Non-Pauli errors in the three-dimensional surface code

Thomas R. Scruby [1,2,3,*] Michael Vasmer [4,5,†] and Dan E. Browne [3,‡]

[1]*Okinawa Institute of Science and Technology, Okinawa 904-0495, Japan*
[2]*National Institute of Informatics, Tokyo 101-8430, Japan*
[3]*Department of Physics and Astronomy, University College London, London WC1E 6BT, United Kingdom*
[4]*Perimeter Institute for Theoretical Physics, Waterloo, Ontario N2L 2Y5, Canada*
[5]*Institute for Quantum Computing, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada*

(Received 10 June 2022; accepted 26 August 2022; published 21 October 2022)

A powerful feature of stabilizer error correcting codes is the fact that stabilizer measurement projects arbitrary errors to Pauli errors, greatly simplifying the physical error correction process as well as classical simulations of code performance. However, logical non-Clifford operations can map Pauli errors to non-Pauli (Clifford) errors, and while subsequent stabilizer measurements will project the Clifford errors back to Pauli errors the resulting distributions will possess additional correlations that depend on both the nature of the logical operation and the structure of the code. Previous work has studied these effects when applying a transversal T gate to the three-dimensional color code and shown the existence of a nonlocal "linking charge" phenomenon between membranes of intersecting errors. In this paper we generalise these results to the case of a CCZ gate in the three-dimensional surface code and find that many aspects of the problem are much more easily understood in this setting. In particular, the emergence of linking charge is a local effect rather than a nonlocal one. We use the relative simplicity of Clifford errors in this setting to simulate their effect on the performance of a single-shot magic state preparation process and find that their effect on the threshold is largely determined by probability of X errors occurring immediately prior to the application of the gate, after the most recent stabilizer measurement.

DOI: 10.1103/PhysRevResearch.4.043052

I. INTRODUCTION

Traditional wisdom in the field of quantum error correction says that, although there are infinitely many possible quantum errors, measurements made as part of the error correction process will project this spectrum of errors to a discrete set and it is sufficient to consider only this set when examining the performance of the code [1–3]. In particular, for stabilizer error correcting codes this set is just the $n-$ qubit Pauli group (up to phases). This fact greatly simplifies the study of these codes, and in particular enables efficient numerical investigation of code performance [4–7].

In recent years further study has shown that things are not quite so simple. For example, *coherent errors* (small rotations on all qubits of a code) will indeed be projected to distributions of Pauli errors, but this projection can also create a phase conditional on the presence/absence of a logical operator and this will cause a rotation in the logical space [8–13]. Another kind of non-Pauli error (the subject of this paper) is produced when Pauli errors in a code are mapped to Clifford errors by the application of a non-Clifford transversal gate [14]. Subsequent stabilizer measurements will map these errors back to Paulis again but not to the same error we started with, instead producing errors with complex nonlocal correlations [15–17]. The effects of Clifford errors and approaches to correcting them have also been studied in more general settings [18,19], and error membranes very similar to those considered in this paper previously appeared in [20].

Previous studies of Clifford errors due to transversal non-Clifford gates have mostly been restricted to the setting of the color code [21–23], where [in the three-dimensional (3D) variant] transversal application of T and $T^\dagger$ can map distributions of X errors to distributions of S and $S^\dagger$ (in addition to the original Pauli error). In Sec. II we review these results and in Sec. III we generalise them to the setting of the 3D surface code, which admits a transversal CCZ [24] gate that can map X errors to CZ errors. We demonstrate similar behavior to what was observed in the color code, and additionally (in Appendix A) we show that our proof technique reproduces previous results when applied to the color code, providing a fresh perspective on the problem of Clifford errors in this setting.

Following this analytical study of these Clifford errors we numerically investigate their impact on code performance in Sec. IV. We simulate a magic state preparation scheme that uses a transversal CCZ gate between three 3D surface codes followed by a dimension jump [25] to prepare the magic state $CCZ|+++\rangle$ in constant time. To assess the impact of Clifford

*thomas.scruby@oist.jp
†mvasmer@perimeterinstitute.ca
‡d.browne@ucl.ac.uk

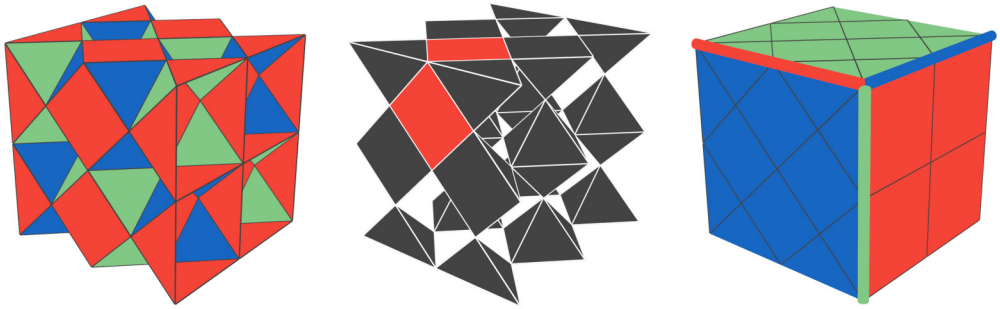


FIG. 1. (a) A rectified lattice supporting three surface codes. (b) One of the three surface codes. X stabilizers are on octahedral cells and Z stabilizers are on square faces. (c) The logical operator structure of the three codes. Logical Xs are sheets and logical Zs are strings. The intersection of logical Xs from any pair of codes is the support of a logical Z in the third.

errors on code performance we also simulate the codestate initialization and dimension jump steps in the absence of the CCZ gate and compare the performance of the two cases. We find that X errors occurring immediately before the CCZ gate (after the most recent stabilizer measurement) can have a large impact on the threshold, while the impact of earlier errors is fairly minimal.

Finally, we discuss the implications of these results and also their generalisations to other codes in Sec. V.

II. BACKGROUND

In this section we provide a review of the 3D surface (Sec. II A) and color (Sec. II B) codes and review previous results regarding Clifford errors in the latter (Sec. II C).

A. Surface codes

A 2D surface code can be defined by placing qubits on the edges of a 2D lattice and then assigning a Z stabilizer generator to each face and an X stabilizer generator to each vertex [26]. A 3D surface code can be defined in an identical fashion using a 3D lattice rather than a 2D one [27]. There also exists a "rotated" description of the 2D surface code, which is more qubit efficient, and assigns qubits to vertices and X and Z stabilizers to a bicoloring of faces [28]. A similar description exists for the 3D surface code, where qubits are placed on the vertices of a "rectified" lattice [24], obtained from the original lattice by placing a vertex at the center of each edge, joining these vertices with edges if they are part of the same face in the original lattice, and then deleting all the vertices and edges of the original lattice.

One such rectified lattice (obtained from a simple cubic lattice) is shown in Fig. 1(a), and an appropriate coloring of this lattice allows us to simultaneously define three distinct 3D surface codes. This is done by placing three qubits at each vertex (one for each code) and then assigning Z stabilizers of code i to faces with color κ_i and X stabilizers to cells that have no κ_i-colored faces [24]. Figure 1(b) shows one such code, where Z stabilizers are on square faces and X stabilizers are on octahedral cells. This code is equivalent to the 3D surface code obtained by assigning Z (X) stabilizers to faces (vertices) in the original simple cubic lattice. The other codes will have Z stabilizers on triangular faces and X stabilizers on cuboctahedral cells. In what follows we refer to these two types of 3D surface code as octahedral and cuboctahedral codes respectively.

The logical X operator of each code is a membrane supported on a boundary of the lattice, while the logical Z is a string supported on the intersection of two boundaries (up to composition with stabilizers). The code distance is therefore the lattice size L (for a lattice with dimensions $L \times L \times L$). An appropriate choice of boundary stabilizers ensures that the logical X operators for the three codes are all supported on different boundaries, and that the logical Z of code k is supported on the intersection of the logical operators of code i and j, as shown in Fig. 1(c). These three surface codes admit a transversal implementation of CCZ [24,29].

B. Color codes

The 2D color code is defined using a trivalent lattice with faces that are 3-colorable and have even numbers of vertices [21]. A qubit is associated with each vertex of this lattice and an X and a Z stabilizer generator are associated with each face. The fact that faces have even numbers of vertices ensures that X and Z stabilizers from the same face commute, while the trivalency/3-colorability of the lattice ensures that neighboring faces always meet at an edge, so X and Z stabilizers from different faces also commute. An example of a distance-3 2D color code is shown in Fig. 2(a). Logical X and Z operators are supported on any boundary of this code.

The 3D color code is defined using a four-valent lattice with four-colorable cells and qubits on vertices. If an edge connects two κ colored cells then we can color that edge κ, meaning that κ color cells will be formed from three colors of edge (the three colors not equal to κ). Each face of the lattice supports a Z stabilizer and each cell supports an X stabilizer. An example of a 15-qubit 3D color code is shown in Fig. 2(b). An implementation of logical X in this code is supported on any boundary of the tetrahedron while an implementation of logical Z is supported on any edge of the tetrahedron. The color code admits a transversal T gate, which involves the application of T and $T^\dagger$ to a bicoloring of qubits in the lattice [30].

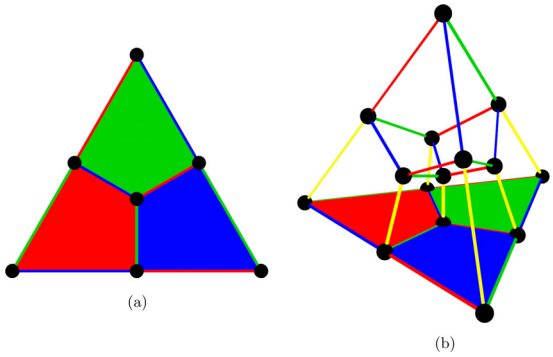


FIG. 2. (a) A 7-qubit 2D color code. (b) A 15-qubit 3D color code. A Z stabilizer is supported on each face and an X stabilizer is supported on each cell. The bottom boundary of this code matches the 2D color code shown in (a).

C. Clifford errors in the color code

The problem of Clifford errors in the 2D color code was examined by Yoshida in [15], although they are not referred to as such and instead are considered in the context of excitations in a symmetry-protected topological phase. Specifically, Yoshida examined the effect of applying a pattern of alternating S and $S^\dagger$ to all qubits within a particular region $\mathcal{R}$ defined by a subset of plaquettes of a particular color as in Fig. 3 . The 2D color code possesses a transveral S gate, which can be implemented via such an application of S and $S^\dagger$ to all qubits in the code and so this error can be thought of as a partial or incomplete logical operator. We can define the boundary of $\mathcal{R}$, $\partial\mathcal{R}$, to be the set of all plaquettes/stabilizer generators partially supported on $\mathcal{R}$. An important observation in this and all subsequent cases is that because logical operators must preserve the codespace (and therefore are not detectable by stabilizers of the code) this region of Clifford errors should only be detected by stabilizers in $\partial\mathcal{R}$. If it could be detected by a stabilizer not in $\partial\mathcal{R}$ then that stabilizer should also detect the logical S gate as these operators are not locally distinguishable except on the boundary of $\mathcal{R}$.

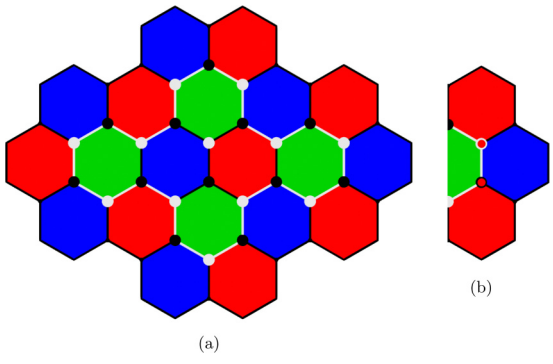


FIG. 3. A region of S errors in the 2D color code. S is applied to all qubits marked with a white circle and $S^\dagger$ is applied to all qubits marked with a black circle. The qubits in this region are those on the vertices of the four green plaquettes inside the loop of white edges while the plaquettes that border this region are either red or blue. (b) Part of the boundary region of (a). A ZZ error acting on the qubits marked with red circles anticommutes with the X stabilizers on the two red plaquettes.

The formally derived result of [15] agrees with this intuition. It says that if we apply the Clifford error shown in Fig. 3, for example, and then measure the stabilizers of the code (specifically we only need to measure the X stabilizers as Z and S commute) we will project the S and $S^\dagger$ errors to a distribution of Z errors that anticommute only with the red and blue stabilizers in $\partial\mathcal{R}$, with each such stabilizer returning a -1 measurement outcome with probability $p = 0.5$. Error strings in the 2D color code must either anticommute with stabilizers of all three colors or anticommute with a pair of stabilizers of the same color. The former is not possible in this case as there are only two colors of plaquette in $\partial\mathcal{R}$ and so the error distribution must be a collection of Z strings supported on qubits of $\mathcal{R}$ that run between same-colored plaquettes in $\partial\mathcal{R}$. Because each such string anticommutes with a pair of plaquettes the total number of -1 outcomes from plaquettes of each color should be even. For example, Fig. 3(b) shows a two-qubit Z error that anticommutes with a pair of red plaquettes in $\partial\mathcal{R}$.

A corresponding analysis of S errors in the 3D color code can be found in [16]. This case is arguably of greater practical relevance because, as mentioned previously, the 3D color code admits a transversal T gate implemented by an application of T and $T^\dagger$ to a bicoloring of the vertices (qubits) of the lattice. This gate will create regions of S and $S^\dagger$ errors wherever we have regions of X errors as $TXT^\dagger = e^{-i\pi/4}SX$ and $T^\dagger XT = e^{i\pi/4}S^\dagger X$.

We can initially consider an X error membrane defined on the vertices of a set of faces of color $\kappa_1\kappa_2$ (by which we mean they are formed from edges of color κ_1 and κ_2). This error will be detected by Z stabilizer generators on faces of color $\kappa_3\kappa_4$ at the boundary of the membrane. An example is shown in Fig. 4(a). When we apply the $\overline{T}$ gate described previously we will create S errors wherever we apply T and $S^\dagger$ errors wherever we apply $T^\dagger$.

Like the 2D color code, the 3D color code admits a transversal S gate. This gate can be implemented by a membrane of S and $S^\dagger$ (using the same coloring as the transversal T) with its edges at the boundaries of the code [31]. As with the 2D case, an S error membrane such as the one in Fig. 4(a) should only be detected by stabilizers at its boundary (i.e., by X stabilizers on cells that the syndrome loop passes through) because it is only locally distinguishable from the logical S operator in this region.

The results of [16] agree with [15] for this case, i.e., we expect measurement outcomes of $+1$ from all stabilizers except for X stabilizers on the membrane boundary, which we expect to return random outcomes but with an even parity of -1s for each color. However, more complex errors are possible in the 3D color code and these are where the results diverge from the 2D case.

Consider a pair of intersecting membranes with linked syndromes as in Fig. 4(b). This error is the product of two X error membranes of the form discussed above, and so one might expect that application of transversal T and measurement of the X stabilizers on the boundaries of these membranes would once again give random outcomes with an even parity of violated stabilizers of each color. However, what is shown in

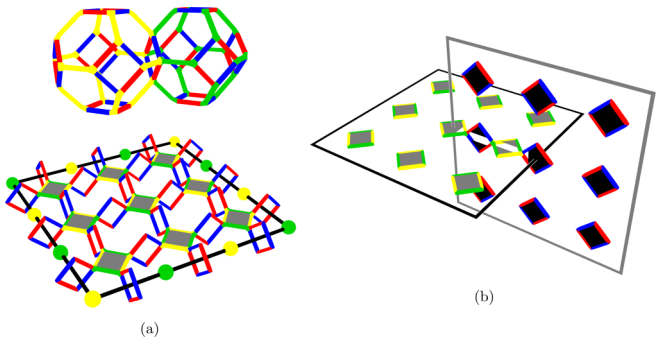


FIG. 4. (a) G (top left) and Y (top right) cells of a 3D color code. These cells meet at an RB face. [(a) bottom] A membrane of X errors in the 3D color code. The error is supported on qubits on the vertices of YG faces (grey) and detected by Z stabilizers on RB faces on the membrane's boundary. The resulting syndrome is shown by the black loop, which passes through the centres of the violated Z stabilizers. The two colors of dots on this syndrome mark the places where it passes through the center of a G or Y cell. In order to improve visual clarity full cells are not shown. (b) Two intersecting membranes of X errors with linked syndromes in the 3D color code. One is defined on YG faces and the other on RB faces. This error is the product of the two individual membranes so errors on the intersection cancel. One face of the YG membrane supports a Z stabilizer that detects the RB membrane and vice versa.

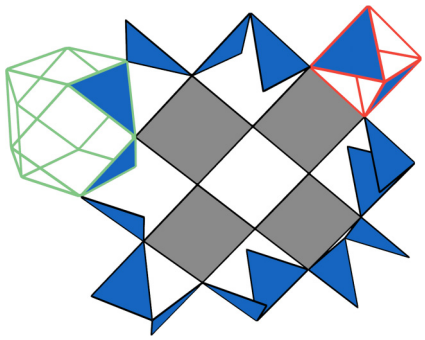


FIG. 5. An X error membrane in one of the cuboctahedral 3D surface codes that can be defined using the rectified lattice. The error is supported on qubits placed on vertices of the grey faces and is detected by Z stabilizers on the blue faces. These faces are part of cells that support X stabilizers in the other two codes (two examples shown, octahedral for one code and cuboctahedral for the other).

[16] is that we actually observe an odd number of violated stabilizers of each color. This is consistent with a distribution of Z errors as described previously plus an additional Z error string running between the two membrane boundaries (such an error string anticommutes with a G and Y cell on one boundary and an R and a B cell on the other boundary). This is termed a "linking charge" of the two membranes, since in the topological phase perspective of the 3D color code the charge distributions on the boundaries of the individual membranes are no longer independent. Previously each boundary was charge-neutral overall, whereas now the distribution for the pair of membranes is charge neutral but the distributions on individual membrane boundaries are not.

In the next section we provide a proof that these same phenomena occur in the 3D surface code, and explain their origin in depth. In Appendix A we show that our proof technique recovers the results of [16] when applied to the 3D color code and also examine the cases of CS and CCZ applied between multiple color codes, so readers wishing to understand the color code case in more detail can consult that Appendix (or alternatively [32]) for a different perspective on the problem.

III. CLIFFORD ERRORS IN THE 3D SURFACE CODE

A. Single error membrane in cleanable code regions

We now consider three 3D surface codes defined on a rectified lattice as in the previous section, and therefore admitting a transversal CCZ. We use notation where X_α^c means X operators on qubits from code $c \in \{1, 2, 3\}$ at vertices in the set α. We start with a single a membrane-like operator X_α^1 detected by Z stabilizers of code 1, which are faces of cells in codes 2 and 3 as in Fig. 5. We assume that this membrane exists in a cleanable region [33] of the code, i.e., for any given logical Pauli operator of the code we can find an implementation of this operator that has trivial intersection with the membrane. Using the commutation relation $(CCZ)(X \otimes I \otimes I) = (X \otimes CZ)(CCZ)$ we see that applying transversal CCZ in the presence of this error has the effect

$$\overline{CCZ} X_\alpha^1 |\overline{\psi}\rangle = X_\alpha^1 CZ_\alpha^{23} |\overline{\psi'}\rangle \tag{1}$$

where $|\overline{\psi}\rangle$ and $|\overline{\psi'}\rangle$ are states in the codespace of the three codes. CZ_α^{23} is a Clifford error analogous to the S error membrane we observed in the 3D color code. As with that error, this CZ error becomes a logical operator if applied to the full support of a logical X operator (as $\overline{CCZ}$ should preserve the codespace if X_α^1 was a logical X operator rather than an error). We therefore expect that, once again, this error should only be detected by stabilizers on the boundaries of the error membrane.

In order to consider the effect of CZ_α^{23} on the codestate $|\overline{\psi'}\rangle$ we can consider its effect individually on basis states. The state $|\overline{000}\rangle$ can be written as

$$|\overline{000}\rangle = \frac{1}{\sqrt{n}} \sum_{ijk} X_{\beta_i}^1 X_{\beta_j}^2 X_{\beta_k}^3 |\mathbf{0}\rangle \tag{2}$$

where X_β^m are X stabilizers of code m and $|\mathbf{0}\rangle$ is the all-zeros state of the qubits of all three codes. Other basis states can be written in a similar way by replacing these X stabilizers with products of X stabilizers and a logical X operator for a given code. However, as we are currently considering an error membrane in a cleanable region of the code we can always choose these logical operators such that they have trivial intersection with the membrane and thus the analysis of any basis state is equivalent to the analysis for $|\overline{000}\rangle$ in this case. We can apply CZ_α^{23} to this state and use the commutation relation $(CZ)(X \otimes I) = (X \otimes Z)(CZ)$ to find

$$\begin{aligned} CZ_\alpha^{23} |\overline{000}\rangle &= \frac{1}{\sqrt{n}} \sum_{ijk} CZ_\alpha^{23} X_{\beta_i}^1 X_{\beta_j}^2 X_{\beta_k}^3 |\mathbf{0}\rangle \\ &= \frac{1}{\sqrt{n}} \sum_{ijk} X_{\beta_i}^1 X_{\beta_j}^2 Z_{\alpha\cap\beta_j}^3 X_{\beta_k}^3 Z_{\alpha\cap\beta_k}^2 |\mathbf{0}\rangle \end{aligned} \tag{3}$$

where we have used that CZ_α^{23} acts trivially on $|\mathbf{0}\rangle$. We can then commute the Z terms to the right and absorb them into $|\mathbf{0}\rangle$ to obtain

$$|\phi\rangle = CZ_\alpha^{23}|\overline{000}\rangle = \frac{1}{\sqrt{n}}\sum_{ijk}(-1)^{|\alpha\cap\beta_j\cap\beta_k|}X^1_{\beta_i}X^2_{\beta_j}X^3_{\beta_k}|\mathbf{0}\rangle. \quad (4)$$

In order to investigate the possible measurement outcomes of a specific stabilizer $X^q_{\beta_l}$ we can split the state into $|\phi\rangle = a|\phi_l^+\rangle + b|\phi_l^-\rangle$ where $X^q_{\beta_l}|\phi_l^+\rangle = |\phi_l^+\rangle$ and $X^q_{\beta_l}|\phi_l^-\rangle = -|\phi_l^-\rangle$. Note that (neglecting normalisation) $|\phi_l^+\rangle$ will be a sum of pairs

$$|\phi_l^+\rangle = \sum_x \left(X^{123}_{\beta_x} + X^q_{\beta_l}X^{123}_{\beta_x}\right)|\mathbf{0}\rangle \quad (5)$$

while $|\phi_l^-\rangle$ will be a sum

$$|\phi_l^-\rangle = \sum_x \left(X^{123}_{\beta_x} - X^q_{\beta_l}X^{123}_{\beta_x}\right)|\mathbf{0}\rangle \quad (6)$$

where $X^{123}_{\beta_x} = X^1_{\beta_i}X^2_{\beta_j}X^3_{\beta_k}$ is a product of stabilizers from all three codes. The case where $q = 1$ is trivial because the CZ error is not supported in this code, and the cases where $q = 2$ and $q = 3$ are identical because of the symmetry of the CZ operator. It is therefore sufficient to consider only the case where $q = 2$. Then the pairs in (5) correspond to cases where $(-1)^{|\alpha\cap\beta_j\cap\beta_k|} = (-1)^{|\alpha\cap(\beta_j+\beta_l)\cap\beta_k|}$, which implies $|\alpha\cap\beta_l\cap\beta_k|$ is even ($\beta_j+\beta_l$ is the pointwise addition of these sets modulo 2). On the other hand, pairs in (6) correspond to cases where $(-1)^{|\alpha\cap\beta_j\cap\beta_k|} = -(-1)^{|\alpha\cap(\beta_j+\beta_l)\cap\beta_k|}$ meaning $|\alpha\cap\beta_l\cap\beta_k|$ must be odd in this case. We now consider three relevant types of stabilizer:

(i) Stabilizers not on the membrane boundary: Recall that if X^1_α was a logical X operator of code 1 then CZ^{23}_α should be a logical CZ of codes 2 and 3 and so must preserve the stabilizer groups of these codes. This means that for any cell of code 2 not on the boundary of the CZ membrane, the intersection of this cell with the CZ membrane must be the support of a Z stabilizer of code 3 or X stabilizers in one code would be mapped to Z errors in the other by a transversal application of CZ. This means that $\alpha\cap\beta_l$ is the support of a Z stabilizer of code 3 and since β_k is the support of an X stabilizer of code 3, $|\alpha\cap\beta_l\cap\beta_k|$ must be even for all such β_l and β_k. Therefore $|\phi\rangle = |\phi_l^+\rangle$ and we are in a $+1$ eigenstate of these stabilizers.

(ii) Stabilizers generators (cells) on the membrane boundary: The error X^1_α is detected by Z stabilizers supported on the faces of these cells as in Fig. 5. Each such face is the intersection between a pair of generators $X^2_{\beta_l}$ and $X^3_{\beta_k}$ and so $|\alpha\cap\beta_l\cap\beta_k|$ must be odd or a Z stabilizer on this face would not detect the error. There are two such faces for every generator on the membrane boundary (because the syndrome is a loop) and so every $X^2_{\beta_l}$ has two $X^3_{\beta_k}$ neighbors for which $|\alpha\cap\beta_l\cap\beta_k|$ is odd, which we will refer to as the "boundary neighbors" of $X^2_{\beta_l}$. The boundary neighbors are disjoint, so if $X^3_{\beta_k}$ is instead the product of the boundary neighbors then $|\alpha\cap\beta_l\cap\beta_k|$ is even. Thus, the terms in $|\phi_l^+\rangle$ are those for which $X^{123}_{\beta_x}$ contains neither or both of the boundary neighbors of $X^2_{\beta_l}$, whereas the terms in $|\phi_l^-\rangle$ are those where $X^{123}_{\beta_x}$ contains a single one of these neighbors. The overall superposition contains an equal number of each type of term so $|\phi\rangle = \frac{1}{\sqrt{2}}(|\phi_l^+\rangle + |\phi_l^-\rangle)$ and we measure a random ±1 outcome from this stabilizer.

(iii) The product of all stabilizer generators on the membrane boundary: For any generator $X^2_{\beta_j}$ on the membrane boundary the intersection $|\alpha\cap\beta_j\cap\beta_k|$ with a neighboring generator $X^3_{\beta_k}$ is odd (as discussed above). This means a Z operator $Z^3_{\alpha\cap\beta_j}$ anticommutes with these $X^3_{\beta_k}$, and for each such $X^3_{\beta_k}$ there are two generators $X^2_{\beta_j}$ that have this property, so if $X^2_{\beta_l}$ is the product of all generators from code 2 on the membrane boundary then $Z^3_{\alpha\cap\beta_l}$ is a stabilizer. This means that $|\alpha\cap\beta_l\cap\beta_k|$ is even for all $X^3_{\beta_k}$ and so $|\phi\rangle = |\phi_l^+\rangle$ for this choice of $X^2_{\beta_l}$.

In summary this gives us an analogous result to what was observed for an isolated membrane in the color code. Stabilizers not on the boundary always give $+1$. Stabilizer generators on the boundary give ±1 randomly but we must get an even number of -1 stabilizers in any given code.

B. Linked error membranes in cleanable code regions

Consider the case where we have X error membranes in codes 1 and 2

$$\overline{CCZ}X^1_\alpha X^2_\gamma|\overline{\psi}\rangle \quad (7)$$

such that $\alpha\cap\gamma$ is nonempty. Then commuting the CCZ to the right we have

$$X^1_\alpha CZ^{23}_\alpha X^2_\gamma CZ^{13}_\gamma|\overline{\psi'}\rangle = X^1_\alpha X^2_\gamma Z^3_{\alpha\cap\gamma}CZ^{23}_\alpha CZ^{13}_\gamma|\overline{\psi}\rangle. \quad (8)$$

So we now have some CZ errors acting on a codestate as before, but we also have a Z string $Z^3_{\alpha\cap\gamma}$ on the intersection of these membranes in code 3. This is the surface code equivalent of the linking charge string described in [16]. Thus we expect that in code 1 we get random outcomes from stabilizers on the boundary of CZ^{13}_γ and $+1$ outcomes from all others, with the total number of -1s even. In code 2 we expect the same thing on the boundary of CZ^{23}_α. In code 3 we expect random outcomes from stabilizers on the boundaries of both membranes, and also expect an odd parity of -1 stabilizers on each boundary due to the linking charge string.

The simplicity of this statement stands in stark contrast to the complexity of the original proof of this phemonenon in the case of the color code as presented in [16], to the extent that it may seem unremarkable to readers who are not familiar with that paper. Additionally, it is not only the mathematical origin of linking charge that is clearer in the surface code but also its physical significance. The transversal CCZ in this code is achieved because the intersection of logical X operators from any pair of codes is the support of a logical Z operator in the third and so the logical action $\overline{CCZ}_{123}\overline{X}_1\overline{X}_2\overline{CCZ}_{123} = \overline{X}_1\overline{X}_2\overline{Z}_3\overline{CZ}_{23}\overline{CZ}_{13}$ is correctly implemented. Linking charge in this case is just another example of this creation of Z strings on X membrane intersections. This is consistent with claims made in [16] that linking charge is important for the correct function of the transversal T gate in the 3D color code.

C. Error membranes in noncleanable regions

Finally we address the case of error membranes in noncleanable regions of the code. Consider

$$\overline{CCZ}X_\alpha^1|\overline{\psi}\rangle = X_\alpha^1 CZ_\alpha^{23}|\overline{\psi'}\rangle \tag{9}$$

where CZ_α^{23} now contains the support of a logical Z operator in code 2 or 3. We choose code 2 but this choice is not important. This means that any implementation of logical X in code 2 must have nontrivial intersection with X_α^1 and this changes our analysis for some codewords. For $|\overline{010}\rangle$ we have

$$\begin{aligned} CZ_\alpha^{23}|\overline{010}\rangle &= \frac{1}{\sqrt{n}}\sum_{ijk} CZ_\alpha^{23} X_{\beta_i}^1 \overline{X}_L^2 X_{\beta_j}^2 X_{\beta_k}^3 |\mathbf{0}\rangle \\ &= \frac{1}{\sqrt{n}}\sum_{ijk} X_{\beta_i}^1 \overline{X}_L^2 Z_{\alpha\cap L}^3 X_{\beta_j}^2 Z_{\alpha\cap\beta_j}^3 X_{\beta_k}^3 Z_{\alpha\cap\beta_k}^2 |\mathbf{0}\rangle \\ &= \frac{Z_{\alpha\cap L}^3}{\sqrt{n}}\sum_{ijk} (-1)^{\alpha\cap\beta_j\cap\beta_k} X_{\beta_i}^1 \overline{X}_L^2 X_{\beta_j}^2 X_{\beta_k}^3 |\mathbf{0}\rangle \end{aligned} \tag{10}$$

where $\overline{X}_L^2$ is a logical X implementation for code 2. We therefore create a global Z error on the intersection of the error membrane support α and the logical operator support L. Both α and L are membranes so we can always choose L such that $\alpha \cap L$ is a string. This does not affect observed syndromes (because $Z_{\alpha\cap L}^3$ runs from one side of the membrane to the other and so anticommutes with a pair of stabilizers on the membrane boundary) and also does not affect the encoded logical information unless $\alpha \cap L$ is the support of a logical Z operator for code 3. Therefore we only get a logical error from applying $\overline{CCZ}$ to X_α if α contains the support of logical operators of both code 2 and code 3.

IV. EFFECTS OF CLIFFORD ERRORS ON CODE PERFORMANCE

To investigate the effects of Clifford errors on the performance of an error correcting code we simulate a single-shot magic state preparation procedure that makes use of the surface code CCZ. Previous work has simulated the effect of Clifford errors arising from the application of transversal T in the color code [17] but did not include the contribution due to linking charge (as it cannot be modelled locally in that setting). On the other hand, that paper used a circuit noise model whereas we use a simpler phenomenological noise model. All code used for our simulations can be found at [34].

Our simulated procedure consists of the following steps:

(1) Prepare the three codes in the logical $|\overline{+}\rangle$ state by preparing all physical qubits in $|+\rangle$, measuring all Z stabilizers that correspond to faces of the lattice and applying a correction. This can be done in constant time as the 3D surface code allows for single-shot decoding of X errors.

(2) Apply transversal CCZ to the three codes to create the magic state $CCZ|+++\rangle$.

(3) Perform a 3D $\rightarrow$ 2D dimension jump in each code, such that we end with three entangled 2D surface codes in the state $CCZ|+++\rangle$. This can also be done in constant time at the cost of reduced code performance.

We apply measurement errors during the initial Z stabilizer measurements and also apply a depolarising error either just before or just after the CCZ gate, all with the same probability p. After all the steps are complete we use perfect stabilizer measurements to calculate a final correction for the 2D codes, check for logical errors and declare success or failure depending on the outcome. The results can be seen in Fig. 6 and Fig. 7, with observed thresholds as shown in Table I. To estimate the error thresholds we use standard finite-size scaling analysis [5]. Specifically, in the vicinity of the threshold we fit the data to the following ansatz:

$$p_{\text{fail}}(x) = a_0 + a_1 x + a_2 x^2, \tag{11}$$

where $x = (p - p_{\text{th}})L^{1/\nu}$ and a_0, a_1, a_2, p_{th} and ν are the parameters of the fit.

We observe that when the depolarising channel is applied after the CCZ gate the inclusion of CZ errors (in this case arising only due to measurement errors) results in only a marginally lower $\overline{Z}$ error threshold estimate in the octahedral surface code, and actually leads to a slight increase in the estimated $\overline{Z}$ error threshold value in the cuboctahedral surface

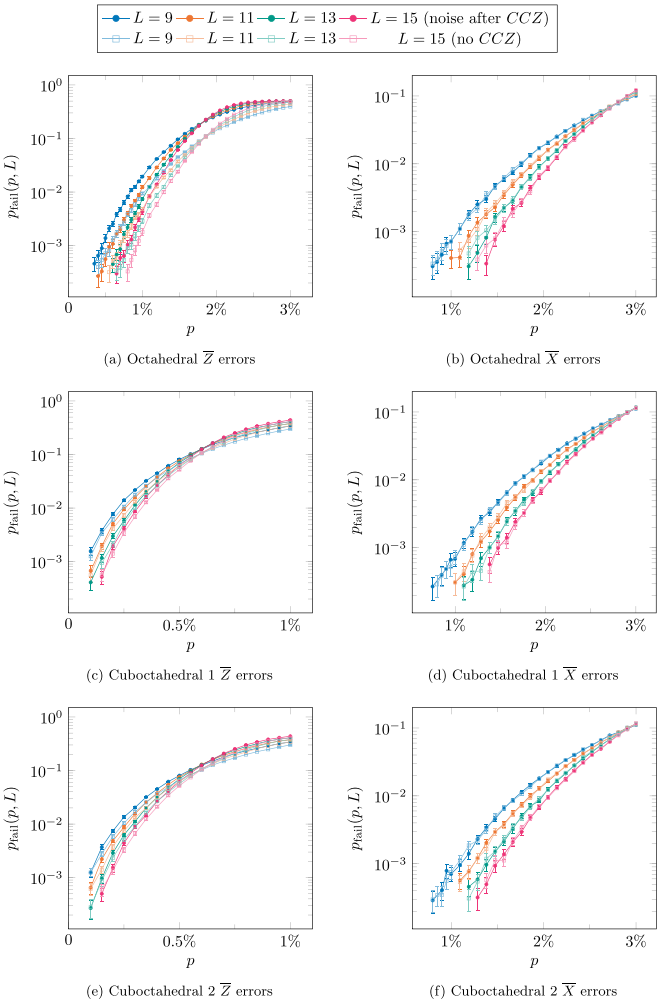
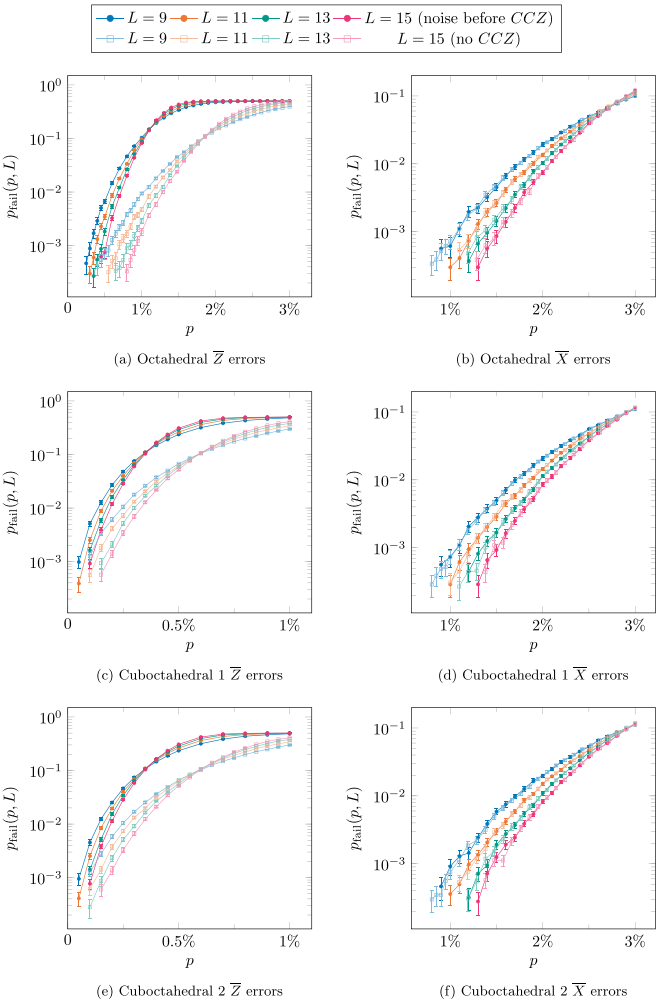


FIG. 6. For each of the three codes, we show a plot of the logical error rate p_{fail} (separately for $\overline{X}$ and $\overline{Z}$ errors) as a function of the depolarising/measurement error rate p for various lattice sizes L (with and without the CCZ gate). For these simulations, the depolarising noise channel was applied *after* the CCZ gate. The error bars show the Agresti-Coull 95% confidence intervals [35,36].



(a) Octahedral $\overline{Z}$ errors (b) Octahedral $\overline{X}$ errors

(c) Cuboctahedral 1 $\overline{Z}$ errors (d) Cuboctahedral 1 $\overline{X}$ errors

(e) Cuboctahedral 2 $\overline{Z}$ errors (f) Cuboctahedral 2 $\overline{X}$ errors

FIG. 7. For each of the three codes, we show a plot of the logical error rate p_{fail} (separately for $\overline{X}$ and $\overline{Z}$ errors) as a function of the depolarising/measurement error rate p for various lattice sizes L (with and without the CCZ gate). For these simulations, the depolarising noise channel was applied *before* the CCZ gate. The error bars show the Agresti-Coull 95% confidence intervals [35,36].

codes. On the other hand, when the depolarising channel is applied just before the CCZ gate the CZ errors (now due to both measurement and qubit errors) have a significant impact on the threshold. In Sec. IV A and Sec. IV B we consider each of these cases in detail, and following this analysis we provide a more detailed description of the above procedure, with explanations of state preparation, CCZ application and $3D \to 2D$ dimension jumping given in Sec. IV C, Sec. IV D, and Sec. IV E, respectively.

A. Noise after CCZ

To understand the minimal effect seen when the depolarising channel comes after the CCZ we first need to observe that the CZ error probability in this case is actually a function of lattice size L (because the CZ errors occur due to uncorrected X errors). When we are above the X threshold the likelihood of CZ errors increases with increasing L, whereas below the X threshold it decreases. We can then define an effective Z error probability

$$p_Z^{\text{eff}} = p_Z + q(L) \tag{12}$$

where p_Z is the per-qubit Z error probability due to the depolarising channel (i.e., $2p/3$ where p is the error probability used in our simulations) and $q(L)$ is the probability of additional Z errors produced by CZ errors + stabilizer measurement. If we consider only values of p_Z close to the Z error threshold then we are below the X error threshold so $q(L)$ will be small and will decrease with increasing L. In this region we can approximate the curves p_{fail}^L as straight lines $p_{\text{fail}}^L = m_L p_Z + c_L$, and if we choose a pair of lines $p_{\text{fail}}^{L_1}$ and $p_{\text{fail}}^{L_2}$ and replace p_Z with p_Z^{eff} then the point where $p_{\text{fail}}^{L_1} = p_{\text{fail}}^{L_2}$ shifts by an amount

$$\Delta p(L_1, L_2) = \frac{m_{L_2} q(L_2) - m_{L_1} q(L_1)}{m_{L_1} - m_{L_2}}. \tag{13}$$

If $L_2 > L_1$ then this shift is positive if $m_{L_2}/m_{L_1} < q(L_1)/q(L_2)$ and negative (or zero) otherwise. m_{L_2}/m_{L_1} is large in codes that deal well with Z errors, while $q(L_1)/q(L_2)$ is large in codes that deal well with X errors. Hence we see a positive shift of the threshold estimate in the cuboctahedral codes (where performance against X errors is very good but performance against Z errors is fairly poor) and a negative shift in the octahedral code (which deals better with Z errors but worse with X errors relative to the cuboctahedral code).

Notice also that although we use the term "threshold" in this discussion these are not true thresholds as are they are not independent of lattice size (the shift Δp depends on L). Estimating a threshold value using a set of relatively small lattices gives values like those in Table I, but if we estimated a threshold using only very large lattices the impact of the CZ errors would be negligible and the calculated value would be almost identical to the case of purely 2D noise. If we consider

TABLE I. Error threshold estimates obtained by fitting the data in Figs. 6 and 7 to the ansatz in (11). The indicated errors were derived via bootstrap resampling.

	Octahedral	Cuboctahedral 1	Cuboctahedral 2
$\overline{Z}$ error threshold (no CCZ)	1.788(8)%	0.525(8)%	0.519(8)%
$\overline{Z}$ error threshold (noise before CCZ)	1.065(9)%	0.278(9)%	0.276(9)%
$\overline{Z}$ error threshold (noise after CCZ)	1.747(14)%	0.542(8)%	0.533(7)%
$\overline{X}$ error threshold (no CCZ)	2.693(12)%	2.864(22)%	2.879(18)%
$\overline{X}$ error threshold (noise before CCZ)	2.630(14)%	2.750(26)%	2.750(31)%
$\overline{X}$ error threshold (noise after CCZ)	2.694(12)%	2.874(21)%	2.886(19)%

the true threshold to be the point below which $p_{\text{fail}}^{L_2} < p_{\text{fail}}^{L_1}$ for all $L_2 > L_1$ then this is the value

$$p_{\text{th}}^{\text{true}} = \min_{L_2 > L_1} \left(p_{\text{th}}^{\text{IID}} + \Delta p(L_1, L_2) \right). \tag{14}$$

Our numerical results suggest that for the cuboctahedral lattices $p_{\text{th}}^{\text{true}} = p_{\text{th}}^{\text{IID}}$ [with the minimum value of $\Delta p(L_1, L_2)$ being $\lim_{L\to\infty} \Delta p(L, L+1) = 0$] while $p_{\text{th}}^{\text{true}}$ will be very close to the value shown in Table I for the octahedral lattice [with the minimum (negative) value of $\Delta p(L_1, L_2)$ being when L_1 and L_2 are the two smallest members of the code family].

B. Noise before *CCZ*

In the previous case the relative immunity of the threshold to *CZ* errors was due to the fact that X error probability in the code fell with increasing lattice size. This is no longer true if additional X errors occur after the Z stabilizer measurements but before the application of the *CCZ* gate, as these errors will not be accounted for by the X correction calculated using those measurements. More explicitly, the per-qubit X error probability at the point in time when the *CCZ* gate is applied will be

$$p_X^{\text{eff}} = p_X + p_{\text{res}} \tag{15}$$

where p_{res} is the probability of a residual X error from decoding using faulty syndrome measurements and $p_X = 2p/3$ $(= p_Z)$ is once again the probability of a specific Pauli error due to the depolarising channel. p_{res} is a function of L (and decreases with increasing L when we are below the X threshold) while p_X is independent of L and so for large L we will have $p_X^{\text{eff}} \approx p_X$, and if we are far enough below the threshold then we expect p_X to dominate even for small L. This means that if we define the same effective Z error probability p^{eff} as in the previous section the quantity $q(L)$ will also be almost constant in L for values of p_Z close to the Z threshold and the quantity $\Delta p(L_1, L_2)$ will then be an almost constant negative shift of this threshold. This is reflected in Fig. 7.

This kind of problem is not unique to non-Clifford gates (and in general we expect any nontrivial logical operation to have a negative impact on the performance of the code [37]). Consider, for example, the simple case of stabilizer measurement → correction → transversal *CNOT* in the 2D surface code. All X errors in the control code, which occurred between measurement and correction will now also be applied to the target code, while all such Z errors in the target will be applied to the control, effectively doubling the X/Z error probability in the target/control for this step of the computation. However, for the case of transversal Clifford gates we can instead consider a process of stabilizer measurement → transversal gate → correction, where the correction we apply to the code is the correction produced by the decoder commuted through the transversal gate. For transversal non-Cliffords things are not so simple because commuting the Pauli correction through the non-Clifford gives a Clifford correction. This will be a valid correction for the Clifford error, which exists in the code, but if we wish to restrict ourselves to only applying Pauli corrections (e.g., because single-qubit gates typically have better fidelities than multiqubit gates) then we must once again measure the stabilizers to project the Clifford error to a Pauli error and there is then no way to deterministically infer a suitable Pauli correction from our previously calculated Clifford correction as the projection is probabilistic.

This may seem like a serious drawback for transversal non-Clifford gates (and particularly multiqubit ones), as the decoding calculation will take a non-negligible amount of time and so it is reasonable to expect that a non-negligible number of errors will occur in the period between measurement of stabilizers and application of a correction. Fortunately, this problem may not be as significant as it seems. Notice that if we consider $p_X^{\text{eff}} \approx p_X$ then we can write p_Z^{eff} as

$$p_Z^{\text{eff}} = p_Z + \alpha p_X \tag{16}$$

for some $0 \leqslant \alpha \leqslant 1$ [in practice we expect $\alpha \approx 0.5$ since most X errors from the depolarising channel will be isolated single-qubit errors that will be mapped to Z errors with (per-code) probability 0.5]. In our simulations we have $p_X = p_Z$ and so we observe a large increase in p_Z^{eff} and a large decrease in the threshold. However, in many architectures we have $p_Z \gg p_X$ [38–42], in which case the increase in p_Z^{eff} due to errors of this type would be much less significant.

Another practical consideration is the trade-off between decoding speed and performance. Faster decoders will allow less time for errors to build up before the gate application, but it is often the case that faster decoders have weaker performance and so if error rates are low enough then using a slower but more accurate decoder may be preferable.

C. State preparation

While the state preparation step is not of particular interest by itself, the details of this step will be relevant when we discuss the generation of the Z error distributions that result from the *CZ* errors. Throughout the simulation we track the locations of both X and Z errors.

To simulate the preparation of the initial state we start with no errors on any qubit (i.e., we assume perfect single-qubit state preparation) and then apply an X error to each qubit with probability 0.5. This reproduces the effect of measuring the Z stabilizers of the code when all qubits are in $|+\rangle$. We then calculate the Z stabilizer syndrome and, for each stabilizer, flip the outcome of this calculation with probability p to simulate measurement errors. We use a minimum-weight perfect matching decoder [27] to generate a correction for these measurement errors and we use BP-OSD [43–46] to calculate a correction for the qubit errors.

In the absence of measurement errors (or if we make no mistakes in calculating a correction for these errors) the pattern of single-qubit Xs after correction will correspond to a random configuration of X stabilizers and possibly an X logical. This pattern of Xs represents one randomly selected term of the superposition

$$\begin{aligned} |\overline{+}\rangle &= \frac{1}{\sqrt{2}}(|\overline{0}\rangle + |\overline{1}\rangle) \\ &= \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{n}}\sum_i X_{\beta_i}|\mathbf{0}\rangle + \frac{1}{\sqrt{n}}\overline{X}_L \sum_i X_{\beta_i}|\mathbf{0}\rangle\right). \end{aligned} \tag{17}$$

In the presence of measurement errors (and if we make mistakes correcting them) this pattern of Xs will differ from a term of this superposition by errors on a subset of qubits [46]. Notice that an "X error" can now mean either an application of X to a qubit where we should have applied identity or the reverse, so the locations of these errors are not explicitly known or tracked by the simulation.

D. Application of *CCZ*

The pattern of X's we have prepared makes the creation of an appropriate Z error configuration extremely simple: We simply apply a Z error to any qubit of code k where we have applied X to the corresponding qubits in codes i and j (i.e., just the natural local action of CCZ). We know from (8) that this will produce linking charge strings so we only need to check that it will produce appropriate error configurations on the boundaries of error membranes and also that it will not produce errors elsewhere.

To check the former we can consider the intersection of a membrane of X errors in code i with an X stabilizer of code j [we assume that there are no X errors in code j so that the pattern of X operators for this code matches a term of (17)]. We know from the analysis in Sec. III that a Z error supported on this intersection in code k anticommutes with a pair of X stabilizers of code k that lie on the boundary of the code i error membrane. We also know that each such stabilizer should be violated with probability 0.5 while the total number of violated stabilizers should be even. Each X stabilizer of code j appears in the pattern of Xs for this code with probability 0.5, and so for each error-stabilizer intersection we create a Z error string with this same probability. The resulting pattern of all these strings will anticommute with the X stabilizers of code k exactly as described above.

To check that we do not create Z errors in other cases we only need to observe that, in the absence of errors, X will only be applied to a qubit if this forms part of a stabilizer or logical in the pattern of X's for this code. The fact that intersections of stabilizers and logicals in codes i and j correspond to the supports of Z stabilizers and logicals in code k is what makes the CCZ gate transversal so no errors can be created in this case.

E. Dimension jump

Full descriptions of dimension jumping for 3D surface codes can be found in in [47,48] but we review the collapse part here for completeness. It will be helpful to once again write our codewords in terms of stabilizers, but this time we use the Z stabilizers rather than the X, so that, for example,

$$|\overline{+}\rangle = \sum_i Z_{\beta_i}|+\rangle. \tag{18}$$

Measuring out all qubits of the code in the X basis will project to one specific term of this superposition. If we split the qubits into two sets M and N and measure out only qubits in M then we project to a subset of terms

$$|\psi\rangle = Z_{\beta_j}\sum_n Z_{\beta_n}|+\rangle \tag{19}$$

where $\beta_j \cap M \neq \emptyset$ while $\beta_n \cap M = \emptyset$. If $\sum_n Z_{\beta_n}|+\rangle^{\otimes|N|}$ is a codeword $|\overline{+}\rangle$ of a smaller code then this code will be subject to a Z error $Z_{\beta_j\cap N}$ and if we wish to transform from the larger code to the smaller one fault-tolerantly then this error must be correctable. A correction can be calculated by finding a stabilizer of the larger code that matches Z_{β_j} on the qubits in M and then applying this operator to qubits in N, but whether or not this reliably corrects the error will depend on the stabilizer structure of the code and on the choice of regions M and N. Additionally, in order to correctly transfer the logical information we require that there is at least one representation Z_L of logical Z for the larger code for which $Z_{L\cap N}$ is a valid logical Z of the smaller code. This means that $|\overline{-}\rangle = Z_L\sum_i Z_{\beta_i}|+\rangle$ in the larger code is mapped to

$$|\psi\rangle = Z_{L\cap M}Z_{\beta_j}\left(Z_{L\cap N}\sum_n Z_{\beta_n}|+\rangle^{\otimes|N|}\right)|+\rangle^{\otimes|M|} \tag{20}$$

where $Z_{L\cap N}\sum_n Z_{\beta_n}|+\rangle^{\otimes|N|}$ will then be the codeword $|\overline{-}\rangle$ in the smaller code.

An extra complication is added when the qubits of M are subject to Z errors prior to measurement. In this case the outcomes of our single-qubit X measurements will not be consistent with any Z stabilizer Z_{β_j}, and before we can calculate a correction for the qubits in N we must first find one for the qubits in M. We can do this by using the single-qubit measurement outcomes along with the measurement outcomes of the X stabilizers of the smaller code to reconstruct an X stabilizer syndrome for the larger code, and then use this syndrome to find a correction.

In our simulation the "larger code" is the full 3D surface code while the "smaller code" is a 2D surface code supported on one of the boundaries of the 3D code. The qubits in sets M and N are referred to as the "inner" and "outer" qubits respectively. Because the logical Z operators of the three different surface codes are all perpendicular there is no way to collapse all three codes to the same boundary, since for any boundary N of the rectified lattice there is always one code where $Z_{L\cap N} \neq \overline{Z}^{2D}$ for every representation Z_L. Instead we choose to collapse the two codes defined on cuboctohedral lattices to one of the half-cuboctahedral boundaries (giving two rotated 2D surface codes) while the code defined on the octahedral lattice collapses to a half-octahedral boundary (giving an unrotated 2D surface code) as shown in Fig. 8. Code deformation can be used to transform the unrotated code into a rotated code that overlaps with the other two.

In each of the three codes the error $Z_{\beta_j\cap N}$ comes from faces of the 3D lattice that meet the boundary at an edge, since these faces support Z stabilizers of the 3D code and when restricted to the 2D code these edges form strings of errors. As mentioned above, a correction can be calculated by finding a Z stabilizer of the 3D code that matches the measurement outcomes from the measured-out qubits, and because each edge of the 2D code corresponds to a single (nonboundary) face of the 3D lattice this correction will exactly match the error in the 2D code. An algorithm for calculating this correction in each of the three codes is given in Appendix B.

If there were Z errors on the measured out qubits (or if the measurements themselves were faulty) we will need to

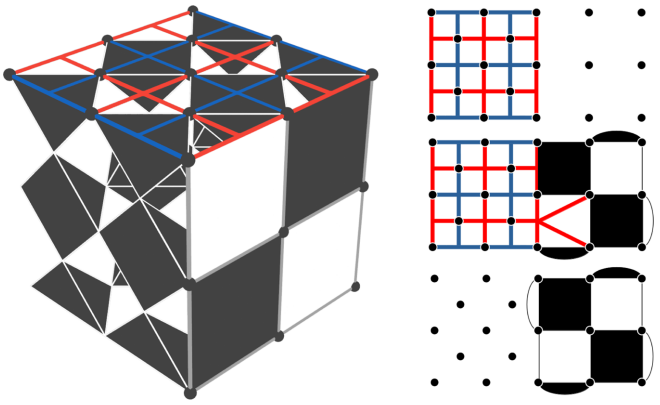


FIG. 8. Code deformation operation for turning an unrotated surface code (supported on a half-octahedral boundary of the rectified lattice) into a rotated surface code (supported on a cuboctahedral boundary of the rectified lattice).

find a correction for these by reconstructing a 3D X stabilizer syndrome. These stabilizers are supported on cells of the 3D lattice so for cells supported entirely on the inner qubits we can simply take the product of all single-qubit measurement outcomes from that cell. If a cell is partially supported on the outer qubits then its restriction to these qubits gives an X stabilizer of the 2D code, so to calculate a measurement outcome for this cell we can take the product of the measurement outcome of this 2D stabilizer and all the single-qubit measurement outcomes from the other qubits in the cell. However, obtaining a reliable set of stabilizer measurement outcomes for the 2D code requires repeated measurements of these stabilizers, and we have promised a single-shot magic state preparation process. Fortunately, there is a way to fault-tolerantly calculate a correction using only the single-qubit measurement outcomes although this comes at the cost of a reduction in code performance. This calculation is described in Appendix C.

V. DISCUSSION

We have generalised results regarding Clifford errors in the color code to the case of the 3D surface code and found that not only do these results translate straightforwardly but they are much more easily understood in this setting. In the surface code the deterministic linking charge error string and the random error strings on the Clifford error boundary have separate mathematical origins, the former being caused by commuting the transversal non-Clifford gate through a pair of Pauli errors and the latter being caused by the action of a Clifford error on the codestate. In the color code this distinction is less clear and these two kinds of Z error have the same origin (the action of the Clifford error on the codestate). The transversal T gate in the color code and transversal CCZ in the surface code are related by a local unitary mapping [49,50] and so we propose that linking charge arising from the T gate in the color code is best understood as a surface code phenomenon whose origin is obscured by the mapping to the color code. On the other hand, we showed in Appendix A that linking charge effects occurring when CS and CCZ are applied between three color codes are identical to the surface code case (i.e., they have a local origin), so we see that the nonlocal linking charge previously observed in the color code is specifically a property of the transversal T, rather than of the color code itself.

Clifford errors from the CCZ gate are more approachable not only from an analytic perspective but also from a numerical one. Full simulation of Clifford errors due to T requires a nonlocal method of detecting linked syndromes and as a result linking charge contributions have been left out of previous numerical works [17]. In contrast, our simulations reproduce the full effect of CZ errors in the 3D surface code with only a simple local check at each lattice site. The results of these simulations show that the occurrence of a large number of X errors before the CCZ but after the most recent stabilizer measurement can result in a significant reduction of the Z threshold for the code, but in practical implementations we expect that the number of such errors will be small when compared to the number of Z errors, which occur naturally during this period.

Finally, we note that in addition to the 3D surface and color codes, our proof techniques can be applied straightforwardly to any CSS code with a transversal non-Clifford that is diagonal in the Z basis. For instance, the 4D surface code admits a transversal $CCCZ$ between four copies of the code [51] and in the same way as (8) we have

$$\begin{aligned}\overline{CCCZ}X_\alpha^1X_\beta^2X_\gamma^3|\overline{\psi}\rangle &= X_\alpha^1CCZ_\alpha^{234}X_\beta^2CCZ_\beta^{134}X_\gamma^3CCZ_\gamma^{124}|\overline{\psi'}\rangle\\ &= X_\alpha^1X_\beta^2CZ_{\alpha\cap\beta}^{34}X_\gamma^3CZ_{\alpha\cap\gamma}^{24}CZ_{\beta\cap\gamma}^{14}|\psi''\rangle\\ &= X_\alpha^1X_\beta^2X_\gamma^3Z_{\alpha\cap\beta\cap\gamma}^4|\psi'''\rangle \end{aligned} \tag{21}$$

and so we observe a Z linking charge string on the intersection of these three X errors in addition to whatever effect the Clifford CZ and non-Clifford CCZ errors have on the codestate. This effect can be calculated in the same way as for the CZ errors shown previously, i.e., by writing the codestates as superpositions of stabilizers, commuting the non-Pauli errors through these stabilizers, then examining the effect of the phases produced on the possible stabilizer measurement outcomes.

ACKNOWLEDGMENTS

T.R.S. acknowledges support from University College London and the Engineering and Physical Sciences Research Council [Grant No. EP/L015242/1] and also the JST Moonshot R&D Grant [Grant No. JPMJMS2061]. Research at Perimeter Institute is supported in part by the Government of Canada through the Department of Innovation, Science and Economic Development Canada and by the Province of Ontario through the Ministry of Colleges and Universities. This research was enabled in part by support provided by Compute Ontario and Compute Canada.

The authors thank B. Brown, M. Kesselring, A. Kubica and M. Beverland for valuable discussions regarding linking charge in the color code.

This work was partially completed while T.R.S. was at University College London and some parts appeared previously in [52].

APPENDIX A: CLIFFORD ERRORS IN THE 3D COLOR CODE

In this Appendix we apply the proof techniques used in the main text to the case of the 3D color code. We recover the result of linking charge for the case of transversal T in this setting, although it requires more effort and is considerably less intuitive than the surface code case. We also examine the cases where the non-Clifford gates CS and CCZ are applied between two and three copies of the code.

1. Single error membranes in cleanable code regions

The color code has a transversal T gate corresponding to an application of T and $T^\dagger$ to white and black vertices in a bicoloring of the lattice. Consider a membrane-like error X_α (Pauli X on all qubits in set α and identity otherwise) supported on a subset of faces of color $\kappa_1\kappa_2$ and detected by Z stabilizers on faces of color $\kappa_3\kappa_4$ at the boundary of the membrane. Using that $TX = e^{-i\pi/4}SXT$ and $T^\dagger X = e^{i\pi/4}S^\dagger X T^\dagger$ we have that

$$\overline{T}X_\alpha|\overline{\psi}\rangle = e^{-i\pi N_w^\alpha/4}e^{i\pi N_b^\alpha/4}A_\alpha X_\alpha|\overline{\psi'}\rangle, \tag{A1}$$

where N_w^α and N_b^α are the numbers of white and black vertices in α and A_α is a tensor product of S on all white vertices of α and $S^\dagger$ on all black vertices of α. $|\overline{\psi}\rangle$ and $|\overline{\psi'}\rangle$ are states in the codespace. Using $SX = YS$, $S^\dagger X = -YS^\dagger$ and $Y = iXZ$ we have

$$\begin{aligned} e^{-i\pi N_w^\alpha/4}e^{i\pi N_b^\alpha/4}A_\alpha X_\alpha|\overline{\psi'}\rangle = e^{-i\pi N_w^\alpha/4}e^{i\pi N_b^\alpha/4}(-1)^{N_b^\alpha}i^{|\alpha|} \\ \times X_\alpha Z_\alpha A_\alpha|\overline{\psi'}\rangle. \end{aligned} \tag{A2}$$

α is a product of faces of the code and faces are cycles in the lattice so must contain an equal number of b and w vertices so $e^{-i\pi N_w^\alpha/4}e^{i\pi N_b^\alpha/4} = 1$. If $|\alpha| = 0 \mod 4$ then $i^{|\alpha|} = (-1)^{N_b^\alpha} = 1$ and if $|\alpha| = 2 \mod 4$ then $i^{|\alpha|} = (-1)^{N_b^\alpha} = -1$. Z_α is a Z stabilizer of the code (since it is a Z operator supported on a set of faces) and commutes with A_α as they are both diagonal in the computational basis. In summary

$$\overline{T}X_\alpha|\overline{\psi}\rangle = X_\alpha A_\alpha|\overline{\psi'}\rangle. \tag{A3}$$

We then want to know what effect A_α has on codestates. Since we are considering an error in a cleanable region of the code it is sufficient to consider only the effect on $|\overline{0}\rangle$ as the analysis for $|\overline{1}\rangle$ will be identical. We have that

$$|\overline{0}\rangle = \frac{1}{\sqrt{n}}\sum_{i=1}^{n} X_{\beta_i}|\mathbf{0}\rangle \tag{A4}$$

where X_{β_i} are stabilizers of the code (which are cells or products of cells of the lattice) and $|\mathbf{0}\rangle$ is the all-zeros state. We can then use the same commutation relations as above to show

$$\begin{aligned} A_\alpha\frac{1}{\sqrt{n}}\sum_{i=1}^{n} X_{\beta_i}|\mathbf{0}\rangle &= \frac{1}{\sqrt{n}}\sum_{i=1}^{n} A_\alpha X_{\beta_i}|\mathbf{0}\rangle \\ &= \frac{1}{\sqrt{n}}\sum_{i=1}^{n}(-1)^{N_b^{\alpha\cap\beta_i}} i^{|\alpha\cap\beta_i|}X_{\beta_i}Z_{\alpha\cap\beta_i}|\mathbf{0}\rangle. \end{aligned} \tag{A5}$$

$Z_{\alpha\cap\beta_i}$ acts trivially on the all-zeros state so we can rewrite this as

$$A_\alpha|\overline{0}\rangle = \frac{1}{\sqrt{n}}\sum_{i=1}^{n} i^{g(\alpha\cap\beta_i)}X_{\beta_i}|\mathbf{0}\rangle \tag{A6}$$

where $g(\alpha\cap\beta_i) = |\alpha\cap\beta_i| + 2N_b^{\alpha\cap\beta_i}$. We can see that

$$\begin{aligned} \left(|\alpha\cap\beta_i| + 2N_b^{\alpha\cap\beta_i}\right) \mod 4 &= \left(N_w^{\alpha\cap\beta_i} + 3N_b^{\alpha\cap\beta_i}\right) \mod 4 \\ &= \left(N_w^{\alpha\cap\beta_i} - N_b^{\alpha\cap\beta_i}\right) \mod 4 \end{aligned} \tag{A7}$$

so $g(\alpha\cap\beta_i)$ can be understood as the difference (mod 4) between the number of b and w vertices in $\alpha\cap\beta_i$. α is a set of faces of the code and the intersection of any face (or product of faces) with a cell (or product of cells) of the 3D color code is even so $i^{g(\alpha\cap\beta_i)} = \pm 1$ and the effect of A_α on $|\overline{0}\rangle$ is to flip the sign of some of the terms in this superposition as in the surface code case. We can then once again write our state as

$$A_\alpha|\overline{0}\rangle = a|\phi_i^+\rangle + b|\phi_i^-\rangle \tag{A8}$$

where $X_{\beta_i}|\phi_i^+\rangle = |\phi_i^+\rangle$ and $X_{\beta_i}|\phi_i^-\rangle = -|\phi_i^-\rangle$. The following lemma will be useful in finding values for a and b.

Lemma A.0.1. $i^{g(\alpha\cap(\beta_i+\beta_j))} \neq i^{g(\alpha\cap\beta_i)}i^{g(\alpha\cap\beta_j)}$ only if $|\alpha\cap\beta_i\cap\beta_j|$ is odd ($\beta_i+\beta_j$ is pointwise addition modulo 2)

If β_i and β_j are disjoint then $\beta_i+\beta_j = \beta_i\cup\beta_j$ and so $\alpha\cap(\beta_i\cup\beta_j) = (\alpha\cap\beta_i)\cup(\alpha\cap\beta_j) = (\alpha\cap\beta_i)+(\alpha\cap\beta_j)$. This means

$$\begin{aligned} g(\alpha\cap(\beta_i+\beta_j)) &= |\alpha\cap(\beta_i+\beta_j)| + 2N_b^{\alpha\cap(\beta_i+\beta_j)} \\ &= |(\alpha\cap\beta_i)+(\alpha\cap\beta_j)| + 2N_b^{(\alpha\cap\beta_i)+(\alpha\cap\beta_j)} \\ &= |\alpha\cap\beta_i| + |\alpha\cap\beta_j| + 2N_b^{\alpha\cap\beta_i} + 2N_b^{\alpha\cap\beta_j} \\ &= g(\alpha\cap\beta_i) + g(\alpha\cap\beta_j) \end{aligned} \tag{A9}$$

where we have used that $|a+b| = |a|+|b|$ for disjoint a and b. Therefore we only have $i^{g(\alpha\cap(\beta_i+\beta_j))} \neq i^{g(\alpha\cap\beta_i)}i^{g(\alpha\cap\beta_j)}$ if $\beta_i\cap\beta_j$ is nonempty. In this case we can write $\beta_i\cup\beta_j = \beta_i' + \beta_j' + \beta_i\cap\beta_j$ where β_i' (β_j') are the elements of β_i (β_j) not in $\beta_i\cap\beta_j$. $\beta_i+\beta_j = \beta_i'+\beta_j'$ and β_i', β_j' and $\beta_i\cap\beta_j$ are all disjoint, so by the same method as above we can show

$$\begin{aligned} g(\alpha\cap(\beta_i+\beta_j)) &= g(\alpha\cap(\beta_i'+\beta_j')) \\ &= g(\alpha\cap\beta_i') + g(\alpha\cap\beta_j') \end{aligned} \tag{A10}$$

and

$$\begin{aligned} &g(\alpha\cap\beta_i) + g(\alpha\cap\beta_j) \\ &\quad= g(\alpha\cap(\beta_i'+\beta_i\cap\beta_j)) + g(\alpha\cap(\beta_j'+\beta_i\cap\beta_j)) \\ &\quad= g(\alpha\cap\beta_i') + g(\alpha\cap\beta_j') + 2g(\alpha\cap\beta_i\cap\beta_j). \end{aligned} \tag{A11}$$

Thus for general β_i and β_j we have

$$i^{g(\alpha\cap\beta_i)}i^{g(\alpha\cap\beta_j)} = (-1)^{g(\alpha\cap\beta_i\cap\beta_j)}i^{g(\alpha\cap(\beta_i+\beta_j))} \tag{A12}$$

and so $i^{g(\alpha\cap(\beta_i+\beta_j))} \neq i^{g(\alpha\cap\beta_i)}i^{g(\alpha\cap\beta_j)}$ only if $g(\alpha\cap\beta_i\cap\beta_j)$ is odd, which means $|\alpha\cap\beta_i\cap\beta_j|$ is also odd. ■

Corollary 1: If $Z_{\alpha\cap\beta_i}$ is a stabilizer then $i^{g(\alpha\cap(\beta_i+\beta_j))} = i^{g(\alpha\cap\beta_i)}i^{g(\alpha\cap\beta_j)}\ \forall\ \beta_j$. This is because the intersection of any X

and Z stabilizer of the code must be even and so $\alpha \cap \beta_i \cap \beta_j$ must be even if $\alpha \cap \beta_i$ is the support of a Z stabilizer.

Now let us reconsider the state

$$|\phi\rangle = A_\alpha|\overline{0}\rangle = \frac{1}{\sqrt{n}}\sum_{i=1}^{n} i^{g(\alpha\cap\beta_i)}X_i|\mathbf{0}\rangle = a|\phi_i^+\rangle + b|\phi_i^-\rangle. \tag{A13}$$

Note that once again $|\phi_i^+\rangle$ must have the form

$$|\phi_i^+\rangle = \sum_j (X_{\beta_j} + X_{\beta_i}X_{\beta_j})|\mathbf{0}\rangle \tag{A14}$$

while

$$|\phi_i^-\rangle = \sum_j (X_{\beta_j} - X_{\beta_i}X_{\beta_j})|\mathbf{0}\rangle. \tag{A15}$$

The pairs in $|\phi_i^+\rangle$ are those for which $i^{g(\alpha\cap\beta_j)} = i^{g(\alpha\cap(\beta_i+\beta_j))}$ while the pairs in $|\phi_i^-\rangle$ are those where $i^{g(\alpha\cap\beta_j)} = -i^{g(\alpha\cap(\beta_i+\beta_j))}$. This means that if $i^{g(\alpha\cap\beta_i)} = 1$ we must have $i^{g(\alpha\cap(\beta_i+\beta_j))} = i^{g(\alpha\cap\beta_i)}i^{g(\alpha\cap\beta_j)}$ for terms of $|\phi_i^+\rangle$ and $i^{g(\alpha\cap(\beta_i+\beta_j))} = -i^{g(\alpha\cap\beta_i)}i^{g(\alpha\cap\beta_j)}$ for terms of $|\phi_i^-\rangle$, whereas if $i^{g(\alpha\cap\beta_i)} = -1$ then the reverse is true.

Now consider the same options for X_i as in the surface code (readers may find it helpful to return to Fig. 4(a), which shows the kind of error we are considering here):

(i) Stabilizers not supported on the membrane boundary: The fact that $i^{g(\alpha\cap\beta_i)} = 1$ in this case is part of the requirement for transversal T (see corollary 7 of [53]). Additionally, the intersection of these X stabilizers with the membrane is either nothing or the support of a Z stabilizer and so we always have $i^{g(\alpha\cap(\beta_i+\beta_j))} = i^{g(\alpha\cap\beta_i)}i^{g(\alpha\cap\beta_j)}$ for this choice of X_i by corollary 1. This means that $|\phi\rangle = |\phi_i^+\rangle$ in this case, and so we are in a $+1$ eigenstate of these stabilizers.

(ii) Stabilizer generators on the membrane boundary: Our error membrane is supported on faces of color $\kappa_1\kappa_2$ and detected by faces of color $\kappa_3\kappa_4$, which are the interfaces of κ_1 and κ_2 cells on the membrane boundary. A κ_1 cell meets the membrane at κ_2 colored edges, which each contain one w and one b vertex and so the total numbers of each in $\alpha \cap \beta_i$ are equal and $i^{g(\alpha\cap\beta_i)} = 1$ for these stabilizers. The intersection of a $\kappa_1\kappa_2$ face, a κ_1 cell and a κ_2 cell is a single vertex (since the κ_1 cell meets the face at a κ_2 edge, the κ_2 cell meets the face at a κ_1 edge and these edges meet at a vertex) and therefore $|\alpha \cap \beta_i \cap \beta_j|$ is odd for a pair of neighboring cells on the membrane boundary. Thus $|\phi_i^+\rangle$ is formed from pairs $(X_{\beta_j} + X_{\beta_i}X_{\beta_j})|\mathbf{0}\rangle$ where X_{β_j} contains either 0 or 2 such neighbors of X_{β_i} while $|\phi_i^-\rangle$ contains all pairs $(X_{\beta_j} - X_{\beta_i}X_{\beta_j})|\mathbf{0}\rangle$ where X_{β_j} contains only one of the neighbors of X_{β_i}. There are equal numbers of each type of pair so $|\phi\rangle = \frac{1}{\sqrt{2}}(|\phi_i^+\rangle + |\phi_i^-\rangle)$ and we expect a random ±1 outcome from a measurement of X_{β_i}.

(iii) All stabilizer generators of one color on the membrane boundary: For any κ_1 cell X_{β_i} and neighboring κ_2 cell X_{β_j} that are both on the membrane boundary we have that $|\alpha \cap \beta_i \cap \beta_j|$ is odd, and so $Z_{\alpha\cap\beta_i}$ is an error string that anticommutes with the two κ_2 colored neighbors of X_{β_i}. If X_{β_i} is instead the product of all κ_1 colored cells on the membrane boundary then each κ_2 cell anticommutes individually with the string from each of its two κ_1 colored neighbors and so commutes with their product. Thus $Z_{\alpha\cap\beta_i}$ is a stabilizer for this choice of X_{β_i}. Additionally, $i^{g(\alpha\cap\beta_i)} = 1$ (since $i^{g(\alpha\cap\beta_i)} = 1$ when X_{β_i} is an individual cell and cells of the same color are disjoint). Therefore we have $|\phi\rangle = |\phi_i^+\rangle$ for this X_{β_i} as well.

We have recovered the expected result for an isolated membrane: If we measure all stabilizer generators of the code then stabilizers not on the membrane boundary will return $+1$. Stabilizers on the membrane boundary return random ±1 outcomes, but the total parity of -1 stabilizers of any color will always be even.

2. Linked error membranes in cleanable code regions

Now we consider a pair of membranes of X errors with one defined on $\kappa_1\kappa_2$ faces and detected by $\kappa_3\kappa_4$ faces and one defined on $\kappa_3\kappa_4$ faces and detected by $\kappa_1\kappa_2$ faces as in Fig. 4(b). We will refer to these as $\alpha_{\kappa_1\kappa_2}$ and $\alpha_{\kappa_3\kappa_4}$ respectively. Following the application of $\overline{T}$ we have a state

$$\begin{aligned}\overline{T}X_{\alpha_{\kappa_1\kappa_2}}X_{\alpha_{\kappa_3\kappa_4}}|\overline{\psi}\rangle &= \overline{T}X_{\alpha_{\kappa_1\kappa_2}+\alpha_{\kappa_3\kappa_4}}|\overline{\psi}\rangle \\ &= X_{\alpha_{\kappa_1\kappa_2}+\alpha_{\kappa_3\kappa_4}}A_{\alpha_{\kappa_1\kappa_2}+\alpha_{\kappa_3\kappa_4}}|\overline{\psi'}\rangle.\end{aligned} \tag{A16}$$

by the same reasoning as (A3) and using the fact that the membranes are individually defined on the supports of Z stabilizers (sets of faces) and so their product is also the support of a Z stabilizer. Notice that, unlike in the surface code, we do not observe the emergence of a linking charge string at this point and in fact we observe no errors on the intersection at all since $\alpha_{\kappa_1\kappa_2} + \alpha_{\kappa_3\kappa_4} = \alpha_{\kappa_1\kappa_2} \cup \alpha_{\kappa_3\kappa_4} - \alpha_{\kappa_1\kappa_2} \cap \alpha_{\kappa_3\kappa_4}$. The linking charge string in this case will come from the action of the Clifford error on the codestate rather than directly from the commutation of the transversal non-Clifford through the original X error.

Much of the analysis from above carries over to this case, and the only difference will be for X_{β_i} partially supported on the intersection of the two membranes. Note that the same method used to prove (A12) can equivalently be used to show

$$i^{g((\alpha_{\kappa_1\kappa_2}+\alpha_{\kappa_3\kappa_4})\cap\beta_i)} = (-1)^{g(\alpha_{\kappa_1\kappa_2}\cap\alpha_{\kappa_3\kappa_4}\cap\beta_i)}i^{g(\alpha_{\kappa_1\kappa_2}\cap\beta_i)}i^{g(\alpha_{\kappa_3\kappa_4}\cap\beta_i)}. \tag{A17}$$

Then we have that

(i) X stabilizers at intersection endpoints: These stabilizers are on the boundary of one membrane and in the interior of the other and contain a single qubit in $\alpha_{\kappa_1\kappa_2} \cap \alpha_{\kappa_3\kappa_4}$. If this cell has color κ_1 then it must meet the $\kappa_1\kappa_2$ membrane at a set of κ_2 colored edges and the $\kappa_3\kappa_4$ membrane at a $\kappa_3\kappa_4$ colored face. Sets of disjoint edges and individual faces both contain equal numbers of b and w vertices so $i^{g(\alpha_{\kappa_1\kappa_2}\cap\beta_i)} = i^{g(\alpha_{\kappa_3\kappa_4}\cap\beta_i)} = 1$. $\alpha_{\kappa_1\kappa_2} \cap \alpha_{\kappa_3\kappa_4} \cap \beta_i$ is a single qubit so $i^{g(\alpha\cap\beta_i)} = -1$ by (A17).

(ii) X stabilizers on the intersection (not endpoints). $Z_{\alpha_{\kappa_1\kappa_2}\cap\beta_i}$ and $Z_{\alpha_{\kappa_3\kappa_4}\cap\beta_i}$ are both Z stabilizers so $i^{g(\alpha_{\kappa_1\kappa_2}\cap\beta_i)} = i^{g(\alpha_{\kappa_3\kappa_4}\cap\beta_i)} = 1$. The product of two Z stabilizers is another Z stabilizer, and all Z stabilizers have even weight so $|(\alpha_{\kappa_1\kappa_2} \cap \beta_i) + (\alpha_{\kappa_3\kappa_4} \cap \beta_i)|$ is even and so $|\alpha_{\kappa_1\kappa_2} \cap \alpha_{\kappa_3\kappa_4} \cap \beta_i|$ is also even and $i^{g(\alpha\cap\beta_i)} = 1$.

For nonendpoint stabilizers everything is as before. For endpoint stabilizer generators we once again have $|\phi\rangle = \frac{1}{\sqrt{2}}(|\phi_i^+\rangle + |\phi_i^-\rangle)$ but we have swapped which pairs of states are in $|\phi_i^+\rangle$ and $|\phi_i^-\rangle$ since, e.g., $|\phi_i^-\rangle$ now contains the pair $|\mathbf{0}\rangle - X_{\beta_i}|\mathbf{0}\rangle$ whereas previously we had $|\mathbf{0}\rangle + X_{\beta_i}|\mathbf{0}\rangle$ in $|\phi_i^+\rangle$. If X_{β_i} is all cells of one color on one of the membrane

boundaries and X_{β_j} is a single cell on this boundary then as before $Z_{\alpha\cap\beta_i}$ is a stabilizer and as before $i^{g(\alpha\cap\beta_i)}$ is the product of $i^{g(\alpha\cap\beta_j)}$ for all individual cells. One of these cells sits at an intersection endpoint and so has $i^{g(\alpha\cap\beta_j)} = -1$ whereas the rest have $i^{g(\alpha\cap\beta_j)} = 1$ and so $i^{g(\alpha\cap\beta_i)} = -1$. Thus we now have $|\phi\rangle = |\phi_i^-\rangle$ whereas before we had $|\phi\rangle = |\phi_i^+\rangle$. This implies that when we measure all the stabilizer generators of the code we will once again get random outcomes from the membrane boundary stabilizers, but instead of having an even parity of each color on each boundary we have an odd parity, and this is consistent with a linking charge string running between the two membrane boundaries.

3. Error membranes in noncleanable regions

Consider an X error membrane in the color code supported on a subset of faces of color $\kappa_1\kappa_2$ and also on the support of a logical Z operator. As before we have

$$\overline{T}X_\alpha|\overline{\psi}\rangle = e^{-i\pi N_w^\alpha/4}e^{-i\pi N_b^\alpha/4}i^{g(\alpha)}X_\alpha Z_\alpha A_\alpha|\overline{\psi'}\rangle. \tag{A18}$$

α is a product of the supports of Z stabilizers and a Z logical. For the former $N_w = N_b$ and for the latter $N_w = N_b + 1$ so $e^{-i\pi N_w^\alpha/4}e^{-i\pi N_b^\alpha/4} = e^{-i\pi/4}$. Also $g(\alpha) = |\alpha| + 2N_b^\alpha = N_w^\alpha + N_b^\alpha + 2N_b^\alpha = 4N_b^\alpha + 1$ so $i^{g(\alpha)} = i$. This gives

$$\overline{T}X_\alpha|\overline{\psi}\rangle = e^{i\pi/4}X_\alpha Z_\alpha A_\alpha|\overline{\psi'}\rangle. \tag{A19}$$

Notice that Z_α is a logical Z operator, whereas for cleanable α it was a stabilizer. Now we want to consider the effect of A_α on codestates. For $|\overline{0}\rangle$ the analysis is the same as before, but for $|\overline{1}\rangle$ we must now consider the interaction of logical X operators with A_α. $|\overline{1}\rangle$ can be written

$$|\overline{1}\rangle = \frac{1}{\sqrt{n}}\sum_{i=1}^{n}\overline{X}X_{\beta_i}|\mathbf{0}\rangle \tag{A20}$$

where $\overline{X}$ is a logical X operator. It does not matter which implementation of $\overline{X}$ we choose, so we choose it to be X on all qubits. We then have that

$$\begin{aligned} A_\alpha|\overline{1}\rangle &= \frac{1}{\sqrt{n}}\sum_{i=1}^{n} i^{g(\alpha)}\overline{X}Z_\alpha i^{g(\alpha\cap\beta_i)}X_{\beta_i}Z_{\alpha\cap\beta_i}|\mathbf{0}\rangle \\ &= \frac{i}{\sqrt{n}}\sum_{i=1}^{n} i^{g(\alpha\cap\beta_i)}\overline{X}X_{\beta_i}|\mathbf{0}\rangle \end{aligned} \tag{A21}$$

where we have used that Z_α is a logical Z operator and so commutes with X_{β_i}, which is a stabilizer. Thus we see that the action of A_α on $|\overline{1}\rangle$ is the same as in the cleanable case except for a global factor of i. This is consistent with a logical S error and so we conclude that in addition to creating distributions of Z errors A_α also applies a logical S to the codestate. Notice that in addition to this, commuting $\overline{T}$ through X_α created a logical Z error Z_α in (A19).

4. *CCZ* and *CS* between multiple color codes

We can also apply CCZ between three copies of the 3D color code and CS between two copies (this follows from the fact that $CNOT$ and T are transversal in this code and CCZ/CS can be synthesised exactly using these gates [3,54]). For CCZ the effect of the resulting CZ errors on terms in the color code codestates will be identical to (4) as this equation is not code-specific (provided the code is CSS and so has codestates, which can be written in this form). Then to find $|\phi_q^+\rangle$ and $|\phi_q^-\rangle$ we can note that the only code properties we assumed in this calculation for the surface code were (a) that an X error membrane is detected by Z stabilizers at its boundary and (b) that the intersection of X stabilizers of codes i and j is a Z stabilizer of code k. Both of these properties are true for the color code, so the result is exactly the same as for the surface code case.

The case of CS (which can be applied transversally using the same bicoloring as the T gate) between two color codes is more interesting. CS has the commutation relations

$$\begin{aligned} (CS)(X\otimes I) &= (X\otimes S)(CZ)(CS), \\ (CS^\dagger)(X\otimes I) &= (X\otimes S^\dagger)(CS). \end{aligned} \tag{A22}$$

If we then consider applying CS to a pair of color codes, one of which contains an X error X_α^1, then we have

$$\overline{CS}X_\alpha^1|\overline{\psi}\rangle = X_\alpha^1 A_\alpha^2 CZ_\alpha^{12}|\overline{\psi'}\rangle \tag{A23}$$

and so we obtain both an S error and a CZ error. We have already examined both of these errors individually so we can use those results to see that the effect on the terms of the codestate will be

$$A_\alpha^2 CZ_\alpha^{12}|\overline{00}\rangle = \frac{1}{\sqrt{n}}\sum_{ij} i^{g(\alpha\cap\beta_j)}(-1)^{|\alpha\cap\beta_i\cap\beta_j|}X_{\beta_i}^1 X_{\beta_j}^2|\mathbf{0}\rangle. \tag{A24}$$

From our previous analysis we know that $i^{g(\alpha\cap\beta_j)} = -1$ when β_j contains the support of two neighboring cells on the membrane boundary while $(-1)^{|\alpha\cap\beta_i\cap\beta_j|} = -1$ when β_i contains the support of a cell on the membrane boundary and β_j contains the support of a single neighbor of that cell (also on the membrane boundary). For a given X stabilizer generator in code 1 only the latter contribution is relevant (because the qubits of this code are acted on only by CZ and not by S) so the argument is identical to the case of errors due to CCZ. For a given X stabilizer generator in code 2 the former contribution depends only on β_j while the latter depends only on β_i and so the two are independent.

Thus we conclude that application of CS to a pair of codes containing error X_α^1 results in random Z error distributions on the boundary of α in both codes 1 and 2, with the former being due just to a CZ error and the latter being a product of errors produced by CZ and S (but still being identical to a distribution that could be produced by either of these errors individually, i.e., a -1 outcome from any given stabilizer with probability $p = 0.5$ but an even number of -1s overall).

We can also examine linking charge for the case of CS. The calculation is simply

$$\begin{aligned} \overline{CS}X_\alpha^1 X_\gamma^2|\overline{\psi}\rangle &= X_\alpha^1 A_\alpha^2 CZ_\alpha^{12}X_\gamma^2 A_\gamma^1 CZ_\gamma^{12}|\overline{\psi'}\rangle \\ &= X_\alpha^1 X_\gamma^2\big((-1)^{N_b^\gamma} i^{|\alpha\cap\gamma|}Z_{\alpha\cap\gamma}^2\big)\big(Z_{\alpha\cap\gamma}^1\big)|\psi''\rangle \end{aligned} \tag{A25}$$

where the first bracketed linking charge term (in code 2) comes from the S part of the error and the second one (in code 1) comes from the CZ part. The phases incurred are global and so not a problem. $|\psi''\rangle$ is the state $|\overline{\psi'}\rangle$ multiplied by the CZ and A Clifford errors.

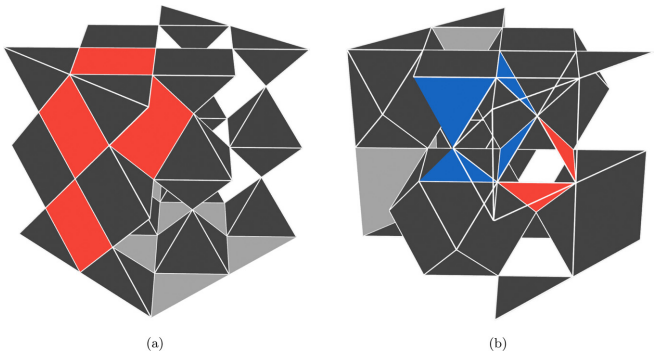


FIG. 9. (a) Z stabilizer structure in the octahedral lattice. (b) Z stabilizer structure in the cuboctahedral lattice

APPENDIX B: ALGORITHMS FOR CALCULATING DIMENSION JUMP CORRECTIONS

In this Appendix we describe the algorithms used in our simulation for calculating the corrections applied to the 2D code after the dimension jump. At this point in the procedure we have measured out all inner qubits of the 3D code in X and calculated a correction for these qubits. We therefore have a (corrected) pattern of ± 1 outcomes from these measurements, which should match some Z stabilizer of the 3D code. The aim of the algorithms in this Appendix is to use these outcomes to find a correction for any error, which may have arisen in the 2D code due to these projective measurements.

We first discuss the simpler case of the octahedral surface code. An example is shown in Fig. 9(a). In this code the Z stabilizer generators are weight-4 (some shown in red) and we wish to collapse to a half-octahedral boundary (shown in grey). To calculate a correction we start at the top boundary (opposite to the boundary we wish to collapse to) and examine the single-qubit measurement outcomes from qubits on this boundary. From Fig. 9(a) we can see that the Z stabilizer generators that touch this boundary are either supported entirely on the boundary or only on a single qubit of it. We assume that any -1 outcomes on this boundary are due to this second type of stabilizer, so that even if we obtain four -1 outcomes in the support of a single Z stabilizer generator we assume that this is due instead to the product of four separate generators. In this way we obtain one generator for each -1 outcome on this boundary. We then flip the recorded signs of the measurement outcomes from qubits within the support of the product of all these generators. This guarantees that the top boundary now contains only $+1$s, and also that the plane of qubits immediately below this will also contain only $+1$s since (up to composition with generators supported only on the top boundary) the pattern of -1s on the top boundary uniquely specifies a pattern of generators that matches the -1s on this plane. The only mistake we can have made is to apply four generators partially supported on the boundary instead of a single generator fully supported on the boundary, as mentioned above. The effect of this is to remove these four -1s from the top plane of qubits but also to flip the four corresponding outcomes in the plane two layers below. This plane is identical to the top boundary and so we can repeat the process, terminating when we arrive at the bottom boundary. The final result will be the removal of all Z errors in the 2D code, which arise due to 3D code stabilizers partially supported on the outer qubits. Any extra Zs applied due to "mistakes" of the kind described above will actually be Z stabilizers of the 2D code.

The cuboctahedral lattices are more complicated. We see one such lattice and some example Z stabilizers in Fig. 9(b), where one cell has been removed so that we can see the interior. The boundary we wish to collapse to is once again shown in grey, and this time it is a half-cuboctahedral boundary. Once again we start at the boundary opposite the one we wish to collapse to, which this time is the front-right boundary. There are no Z stabilizer generators fully supported on this boundary, and partially supported generators always meet it at a single qubit (examples shown in red). These generators come in pairs, so unlike in the octahedral lattice we cannot uniquely specify a stabilizer generator based on a -1 outcome of a qubit on this boundary. Instead we choose one of these two generators at random. Flipping the signs of the associated measurement outcomes as before means this boundary now contains only $+1$s and although we may have mistakenly chosen the "wrong" generator this will turn out not to matter.

In the next plane the Z stabilizer generators are supported on two qubits of the plane rather than one. Notice also that if we consider only the generators that "lead towards" the boundary we wish to collapse to (shown in blue) and ignore the ones that "lead away" (red) then they come in sets of four. Each blue generator shares qubits (in this plane) only with other blue generators in this set of four and so each set can be considered individually. The component of the correction based on this layer is calculated by proceeding clockwise around each set of four qubits. If the measurement outcome from a given qubit is recorded as -1 then we check the outcomes of the clockwise and anticlockwise neighbors of this qubit. If exactly one of these two outcomes is -1 then we identify the generator supported on this pair of qubits and flip all recorded signs for measurement outcomes in its support. If both or neither of the neighbors of this qubit have outcome -1 then we randomly choose one of the two generators with this qubit in its support and flip the associated measurement outcomes. Examples are shown in Fig. 10. Once this process has been completed for all quadruples of qubits in the plane all recorded outcomes from qubits within this plane will be $+1$. The plane below this matches the front boundary plane and so the process starts again and repeats until we reach the final boundary where the 2D code is supported.

Now we can consider the effects of making the wrong selection during the random choices of generator involved in this algorithm. Firstly, consider the case were we choose the wrong generator while finding a correction for the front boundary (or any other matching plane). These are the red faces in Fig. 9(b), so we can see that the product of the two generators that share a qubit of this boundary is a four-qubit operator supported on two different qubit quadruples on the next plane. This will lead us to flip the outcomes from qubits associated with two generators (one supported on each quadruple) that meet at a single qubit in the subsequent plane. This means that the outcome from this qubit will be flipped twice, and so not flipped at all and thus the original

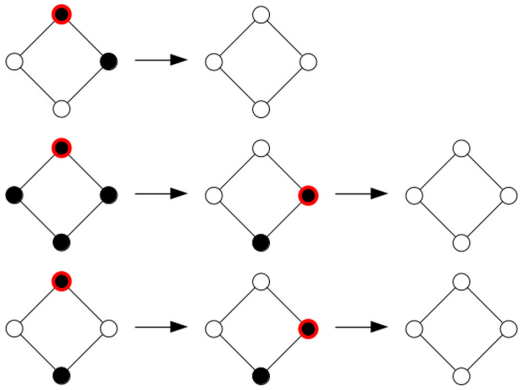


FIG. 10. Examples of corrections calculated for −1 measurement outcomes from quadruples of qubits in the cuboctahedral lattice. +1 (−1) outcomes are shown by empty (filled) circles. At each step we examine the neighbors of the qubit marked with a red circle and modify the recorded measurement outcomes according to the algorithm described in the text.

"incorrect" choice of generator causes no problems. For the random choices involved in finding corrections from the quadruples, the different choices of generator lead to corrections for the final 2D code that differ only by stabilizers of the code, and so once again cause no issues.

APPENDIX C: CONSTANT-TIME DECODING AT THE 2D/3D CODE BOUNDARY

To calculate this correction we assume that all 2D stabilizer measurement outcomes will return a value of +1. This means that the measurement outcome we calculate for cells that are partially supported on the outer qubits is just the product of all the outcomes of the measured-out qubits of this cell. To understand why this works we can consider the example in Fig. 11. In Fig. 11(a) we have a two-qubit Z error on the marked qubits, for which the correct 3D syndrome would be a −1 outcome from the cells labeled A and C. We imagine that the bottom boundary of the figure is the 2D code we wish to collapse to, so a syndrome calculation that uses only the inner (nonboundary) qubits will give −1 outcomes for cells A and B, for which a valid correction would be the qubit in the intersection of these two cells. In other words, calculating stabilizer outcomes using only inner qubit measurements will only give corrections for inner qubit errors. This may seem like a rather trivial statement, but the fact that it can be made consistent with the rest of the dimension jump such that the entire procedure remains fault-tolerant actually relies on a rather specific feature of the codes we have chosen. To see this, we can consider Fig. 11(b) where the 4-qubit Z error is actually a Z stabilizer of the 3D code. In this case, calculating a 3D X syndrome using only the inner qubits will give a −1 outcome from cells B and C even though in reality we should have no syndrome at all. It is important that we do not try to correct this "error" because it will be

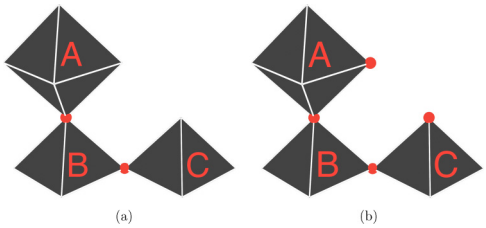


FIG. 11. (a) A two-qubit Z error, which anticommutes with the X stabilizers on cells labeled A and C. (b) A four-qubit Z stabilizer.

dealt with separately in the next step of the dimension jump when we find a 3D Z stabilizer that matches the single-qubit measurement outcomes and apply its restriction to the 2D code (as described in the previous Appendix). For example, a decoder given this syndrome would return a correction supported on the outer qubit that is in the intersection of cells B and C, and this same correction will be returned in the next step of the jump when we identify the other three single-qubit Zs as the restriction of this Z stabilizer to the inner qubits. As a result, the error in the outer code will be corrected twice and so will not be corrected at all.

The way around this is to recall that we began by assuming that all 2D stabilizer measurements would return an outcome of +1, meaning that there are no Z errors on the outer qubits. For the code and boundary depicted above all 3D Z stabilizers are either also 2D Z stabilizers or intersect the 2D code at a single qubit. As a result, a MWPM decoder given a syndrome calculated from only the inner qubits will always return a correction supported on the outer code if the "error" that caused the syndrome was actually a 3D Z stabilizer [since, e.g., for the case of Fig. 11(b) a correction supported on the outer qubits is weight-1 while a correction supported on the inner qubits is weight-3]. We can therefore add an extra rule to our 3D X decoder, which says that after calculating a correction we only apply the parts of the correction that are supported on the inner qubits of the 3D code, and this ensures that "errors" due to 3D Z stabilizers are left untouched. Notice that for the other two 3D surface codes (where the Z stabilizers are supported on the triangular faces of cells A, B, and C) the 3D Z stabilizers at the boundary shown in Fig. 11 are supported on two outer qubits and one inner qubit, so this modification of the decoder would not work if we tried to collapse these codes to this boundary. Fortunately however, we do not try to do this. Instead we collapse them to the half-cuboctahedral boundaries where the Z stabilizers are supported on one outer qubit and two inner qubits.

This decoding strategy is not without cost. We have created a large number of new −1 X stabilizer outcomes at one boundary of the 3D code, and while these can be corrected properly in isolation 3D Z errors, which occur close to this boundary can interfere with this and the performance of the MWPM decoder will suffer as a result. Despite this, we still observe thresholds for the dimension jump in all three codes.

[1] P. W. Shor, Scheme for reducing decoherence in quantum computer memory, Phys. Rev. A **52**, R2493 (1995).

[2] A. Ekert and C. Macchiavello, Quantum Error Correction for Communication, Phys. Rev. Lett. **77**, 2585 (1996).

[3] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, New York, 2011).

[4] D. Gottesman, The Heisenberg representation of quantum computers, arXiv:quant-ph/9807006.

[5] C. Wang, J. Harrington, and J. Preskill, Confinement-Higgs transition in a disordered gauge theory and the accuracy threshold for quantum memory, Ann. Phys. (NY) **303**, 31 (2003).

[6] D. K. Tuckett, Tailoring surface codes: Improvements in quantum error correction with biased noise, Ph.D. thesis, University of Sydney, 2020 (qecsim: https://github.com/qecsim/qecsim).

[7] C. Gidney, Stim: A fast stabilizer circuit simulator, Quantum **5**, 497 (2021).

[8] J. Wallman, C. Granade, R. Harper, and S. T. Flammia, Estimating the coherence of noise, New J. Phys. **17**, 113020 (2015).

[9] R. Kueng, D. M. Long, A. C. Doherty, and S. T. Flammia, Comparing Experiments to the Fault-Tolerance Threshold, Phys. Rev. Lett. **117**, 170502 (2016).

[10] D. Greenbaum and Z. Dutton, Modeling coherent errors in quantum error correction, Quantum Sci. Technol. **3**, 015007 (2017).

[11] S. Bravyi, M. Englbrecht, R. König, and N. Peard, Correcting coherent errors with surface codes, npj Quantum Inf. **4**, 55 (2018).

[12] S. J. Beale, J. J. Wallman, M. Gutiérrez, K. R. Brown, and R. Laflamme, Quantum Error Correction Decoheres Noise, Phys. Rev. Lett. **121**, 190501 (2018).

[13] J. K. Iverson and J. Preskill, Coherence in logical quantum channels, New J. Phys. **22**, 073066 (2020).

[14] D. Gottesman and I. L. Chuang, Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations, Nature (London) **402**, 390 (1999).

[15] B. Yoshida, Topological color code and symmetry-protected topological phases, Phys. Rev. B **91**, 245131 (2015).

[16] H. Bombín, Transversal gates and error propagation in 3D topological codes, arXiv:1810.09575.

[17] M. E. Beverland, A. Kubica, and K. M. Svore, Cost of universality: A comparative study of the overhead of state distillation and code switching with color codes, PRX Quantum **2**, 020341 (2021).

[18] C. Chamberland, P. Iyer, and D. Poulin, Fault-tolerant quantum computing in the Pauli or Clifford frame with slow error diagnostics, Quantum **2**, 43 (2018).

[19] C. Chamberland, J. J. Wallman, S. Beale, and R. Laflamme, Hard decoding algorithm for optimizing thresholds under general Markovian noise, Phys. Rev. A **95**, 042332 (2017).

[20] G. Zhu, T. Jochym-O'Connor, and A. Dua, Topological Order, Quantum Codes, and Quantum Computation on Fractal Geometries, PRX Quantum **3**, 030338 (2022).

[21] H. Bombín and M. A. Martin-Delgado, Topological Computation without Braiding, Phys. Rev. Lett. **98**, 160502 (2007).

[22] H. Bombín and M. A. Martin-Delgado, Topological Quantum Distillation, Phys. Rev. Lett. **97**, 180501 (2006).

[23] A. Kubica and M. E. Beverland, Universal transversal gates with color codes—A simplified approach, Phys. Rev. A **91**, 032330 (2015).

[24] M. Vasmer and D. E. Browne, Three-dimensional surface codes: Transversal gates and fault-tolerant architectures, Phys. Rev. A **100**, 012312 (2019).

[25] H. Bombín, Dimensional jump in quantum error correction, New J. Phys. **18**, 043038 (2016).

[26] A. Yu. Kitaev, Fault-tolerant quantum computation by anyons, Ann. Phys. (NY) **303**, 2 (2003).

[27] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Topological quantum memory, J. Math. Phys. **43**, 4452 (2002).

[28] C. Horsman, A. G. Fowler, S. Devitt, and R. Van Meter, Surface code quantum computing by lattice surgery, New J. Phys. **14**, 123011 (2012).

[29] A. Kubica, B. Yoshida, and F. Pastawski, Unfolding the color code, New J. Phys. **17**, 083026 (2015).

[30] H. Bombín, Gauge color codes: Optimal transversal gates and gauge fixing in topological stabilizer codes, New J. Phys. **17**, 083002 (2015).

[31] This membrane of qubits also supports a logical X operator. We can see that this configuration of S and $S^\dagger$ should implement transversal S because it will be created by applying transversal T to a code in the logical $|1\rangle$ state, and in the logical space we should have $\overline{TXT}^\dagger = e^{-i\pi/4}\overline{SX}$.

[32] B. Brown and M. Kesselring (unpublished).

[33] S. Bravyi and B. Terhal, A no-go theorem for a two-dimensional self-correcting quantum memory based on stabilizer codes, New J. Phys. **11**, 043029 (2009).

[34] https://github.com/tRowans/clifford-errors.

[35] A. Agresti and B. A. Coull, Approximate is better than "exact" for interval estimation of binomial proportions, Am. Stat. **52**, 119 (1998).

[36] A. DasGupta, T. Tony Cai, and L. D. Brown, Interval estimation for a binomial proportion, Stat. Sci. **16**, 101 (2001).

[37] H. Bombin, C. Dawson, R. V. Mishmash, N. Nickerson, F. Pastawski, and S. Roberts, Logical blocks for fault-tolerant topological quantum computation, arXiv:2112.12160.

[38] P. Aliferis, F. Brito, D. P. DiVincenzo, J. Preskill, M. Steffen, and B. M. Terhal, Fault-tolerant computing with biased-noise superconducting qubits: A case study, New J. Phys. **11**, 013061 (2009).

[39] M. D. Shulman, O. E. Dial, S. P. Harvey, H. Bluhm, V. Umansky, and A. Yacoby, Demonstration of entanglement of electrostatically coupled singlet-triplet qubits, Science **336**, 202 (2012).

[40] D. Nigg, M. Müller, E. A. Martinez, P. Schindler, M. Hennrich, T. Monz, M. A. Martin-Delgado, and R. Blatt, Quantum computations on a topologically encoded qubit, Science **345**, 302 (2014).

[41] R. Lescanne, M. Villiers, T. Peronnin, A. Sarlette, M. Delbecq, B. Huard, T. Kontos, M. Mirrahimi, and Z. Leghtas, Exponential suppression of bit-flips in a qubit encoded in an oscillator, Nat. Phys. **16**, 509 (2020).

[42] A. Grimm, N. E. Frattini, S. Puri, S. O. Mundhada, S. Touzard, M. Mirrahimi, S. M. Girvin, S. Shankar, and M. H. Devoret, Stabilization and operation of a Kerr-cat qubit, Nature (London) **584**, 205 (2020).

[43] P. Panteleev and G. Kalachev, Degenerate quantum LDPC codes with good finite length performance, Quantum **5**, 585 (2021).

[44] J. Roffe, D. R. White, S. Burton, and E. Campbell, Decoding across the quantum low-density parity-check code landscape, Phys. Rev. Res. **2**, 043423 (2020).

[45] A. O. Quintavalle, M. Vasmer, J. Roffe, and E. T. Campbell, Single-shot error correction of three-dimensional homological product codes, PRX Quantum **2**, 020340 (2021).

[46] Of course, it technically differs from all terms of the superposition by errors on a subset of qubits, but one (or perhaps a small number) of the terms will be the closest in terms of Hamming distance.

[47] B. J. Brown, A fault-tolerant non-Clifford gate for the surface code in two dimensions, Sci. Adv. **6**, aay4929 (2020).

[48] T. R. Scruby, D. E. Browne, P. Webster, and M. Vasmer, Numerical implementation of just-in-time decoding in novel lattice slices through the three-dimensional surface code, Quantum **6**, 721 (2021).

[49] A. M. Kubica, The ABCs of the color code: A study of topological quantum codes as toy models for fault-tolerant quantum computation and quantum phases of matter, Ph.D. thesis, California Institute of Technology, 2018, http://dx.doi.org/10.7907/059V-MG69.

[50] M. Vasmer and A. Kubica, Morphing quantum codes, PRX Quantum **3**, 030319 (2021).

[51] T. Jochym-O'Connor and T. J. Yoder, Four-dimensional toric code with non-Clifford transversal gates, Phys. Rev. Res. **3**, 013118 (2021).

[52] T. R. Scruby, Logical gates by code deformation in topological quantum codes, Ph.d. thesis, University College London, 2021, https://discovery.ucl.ac.uk/id/eprint/10135040.

[53] N. Rengaswamy, R. Calderbank, M. Newman, and H. D. Pfister, On optimality of CSS codes for transversal T, IEEE J. Sel. Areas Inf. Theory **1**, 499 (2020).

[54] M. Beverland, E. Campbell, M. Howard, and V. Kliuchnikov, Lower bounds on the non-Clifford resources for quantum computations, Quantum Sci. Technol. **5**, 035009 (2020).