Cleaning lemma for stabilizer codes

The *cleaning lemma* for stabilizer codes says the following: For an $[[n, k]]$ stabilizer code, let M denote a subset of the n qubits in the code block, and let M^c denote the complementary set of qubits. If x is one of the code's logical Pauli operators, we say that x can be *cleaned* on M if there is a logically equivalent Pauli operator $x' = xy$ (where y is an element of the code stabilizer S) such that x' acts nontrivially only on M^c:

$$x' = I_M \otimes Q_{M^c}. \tag{1}$$

We say that x can be *supported* on M if it can be cleaned on M^c. Let $g(M)$ denote the number of independent logical Pauli operators that can be supported on M and let $g(M^c)$ denote the number of independent Pauli operators that can be supported on M^c. Then the cleaning lemma asserts that

$$g(M) + g(M^c) = 2k. \tag{2}$$

In particular, therefore, if no logical operator can be supported on M, then the complete k-qubit logical Pauli group can be supported on its complement.

We say that the subset M is *correctable* if erasure of the qubits in M is a correctable error. From the error correction conditions for stabilizer codes, we may say that if M is correctable, then any Pauli operator supported on M either anticommutes with the stabilizer or is contained in the stabilizer. Conversely, if M is not correctable, then there is a nontrivial Pauli operator supported on M which commutes with the stabilizer and is not contained in the statilizer; that is, if M is not correctable, then there is a nontrivial logical operator supported on M.

To prove the cleaning lemma we proceed as follows. We regard the abelianized n-qubit Pauli group P as a $(2n)$-dimensional vector space over the binary field $\mathbb{F}_2$, and say that the vectors x and y are orthogonal if the corresponding elements of P commute. Let P_M denote the subspace of P which is supported on the subset M of the n qubits. Let S denote the stabilizer of an $[[n, k]]$ quantum stabilizer code. Let $[T]$ denote the dimension of a subspace T.

We may express S as

$$S = S_M \oplus S_{M^c} \oplus S'. \tag{3}$$

Here $S_M = S \cap P_M$ is the subspace of S which is supported on M, $S_{M^c} = S \cap P_{M^c}$ is the subspace of S which is supported on M^c, and S' is a third subspace of S. For any vector x, we may consider its *restriction* $x|_M$ to the set M. For a Pauli operator $x = x_1 \otimes x_2$, with x_1 supported on M and x_2 supported on M^c, its restriction to M is $x_1 \otimes I$.

Now consider the restriction $S'|_M$ of S' to M. We claim that $[S_M \oplus S'|_M] = [S_M \oplus S'] = [S_M] + [S']$. That is, linearly independent vectors in $S_M + S'$ remain linearly independent when restricted to M. If this were not so, then a nontrivial linear combination of these vectors would have a trivial restriction to M, and would therefore be contained in S_{M^c}.

Let's compute the dimension $[\left(S^\perp\right)_M]$ of $\left(S^\perp\right)_M = S^\perp \cap P_M$, where $S^\perp$ is the orthogonal complement of S in P. Note that, because S_{M^c} is trivially orthogonal to P_M, $\left(S^\perp\right)_M$ is the

orthogonal complement in P_M of the restriction $(S_M \oplus S')\,|_M$ of $S_M \oplus S'$ to M. Therefore, by counting dimensions,

$$[\left(S^\perp\right)_M] = [P_M] - [S_M \oplus S'|_M] = [P_M] - [S_M] - [S']. \tag{4}$$

By applying the same reasoning with M replaced by M^c, we have

$$[\left(S^\perp\right)_{M^c}] = [P_{M^c}] - [S_{M^c} \oplus S'|_{M^c}] = [P_{M^c}] - [S_{M^c}] - [S']. \tag{5}$$

Logical operators supported on M are elements of P_M which commute with the stabilizer and are not contained in the stabilizer (and same for M^c); therefore

$$\begin{aligned} g(M) &= [\left(S^\perp\right)_M] - [S_M] = [P_M] - 2[S_M] - [S'], \\ g(M^c) &= [\left(S^\perp\right)_{M^c}] - [S_{M^c}] = [P_{M^c}] - 2[S_{M^c}] - [S'], \end{aligned} \tag{6}$$

and hence

$$\begin{aligned} g(M) + g(M^c) &= [P_M] + [P_{M^c}] - 2\left([S_M] + [S_{M^c}] + [S']\right), \\ &= [P] - 2[S] = 2n - 2(n-k) = 2k, \end{aligned} \tag{7}$$

as we wanted to show.

An important caveat: For a logical operator $x \in S^\perp \setminus S$, there might exist y, y' in S such that xy is supported on M and xy' is supported on M^c. Therefore, we may not conclude from eq.(2) that *each* of the code's $2k$ independent logical Pauli operators can be supported on either M or M^c. Consider, for example, the toric code with $k = 2$, where M is a narrow strip of qubits that winds around one cycle of the torus. This strip supports the code's string logical operators X_1 and Z_2, where 1,2 denote the two protected qubits, and M^c also supports these same two logical operators; hence $g(M) = 2$ and $g(M^c) = 2$. But neither M nor M^c supports X_2 or Z_1, the string logical operators that wind around the complementary cycle of the torus.

If erasure of M is correctable, then $g(M) = 0$ and we conclude that $g(M^c) = 2k$, so that all logical Pauli operators may be supported on M^c. We will use this consequence of the cleaning lemma in the arguments below.

A bound on topological stabilizer codes

We say that a stabilizer code family is *local* if we may choose all stabilizer generators to be geometrically local in D spatial dimensions, for some finite D. If in addition the code distance d increases without bound as the code length n increases, we say that the code is *topological*. For example, if the qubits are arranged in a D-dimensional (hyper)cubic lattice, the code is local if each stabilizer generator has its support on a (hyper)cube with side length r, where r is a constant independent of the length n of the code. When a code is local in D dimensions, we may call it a *D-dimensional code*, dropping the word "local" which is understood. We say that r is the *range* of the code generators. Note that for a topological code family, any set of qubits of constant size is correctable for sufficiently large n.

Here we prove a bound on the code parameters $[[n, k, d]]$ for two-dimensional (2D) stabilizer codes: $kd^2 = O(n)$.

The proof of the bound makes use of three lemmas — the cleaning lemma, the union lemma, and the expansion lemma. The cleaning lemma for stabilizer codes asserts that for any correctable set M and logical Pauli operator x, we may choose x to be supported on the complement M^c of M. That is, for any logical operator x (commuting with the code stabilizer S) there exists $y \in S$ such that xy is supported on M^c. A proof of the cleaning lemma is provided in the previous section.

The union lemma. We say that two sets of qubits are *separated* if no stabilizer generator has support on both sets. For a 2D code this is ensured if no $r \times r$ square intersects with both sets. The union lemma says that if M_1 and M_2 are separated and both are correctable, then their union $M_1 \cup M_2$ is also correctable. We denote this union as M_1M_2. To prove the union lemma by contradiction, suppose that M_1M_2 is not correctable, in which case there exists a nontrivial logical operator x supported on M_1M_2, of the form

$$x = (y_1)_{M_1} \otimes (y_2)_{M_2} \otimes I_{(M_1M_2)^c}. \tag{8}$$

Because x is logical, it commutes will all stabilizer generators, and since no generator has support on both M_1 and M_2, it must be that

$$x_1 = (y_1)_{M_1} \otimes I_{M_2} \otimes I_{(M_1M_2)^c} \quad \text{and} \quad x_2 = I_{M_1} \otimes (y_2)_{M_2} \otimes I_{(M_1M_2)^c} \tag{9}$$

also commute with all stabilizer generators, and hence are logical. Furthermore, since $x = x_1x_2$, and x is nontrivial, either x_1 or x_2 must be nontrivial. This contradicts our assumption that M_1 and M_2 are correctable, proving the lemma.

An alternative way to formulate the proof is to note that if x is any logical operator, then we can clean either M_1 or M_2, because both are correctable. But because M_1 and M_2 are separated, the stabilizer generators that clean M_1 do not act on M_2, and the stabilizer generators that clean M_2 do not act on M_1. Therefore we can clean M_1 and M_2 simultaneously; i.e., we can clean the union M_1M_2. This means that all logical operators can be supported on $(M_1M_2)^c$, $g((M_1M_2)^c) = 2k$, and hence by the cleaning lemma $g(M_1M_2) = 0$. We conclude that M_1M_2 supports no nontrival logical operators and is therefore correctable.

The expansion lemma. The expansion lemma requires a bit more explanation. Let M' denote the support of all stabilizer generators which act nontrivially on M. The *external boundary* ∂_+M of M is $M' \cap M^c$, and the *internal boundary* ∂_-M of M is $(M^c)' \cap M$. For a local code with range r, we may visualize ∂_+M as a thin shell surrounding M (with thickness no greater than r), while ∂_-M is a thin shell surrounding M^c. The expansion lemma specifies conditions under which we can slightly augment the size of a correctable set while preserving its correctability. It asserts the following: Suppose that $M = NA$ (that is, $M = N \cup A$, where N and A are disjoint). Suppose further that A contains ∂_-M, and that N, A, and ∂_+M are correctable. Then M is correctable.

To prove the expansion lemma by contradiction, suppose that M is not correctable, in which case there is a nontrivial logical Pauli operator x which is supported on M. Furthermore, because A is correctable, there is a stabilizer element y, which we may choose to be

supported on M', such that $z = xy$ is cleaned on A:

$$z = w_N \otimes I_A \otimes v_{\partial_+ M} \otimes I_{(M')^c}. \tag{10}$$

Because z is logical, it commutes with all stabilizer generators, and because $A \supseteq \partial_- M$, no stabilizer generator acts nontrivially on both $N = M \setminus A$ and $\partial_+ M$. Therefore, the restriction z_1 of z to N is logical, and the restriction z_2 of z to $\partial_+ M$ is also logical. Furthermore, since $z = z_1 z_2$ and z is nontrivial, either z_1 or z_2 must be nontrivial. This contradicts our assumption that N and $\partial_+ M$ are correctable, proving the lemma.

The holographic principle. Next, we use the cleaning lemma and expansion lemma to infer the *holographic principle* for 2D codes, which asserts that, for some constant α, any square on the lattice with side length $\ell = \alpha d$ is a correctable set of qubits. Note that this is not obvious, since the number of qubits contained in the square is $\Omega(d^2)$, and hence, for a topological code family becomes much larger than the code distance d (the size of the smallest noncorrectable set) for sufficiently large n.

To prove the holographic principle, consider two concentric squares N and M, where N has side length $\ell - 4r$, and M has side length $\ell - 2r$; therefore, M' is contained in a square with side length ℓ, and $A = M \setminus N$ contains $\partial_- M$. The expansion lemma ensures that M is correctable if N, A, and $\partial_+ M$ are all correctable. By definition of distance a set is correctable if it contains fewer than d qubits, and therefore $\partial_+ M$ and A are both correctable provided that

$$|\partial_+ M| \leq |M'| - |M| \leq \ell^2 - (\ell - 2r)^2 < 4r\ell < d. \tag{11}$$

Now imagine starting with a small square, with area less than d so that the square is correctable, and gradually increasing the side length step-by-step, where the length grows by $2r$ in each step. According to eq.(11), the square continues to be correctable as long as the side length ℓ remains less than $d/4r$. This proves the holographic principle. We call this the holographic principle, somewhat whimsically, because if logical information is encoded in the square, then some of that logical information (though perhaps not all) must be accessible near the square's boundary.

Applying the decoupling principle. To prove our bound on $[[n, k, d]]$ we need one more result, which applies to more general binary quantum codes, not just to stabilizer codes. Suppose the code block is divided into three regions ABC, where A and B are both correctable. Then the number k of encoded qubits can be no larger than $|C|$, the number of qubits in C.

We obtain this result from the decoupling principle. Express the code block as MM^c, where M is a correctable set and M^c is its complement. Introduce a reference system R, of dimension 2^k, which is maximally entangled with the code's k logical qubits. Consider the Stinespring dilation of erasure channel applied to M. The dilation discards M, mapping it to the environment E; hence we may consider M itself to be the environment of the channel. The decoupling principle asserts that if the erasure channel is correctable, then the marginal density operator for RM is a product state, whose entropy is additive — the reference system

is uncorrelated with the erased subsystem:

$$H(RM) = H(R) + H(M) \implies k = H(M^c) - H(M), \tag{12}$$

where in the second equation we have replaced the entropy $H(R)$ of the maximally mixed reference system by its number of qubits k, and have used the feature that the state of RMM^C is pure, hence $H(RM) = H(M^c)$.

Now, recalling that A is correctable we apply eq.(12) to $M = A$ and its complement $M^c = BC$, obtaining

$$k = H(BC) - H(A) \leq H(B) + H(C) - H(A). \tag{13}$$

Likewise, since B is also correctable, we apply eq.(12) to $M = B$ and its complement $M^c = AC$, obtaining

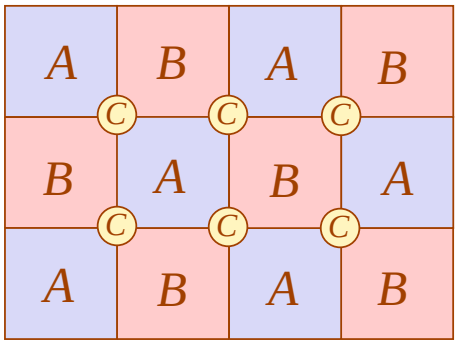
$$k = H(AC) - H(B) \leq H(A) + H(C) - H(B). \tag{14}$$

Combining these two inequalities we find

$$k \leq H(C) \leq |C|, \tag{15}$$

as desired.

Proof of the bound. Now we are ready to prove our bound on 2D topological codes. For a 2D stabilizer code, consider partitioning the code block into three sets ABC as shown here:



Each connected component of A is a square with side length less than $d/4r$, with corners clipped off as shown, and same for B. Each connected component of C has constant size, but is sufficiently large to separate the components of A by a distance larger than the range r of the stabilizer generators (and same for B). By the holographic principle, each component of A is correctable, and because the components are separated, A is correctable by the union

lemma. Likewise, B is also correctable. The number of components of C is $O(n/d^2)$, and each component has constant size. We conclude that

$$k \leq |C| = O(n/d^2), \tag{16}$$

which proves the bound. Though in the figure we have not indicated any holes punched through the code block, adding holes would not modify the conclusion.

We can also see that the bound is tight — 2D code families exist with kd^2 linear in n. To show this, consider a planar surface code defined on a rectangle with k holes punched through it, and with boundary conditions such that e anyons can "condense" (i.e., disappear) at any boundary. Nontrivial logical operators are e strings that connect distinct boundaries, and m strings that enclose one or more holes. The code distance (the smallest weight of a nontrivial logical operator) will be at least d if each hole has a perimeter at least d, and no two boundaries are closer together than distance d. We can easily arrange k square holes, each with side length $d/4$ such that all holes are distance d apart, and also distance d from the outer boundary, on a rectangular lattice $2d$ sites high and $2kd$ sites wide, hence with $4kd^2$ sites; the corresponding surface code has $n = O(kd^2)$ physical qubits.

Our bound shows that a 2D code family cannot be "good" — it cannot have both k and d scaling linearly with n. In fact, if k scales linearly with n, then the distance d must be a constant. To do better, we need to either allow non-Euclidean geometry, or to allow stabilier generators that are not geometrically local.

2D stabilizer codes have stringlike logical operators

Using the cleaning lemma and the union lemma we can see that any 2D stabilizer code has a nontrivial logical operator supported on a strip of constant width. A similar argument shows that a stabilizer code in D simensions has a nontrivial logical operator supported on a $(D-1)$-dimensional slab of constant thickness. (Note that the argument does not show that each logical operator is supported on a slab — only that at least one nontrivial logical operator has this structure.)

For qubits arranged on a square, we may argue as follows. Cover the code block with vertical strips, where each strip has constant thickness greater than r, the range of the stabilizer generators. Counting from the left edge, color the odd-numbered strips black and the even-numbered strips white. We claim that at least one of these strips supports a nontrivial logical operator. To prove the claim by contradiction, suppose that no nontrivial logical operator is supported on any of the black strips. Thus each black strip is correctable, and because the strips are separated by distance greater than r, the union of all black strips is also correctable and therefore cleanable. Consider a nontrivial logical operator; after cleaning this logical operator on the black strips, we obtain a nontrival logical operator x supported on the union of all white strips. Thus x commutes with all stabilizer generators. Because the while strips are separated by distance greater than r, no stabilizer generator has support on more than one strip. It follows that all stabilizer generators commute with the restriction of x to a single strip; that is, x is a product of strip operators, each supported on a vertical strip of constant width, and each of which is logical (commutes with the stabilizer). Because

x is a nontrivial logical operator, at least one of these strip operators must be both logical and nontrivial. This contradicts our assumption that no strip supports a nontrivial logical operator, hence proving, as desired, that there is a nontrivial logical operator supported on a single vertical strip with constant thickness.

Of course, we did not need to choose the strips to be vertical. The same argument applies to any way of dividing the code block into regions that can be colored white and black, such that distinct black strips are separated by distance greater than r and distinct white strips are separated by distance greater than r. In particular, it follows that there is a nontrivial logical operator supported on a *horizontal* strip of constant thickness as well.

2D local stabilizer codes are not self correcting

Results that specify the geometry of a region that supports a logical operator have implications for the physical robustness of quantum memories.

Consider a stabilizer code with geometrically local generators $\{S_a\}$, where the qubits reside at the sites of a D-dimensional hypercubic lattice with linear size L. The generating set $\{S_a\}$ might be overcomplete, but we assume that the number of local generators acting nontrivially on each qubit is a constant independent of L. The local Hamiltonian

$$H = -\sum_a \frac{1}{2}\left(S_a - I\right), \tag{17}$$

has a 2^k-fold degenerate ground state with energy $E = 0$, where k is the number of protected qubits — each ground state is a simultaneous eigenstate with eigenvalue one of all elements of $\{S_a\}$. If a quantum memory governed by this Hamiltonian is subjected to thermal noise, how well protected is the 2^k-dimensional code space?

If $|\psi\rangle$ is a zero-energy eigenstate of H and x is a Pauli operator, then $x|\psi\rangle$ is an eigenstate of H with eigenvalue $E(x)$, where $E(x)$ is the number of elements of $\{S_a\}$ that anticommute with x. Thermal fluctuations may excite the memory, but excitations with energy cost E are suppressed by the Boltzmann factor $e^{-E/\tau}$ where τ is the temperature (and Boltzmann's constant k_B has been set to one). We suppose that the environment applies a sequence of weight-one Pauli operators to the system, so that the error history after t steps can be described as a walk on the Pauli group, starting at the identity:

$$\{x_i \in P, i = 0, 1, 2, 3, \ldots t\}, \tag{18}$$

where $x_0 = I$, and $x_{i+1}x_i^{-1}$ has weight one. Let $\mathcal{P}(z)$ denote the set of all such walks, with any number of steps, that start at I and terminate at $z \in P$. We define

$$\Delta(z) \equiv \min_{\gamma \in \mathcal{P}(z)} \max_{x \in \gamma} E(x), \tag{19}$$

the minimum energy barrier that must be surmounted by any walk that reaches Pauli operator z. Thus such walks occur with a probability per unit time suppressed by the Boltzmann

factor $e^{-\Delta(z)/\tau}$. We also define

$$\Delta_{\min} \equiv \min_{x \in S^{\perp} \setminus S} \Delta(x), \tag{20}$$

$$\tag{21}$$

Here $\Delta_{\min}$ is the lowest energy barrier protecting any nontrivial logical operator (representing a nontrivial coset of $S^{\perp}/S$).

We say that a quantum memory is *self correcting* if $\Delta_{\min}$ grows faster than logarithmically with L. In that case *all* nontrivial logical operators are suppressed by a Boltzmann factor whose reciprocal grows super-polynomially with L. Though the Pauli walk may not be a particularly accurate description of noise in realistic systems, it allows us to define the notion of barrier height precisely, and to state the criteria for self correction simply. Furthermore, we expect the Boltzmann factor $e^{-\Delta/\tau}$ suppressing the Pauli walk to provide a reasonable (though crude) estimate of the logical error rate for more realistic noise models, assuming that the system attains thermal equilibrium.

Our finding that 2D stabilizer codes always admit strip logical operators has implicatioins regarding the self correction properties of these codes. For any logical operator x supported on a vertical strip of constant width, we may build a Pauli walk that starts at I and ends at x by starting at the bottom of the strip, and then building the vertical string operator row by row. At each stage of this walk, any "excited" local stabilizer generator S_a such that $S_a = -1$ acts only on qubits which are within distance r of the boundary of the walk, where r is the range of the stabilizer generators The number of such qubits is $O(1)$ and the total number of stabilizer generators acting on these qubits is $O(1)$. Therefore, the energy cost of the partially completed walk, and hence $\Delta_{\min}$, are $O(1)$. We conclude the 2D stabilizer codes are not self correcting.

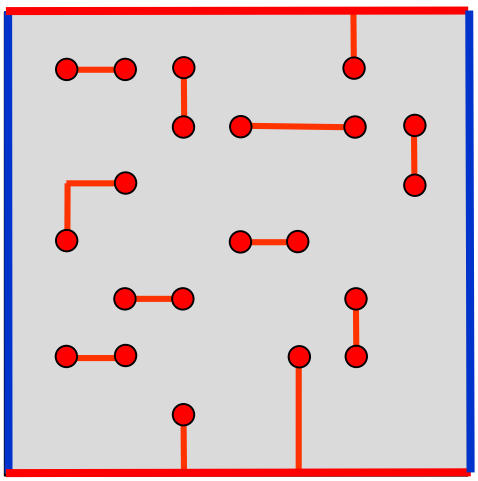
Surface code accuracy threshold

To create a stable quantum memory, we need not synthesize a topologically ordered material; instead we can *simulate* the material using whatever quantum computing hardware we prefer. Kitaev constructed a simple two-dimensional lattice model (the *surface code*), with a qubit at each lattice site, that exhibits $\mathbb{Z}_2$ topological order. Though it was first proposed 25 years ago, the surface code still offers a particularly promising route toward scalable fault-tolerant quantum computation. It has two major advantages. First, the quantum processing needed to diagnose and correct errors is remarkably simple. Second, and not unrelatedly, it can tolerate a relatively high gate error rate.

Errors afflicting a quantum memory can be expanded in terms of multi-qubit Pauli operators, and each such Pauli operator can be expressed as a product of an X-type error, where either X or the identity acts on each qubit, and a Z-type error, where either Z or the identity acts on each qubit. (A $Y = -iZX$ error is just the case where both X and Z act on the same qubit.) Therefore, our quantum memory will be well protected if we can correct both X-type and Z-type errors with high success probability. In the case of the surface code, there are two separate procedures for correcting X errors and correcting Z errors, and

both work in essentially the same way, so it will suffice to discuss only how the Z errors are corrected.

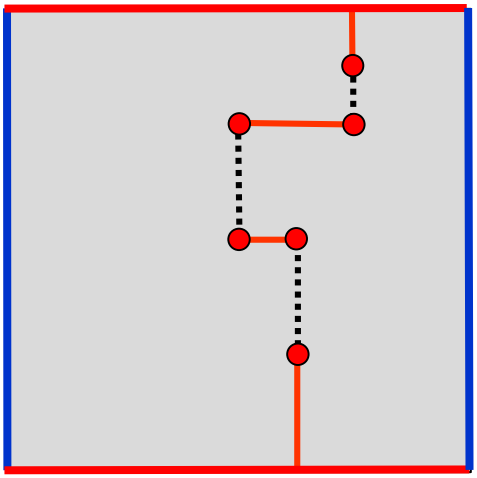
In (one version of) the surface code, the physical qubits reside on edges of a square lattice, and e anyons may reside on the sites of the lattice. Suppose an unknown quantum state $\alpha|\bar{0}\rangle + \beta|\bar{1}\rangle$ has been stored in the code space, where $|\bar{0}\rangle$ and $|\bar{1}\rangle$ are the encoded $\bar{Z}$ eigenstates. After this state is encoded, Z errors occur on some of the qubits, knocking the state out of the code space by creating e anyons. A snapshot of a typical error configuration is shown here:



Edges on which the qubits have Z errors (colored red) define a set of connected "error chains," and pairs of anyons appear at the endpoints of each error chain. The positions of the anyons (and hence the endpoints of the error chains) can be identified by a simple quantum computation. After finding their positions we can remove these anyons two at a time; we select a pair of anyons, and apply Z to all the qubits along a "recovery chain" that connects the pair, in effect bringing the pair of anyons together to annihilate. Alternatively, we can remove a single anyon by choosing a recovery chain that connects that anyon to the top or bottom edge. Our goal is to remove all of the anyons, returning the state to the code space, and (we hope) restoring the initial encoded state.

The anyon positions are said to constitute an error "syndrome" because they help us to diagnose the damage sustained by the physical qubits in the code block. Even though this syndrome locates the boundary points of the error chains, we don't know the configuration of the error chains themselves, so our recovery chains won't necessarily coincide with the error chains, or even connect together the same pairs of anyons. But they don't have to. If each connected path resulting from combining the error chain with the recovery chain forms a closed loop in the bulk, or an open path with both its endpoints lying on the same edge (either top or bottom), then error recovery is successful. This works because of the properties of the topologically ordered medium: creation of a pair of anyons followed by pair annihilation, or creation of a single anyon at the bottom (top) edge followed by annihilation at the bottom (top) edge are processes that act trivially on the code space. On

the other hand, suppose that the error chain combined with the recovery chain produces a path connecting the bottom and top edges as shown here:



Then (if there are an odd number of such paths) a logical $\bar{Z}$ error occurs and our recovery procedure fails.

To keep things simple, consider a stochastic independent noise model, in which each qubit in the code block experiences a Z error with probability ϵ. Suppose we choose our recovery chains to have the minimal possible weight; that is, we return to the code space by applying Z to as few qubits as possible. Given the known positions of the anyons, this minimal chain can be computed efficiently with a classical computer. For this recovery procedure, we can find an upper bound on the probability of a logical error by the following argument.

We denote by d the minimal weight of a connected path from the bottom edge to the top edge; that is, d (the *distance* of the code) is the minimal weight of a $\bar{Z}$ logical operator. If our attempt to recover resulted in a logical $\bar{Z}$ error, there must be a path connecting the bottom and top edges of the code block such that each edge of the lattice on this path is in either an error chain or a recovery chain. Let's say this connected path has length $\ell \geq d$, and denote the path by C_ℓ. The number of errors on C_ℓ must be at least $\ell/2$ if ℓ is even, or $(\ell+1)/2$ if ℓ is odd; otherwise we could have found a lower weight recovery chain by applying Z on the error chains contained in C_ℓ, rather than to the qubits on C_ℓ which are complementary to the error chains on C_ℓ. The number of ways that the edges with Z errors could be distributed along C_ℓ is no more than 2^ℓ (each qubit on C_ℓ either has an error or does not). Since, for each physical qubit, Z errors occur with probability ϵ, the probability that C_ℓ is contained in the union of error chains and recovery chains obeys

$$P(C_\ell) \leq 2^\ell \epsilon^{\ell/2}. \tag{22}$$

Let N_ℓ denote the number of paths connecting the bottom and top edges with length ℓ. For a logical error to occur, the combination of error chains and recovery chains must produce at least one path connecting the bottom and top edges. Using the upper bound eq.(22)

on the probability of each such path, and applying the union bound, we conclude that the probability of a logical $\bar{Z}$ error satisfies

$$P_{\text{logical}} \leq \sum_{\ell=d}^{n} N_\ell 2^\ell \epsilon^{\ell/2}. \tag{23}$$

The lower limit on the sum is $\ell = d$, the length of the shortest path connecting the bottom and top edges. The upper limit is n, the total number of qubits in the code block, which is therefore the maximum length of any path.

We can also find a simple upper bound on N_ℓ. Let's say our square lattice is $d \times d$. A path from the bottom to the top edge can begin at any one of d positions along the bottom edge, and in each of the ℓ steps along the path, there are three possible moves: straight ahead, left turn, or right turn. Therefore (even if we don't insist that the path reach the top edge), we have

$$N_\ell \leq d\ 3^\ell \implies P_{\text{logical}} \leq d \sum_{\ell=d}^{n} (36\epsilon)^{\ell/2}. \tag{24}$$

Now suppose that $\epsilon < 1/36$, so that the terms in the sum over ℓ decrease as ℓ increases. For a square lattice, the number of edges (qubits) in the code block is $n = O(d^2)$, so the number of terms in the sum is also $O(d^2)$, and we conclude that

$$P_{\text{logical}} \leq O(d^3)\,(\epsilon/\epsilon_0)^{d/2} \quad \text{for} \quad \epsilon < \epsilon_0 = 1/36 \approx .028\ . \tag{25}$$

Thus, this argument establishes that the surface code is a quantum memory with an *accuracy threshold* — for any constant $\epsilon < \epsilon_0$, the probability of a logical error decays exponentially as the code distance d increases (apart from a possible polynomial prefactor). If the physical error rate is below the threshold value ϵ_0, we can make the logical error rate arbitrarily small by choosing a sufficiently large code block. Unsurprisingly, in view of the crudeness of this argument, the actual value of the error threshold ϵ_0 is larger than we estimated. Monte Carlo simulations find $\epsilon_0 \approx .103$ for this minimum-weight decoding method.

Scalable quantum computing

To draw quantitative conclusions about the overhead cost of fault-tolerant quantum computing, refinements of this argument are needed. First of all, we implicitly assumed that the error syndrome measurements are perfect. In fact measurement errors occur, which means we need to repeat the measurement $O(d)$ times to acquire sufficiently trustworthy information about where the anyons are located. Secondly, we did not take into account the structure of the quantum circuit used to make these measurements. To determine whether an anyon is present at a particular site, four entangling two-qubit gates are needed, any one of which could be faulty, and a single fault can cause both an error in the measurement outcome and errors in the data qubits. A more complete analysis shows that the threshold error rate for the two-qubit gates is close to 1%. Numerical simulations find that for each round of syndrome measurement, the probability of a logical error rate scales roughly like

$$P_{\text{logical}} \approx 0.1\,(100p)^{(d+1)/2}, \tag{26}$$

where now p denotes the two-qubit gate error rate, and we have assumed that d is odd.

So far we have considered only the probability of a storage error for one protected qubit, but in a scalable fault-tolerant quantum computer we will need many protected qubits, and we will need to perform highly reliable universal quantum gates that act on these qubits. One can envision an architecture in which the logical qubits are arranged like square tiles on a surface, with buffer qubits filling gaps between the tiles. I won't go into the details of how the logical gates are executed, but it is helpful to realize that much of the logical processing can be executed by performing entangling measurements on pairs of logical blocks. For example, we can measure $\bar{X}_1 \otimes \bar{X}_2$, where blocks 1 and 2 reside on adjacent tiles, by fusing the blocks together along their red edges and then cutting the blocks apart again, a process called "lattice surgery." The fusing and cutting are achieved by measurements that activate the buffer qubits in between the edges of the two blocks, followed by measurements that decouple the buffer qubits.

The good news is that the error rates for logical gates are not much worse than the storage error rates we have already discussed, except we should keep in mind that we need to repeat the syndrome measurement $O(d)$ times in each logical gate cycle. The bad news is that eq.(26) indicates that we'll need a rather large code distance if we want to make the logical error rate very small. Suppose, for example, that we would like to run Shor's algorithm to factor a 2048-bit number, which would break the RSA cryptosystem, and suppose that the physical two-qubit gate error rate is 10^{-3}, better than in current multi-qubit devices. A recent analysis calls for a logical error probability $\approx 10^{-15}$ per round of syndrome measurement, and hence a code distance of $d = 27$. The number of physical qubits per code block, including ancilla qubits needed for syndrome measurement and lattice surgery, is $2(d+1)^2 = 1568$, and the total number of logical qubits used in this version of the factoring algorithm is about 14,000, pushing the physical qubit count above 20 million. That's a lot!

There are many challenges to making large-scale fault-tolerant quantum computing practical, including serious systems engineering issues. There are also issues of principle to consider — what is required for a fault-tolerant scheme to be scalable, and what conditions must be satisfied by the noise model? One essential requirement is some form of cooling, to extract the entropy introduced by noise. In the protocol described above, entropy is extracted by measuring and resetting ancilla qubits in each round of syndrome measurement. Parallel operations are also necessary, so noise can be controlled in different parts of the computer simultaneously.

The analysis leading to eq.(26) is based on a simple noise model in which gate errors are stochastic (rather than coherent) and there are no correlations among errors in different gates. The fault-tolerant methods should work for more realistic noise models, as long as the errors are sufficiently weak and not too strongly correlated. By benchmarking logical error rates using relatively small quantum codes in near-term devices, we will gain valuable insight into how effectively quantum error correction protects computations performed on actual quantum hardware.