A Study of Topological Quantum Error Correcting Codes
Part II: Surface Codes

Preetum Nakkiran
preetum@berkeley.edu

May 07, 2015

Abstract

Following Part I, here we present and analyze topological surface codes. We will see how they inherit some ideas from CSS codes, and leverage topological properties to achieve good distance. We also include a discussion about some subtleties of quantum mechanics.

Contents

1 Revisiting QM

Before developing more theory, let us briefly revisit some subtle aspects of QM.

1.1 Commuting Operators

There is a subtlety in measuring multiple observables (eg, stabilizers) on the same system, which we safely ignored before, but will now need to consider. Say we have a stabilizer QECC with stabilizers A, B. In order to decode, we want to "measure both stabilizers A and B". However, in our analysis so far, we have considered only the effect of individually measuring A or B on received states. We need to make sure that the two operators respect each other, in the sense that measuring A then B has the same effect as measuring B then A (same distribution of measurements, and same effect on the system collapse). This implies that their marginals are the same as well, ie measuring just A has the same effect as measuring B then A ($\equiv A$ then B), so our previous analysis holds.

In general, this "simultaneous measurement" is possible iff operators A and B commute. [1] From linear algebra, $AB = BA \implies A, B$ are *simultaneously diagonalizable*: they share a common eigenbasis. (In matrix notation, there exists an eigenbasis U s.t. $A = U\Lambda_A U^{-1}$ and $B = U\Lambda_B U^{-1}$ for diagonal Λ_A, Λ_B). We have already seen examples of non-commuting operators which cannot be simultaneously measured. In Section 2.2 of Part I, we saw that starting with a qubit in state $|0\rangle$, measuring first Z then X is different from measuring first X, then Z (since the first pair will leave the system in state $|+\rangle$ or $|-\rangle$, and the second will leave it in $|0\rangle$ or $|1\rangle$).

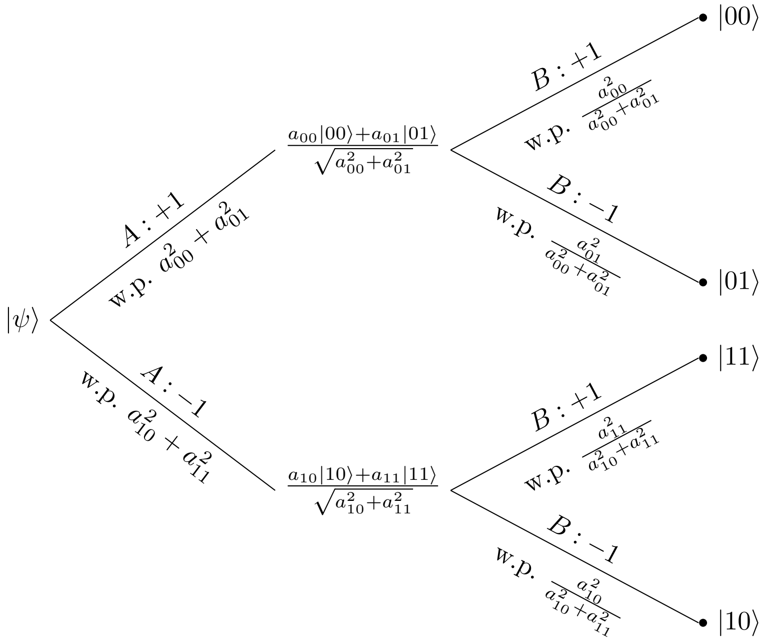
Now let us see an explicit example of measuring two commuting operators. Consider a two-qubit system, starting in the general state

$$|\psi\rangle = a_{00}|00\rangle + a_{01}|01\rangle + a_{10}|10\rangle + a_{11}|11\rangle$$

Define operators $A = Z \otimes I$ and $B = Z \otimes Z$. Notice these commute, and have eigenvalues ±1. For example, the $+1$ eigenspace of A is $\{|00\rangle, |01\rangle\}$, and the $+1$ eigenspace of B is $\{|00\rangle, |11\rangle\}$. We first consider measuring A then B. If we measure A on $|\psi\rangle$, we get $+1$ w.p. $a_{00}^2 + a_{01}^2$, and collapse to state proportional to $a_{00}|00\rangle + a_{01}|01\rangle$ (normalized). If we measure B from this state, we get $+1$ w.p. $\frac{a_{00}^2}{a_{00}^2+a_{01}^2}$, and collapse to $|00\rangle$. Thus, we get outcome $(+1, +1)$ from measuring (A, B) with probability a_{00}^2. Similarly, we will get $(+1, +1)$ with the same probability if we measured B first, then A (and we would collapse to the same state). All other outcomes will similarly be the same, regardless of order.

[1] In fact, there is a general *uncertainty relation* in QM relating the degree to which operators commute to the precision with which they can be simultaneously measured. Heisenberg uncertainty is a special case, for the non-commuting operators position and momentum.

Figure 1: Measuring A, then B on $|\psi\rangle$


$|\psi\rangle$
$A:+1$
w.p. $a_{00}^2 + a_{01}^2$
$A:-1$
w.p. $a_{10}^2 + a_{11}^2$
$\frac{a_{00}|00\rangle + a_{01}|01\rangle}{\sqrt{a_{00}^2+a_{01}^2}}$
$\frac{a_{10}|10\rangle + a_{11}|11\rangle}{\sqrt{a_{10}^2+a_{11}^2}}$
$B:+1$
w.p. $\frac{a_{00}^2}{a_{00}^2+a_{01}^2}$
$B:-1$
w.p. $\frac{a_{01}^2}{a_{00}^2+a_{01}^2}$
$B:+1$
w.p. $\frac{a_{11}^2}{a_{10}^2+a_{11}^2}$
$B:-1$
w.p. $\frac{a_{10}^2}{a_{10}^2+a_{11}^2}$
$|00\rangle$
$|01\rangle$
$|11\rangle$
$|10\rangle$

This example can easily be turned into a proof that commuting operators can be simultaneously measured. As above, the key idea is that commuting operators respect each other's eigenspaces, so the probabilistic collapse of measuring one operator "factors through" measuring the other. In QECCs, we require that all stabilizers commute with each other precisely because we want to simultaneously measure them.

1.2 The Tensor Product

This section explores some physical intuition, and can be safely skipped.

The tensor product is often claimed to be the "natural" way of describing the state space of a joint system. But the multilinearity of the tensor product seems unnatural: Consider a 3-qubit system, in the "pure state" of all qubits $|0\rangle$. Why should we expect $(|0\rangle, |0\rangle, |0\rangle) \sim (-|0\rangle, -|0\rangle, |0\rangle)$? [2] That is, why should we expect negating the first two qubits is the same as negating none? If we assume tensor-product structure, this follows from multilinearity: $|0\rangle \otimes |0\rangle \otimes |0\rangle = (-|0\rangle) \otimes (-|0\rangle) \otimes |0\rangle$

We can perhaps gain insight into the multilinearity of joint systems by considering an explicit example of a separable system.

We will have to move from the purely linear-algebraic formalism of Dirac to the [linear] differential-equation formalism of Schrödinger ("wave mechanics"). Here the "wavefunction" $\Psi(q,t)$ is a complex-valued function of system coordinates q and time t. We will use state-space coordinates[3], where q is positions. We can then interpret Ψ as roughly a "distribution" on the state-space. For example, for a single particle in 3D, $q = (x, y, z)$ and then $|\Psi(x, y, z, t)|^2$ is the probability density function of finding the particle at location (x, y, z) at time t. (This is like using a "position basis" in Dirac notation: $|x\rangle \sim \Psi(x)$). For N particles in 3D, the state-space will be $3N$-dimensional, and $q = (x_1, y_1, z_1, x_2, y_2, \dots z_3)$.

[2] Writing the state as a tuple to avoid tensor-product notation.

[3] As opposed to phase-space coordinates, ie momentums.

Recall, the general Schrödinger equation: $i\hbar\frac{\partial}{\partial t}\Psi = H\Psi$. For a single particle moving in an potential $V(q)$, the Hamiltonion H takes the form:

$$H = \frac{-\hbar^2}{2\mu}\nabla^2 + V(\mathbf{q})$$

In the case where the Hamiltonian does not depend on time (as above), this equation admits "separable" solutions of the form $\Psi(q,t) = \psi(q)\exp(-iEt/\hbar)$. For constant E, and ψ a solution to the *time-independent Schrödinger equation*:

$$H\psi = E\psi \tag{1}$$

Now consider a joint system of 3 particles in 1D, with time-independent wavefunction $\psi(x_1, x_2, x_3)$. Further, assume each subsystem (particle) is isolated from the others – there is no interaction between them. Mathematically, this means the Hamiltonion will separate, as

$$H = H_1 + H_2 + H_3$$

Where H_i depends only on the coordinates of the i-th particle. For example, we may have $H_1 = \frac{-\hbar^2}{2\mu}\frac{\partial^2}{\partial^2 x_1} + x_1^2$, which depends only on the coordinates of particle 1.

Say we have wavefunctions $X_1(x_1), X_2(x_2), X_3(x_3)$ which satisfy

$$H_i X_i = E_i X_i \tag{2}$$

That is, X_i is a solution of (1) for the i-th subsystem, treated independently. We claim that

$$\psi(x_1, x_2, x_3) = X_1(x_1)X_2(x_2)X_3(x_3) \tag{3}$$

is a solution to (1) for the joint system.

Notice:

$$\begin{aligned}
H_1[\psi] &= H_1[X_1(x_1)X_2(x_2)X_3(x_3)] \\
&= X_2(x_2)X_3(x_3)H_1[X_1(x_1)] && (H_1 \text{ independent of } x_2, x_3) \\
&= X_2(x_2)X_3(x_3)E_1X_1(x_1) && (\text{by } (2)) \\
&= E_1\psi && (E_1 \text{ constant})
\end{aligned}$$

Thus, we have:

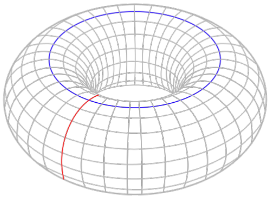
$$\begin{aligned}
H\psi &= (H_1 + H_2 + H_3)[\psi] \\
&= H_1\psi + H_2\psi + H_3\psi \\
&= E_1\psi + E_2\psi + E_3\psi \\
&= (E_1 + E_2 + E_3)\psi
\end{aligned}$$

Which is a solution to (1) for the joint Hamiltonion H, with eigenvalue $E = E_1 + E_2 + E_3$.

Therefore, the product of individual wavefunctions X_i forms a wavefunction ψ for the joint system. This product is obviously multilinear, and perhaps gives some intuition about the multilinearity of joint systems (in this example, taking $X_1 \to -X_1$ and $X_2 \to -X_2$ leaves the product ψ unchanged). In general, we need to take the tensor-product construction of joint systems as a postulate of QM.

Remark: I don't know if there is a way to extend this intuition to non-separable systems. Or in general, if there is a better way of physically motivating the multilinearity...

2 Surface Codes



We are now ready to study topological QECCs. As we have seen, ECCs are designed such that their codewords are "far apart", so it is difficult for an error to take one codeword to another. At a high level, surface codes achieve this by choosing codewords as homology classes on a nontrivial surface. For example, the two homology classes on the torus (represented by red and blue cycles) are fundamentally different, and intuitively it is "hard" to turn one into the other. One motivation for surface codes is that their stabilizers are "local" (only involve a local neighborhood of qubits), which is nice for physical implementations of quantum computers.

This section is based heavily on the chapter [1], with some additional exposition and proofs.

2.1 Notation

We will be dealing with meshed surfaces (formally, cell-complexes), and will use standard notation from algebraic topology. The groups C_0, C_1, C_2 are the $0, 1, 2$-chains respectively (comprised of vertices, edges, and faces respectively). We think of 1-chains $c \in C_1$ as vectors in $\mathbb{Z}_2^{|E|}$ (as an additive group), with $c_i = 1$ if edge i is in c, and 0 otherwise. We have homomorphic boundary maps between them:

$$\partial_2 : C_2 \to C_1, \qquad \partial_1 : C_1 \to C_0$$

Define $Z_1 \subset C_1$ as the kernel of ∂_1 – that is, 1-chains which have no boundary ("cycles"). Define $B_1 \subset C_1$ as the image of ∂_2 – the boundary of faces. The boundary of a boundary is empty:

$$\partial_2 \partial_1 = 0$$

Therefore $B_1 \subset Z_1$ (boundaries of faces are cycles). Notice, not every cycle is necessarily the boundary of a face (for example, the two cycles on the torus above).

Finally, the homology class H_1 is the quotient group:

$$H_1 := Z_1 / B_1$$

Each class in H_1 consists of 1-cycles equivalent up to boundaries of faces.

2.2 Basis Construction

This construction will be similar to CSS codes, except instead of codewords being uniform superpositions of subcodes, here they will be uniform superpositons over homology classes.

Given a meshed surface, we assign qubits to the edges, and interpret 1-chains $c \in C_1$ as a elements $|c\rangle$ of the computational basis in the natural way: $|c\rangle = \bigotimes_i |c_i\rangle$. We can also form products of X and Z operators along 1-chains: For $c \in C_1$, define X_c as the product of X operators along the edges in c (and the identity on other edges). And define Z_c similarly for Z operators. Notice these are homomorphisms $C_1 \to \mathcal{P}_n$, ie $X_c X_{c'} = X_{c+c'}$ (this holds because $X^2 = I$, so edges which overlap between c, c' cancel in $X_c X_{c'}$, just as in C_1). Further, notice $X_c |b\rangle = |b + c\rangle \ \forall b, c \in C_1$.

For a message $z \in Z_1$, the surface-encoded codeword is:

$$|\overline{z}\rangle := \sum_{b \in B_1} |z + b\rangle \tag{4}$$

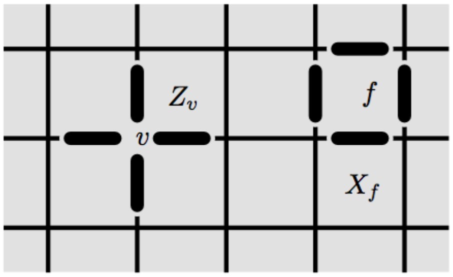


Figure 2: The support of vertex and face stabilizer generators. (Figure from [1]).

Where z, b are considered as vectors in $\mathbb{Z}_2^{|E|}$ (so $z+b$ is mod 2, as usual). As promised, $|\overline{z}\rangle$ is a uniform superposition over the homology class of z (which we denote $\overline{z} \in H_1$, abusing notation).

What is the dimension of our message-space? For any $\overline{z} \neq \overline{z}'$, we have $\langle \overline{z} | \overline{z}' \rangle = 0$, since homology classes partition cycles (there is no overlap between the sums). [4] Therefore there are as many distinct codewords as homology classes. For a surface of genus g, the first homology $H_1 \cong \mathbb{Z}_2^{2g}$, so $|H_1| = 2^{2g}$, and we have encoded $k = 2g$ qubits.

As in classical and CSS codes, the distance of this code w.r.t. bit-flips is the minimum-weight codeword, which in this case corresponds to the shortest non-trivial cycle. More formally: By definition, the bit-flip distance of this code is the minimum weight of an operator X_c that takes some codeword $|\overline{z}\rangle$ to some other codeword $|\overline{z}'\rangle$ for $\overline{z}' \neq \overline{z}$. That is, $X_c |\overline{z}\rangle = |\overline{z} + \overline{c}\rangle = |\overline{z}'\rangle$ for $z \neq z' \in Z_1$. If $\partial c \neq 0$, then $X_c |\overline{z}\rangle = |\overline{z+c}\rangle$ is not a codeword, because $\partial c \neq 0 \implies \partial(z+c) \neq 0$, so $z+c$ is not a cycle (not in Z_1). Further, if c is a boundary ($c \in B_1$), then

$$X_c |\overline{z}\rangle = X_c \sum_{b \in B_1} |z+b\rangle = \sum_{b \in B_1} |z+b+c\rangle = \sum_{b' \in B_1} |z+b'\rangle = |\overline{z}\rangle$$

So $X_c : c \in B_1$ does not affect the codeword. Thus the only X_c which can take a codeword $|\overline{z}\rangle$ to some other codeword $|\overline{z}'\rangle$ are those for which c is a nontrivial cycle (a cycle that is not a boundary). The minimum weight among these is the bit-flip distance.

To analyze this code under phase-flip errors, we will consider the stabilizer representation.

2.3 Stabilizer Construction

Here we will present the surface code defined above in terms of its stabilizers.

For a vertex v and face f, define the face and vertex Pauli operators as:

$$X_f := \prod_{e \in \partial_2 f} X_e, \qquad Z_v := \prod_{e : v \in \partial_1 e} Z_e \tag{5}$$

As illustrated in Figure , the face operator bit-flips the qubits surrounding a face, while the vertex operator phase-flips qubits adjacent to a vertex. For example, $Z_v |c\rangle = -|c\rangle$ if 1-chain c has an odd number of edges incident on vertex v, and $Z_v |c\rangle = |c\rangle$ otherwise.

We claim that these vertex and face operators generate the stabilizer group of the surface code.

First, notice that vertex and face operators commute with each other: If X_f, Z_v involve disjoint edges/qubits, then they obviously commute. Otherwise, their supports intersect if vertex v lies on the boundary of face f.

[4] Recall, in the computational basis $|i\rangle$ and $|j\rangle$ are orthogonal iff $i \neq j$. $\langle \overline{z} | \overline{z}' \rangle = 0$ does not mean considering $\overline{z}, \overline{z}'$ as vectors.

In this case, the boundary of f has exactly two edges incident on vertex v. Considering only these qubits, we have $X_f = \ldots \otimes X \otimes X \otimes \ldots$ and $Z_v = \ldots \otimes Z \otimes Z \otimes \ldots$. In the product $X_f Z_v$, each $XZ = -ZX$ pair picks up a -1 phase, and there are exactly two such shared qubits, so:

$$X_f Z_v = \ldots \otimes XZ \otimes XZ \otimes \cdots = \ldots \otimes (-ZX) \otimes (-ZX) \otimes \cdots = \ldots \otimes ZX \otimes ZX \otimes \cdots = Z_v X_f$$

(For the other qubits, at least one of X_f or Z_v are the identity, so we don't need to consider them).

Now we must check that the code defined by stabilizers (5) is the same as the code with basis (4). To do this, we will show that projection onto the subspace defined by stabilizers (5) is a map that is surjective on the code spanned by (4).

For a chain $c \in C_1$, define the state

$$|\tilde{c}\rangle := \prod_f \frac{1 + X_f}{2} \prod_v \frac{1 + Z_v}{2} \; |c\rangle \tag{6}$$

This is a projection onto the stabilizer codespace (ie, the simultaneous $+1$ eigenspace of all stabilizers): For any operator O with all eigenvalues ± 1, the operator $\frac{1}{2}(I + O)$ is the projection onto the $+1$ eigenspace of O (simply consider its action on the eigenbasis).

Further, notice that the action of Z_v is essentially the same as the boundary operator ∂_1: a vertex $v \in \partial_1 c$ iff the chain c includes an odd number of edges incident on v, and $|c\rangle$ is in the -1 eigenspace of Z_v iff the same criteria holds.

$$Z_v \, |c\rangle = - \, |c\rangle \iff v \in \partial_1 c \tag{7}$$

So the stabilizer condition $Z_v \, |c\rangle = |c\rangle$ is equivalent to $\partial_1 c = 0 \iff c \in Z_1$.

Now consider the span of all such $|\tilde{c}\rangle$. By the above, we have $\partial c \neq 0 \implies |\tilde{c}\rangle = 0$ (since c is the -1 eigenspace of at least one Z_v, it will be destroyed in the projection). The projection onto X_f eigenspaces in (6) can be expanded as a sum over subsets of faces $\{f_i\}$:

$$\prod_f (1 + X_f) = \sum_{\{f_i\}} \prod_i X_{\partial_2 f_i} = \sum_{\{f_i\}} X_{\partial_2 (\sum_i f_i)} = \sum_{c_2 \in C_2} X_{\partial_2 c_2}$$

Since $\partial_2 : C_2 \to C_1$ is a group homomorphism, we have $\operatorname{Im} \partial_2 = B_1 \cong C_2 / \ker \partial_2$. Therefore, summing over the boundary of 2-chains is proportional (by $|\ker \partial_2|$) to summing over the 1-chains in B_1:

$$\sum_{c_2 \in C_2} X_{\partial_2 c_2} \propto \sum_{b \in B_1} X_b$$

For example, the boundary of a face f on a torus can be written as a boundary two different ways, since it is also the boundary of all faces except f. Thus the proportionality factor is 2, and notice also $|\ker d_2| = 2$ (it contains 0 and the 2-chain of all faces).

Consider acting the combined projection in (6) on $|c\rangle$. The Z_v projections destroy any c with $\partial_1 c \neq 0$, enforcing $c \in Z_1$. Then for $c \in Z_1$, the X_f projections yield:

$$|\tilde{c}\rangle = \prod_f \frac{1 + X_f}{2} \, |c\rangle \propto \sum_{b \in B_1} X_b \, |c\rangle = \sum_{b \in B_1} |c + b\rangle$$

This is exactly the surface code basis $|\overline{c}\rangle$ from (4)! Therefore the code defined by the stabilizers is the same as the one defined by the basis.

Intuitively, projection onto the Z_v stabilizers enforces that codewords are cycles. And projection onto the X_f stabilizers enforce that codewords are uniform superpositions over a homology class: X_f flips the bits on the boundary of face f, so if $X_f \, |\overline{c}\rangle = |\overline{c}\rangle$, then $|\overline{c}\rangle$ must be unchanged by flipping the boundary of a face.

(This is very similar to how uniform superpositions over a linear subspace (as in CSS codes) are unaffected by translation within the subspace.) The picture is:

$$C_1 \xrightarrow{Z_v} Z_1 \xrightarrow{X_f} H_1$$

That is, the Z_v stabilizers restrict C_1 to Z_1, and the X_f stabilizers enforce superpositions over homology, reducing to H_1.

Dimensions. We know the dimensions of the involved spaces from algebraic topology:

$$\begin{array}{ccccc} C_1 & \to & Z_1 & \to & H_1 \\ |E| & & |E|-|V|+1 & & |E|-|V|-|F|+2 \end{array}$$

We have $|V|+|F|$ total stabilizers, but our subspace is reduced in dimension by only $\dim C_1 - \dim H_1 = |V|+|F|-2$. This redundancy is exactly explained by the following two relations:

$$\prod_f X_f = 1, \qquad \prod_v Z_v = 1.$$

2.4 Phase-flips and Duality

We are now ready to analyze our surface code under phase-flips. We know we can consider phase-flips on a code $\mathcal{C}$ by equivalently considering bit-flips on the transformed code $H^{\otimes n}\mathcal{C}$. In CSS codes, we analyzed $H^{\otimes n}\mathcal{C}$ directly, in terms of the generators of $\mathcal{C}$. Here we will instead consider the code $H^{\otimes n}\mathcal{C}$ in terms of its stabilizers.

Notice that if code $\mathcal{C}$ has stabilizers $\{S_i\}$, then code $\widetilde{\mathcal{C}} = H^{\otimes n}\mathcal{C}$ has stabilizers which are "conjugated": $\{\widetilde{S_i}\} = \{H^{\otimes n} S_i H^{\otimes n}\}$. Since:

$$\widetilde{S_i}\widetilde{\mathcal{C}} = (H^{\otimes n} S_i H^{\otimes n})(H^{\otimes n}\mathcal{C}) = H^{\otimes n}\mathcal{C} = \widetilde{\mathcal{C}}$$

So let us conjugate the stabilizers (5) of our surface code. Recall that the Hadamard operator conjugates between X, Z: $HZH = X$, $HXH = Z$, and $H^2 = I$. Then:

$$\begin{aligned} \widetilde{X}_f &:= H^{\otimes E} X_f H^{\otimes E} = \prod_{e \in \partial_2 f} H X_e H = \prod_{e \in \partial_2 f} Z_e \\ \widetilde{Z}_v &:= W^{\otimes E} Z_v W^{\otimes E} = \prod_{e : v \in \partial_1 e} H Z_e H = \prod_{e : v \in \partial_1 e} X_e \end{aligned}$$

There is an interesting duality when we compare these to our original stabilizers:

$$X_f := \prod_{e \in \partial_2 f} X_e, \qquad Z_v := \prod_{e : v \in \partial_1 e} Z_e$$

In fact, the supports of X_f, Z_v are in some (topological) sense duals of each other: X_f operates on the edges adjacent to a face, and Z_v operates on edges adjacent to a vertex. Consider the *dual lattice* on our surface, where vertices f^* correspond to faces f of our original lattice, with edges e^* between adjacent faces. There is a natural correspondence between edges e^* and e, since an edge e in the original lattice borders two faces f_1, f_2, and so induces an edge $e^* = (f_1^*, f_2^*)$ in the dual).

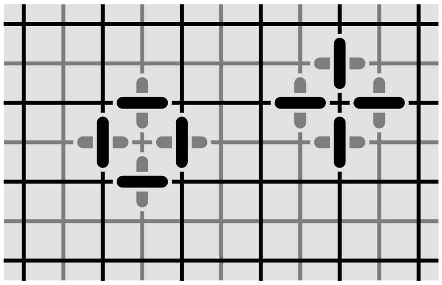


Figure 3: A lattice (black) and its dual (gray). Taking the dual interchanges vertices and faces, as well as vertex and face operators. Reproduced from [1].

With this correspondence, we have [5]

$$Z_{f^*} = \prod_{e^*: f^* \in \partial_1 e^*} Z_{e^*} = \prod_{e \in \partial_2 f} Z_e$$

But this is exactly $\widetilde{X}_f$! We can similarly find X_{v^*}, and we notice:

$$\widetilde{X}_f = Z_{f^*}, \qquad \widetilde{Z}_v = X_{v^*}$$

That is, conjugating the stabilizers takes them to stabilizers on the dual lattice. Therefore, the transformed code $\widetilde{C} = H^{\otimes n}\mathcal{C}$ of a surface code $\mathcal{C}$ is just a surface code on the dual lattice! Thus the phase-flip distance of the surface code is the length of the shortest non-trivial cycle in the *dual lattice*.

2.5 Concrete Example

If we take the original surface to be a $d \times d$ square lattice, then the dual lattice is again a $d \times d$ square lattice. The shortest nontrivial cycle on both of these have length d, so the distance of this code is d (both phase-flip and bit-flip). There are a total of $2d^2$ involved qubits (edges in the lattice), and the dimension of the codespace is $|H_1| = 2$. So this forms a $[[2d^2, 2, d]]$ quantum error-correcting code.

3 Sketch: Logical Operators on Codewords

Here we briefly mention another interesting idea in QECCs: logical operators on codewords. This arises in the setting of *fault-tolerant computing*: Here, we want not only to use a QECC to protect a state, but we want to *perform computational operations on the protected state.* That is, we want our entire quantum computation to operate on states protected by a QECC. To do this, we will need to find operators on codewords which correspond to operators on the original message.

An Example. In our repetition code, for example, the operator $X \otimes X \otimes X$ flips between $|000\rangle$ and $|111\rangle$, and thus acts as a "logical X" operator on our message: it takes an encoding of $|0\rangle$ to an encoding of $|1\rangle$. That is (let f be the encoding transform):

$$X \otimes X \otimes X f(|\psi\rangle) = f(X|\psi\rangle)$$

[5] We are loosely using ∂_1 to map $C_1^* \to C_2^*$ by using the primal-dual correspondences, although technically this should be called ∂_2^*.

Similarly, the operator $Z \otimes I \otimes I$ (or also $I \otimes Z \otimes I$, etc) acts as a "logical Z" on the codespace, since it takes $|111\rangle$ to $-|111\rangle$ (taking an encoding of $|1\rangle$ to an encoding of $-|1\rangle$), but leaves $|000\rangle$ unchanged.

Algebra. There is a nice algebraic characterization of logical operators for stabilizer codes. Let a code $\mathcal{C}$ be defined by stabilizers $\{S_i\}$, generating stabilizer subgroup $\mathcal{S}$.

We are only interested in operators A which take the codespace $\mathcal{C}$ to itself, ie: $A\mathcal{C} = \mathcal{C}$. (Other operators cannot possibly correspond to logical operations on the message). Further, let us consider only operators in the Pauli group $\mathcal{P}_n$. From our discussion in Section 2.4, we know that the code $A\mathcal{C}$ has stabilizers $\{AS_iA^\dagger\}$, generating stabilizer group $A\mathcal{S}A^\dagger$. For $A\mathcal{C} = \mathcal{C}$, we require exactly that the transformed stabilizer group is the same as the original $\mathcal{S}$:

$$A\mathcal{S}A^\dagger = \mathcal{S}$$

Such operators are, by definition, exactly those in the normalizer $\mathcal{N}(\mathcal{S})$ of $\mathcal{S}$ in $\mathcal{P}_n$. However, $S \subseteq \mathcal{N}(\mathcal{S})$, so some of the operators in the normalizer correspond to trivial operations (eg, applying a stabilizer $S_i \in \mathcal{S}$ will not affect codewords, by definition). To find the nontrivial operators, we must consider operators in the normalizer that are not in the stabilizer: $\mathcal{N}(\mathcal{S}) - \mathcal{S}$. However, some of these nontrivial operators may in fact correspond to the same logical operation. That is, we may have some $N \neq N' \in \mathcal{N}(\mathcal{S})$ s.t.:

$$Nc = N'c \quad \forall c \in \mathcal{C}$$

But then:

$$N'^\dagger Nc = c \quad \forall c \in \mathcal{C} \iff N'^\dagger N \in \mathcal{S}$$

Thus the correct way to describe all distinct nontrivial logical Pauli operators on codewords is by the quotient group $\mathcal{N}(\mathcal{S})/\mathcal{S}$.

For a comprehensive and very readable treatment of quantum stabilizer codes, see Daniel Gottesman's thesis [2].

Topology. *It turns out* that the group of logical operators for a surface code is $\cong H_1 \times H_1^*$. This can be shown algebraically, by considering $\mathcal{N}(\mathcal{S})/\mathcal{S}$ for the surface code stabilizers $\mathcal{S}$: *it turns out* that $\mathcal{N}(\mathcal{S}) \cong Z_1 \times Z_1^*$ and $\mathcal{S} \cong B_1 \times B_1^*$. Intuitively, the B_1 component corresponds to X_f stabilizers, and B_1^* (on the dual lattice) corresponds to Z_v stabilizers.

For further discussion (and also more types of topological codes), see the chapter [1].

4 References

[1] H. Bombin, "An Introduction to Topological Quantum Codes," *ArXiv e-prints*, Nov. 2013. [Online]. Available: http://arxiv.org/abs/1311.0277

[2] D. Gottesman, "Stabilizer codes and quantum error correction," 1997. [Online]. Available: http://arxiv.org/pdf/quant-ph/9705052v1.pdf