & & \\
 & & & & & & & & & & Z & X & I & X & Z & & & & & & \\
 & & & & & & & & & & & & & & & X & Z & Z & X & I \\
 & & & & & & & & & & & & & & & I & X & Z & Z & X \\
 & & & & & & & & & & & & & & & X & I & X & Z & Z \\
 & & & & & & & & & & & & & & & Z & X & I & X & Z \\
X & X & X & X & X & X & X & X & X & X & X & X & X & X & X & X & X & X & X & X \\
Z & Z & Z & Z & Z & Z & Z & Z & Z & Z & Z & Z & Z & Z & Z & Z & Z & Z & Z & Z
\end{array} \tag{3}$$

b) Clearly $n = n_1 n_2$, since each of the n_1 physical qubits of S_1 is replaced by n_2 physical qubits for S_2. Also, $k = k_1$. This is because we originally encoded k_1 qubits in S_1 and doing the additional coding into S_2 does not add or subtract any encoded qubits. (This is consistent with our counting of generators in part a: $n_1 n_2 - k_1 = n_1(n_2 - 1) + (n_1 - k_1)$.)

A more difficult question is the distance d of the code. We wish to find the minimum weight of a Pauli in $N(S) \setminus S$. Let us first consider a Pauli operator P acting only on the qubits of a single block of n_2 qubits formed from encoding a single physical qubit of S_1 with S_2. If P is outside $N(S_2)$ (tracing over the qubits outside the block), then P is also outside $N(S)$, as S_2, shifted appropriately, is included in S. By the same token, if $P \in S_2$, then $P \in S$.

Thus we need only consider $P \in N(S_2) \setminus S_2$. We can reinterpret P as a logical Pauli $\overline{Q}$ for S_2, which we know will anticommute with the other logical Paulis for S_2. (Recall that $k_2 = 1$.) But if S_1 has distance 2 or higher, it must either include Q (a single-qubit Pauli operator), in which case $\overline{Q} \in S$, or have an element M which anticommutes with Q, in which case the element of S corresponding to M also anticommutes with $\overline{Q}$, and $\overline{Q} \notin N(S)$. Therefore, if $d_1 \geq 2$, no Pauli acting on a single block of n_2 qubits can be in $N(S) \setminus S$.

Now let us consider the general Pauli operator, which we write as $P = \bigotimes P_i$, with P_i a Pauli acting on the ith block of n_2 qubits. By the logic above, if $P \in N(S) \setminus S$, then each $P_i \in N(S_2) \setminus S_2$. As before, we reinterpret $P_i = \overline{Q_i}$, with $\overline{Q_i}$ a logical Pauli for S_2. It makes sense at this stage to imagine decoding S_2 on each block of n_2 qubits. This decoding procedure therefore maps $P \mapsto Q = \bigotimes Q_i$, and leaves only the stabilizer S_1. Decoding preserves commutation and anticommutation relations, so in order for $P \in N(S) \setminus S$, we need $Q \in N(S_1) \setminus S_1$.

We have thus shown that when $P \in N(S) \setminus S$, it follows that $P = \bigotimes \overline{Q_i}$, with $\overline{Q_i} \in N(S_2) \setminus S_2$ and $Q = \bigotimes Q_i \in N(S_1) \setminus S_1$. It is immediate to look at the stabilizer and see that the converse is true also, that P of this form is in $N(S) \setminus S$. Thus, we see that minimum weight of $P \in N(S) \setminus S$ must be at least the minimum weight of Q (which is d_1) times the minimum weight of each nontrivial $\overline{Q_i}$ (which is d_2), giving $d \geq d_1 d_2$. The concatenated code is thus a $[[n_1 n_2, k_1, d_1 d_2]]$ code.

It is also possible, however, for the distance to be larger than $d_1 d_2$, if the smallest-weight elements of $N(S_1)$ only use Paulis Q_i which correspond to large-weight $\overline{Q_i}$. An example of this is the 9-qubit code, which can be viewed as a concatenated phase error correcting code with a bit flip error correcting code. Each code, viewed as a quantum code, has distance 1, but they compensate for each others' shortcomings, so the overall code has distance 3.

c) Now the concatenated code has a blocks of size n_2, so $n = a n_2 = n_1 n_2 / k_2$. Again, $k = k_1$, since encoding again with S_2 does not change the number of logical qubits. The distance, however, is worse. It is again true that $P \in N(S) \setminus S$ iff $P = \bigotimes \overline{Q_i}$, with $Q = \bigotimes Q_i \in N(S_1) \setminus S_1$. However, each Q_i can now act on up to k_2 qubits of S_1. Therefore, the minimum number of Q_is that have to be nontrivial is no longer d_1, but instead could be as low as d_1/k_2. It is still true that each $\overline{Q_i}$ must act on at least d_2 qubits, though, so we find that $d \geq d_1 d_2 / k_2$. In the general case, therefore, we have a $[[n_1 n_2/k_2, k_1, d_1 d_2/k_2]]$ code.

However, we might note that while a single Q_i can act on k_2 qubits, it only acts on the set of k_2 qubits within a block. Therefore, we can gain if we consider S_1 as a code over qudits of size $D = 2^{k_2}$. In that case, the interesting number for distance is the number of such qudits that must be changed in order to have an undetectable error, and we again recover the original formula for the distance of a concatenated code, since each Q_i acts on only one such qudit. That is, when S_1 is a $[[n_1, k_1, d_1]]_D$ code and S_2 is a $[[n_2, k_2, d_2]]$ code (with $D = 2^{k_2}$), then the concatenated code is a $[[n_1 n_2, k_1, d_1 d_2]]$ qubit code. (The point of doing this is that for large D, it is possible to achieve a better ratio of d_1/n_1.)

Problem #2. Threshold for Classical Reversible Computation

a) This part is straightforward: The $\overline{0}$ and $\overline{1}$ preparations each consist of just 3 state preparations (of 0 and 1, respectively), followed by an EC step, so $A_{\text{prep}} = 3 + B$, where B is the number of locations in an EC step.

The Toffoli gate gadget contains 3 physical Toffolis to implement the transversal Toffoli gate, plus an EC step before and after on each of the three bits, for a total of $A_{\text{Tof}} = 3 + 6B$.

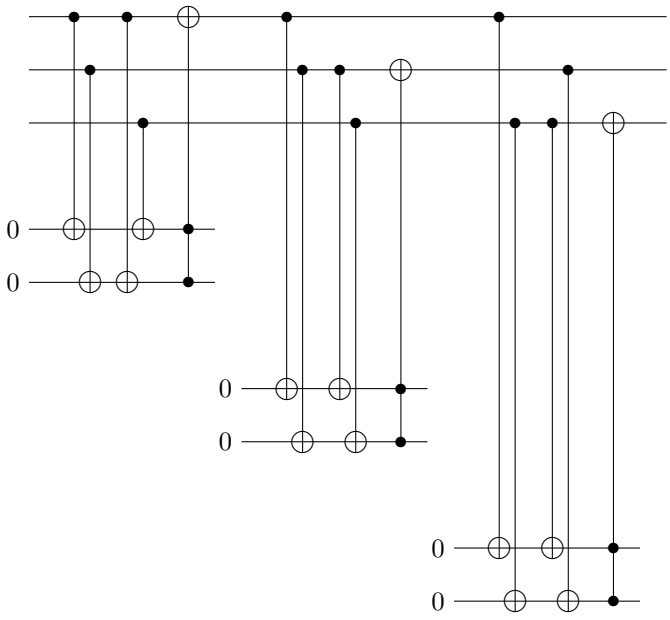
b) We recall that for the 3-bit repetition code, we need to make parity checks on the first two bits and the last two bits. We then decode the syndromes as follows:

$$00 \rightarrow \text{no error}$$
$$10 \rightarrow \text{1st bit}$$
$$11 \rightarrow \text{2nd bit}$$
$$01 \rightarrow \text{3rd bit}$$

We can measure the syndrome straightforwardly — each syndrome bit can be measured by taking one ancilla bit starting in the state 0 and doing CNOTs from the two bits whose parity we want to measure. We do not have to worry about multiple errors appearing in the data block for two reasons: First, because we are only taking the parity of two bits, an error in the middle of the procedure can only affect one bit, and second, because this is classical computation, we don't have to worry about phase errors propagating backwards along CNOTs anyway.

The difficult part is decoding the syndrome to actually correct the error. A non-fault-tolerant method of correcting the data based on the syndrome would use three Toffoli-like gates, triggered by the syndrome being 01, 11, and 01, respectively to flip the first, second, and third bits in the data block. As in the quantum case, however, a single error on an ancilla can ruin a syndrome bit, which could have serious consequences if we reuse the syndrome bits.

One straightforward solution is to re-extract the syndrome for each Toffoli-like gate, giving us the circuit given below:



We have changed the exact syndrome measurement procedure for each iteration. Instead of having just the first pair and last pair of bits for parity checks, we could have taken any two pairs, and we choose the two pairs that include the bit to be corrected. One reason for this is that we can then use the regular Toffoli gate to correct the errors, without having to perform any extra NOT gates. The other reason is explained below.

It is not immediately apparent this circuit actually is fault-tolerant — one might expect to have to repeat syndrome measurements. EC property 1 is trivial, since any bit string is at most one error away from a codeword, but EC property 2 needs to be satisfied. The case with one input error and no faults during the EC step works, but we must check the case with no input errors and one fault during the error correction procedure. An error during one of the Toffoli gates can only affect a single data bit, which is allowed, but an error during a CNOT gate could affect an ancilla bit, a data bit, or both.

However, it turns out that we do not need to repeat the syndrome measurements. First, note that if the faulty CNOT only causes an error on the ancilla, that is automatically acceptable. (An error on any ancilla can result in at most one error introduced into a data bit.) Thus, we need only consider the cases where there is an error on just the data or on both the data and ancilla bit involved in the faulty CNOT. However, for this classical code, there is only one kind of error: a bit flip error. (Since the state is a classical one, we can assume a definite value of the correct state and define bit flips relative to that.) Bit flip errors propagate forwards along CNOTs, so a CNOT with a bit flip error on both data and ancilla is the same as a bit flip error on the data (control bit) followed by a perfect CNOT. That is, we only need to really consider errors on the data bits before or after the CNOTs in the circuit.

Furthermore, an error on a data bit at the start or end of a syndrome measurement will clearly be acceptable too, as that means all the syndrome measurements themselves will be correct. Thus, we need only consider errors during the syndrome measurements, and the only place that is actually in the middle of a syndrome measurement is between the two CNOT gates acting on the same data bit. However, we chose the syndrome measurements so that the pair of CNOT gates which share a data bit act on the same data bit as the one being corrected by the syndrome measurement. Thus, the possible syndrome error and the data error coincide (which in this case means that the data error is not corrected, but at least no additional bits go bad). Thus there is no single location in the circuit that can cause more than one bit to get flipped in the final state, and the gadget satisfies EC property 2.

In each syndrome measurement, we count 2 bit preparations, 4 CNOT gates, and 1 Toffoli gate. The wait steps are not marked explicitly, but for each stage of CNOTs, one data bit must wait, and during each Toffoli gate, two data bits wait, so there are 4 wait steps in each syndrome measurement. There are thus a total of 11 locations in each syndrome measurement, for a total of $B = 33$ locations in the EC gadget as a whole.

c) Clearly the Toffoli gate extended rectangle is the largest. It contains a total of $A = 3 + 6B = 201$ locations. The probability of failure is thus bounded by

$$p(\text{Bad}) \leq \binom{A}{2} p^2 = 20100p^2. \tag{4}$$

The threshold is then at least $1/20100 \approx 5 \times 10^{-5}$. For a variety of reasons, the true threshold of the concatenated classical repetition code is much larger than this, but this does provide a lower bound.