Solution Set #5

CO 639: Quantum Error Correction
Instructor: Daniel Gottesman

Problem 1. Fault-tolerance in higher dimensions

a) It is a straightforward calculation to determine what transformation a given gate does on the generalized Pauli matrices. Then we can analyze how the transversal form of the gate transforms the stabilizer, just as in the binary case. To determine the encoded operation, we look at the transformation of $\overline{X} = X \otimes X^{-1} \otimes I$ and $\overline{Z} = Z \otimes I \otimes Z^{-1}$.

$$\begin{aligned} F : |j\rangle &\mapsto \sum_{k=0}^{d-1} \omega^{jk} |k\rangle && (1) \\ X &\to FXF^\dagger = Z && (2) \\ Z &\to FZF^\dagger = X^{-1}. && (3) \end{aligned}$$

Therefore $X \otimes X \otimes X \mapsto Z \otimes Z \otimes Z$, which is in the stabilizer, and $Z \otimes Z \otimes Z \mapsto X^{-1} \otimes X^{-1} \otimes X^{-1}$, which is also in the stabilizer (which is a group). Therefore, $F^{\otimes 3}$ is a valid transversal gate; it maps

$$\overline{X} \mapsto Z \otimes Z^{-1} \otimes I = Z^2 \otimes I \otimes Z = \overline{Z}^{-1} \quad (4)$$

and

$$\overline{Z} \mapsto X^{-1} \otimes I \otimes X = X^{-2} \otimes X^{-1} \otimes I = \overline{X}. \quad (5)$$

In both equations, we first multiply by a element of the stabilizer, and then use the fact that $d = 3$ (so X and Z have order 3). We find that the logical operation is F^{-1}.

$$\begin{aligned} P : |j\rangle &\mapsto \omega^{j(j+1)/2} |j\rangle && (6) \\ X &\to ZX = \omega XZ && (7) \\ Z &\to Z. && (8) \end{aligned}$$

Thus, $X \otimes X \otimes X \mapsto ZX \otimes ZX \otimes ZX$, which is in the stabilizer, and $Z \otimes Z \otimes Z$ remains unchanged. Thus, $P^{\otimes 3}$ is a valid transversal gate; it maps

$$\overline{X} \mapsto ZX \otimes X^{-1}Z^{-1} \otimes I = \omega \overline{X}(Z \otimes Z^{-1} \otimes I) = \omega \overline{X}\,\overline{Z}^{-1}. \quad (9)$$

and

$$\overline{Z} \mapsto \overline{Z}. \quad (10)$$

Since $P^{-1} : X \mapsto \omega^{-1} X Z^{-1}$, we identify the logical operation as $Z^2 P^{-1} = Z^{-1} P^{-1}$.

$$\begin{aligned} S_c : |j\rangle &\mapsto |cj\rangle && (11) \\ X &\to X^c && (12) \\ Z &\to Z^{c^{-1}}. && (13) \end{aligned}$$

This is only a unitary transformation when $c \neq 0$, but in this case, $X \otimes X \otimes X \mapsto X^c \otimes X^c \otimes X^c$ and $Z \otimes Z \otimes Z \mapsto Z^{c^{-1}} \otimes Z^{c^{-1}} \otimes Z^{c^{-1}}$, so $S_c^{\otimes 3}$ is a valid transversal gate. It maps

$$\overline{X} \mapsto X^c \otimes X^{-c} \otimes I = \overline{X}^c \tag{14}$$

and

$$\overline{Z} \mapsto Z^{c^{-1}} \otimes I \otimes Z^{-c^{-1}} = \overline{Z}^{c^{-1}}, \tag{15}$$

so the logical operation is S_c.

$$\begin{aligned} \mathrm{SUM} : |j\rangle|k\rangle &\mapsto |j\rangle|k+j\rangle. && (16)\\ X \otimes I &\rightarrow X \otimes X && (17)\\ I \otimes X &\rightarrow I \otimes X && (18)\\ Z \otimes I &\rightarrow Z \otimes I && (19)\\ I \otimes Z &\rightarrow Z^{-1} \otimes Z. && (20) \end{aligned}$$

We can see that $X \otimes X \otimes X$ is mapped to either itself or to two copies of itself, and $Z \otimes Z \otimes Z$ is mapped to itself or to the tensor product of its inverse with itself. Therefore, $\mathrm{SUM}^{\otimes 3}$ is a valid transversal gate. It maps

$$\begin{aligned} \overline{X} \otimes I &\mapsto \overline{X} \otimes \overline{X} && (21)\\ I \otimes \overline{X} &\mapsto I \otimes \overline{X} && (22)\\ \overline{Z} \otimes I &\mapsto \overline{Z} \otimes I && (23)\\ I \otimes \overline{Z} &\mapsto \overline{Z}^{-1} \otimes \overline{Z}. && (24) \end{aligned}$$

We thus identify the logical operation as SUM.

b) Since F interchanges X and Z, and P mixes them, these two gates cannot be performed transversally on a general CSS code.

SUM can be, just as in the qubit case: $X^a \otimes I \mapsto X^a \otimes X^a$, so any generator consisting of powers of X gets mapped either to itself or to two copies of itself. Also, as Z^{-1} is a power of Z, Z generators get mapped either to themselves or a tensor product of their inverses with themselves.

In the qudit case, S_c is also a valid transversal gate for any CSS code: Any X generator gets mapped to the cth power of itself, and any Z generator gets mapped to the c^{-1}th power of itself. That is, any linear transformation on the qudits is a valid transversal gate for a higher-dimensional CSS code.

c) The one-qudit Pauli group consists of products $\omega^a X^b Z^c$. Ignoring the power of ω, Clifford group operations map $X \mapsto X^i Z^j$ and $Z \mapsto X^k Z^l$. The images of all other Pauli group elements are then constrained by the group multiplication law. To preserve the commutation relations, we therefore need that

$$\alpha(X^i Z^j, X^k Z^l) = \alpha(X, Z). \tag{25}$$

Note that

$$\alpha(X^i Z^j, X^k Z^l) = \alpha(Z^j, X^k) + \alpha(X^i, Z^l). \tag{26}$$

Also,

$$\alpha(Z^j, X^k) = j\alpha(Z, X^k) = jk\alpha(Z, X) = jk, \tag{27}$$

and, similarly,

$$\alpha(X^i, Z^l) = -il, \tag{28}$$

so $\alpha(X^i Z^j, X^k Z^l) = jk - il$. A general Clifford group element is therefore an element of the Pauli group (to choose the phases ω^a for the images of X and Z), times an operation with general (i, j, k, l) satisfying $il - jk = 1$.

Suppose we have available the Pauli operators, F, P, and S_c (for all $c \neq 0$). Because of the Pauli operators, we do not need to worry about the phase ω^a. We can perform the operation $Q = FPF^{-1}$, which maps $X \mapsto X$ and $Z \mapsto X^{-1}Z$. $P^m Q^n$ maps $X \mapsto XZ^m$ and $Z \mapsto X^{-n}Z^{1-mn}$ (with some phase).

If $i \neq 0$, the operation $S_i P^m Q^n$ maps $X \mapsto X^i Z^{mi^{-1}}$ and $Z \mapsto X^{-in} Z^{i^{-1}(1-mn)}$. If we choose $m = ij$, and $n = -i^{-1}k$, then $X \mapsto X^i Z^j$ and $Z \mapsto X^k Z^l$, where $l = i^{-1}(1 + jk)$, as required by the constraint $il - jk = 1$. Given the Pauli operators, F, P, and S_c (for all $c \neq 0$), we can therefore perform a general one-qudit Clifford group gate with $i \neq 0$.

We can also perform all cases where $i = 0$: Now $jk = 1$, so in particular $j \neq 0$. We must map $X \mapsto Z^j$ and $Z \mapsto X^k Z^l$. We could do this by first performing $X \mapsto X^j$ and $Z \mapsto Z^{-k} X^l$ (which we already know is possible), and then performing F.

However, this set of operators is not minimal: We can, for instance, eliminate all but one S_c, keeping one c such that c generates the multiplicative group for GF(p). (That is, c raised to some power gives us every non-zero number modulo p.) Such a c always exists.

We cannot eliminate F, since otherwise all operations will leave the subgroup $\langle Z^a \rangle$ of the Pauli group invariant. We cannot eliminate P, since otherwise all operations will map X to something either of the form X^a or of the form Z^b, and never $X^a Z^b$.

Note that $P^r Q^s P^m Q^n$ maps

$$\begin{aligned} X &\mapsto X^{1-ms} Z^{m+r-mrs} && (29) \\ Z &\mapsto X^{-s-n+mns} Z^{1-mn-rs-rn+mnrs} && (30) \end{aligned}$$

(with some phases). Let us choose $mrs = m + r$; then

$$\begin{aligned} X &\mapsto X^{1-ms} && (31) \\ Z &\mapsto X^{-s-n+mns} Z^{1-rs}. && (32) \end{aligned}$$

Choose $mns = s + n$ and $X \mapsto X^{1-ms}$, $Z \mapsto Z^{1-rs}$. Let us let $m = 1$ and $s = 1 - c$, with $c \neq 0, 1$. We should also choose $r = -c^{-1}$ and $n = 1 - c^{-1}$ to give us $mrs = m + r$ and $mns = s + n$. Then $X \mapsto X^c$ and $Z \mapsto Z^{c^{-1}}$, so this implements S_c. That is, we can eliminate S_c from the generating set completely.

We can also eliminate the Pauli operators: $F^2 = S_{-1}$, so $F^2|j\rangle = |-j\rangle$. Then

$$F^2 P^{-1} F^2 P|j\rangle = F^2 P^{-1} \omega^{j(j+1)/2}|-j\rangle = \omega^{j(j+1)/2 + j(1-j)/2}|j\rangle = \omega^j |j\rangle = Z|j\rangle. \quad (33)$$

Thus, with only F and P, we can generate Z, and we can get X^{-1} via $X^{-1} = FZF^{-1}$, and thus the whole Pauli group. Therefore, F and P generate the single-qudit Clifford group.

Problem 2. Quantum MDS Codes

There are three errata in this problem: First, α should generate the multiplicative group of GF(p), so that all the points $\alpha, \alpha^2, \ldots, \alpha^{p-2}$ are distinct. Second, in part a, you should show that $C^{p,\mu}$ corrects $p-\mu-1$ erasure errors, not μ erasure errors. Third, in part e, the MDS code should be linear.

a) The classical codewords of $C^{p,\mu}$ consist of all the polynomials of degree μ evaluated at all the points of GF(p). However, a general polynomial of degree μ is completely determined by its value at $\mu+1$ distinct points. Therefore, given only that many registers of the code, we can reconstruct the polynomial and thus the full codeword. That is, we can correct for the erasure of $p - (\mu + 1)$ registers. The code thus has distance at least $p-\mu$. It has $\mu+1$ encoded bits, corresponding to the coefficients of all polynomials of degree μ. By the classical Singleton bound, $d \leq n - k + 1 = p - \mu$. Therefore, the distance is exactly μ, and the code is a $[p, \mu + 1, p - \mu]_p$ code.

b) Let us calculate the inner product of the vector (α^{jr}) with the vector (α^{js}), where in both cases $j = 0, \dots, p-2$:

$$\sum_{j=0}^{p-2} \alpha^{rj}\alpha^{sj} = \sum_{j=0}^{p-2} (\alpha^{r+s})^j. \tag{34}$$

Suppose $r + s \leq p - 2$. Then α^{r+s} is a nonidentity element of GF(p), and we can evaluate the above geometric sum as

$$\frac{(\alpha^{r+s})^{(p-1)} - 1}{\alpha^{r+s} - 1} = 0, \tag{35}$$

since $\alpha^{p-1} = 1$. That is, the two vectors are orthogonal. Also, note that the vector (α^{js}) is orthogonal to the all-1s vector by the same argument:

$$\sum_{j=0}^{p-2} \alpha^{sj} = \frac{(\alpha^{s})^{(p-1)} - 1}{\alpha^{r+s} - 1} = 0. \tag{36}$$

The p-component all-1s vector is orthogonal to itself. Therefore, the dual of $C^{p,\mu}$ contains $C^{p,\nu}$ whenever $\mu + \nu \leq p - 2$, in particular for $\nu = p - \mu - 2$. The dual code to $C^{p,\mu}$ encodes $p - \mu - 1$ bits, and $C^{p,p-\mu-2}$ encodes $p - \mu - 1$ bits; therefore $(C^{p,\mu})^{\perp} = C^{p,p-\mu-2}$.

c) The CSS construction uses two codes C_1 and C_2, and we take both of them in this case to be $C^{p,\mu}$. The construction requires that $C_1^{\perp} \subseteq C_2$, which in this case requires that $p - \mu - 2 \leq \mu$, meaning $\mu \geq (p-2)/2$. We form the stabilizer of the code by converting the parity check matrix of $C^{p,\mu}$ into generators of the stabilizer: First we replace each entry a with X^a, and then we replace each entry a with Z^a. The duality condition guarantees that these operators generate an Abelian group.

The stabilizer has $2(p - \mu - 1)$ generators, so the quantum code encodes $k = 2\mu + 2 - p$ qudits. It corrects up to $p - \mu - 1$ X erasures and up to $p - \mu - 1$ Z erasures, so its distance is at least $p - \mu$. By the quantum Singleton bound, the distance d of a QECC is bounded above by $2d \leq n - k + 2 = 2p - 2\mu$, so the distance is exactly $p - \mu$. The code is a $[[p, 2\mu + 2 - p, p - \mu]]_p$ QECC.

d) The dual code $C^{\perp}$ of a $[n, k, n-k+1]$ code C has parameters $[n, n-k, d']$; we wish to show $d' \geq k+1$, in which case the classical Singleton bound will show that it is exactly $k+1$, proving the result. To prove that $d' \geq k+1$, let us suppose we have a set of k columns of the parity check matrix for $C^{\perp}$, which is the generator matrix of C. Since C can correct for the erasure of $n-k$ registers, the data in these k columns is sufficient to reconstruct the codeword, and there are k different independent codewords we could reconstruct. Therefore, the rank of the k-column submatrix must be at least k, which means it is exactly k, and the k columns are linearly independent. This is true for any k columns, so $d' \geq k+1$.

e) Let me first confess that this problem is harder than I had really intended. (I should have given you a pair of MDS codes one contained in the other.) However, it is still solvable, provided the distance of the MDS code is not 1 or n. I should also have specified that the MDS code was linear, as otherwise the CSS construction is not defined. If the distance is 1, then the dual has distance n, and if the distance is n, the code is clearly equivalent to a repetition code. We have various options to turn a repetition code into a CSS code, but the most straightforward is to use Shor's construction to concatenate the code with a Fourier-transformed version of itself to get a $[[n^2, 1, n]]$ code.

One trivial construction we can make is to take C_1 to be the MDS code and $C_2 = C_1^{\perp}$. The CSS construction then gives us a $[[n, 0, d_Q]]_p$ quantum code, with $d_Q = \min(n - k + 1, k + 1) \geq 2$. Recall from problem set 1 that a stabilizer code encoding 0 qubits must be nondegenerate by definition, and we should make the same requirement for qudits. However, the CSS construction does give us a nondegenerate code.

One might reasonably object that this is not a sensible QECC, and that we should try to find one that actually encodes some qudits. However, we can directly create one from the $[[n, 0, d_Q]]_p$ code Q by discarding a register, as described below.

The code Q is nondegenerate and has distance at least 2. Therefore, it detects all errors $X^i Z^j$ on the last qudit, so the stabilizer must contain elements M_1 and M_2 which act as $X^a Z^b$ and $X^{a'} Z^{b'}$ for some nonzero $(a,b),(a',b')$ such that if $a = ca'$, then $b \neq cb'$. Consequently, not both a and a' are 0 and not both b and b' are 0. Assume without loss of generality $a' \neq 0$. Then $N_1 = M_1^{a'} M_2^{-a}$ acts as $Z^{a'b-ab'} = Z^r$ on the last qudit, and we know that $b \neq ab'/a'$, so $r \neq 0$. Similarly, $N_2 = M_1^{b'} M_2^{-b}$ acts as $X^{ab'-a'b} = X^s$ on the last qudit, with $s \neq 0$. Then $N_1^{r^{-1}}$ is in the stabilizer and acts as X on the last qudit, and $N_2^{s^{-1}}$ is in the stabilizer and acts as Z on the last qudit. We can choose generators for the stabilizer of Q starting with N_1 and N_2, and ensure that all the other generators act as the identity on the nth register by multiplying generators that don't have this property by appropriate powers of N_1 and N_2.

Now let us discard the last register, and with it the generators N_1 and N_2. The remaining generators act as the identity on the last register, so they remain unchanged, and in particular, they still commute, and therefore generate a stabilizer on $n-1$ qudits. There are $n-2$ generators, and therefore the code encodes $(n-1)-(n-2) = 1$ qudit. The code has experienced effectively 1 erasure, and thus can still correct $d_Q - 2$ more, so its distance is at least $d_Q - 1$: we have a $[[n-1, 1, d_Q - 1]]_p$ code.

An equivalent construction can be obtained by puncturing the classical code: Take the code C_1 to be all codewords of the original classical MDS code without the last register. The parameters of C_1 are $[n-1, k, n-k]_p$: we have the same number of encoded registers, and have experienced one erasure, so the distance has decreased by 1. This is still an MDS code. The code $C_2^\perp \subset C_1$ is the set of all codewords of C_1 for which the original MDS code had a 0 in the last register. The distance of this code must be at least as large as the distance of the original MDS code, as the final register is predetermined, so we can still correct the same number of erasures. The code $C_2^\perp$ encodes $k-1$ registers, as we have effectively added one parity check (parity of the last register alone) to the original code's parity check matrix. Therefore, the parameters of $C_2^\perp$ are $[n-1, k-1, n-k+1]_p$, which is also MDS, and by part d, the parameters of C_2 are $[n-1, n-k, k]_p$. Therefore, with the CSS construction, we have an $[[n-1, 1, \tilde{d}_Q]]_p$ QECC, with $\tilde{d}_Q = \min(n-k, k)$.

Problem 3. Threshold with Local Gates

For part a, I was considering the distance D to be the distance from the data block to the most distant ancilla, which actually means the distance to a level L ancilla should be $(2D)^L/2$. This is taken into account in part b, but there are other problems, so the recursion relation in part b should actually read $P_L = 4C(2D)^{2L} P_{L-1}^2$.

a) We arrange our ancillas so that the ancillas needed for level 1 error correction are immediately adjacent to the data block; we can do this keeping all of them within a distance D by assumption. The level 2 ancillas, needed for level 2 error correction, must be further away. Furthermore, those ancillas are themselves encoded with one level of the QECC, so they require their own level 1 ancillas. We surround each level 2 ancilla with an identical collection of level 1 ancillas, which can therefore all be placed within a distance D of the level 2 ancilla. We can imagine that each level 2 ancilla and its associated level 1 ancillas form big indivisible block, and we arrange these blocks just as we did the level 1 ancillas. Similarly, each level 3 ancilla needs a collection of level 2 ancillas, each of which has its own collection of level 1 ancillas. We consider the complete collection as an even bigger indivisible block and continue moving up levels in this way. In 3 dimensions, we would get a layout something like what is shown in figure 1, for instance.

The upshot is that the block associated with a level 2 ancilla has size D, and that when we arrange these blocks, we should therefore scale all distances by $2D$, as this is the minimum distance between the centers of these blocks. The assortment of level 2 blocks, when put together, would therefore all be within a distance $2D^2$ of the data block. When we put together the collections of level 2 ancillas, we should therefore scale by $4D^2$, so the size of a level 3 collection is $4D^3$. In general, a level L collection consists of a bunch of level $L-1$ collections, each of size $(2D)^{L-1}/2$; therefore the level L collection has size $(2D)^L/2$.

a) b)

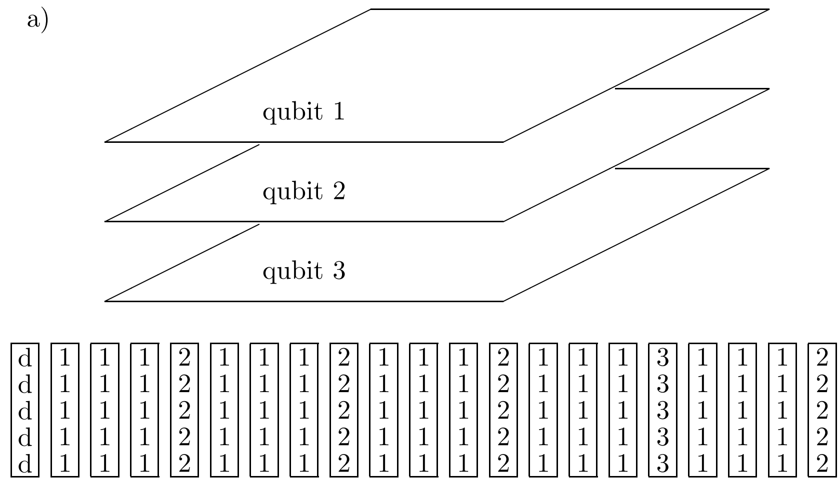

Figure 1: a) The logical qubits of the computer lie on separate planes. b) Each plane has the data on one line, adjacent to ancillas at various levels. The letter d represents a data qubit, 1 is a qubit from a level 1 ancilla, 2 is from a level 2 ancilla, and so on.

b) At level L all distances are scaled by $(2D)^L$. In order to perform a gate at level L, we must move the level L block this distance, do the gate, and then move the block back to where it started, a total distance of $2(2D)^L$. This increases the effective error rate per gate by this factor, so the effective error rate P_{L-1} for a nearest-neighbor gate at level $L-1$ becomes $2(2D)^L P_{L-1}$. In the absence of geometry, the recursion relation is $P_L = CP_{L-1}^2$; taking into account the effects of distance, it should therefore be $P_L = C[2(2D)^L P_{L-1}]^2$.

c) Assume we have the recursion relation $P_L = AB^L P_{L-1}^2$. Then

$$\begin{aligned} P_L &= AB^L (AB^{L-1})^2 P_{L-2}^4 && (37)\\ &= AB^L A^2 B^{2(L-1)} A^4 B^{4(L-2)} P_{L-3}^8 && (38)\\ &= \left(\prod_{i=0}^{L-1} A^{2^i} B^{2^i(L-i)}\right) P_0^{2^L}. && (39) \end{aligned}$$

Now,

$$\prod_{i=0}^{L-1} A^{2^i} = A^{\sum_{i=0}^{L-1} 2^i} = A^{2^L-1}, \tag{40}$$

and

$$\begin{aligned}
\log_B \prod_{i=0}^{L-1} B^{2^i(L-i)} &= \sum_{i=0}^{L-1} 2^i (L-i) && (41)\\
&= L(2^L - 1) - \sum_{i=0}^{L-1} i 2^i && (42)\\
&= L(2^L - 1) - \sum_{i=0}^{L-1} \sum_{j=0}^{i-1} 2^i && (43)\\
&= L(2^L - 1) - \sum_{j=0}^{L-2} \sum_{i=j+1}^{L-1} 2^i && (44)\\
&= L(2^L - 1) - \sum_{j=0}^{L-2} (2^L - 2^{j+1}) && (45)\\
&= L(2^L - 1) - (L-1)2^L + 2(2^{L-1} - 1) && (46)\\
&= 2^{L+1} - L - 2. && (47)
\end{aligned}$$

Thus,

$$\begin{aligned}
P_L &= A^{2^L} B^{2^{L+1}-L} P_0^{2^L} / AB^2 && (48)\\
&\leq (AB^2 P_0)^{2^L} / AB^2. && (49)
\end{aligned}$$

We thus find a threshold $P_c = 1/(AB^2)$. If $P_0 < P_c$, then $P_L \to 0$ as $L \to \infty$.

d) The problem is that the SWAP gate is a two-qubit gate, and therefore an error on a single SWAP gate can cause errors on two qubits. If we are not careful, we may well interact two qubits in a single block of the code, causing an uncorrectable error with only a single gate error.

In two or more dimensions, we can solve this problem by arranging things so that there is room to move computational qubits out of the way to let them pass each other. A possible arrangement is shown in figure 2. That way, the SWAP gate never acts on two useful qubits at the same time — at least one of them is always a filler qubit whose value does not matter. Since the SWAP gate does not propagate errors (it swaps them along with the data), we don't have to worry about errors leaking from the filler qubits into the data qubits.

e) On a truly one-dimensional lattice, we can't move qubits out of the way, so the solution from the last problem won't help. We could manage in a lattice with next-to-nearest neighbor interactions by alternating filler qubits with regular qubits, or with nearest-neighbor interactions in a system consisting of two adjacent lines of qubits, or even a line with periodic single-qubit alcoves off to the side.

However, if none of these geometries is available, we do still have a threshold: The SWAP gate does not propagate errors, so the worst thing it can do is cause two errors with one bad gate. Therefore, if we concatenate using a code that corrects two errors rather than one, we can still correct for the failure of a single SWAP gate. The basic logic of parts a-c holds unchanged — qubits or blocks have to move a distance D to interact with their ancillas; moving a distance D increases the error rate linearly with the distance, so the recursion relation has an exponential factor in the level. The only difference is that we have a code that ideally should correct two errors, but we still have a recursion relation in P_{L-1}^2 since two gate errors could cause a problem if at least one is in a SWAP gate.

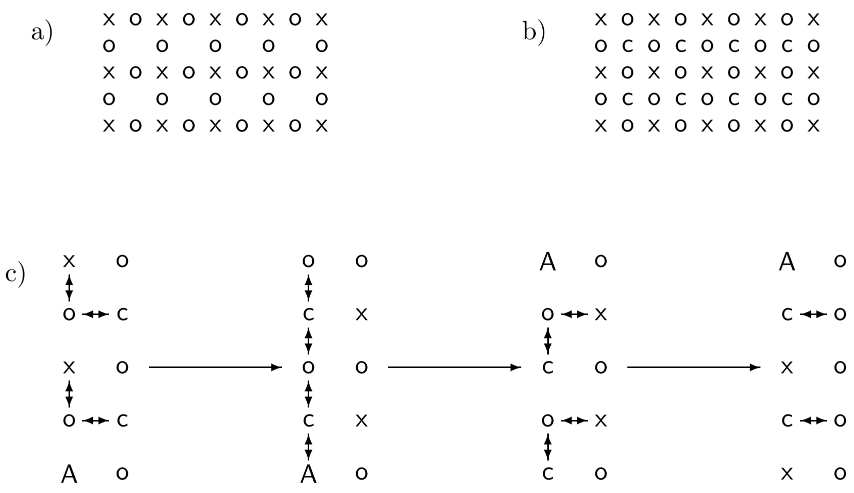

Figure 2: a) Computational qubits (x) arranged on a square lattice, interspersed by auxiliary qubits (o). b) adds cul-de-sacs (c) for moving qubits out of the way. c) Moving a data qubit (A) two positions.

Problem 4. Ancilla Purification for Toffoli Gates

a)

$$\begin{aligned}(X \otimes \text{C-Z})|\Psi_{000}\rangle &= \sum_{ijk}(-1)^{ijk+jk}|i\oplus 1, j, k\rangle && (50)\\ &= \sum_{i'jk}(-1)^{(i'\oplus 1)jk+jk}|i'jk\rangle && (51)\\ &= \sum_{i'jk}(-1)^{i'jk}|i'jk\rangle && (52)\\ &= |\Psi_{000}\rangle. && (53)\end{aligned}$$

The state $|\Psi_{000}\rangle$ is cyclic, so as it is a $+1$-eigenstate of $M_1 = X \otimes$ C-Z, it is also an eigenstate of its cyclic permutations M_2 and M_3. Note that the three operators M_1, M_2, and M_3 commute.

Since $Z \otimes I$ and $I \otimes Z$ both commute with C-Z, and X anticommutes with Z, $|\Psi_{abc}\rangle$ has eigenvalue $(-1)^a$ for M_1, $(-1)^b$ for M_2 (which has X on the second qubit) and $(-1)^c$ for M_3 (which has X on the third qubit). There are 8 states $|\Psi_{abc}\rangle$ and they all have different eigenvalues for the set of three commuting operators M_1, M_2, M_3, and therefore are all orthogonal to each other. They therefore form a basis for the 8-dimensional Hilbert space of 3 qubits.

b) We can write a general pure state by expanding it in the basis $|\Psi_{abc}\rangle$:

$$|\phi\rangle\langle\phi| = \sum_{abca'b'c'} q_{abc}q^*_{a'b'c'}|\Psi_{abc}\rangle\langle\Psi_{a'b'c'}|. \quad (54)$$

Then

$$\begin{aligned}\sum_{rst} M_1^r M_2^s M_3^t|\phi\rangle\langle\phi|M_1^r M_2^s M_3^t &= \sum_{abca'b'c'rst} q_{abc}q^*_{a'b'c'} M_1^r M_2^s M_3^t|\Psi_{abc}\rangle\langle\Psi_{a'b'c'}|M_1^r M_2^s M_3^t && (55)\\ &= \sum_{abca'b'c'rst} q_{abc}q^*_{a'b'c'}(-1)^{ra+sb+tc+ra'+sb'+tc'}|\Psi_{abc}\rangle\langle\Psi_{a'b'c'}| && (56)\\ &= \sum_{abca'b'c'} q_{abc}q^*_{a'b'c'}\delta_{aa'}\delta_{bb'}\delta_{cc'}|\Psi_{abc}\rangle\langle\Psi_{a'b'c'}| && (57)\\ &= \sum_{abc}|q_{abc}|^2|\Psi_{abc}\rangle\langle\Psi_{abc}|. && (58)\end{aligned}$$

This sort of operation — averaging over some operations to make a more symmetric density matrix — is frequently known as a "twirl".

c) After the CNOTs, we have the state

$$\sum_{ijki'j'k'} (-1)^{ijk+i'j'k'+ia+jb+kc+i'a'+j'b'+k'c'} |ijk\rangle |i \oplus i', j \oplus j', k \oplus k'\rangle. \tag{59}$$

If we measure the fifth qubit, getting the result $\tilde{j} = j \oplus j'$, and perform the appropriate conditional operation, we have the state

$$\sum_{ijki'k'} (-1)^{ijk+i'(j\oplus\tilde{j})k'+\tilde{j}(ik)+ia+jb+kc+i'a'+(j\oplus\tilde{j})b'+k'c'} |i, j \oplus \tilde{j}, k\rangle |i \oplus i', \tilde{j}, k \oplus k'\rangle. \tag{60}$$

Then we measure the sixth qubit, getting the result $\tilde{k} = k \oplus k'$ and perform the appropriate conditional operation. We now have the state

$$\sum_{ijki'} (-1)^{ijk+i'(j\oplus\tilde{j})(k\oplus\tilde{k})+\tilde{j}(ik)+\tilde{k}i(j\oplus\tilde{j})+ia+jb+kc+i'a'+(j\oplus\tilde{j})b'+(k\oplus\tilde{k})c'} |i, j \oplus \tilde{j}, k \oplus \tilde{k}, i \oplus i'\rangle, \tag{61}$$

where we have dropped the last two qubits. If we change the summed variables i', j, and k to $i'' = i \oplus i'$, $j'' = j \oplus \tilde{j}$ and $k'' = k \oplus \tilde{k}$, we can write the state as

$$\sum_{ij''k''i''} (-1)^{i''j''k''+ia+(j''\oplus\tilde{j})b+(k''\oplus\tilde{k})c+(i\oplus i'')a'+j''b'+k''c'} |i, j'', k'', i''\rangle. \tag{62}$$

We can discard the overall phase $(-1)^{\tilde{j}b+\tilde{k}c}$ from this expression, giving us

$$\sum_{ij''k''i''} (-1)^{i(a+a')+i''j''k''+j''(b+b')+k''(c+c')+i''a'} |i\rangle |j'', k'', i''\rangle = \sum_i (-1)^{i(a+a')} |i\rangle |\Psi_{b+b',\, c+c',\, a'}\rangle. \tag{63}$$

When we perform the Hadamard on the first qubit and measure it, we get $a \oplus a'$, as we wanted to show.

d) We keep the state if $a \oplus a' = 0$, which happens if both a coordinates are errors or if neither is. The probability of having $a \oplus a' = 0$ is thus $(1-p_a)^2 + p_a^2$ and the probability of having an a error conditioned on having $a \oplus a'$ is $p_a^2/[(1-p_a)^2 + p_a^2]$.

The probability of having a b error, regardless of whether or not we keep the state, is the probability of having exactly one b error among the first two states: $2p_b(1-p_b)$. Since the a and b errors are uncorrelated, the probability of having a b error remains $2p_b(1-p_b)$ even after we condition on keeping the state. Similarly, the probability of having a c error conditioned on keeping the state is $2p_c(1-p_c)$.

Thus, taking into account the cyclic shift between a, b, and c, we have the new probabilities

$$(p'_a, p'_b, p'_c) = (2p_b(1-p_b), 2p_c(1-p_c), p_a^2/[(1-p_a)^2 + p_a^2]. \tag{64}$$

e) The error probabilities for b and c nearly double after a purification step, but the error probability for a decreases as the square. Therefore, let us perform the purification step three times, giving us the chance to decrease all three probabilities.

We can simply the calculations by noting that $(1-p_a)^2 + p_a^2$ has minimum value $1/2$ (achieved at $p_a = 1/2$), so

$$(p'_a, p'_b, p'_c) \leq (2p_b, 2p_c, 2p_a^2). \tag{65}$$

After we perform two steps, we have error probabilities

$$(p''_a, p''_b, p''_c) \leq (4p_c, 4p_a^2, 8p_b^2), \tag{66}$$

and after three steps of purification, we have error probabilities

$$(p_a''', p_b''', p_c''') \leq (8p_a^2, 16p_b^2, 32p_c^2). \tag{67}$$

If $p_a < 1/8$, $p_b < 1/16$, and $p_c < 1/32$, the error probabilities have all decreased. By repeating this cycle many times, we can drive the error probabilities to arbitrarily low values, meaning that an initial error rate of $1/32$ is a threshold for purification.

We can do better in two ways: First by noting that the threshold is well below $1/2$, so that we have a much better bound on $(1-p_a)^2+p_a^2$. Indeed, in the above argument, we never have an error probability greater than $1/8$, so $(1-p_a)^2+p_a^2 \geq 25/32 = 1/1.28$. Of course, this would mean that the c error rate does increase above $1/8$ before it is decreased again, so we have to be more careful. $1/5$ is a better bound to use: This gives us $(1-p_a)^2+p_a^2 \geq 17/25 > 2/3$. We then have probabilities after three steps of recursion

$$(p_a''', p_b''', p_c''') \leq (6p_a^2, 12p_b^2, 24p_c^2), \tag{68}$$

giving a threshold value of $1/24$.

Second, we can note that even if the three error rates start off equal in value, after this recursion, they rapidly become unequal. It is worth our while to rearrange the qubits in the state so that we are always reducing the error rate for the error type that has the largest current error rate. For instance, if we start out with all error rates equal to $1/16$, we can still decrease them indefinitely: $(1/16, 1/16, 1/16) \rightarrow (1/8, 1/8, 1/128) \rightarrow (1/4, 1/128, 1/32) \rightarrow (1/64, 1/16, 1/8) \rightarrow (1/8, 1/32, 1/32) \rightarrow (1/16, 1/16, 1/32) \rightarrow (1/8, 1/16, 1/128) \rightarrow (1/8, 1/64, 1/32) \rightarrow (1/32, 1/16, 1/32) \rightarrow (1/16, 1/16, 1/128) \rightarrow (1/8, 1/64, 1/128) \rightarrow (1/32, 1/64, 1/32)$ and so on. The threshold is therefore definitely above $1/16$.

Indeed, it turns out that you can increase the threshold substantially above this by purifying other states. Single-qubit states are preferred, as there is only one error rate to purify there, but the procedures are more complicated. See quant-ph/0403025 by Bravyi and Kitaev for details.

f) If we have a base error rate of p_g per gate or measurement, we get additional terms in the recursion relation. We will want to perform a new twirl (as per part b) after each purification step in order to retain a mixture of $|\Psi_{abc}\rangle$ states, as opposed to a superposition which might be created by the errors. There is also some trouble because the errors can cause the a, b, and c error rates to become correlated. (E.g., a measurement error causes us to perform the wrong operation on multiple qubits.)

However, the basic conclusion still holds: The revised recursion relations are

$$(p_a', p_b', p_c') \leq (2p_b + 11p_g, 2p_c + 11p_g, 2p_a^2 + 11p_g). \tag{69}$$

(We have 11 gates and measurements in the circuit, and if one goes wrong, we allow it to cause any set of errors.) Given low enough error rates, the squaring will still beat the doubling; however, rather than shrink asymptotically to zero, the error rates will have some limiting value due to the constant influx of new gate errors.