Tailoring Three-Dimensional Topological Codes for Biased Noise

Eric Huang,[1,2,†] Arthur Pesah,[3,†] Christopher T. Chubb,[4,5] Michael Vasmer,[1,6] and Arpit Dua[7,*]

[1] *Perimeter Institute for Theoretical Physics, Waterloo, Ontario N2L 2Y5, Canada*
[2] *University of Maryland, College Park, Maryland 20742, USA*
[3] *Department of Physics and Astronomy, University College London, London WC1E 6BT, United Kingdom*
[4] *Institut quantique and Département de physique, Université de Sherbrooke, Sherbrooke QC J1K 2R1, Canada*
[5] *Institute for Theoretical Physics, ETH Zürich, Zürich 8093, Switzerland*
[6] *Institute for Quantum Computing, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada*
[7] *Department of Physics and Institute for Quantum Information and Matter, California Institute of Technology, Pasadena, California 91125, USA*

(Received 6 December 2022; revised 8 August 2023; accepted 15 August 2023; published 20 September 2023)

Tailored topological stabilizer codes in two dimensions have been shown to exhibit high-storage-threshold error rates and improved subthreshold performance under biased Pauli noise. Three-dimensional (3D) topological codes can allow for several advantages including a transversal implementation of non-Clifford logical gates, single-shot decoding strategies, and parallelized decoding in the case of fracton codes, as well as construction of fractal-lattice codes. Motivated by this, we tailor 3D topological codes for enhanced storage performance under biased Pauli noise. We present Clifford deformations of various 3D topological codes, such that they exhibit a threshold error rate of 50% under infinitely biased Pauli noise. Our examples include the 3D surface code on the cubic lattice, the 3D surface code on a checkerboard lattice that lends itself to a subsystem code with a single-shot decoder, and the 3D color code, as well as fracton models such as the X-cube model, the Sierpiński model, and the Haah code. We use the belief propagation with ordered statistics decoder (BP OSD) to study threshold error rates at finite bias. We also present a rotated layout for the 3D surface code, which uses roughly half the number of physical qubits for the same code distance under appropriate boundary conditions. Imposing coprime periodic dimensions on this rotated layout leads to logical operators of weight $O(n)$ at infinite bias and a corresponding $\exp[-O(n)]$ subthreshold scaling of the logical failure rate, where n is the number of physical qubits in the code. Even though this scaling is unstable due to the existence of logical representations with $O(1)$ low-rate and $O(n^{2/3})$ high-rate Pauli errors, the number of such representations scales only polynomially for the Clifford-deformed code, leading to an enhanced effective distance.

DOI: 10.1103/PRXQuantum.4.030338

I. INTRODUCTION

Fault-tolerant quantum computation is a crucial ingredient for building a scalable quantum computer. Topological stabilizer codes are a highly prized family of low-density parity-check (LDPC) codes due to their geometrically local parity checks, high-storage-threshold error rates and low-overhead fault-tolerant logical gate implementations. It has been found that topological codes can be tailored to noise to achieve higher success rates and threshold error rates [1,2]. Generally, for evaluating the performance of a Pauli-stabilizer code, a Pauli-noise model is considered due to its efficient simulability. It has recently been suggested that Pauli noise can be biased toward dephasing in certain realistic laboratory qubits or can be engineered to be so [3–6]. For biased Pauli noise, Clifford-deformed surface codes such as the XZZX, XY, and $(XYZ)^2$ surface codes, the XYZ color code, and families of randomly Clifford-deformed surface codes have been shown to exhibit high threshold error rates and enhanced subthreshold performance [2,7–11].

Three-dimensional (3D) topological codes offer several advantages over all the known two-dimensional (2D) topological codes. Unlike 2D stabilizer codes, 3D stabilizer codes such as the 3D surface code allow for a transversal implementation of a non-Clifford gate and, overall, a

*Corresponding author. adua@caltech.edu
†These authors contributed equally to this work.

fault-tolerant universal gate set [12–16]. Using 3D codes for storing logical information can also be advantageous in terms of decoding. For instance, the looplike syndromes associated with 3D stabilizer codes such as the surface code and color code can be decoded using single-shot decoding strategies [17–22]. Furthermore, 3D-subsystem codes such as the gauge color code [13,23,24] and the 3D-subsystem surface code [25] allow for a single-shot decoding strategy for general Pauli noise, where the noisy error syndrome need only be measured once [26]. Such single-shot decoding strategies not only reduce the time overhead but also are more resilient to time-correlated noise [27]. 3D fracton topological codes such as the X-cube model (partially) allow for parallelized decoding in submanifolds of the lattice, due to the mobility of syndromes being restricted to these submanifolds [28]. Another recently discovered advantage of 3D codes such as the surface code is that they can be used to construct fractal-lattice codes by punching holes with appropriate boundary conditions [29,30]. Such fractal-lattice surface codes allow for fault-tolerant universal quantum computation with a reduced space overhead and a single-shot decoding strategy that can be used for looplike syndromes on the fractal geometry [30]. Even though designing a qubit architecture with a 3D connectivity is a serious experimental challenge on many quantum computing platforms, several recent advances have made the prospect of a 3D architecture more amenable to near-term experiments. For instance, qubit shuttling has recently been shown to enable 3D connectivity on a 2D layout [31] and could, for instance, be implemented using silicon qubits [32], ion traps [33], or neutral atoms [34]. Other platforms, such as 3D integrated superconducting qubits [35–37] and photonic qubits [38–42] could also allow the realization of 3D codes.

Motivated by the advantages of 3D topological codes, we tailor them for enhanced storage performance in the presence of biased Pauli noise. We construct Clifford-deformed codes from the 3D surface code (cubic lattice) [43], the 3D color code [12,44], the X-cube fracton model [45], the Sierpiński fracton model [46], and the Haah code [47]. We also propose a Clifford deformation of the 3D surface code on the checkerboard lattice, which lends itself to a subsystem code with a single-shot decoding strategy. All of our Clifford-deformed codes allow for decoding strategies with a threshold error rate of 50% at infinite dephasing bias. These Clifford-deformed codes are constructed to have materialized linear symmetries that allow for the first step of decoding to be a minimum-weight perfect-matching (MWPM) decoder [43,48] in submanifolds supporting those symmetries. The resulting models after this first step have further linear symmetries that allow for another round of the matching decoder. The combination of these steps gives the full matching decoder with a threshold error rate of 50%.

We note that our Clifford deformations are single-qubit operators that permute the Pauli operators acting on each qubit. Thus, any result on the computational power of gate implementation for a given code, specifically transversal gates, does not change. Moreover, any other topological properties of our codes such as the number of logical qubits on closed manifolds and shapes or weights of general logical operators do not change. What changes is the number of representations of logical operators containing certain kinds of Pauli operators, such as pure-Z logical operators, and this change is what is relevant for the code performance under biased noise.

For a subset of the codes mentioned above, we use the belief propagation with ordered statistics decoder [49,50] (BP OSD) to numerically find the threshold error rates at finite bias. We discover that the threshold error rate increases with the bias for both original and Clifford-deformed codes. Beyond a critical value of bias, the threshold error rate of the Clifford-deformed codes exceeds that of the original codes. However, we also find some limitations of the BP OSD: for the X-cube model and the 3D surface code on the checkerboard lattice, the apparent threshold error rate tends to recede when increasing the system size, showing either a lower infinite-size threshold error rate or no threshold at all. This effect, previously observed on 2D surface codes and color codes [22], is analyzed in more detail in the context of 3D topological codes. We also compare the BP OSD with the sweep-matching decoder—a combination of MWPM and the sweep decoder [19,20]—for the 3D surface code (cubic lattice) and find better threshold error rates with the BP OSD. Even though Pauli noise is not completely realistic, it is efficiently simulable and is an initial testing ground for any quantum stabilizer code. Moreover, it has been shown that under coherent errors, decoding for Pauli errors is sufficient to give a nonzero threshold rotation [51].

Lastly, we define a rotated layout for the 3D Calderbank-Shor-Steane (CSS) surface code, which offers the same distances (d_X, d_Z) for both Pauli-X and Pauli-Z logical operators as in the standard layout, while using roughly half the number of physical qubits, in analogy with the 2D case [52]. Using this rotated layout for the Clifford-deformed surface code and imposing coprime dimensions with appropriate boundary conditions leads to weight $O(n)$ logical operators at infinite dephasing bias. As a result, the subthreshold performance of the Clifford-deformed surface code is enhanced, with a logical error rate scaling as $e^{-O(n)}$ with the number of qubits n, in comparison to $e^{-O(n^{2/3})}$ for the original code.

The paper is structured as follows. In Sec. II, we review biased noise models and Clifford-deformed codes. We discuss the notions of materialized symmetries in the context of the 2D XZZX and XY surface codes, showing that both codes have a 50% threshold error rate at infinite bias. For

the XY surface code, we introduce a technique called the weight-reduction technique, which we also use for some of the 3D topological codes studied in this paper. In Sec. III, we present CSS and Clifford-deformed 3D topological codes, including the surface code on a cubic lattice, the surface code on a checkerboard lattice, the color code, and fracton models (the X-cube model, the Sierpiński fracton model, and the Haah code). We describe the symmetries and prove the 50% threshold error rates at infinite bias for each of our Clifford-deformed codes. In Sec. IV, we present numerical threshold-error-rate estimates for some CSS and Clifford-deformed 3D codes under finite bias, using the BP OSD and sweep-matching decoders. We also demonstrate some of the limitations of the BP OSD in the context of 3D codes. In Sec. V, we explore further optimizations including a rotated layout that reduces the number of physical qubits n and coprime lattice sizes that achieve logical error rates scaling as $e^{-O(n)}$ in the subthreshold regime. Finally, in Sec. VI, we discuss the implications of our findings.

In Appendix A, we prove that the Clifford-deformed surface code on a checkerboard lattice has a 50% infinite-bias threshold error rate. In Appendix B, we discuss the decoders used in the numerical simulations for the threshold error rates at finite bias. In Appendix C, we discuss the numerical methods used to estimate the threshold error rates at finite bias.

II. BACKGROUND

A. Biased Pauli noise

In order to evaluate the performance of a code, it is conventional to choose a noise model where each qubit is independently subjected to quantum noise. Upon the Pauli-stabilizer measurements, such quantum noise is digitized to Pauli errors up to coherent rotation [53–57]. In general, it is hard to incorporate the coherent rotation. However, the Pauli-twirling approximation has been found to yield estimates of threshold error rates that are close to those found by including an efficiently tractable choice of coherent rotation [51]. For our purposes, we consider a general single-qubit Pauli-noise channel of the form

$$\rho \to (1-p)\rho + p\,(r_X X\rho X + r_Y Y\rho Y + r_Z Z\rho Z), \quad (1)$$

where $p \in [0,1]$ is the single-qubit *physical error rate* and the weights r_X, r_Y, and r_Z describe the relative probability of Pauli errors X, Y, and Z, respectively, such that $r_X, r_Y, r_Z \geq 0$ and $r_X + r_Y + r_Z = 1$. For instance, a depolarizing channel, with $r_X = r_Y = r_Z = 1/3$, causes Pauli errors X, Y, and Z with equal probabilities and thus describes *unbiased Pauli noise*. On the other hand, a dephasing channel, with $r_X = r_Y = 0$ and $r_Z = 1$, results in Z errors alone and describes *infinitely biased Pauli noise*. The *bias* η_Z for dephasing is formally defined [52] as the ratio

$$\eta_Z := \frac{r_Z}{r_X + r_Y}, \quad (2)$$

such that (unbiased) depolarizing noise corresponds to $\eta_Z = 0.5$ while infinitely biased noise corresponds to $\eta_Z = \infty$. In this work, we restrict our attention to noise channels with symmetric dephasing bias such that $r_X = r_Y$ and, hence, the noise is characterized by physical error rate p and dephasing bias η_Z.

B. Clifford-deformed codes

We consider a Pauli-stabilizer code $\mathcal{C}$ with stabilizer group generated by stabilizers $\{S_i\}$ acting on the physical qubits Q. We can construct a new code $\widetilde{\mathcal{C}}$ by applying a Clifford circuit U_C consisting of single-qubit Clifford operations on the physical qubits Q. We refer to this new code $\widetilde{\mathcal{C}}$ as a Clifford-deformed code. Under such a Clifford circuit, the generators S_i are modified to

$$\widetilde{S}_i = U_C^\dagger S_i U_C, \quad (3)$$

which also form a set of commuting Pauli operators. For biased Pauli-noise models, it can be advantageous to use a Clifford-deformed code. Intuitively, if more stabilizer generators anticommute with the more common errors, the resulting increase in nontrivial syndrome bits provides more information to the decoder to better estimate the correction operators. Clifford deformations also have the effect of drastically reducing the number of Z-only, or mostly Z, logical operators. As a result, this reduces the degeneracy of possible errors causing a given syndrome, which can be exploited by decoders for superior performance.

Pioneering studies on Clifford-deformed codes for biased noise have considered the XY surface code [2] and the XZZX surface code [7] in two spatial dimensions. The former is obtained from the CSS surface code in 2D by replacing all Pauli-Zs by Pauli-Ys in the stabilizer generators, while the latter is obtained (from the CSS surface code) by applying a Hadamard operation on half of the qubits such that all stabilizer generators become $X \otimes Z \otimes Z \otimes X$ (see Fig. 1). Both of these Clifford-deformed codes have threshold error rates that track the hashing bounds at finite dephasing bias and have a threshold error rate of 50% at infinite dephasing bias. However, unlike the XY code, the XZZX-code threshold error rates track the hashing bound for noise biased toward Pauli errors X and Y as well. Recently, the performance of random Clifford deformations of the 2D surface code subjected to noise biased toward dephasing has also been investigated [8]. A phase of 50% infinite-bias threshold error rates has been found in the parameter space of (Π_{XZ}, Π_{YZ}), where $\Pi_{XZ}(\Pi_{YZ})$ is the probability of a Clifford operation

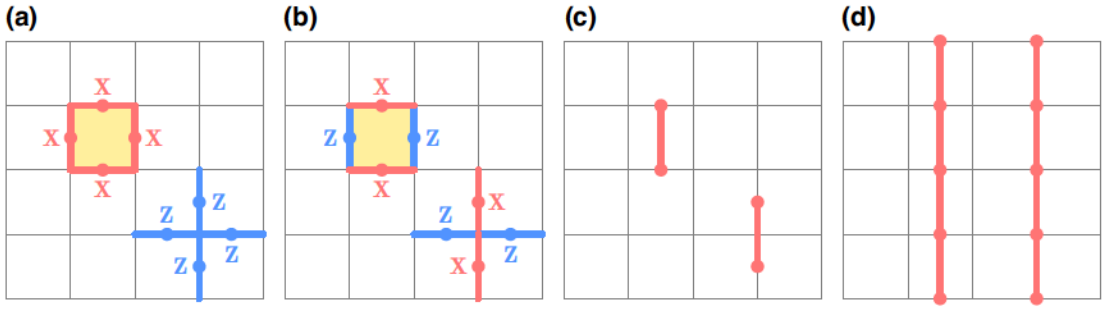

FIG. 1. An illustration of the XZZX surface code. (a) The original surface-code stabilizers, on a square lattice with periodic boundary conditions. Qubits are on edges and stabilizers are on faces and vertices. (b) XZZX surface-code stabilizers, obtained by applying a Hadamard operation on all the vertical qubits. (c) At infinite Z bias, we can ignore the Z part of the stabilizers to form a classical code, the parity-check operators of which (red edges) are supported on two qubits (red dots). (d) The product of weight-2 parity checks along a vertical line is equal to the identity operator. This means that each vertical line contains an even number of excitations, which can be independently decoded by matching. We call this type of relation a materialized linear symmetry.

that implements the permutation $X \leftrightarrow Z$ ($Y \leftrightarrow Z$). This phase can be explained intuitively via a mapping to percolation problems. Moreover, certain randomly Clifford-deformed surface codes on odd $\times$ odd dimensions, have been found to outperform the XZZX and XY codes with the same dimensions in the scaling of the subthreshold logical failure rate.

A suitable choice of Clifford deformation can result in a higher threshold error rate and can also improve subthreshold failure rates for finite system sizes. But subthreshold failure rates for finite system sizes are also sensitive to the dimensions and boundary conditions (note the mention of odd $\times$ odd dimensions in context of subthreshold failure rates in the previous paragraph). For instance, choosing coprime periodic dimensions for the XZZX code results in a subthreshold failure rate scaling of $e^{-O(n)}$ in comparison to that of $e^{-O(\sqrt{n})}$ for the CSS surface code.

For translation-invariant deformations such as the XZZX and XY surface codes, the 50% threshold error rates at infinite bias can be understood in terms of the symmetries of the stabilizer group that appear in this noise regime. We review this for the XZZX and XY codes below.

C. XZZX code: Materialized symmetries and conserved quantities under biased noise

The XZZX surface code can be obtained by applying a Hadamard operator on every vertical qubit of the CSS surface code. The resulting stabilizers are shown in Fig. 1(b). At infinite Z bias, the action of noise on the qubits where a stabilizer acts as Z results in no syndrome. Hence, the stabilizer effectively acts as identity on these qubits for infinite-bias noise. As a result, the code becomes equivalent to a classical code made of two-body parity-check operators B_f and A_v, as illustrated in Fig. 1(c).

This resulting effective model has the following relations on each vertical line ℓ of the lattice:

$$\prod_{f\in\ell} B_f = I, \tag{4a}$$

$$\prod_{v\in\ell} A_v = I. \tag{4b}$$

These relations, represented in Fig. 1(d), are referred to as *materialized subsystem symmetries* [28,48]. Because of these symmetries, the syndrome values b_f and a_v of all the stabilizers under infinite-bias noise obey

$$\prod_{f\in\ell} b_f = 1, \tag{5a}$$

$$\prod_{v\in\ell} a_v = 1, \tag{5b}$$

which are the conservation laws associated with the materialized subsystem symmetries. This implies that at infinite Z bias, the number of excitations of the XZZX stabilizer generators along any vertical line is even. In other words, single-stabilizer-generator syndromes can only "move" along vertical lines at infinite bias under application of Z errors. This leads to a simple decoding strategy for an XZZX code subjected to Z errors: we match the syndromes along each line independently. This is equivalent to decoding $2L$ independent classical repetition codes, each of size L. Since the repetition code has a threshold error rate of 50%, the success rate of this infinite-bias decoder is lower bounded by

$$p_{\text{success}} \geq \left(1 - Ae^{-\alpha L}\right)^{2L} \approx 1 - 2LAe^{-\alpha L} \xrightarrow[L\to\infty]{} 1 \tag{6}$$

for any fixed physical error rate below 50%, where A and α are two positive constants. The reason why this

expression is a lower bound and not an equality is that failing to decode an even number of repetition codes also results in successful decoding. Thus this strategy results in a threshold error rate of 50% [7].

D. XY code: Weight-reduction technique

The XY surface code [2] is another example of a Clifford-deformed surface code that has a threshold error rate of 50% at infinite bias. It is formed by applying an S gate and a Hadamard gate on every qubit, having the effect of turning all Z stabilizers into Y stabilizers, as shown in Fig. 2(a). As a result, at infinite Z bias, the code becomes a classical code where the parity checks are four-body terms supported on every square.

To prove that this code has a 50% infinite-bias threshold error rate, we need to generalize the technique developed for the XZZX surface code. As can be seen in Fig. 2(b), the checks form linear symmetries along both the vertical and horizontal directions. However, in contrast to the XZZX code, these symmetries involve weight-4 checks, making the underlying decomposition into repetition codes less obvious to see.

To decompose the code into repetition codes, we use a two-step decoding strategy that we call the *weight-reduction technique*, which we use extensively in our study of 3D codes.

In the first step, we start by writing each square check as the parity of its two incident horizontal edge checks, where an edge check is defined as the parity of its two incident vertices. In Fig. 2(b), these horizontal edge

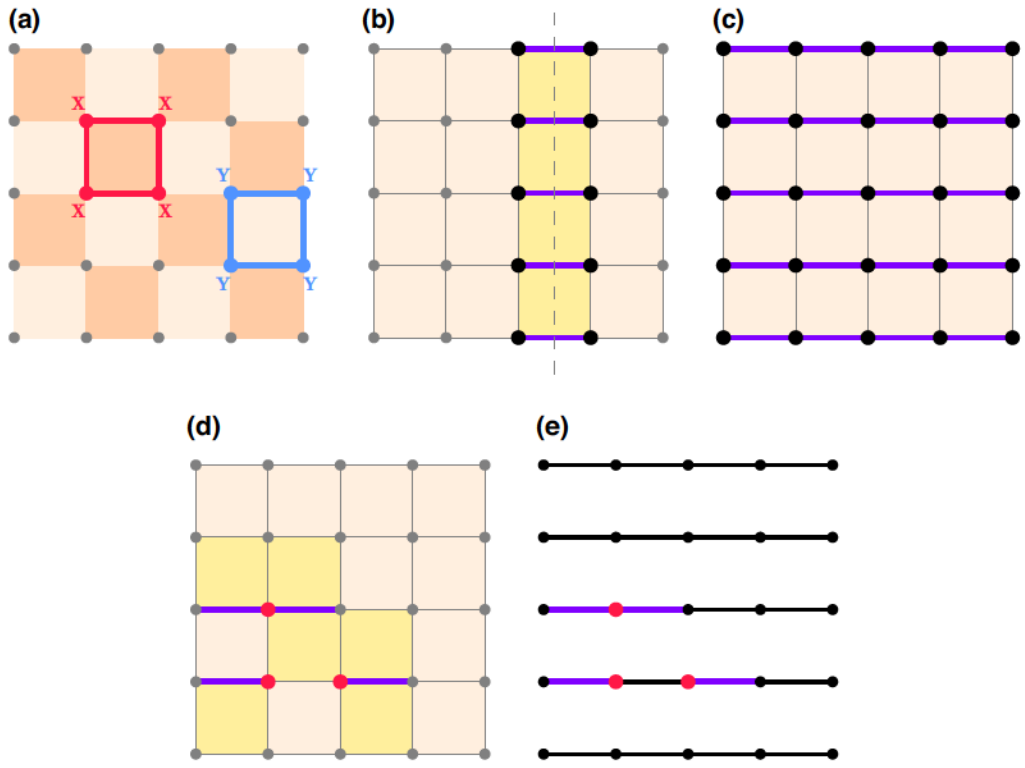

FIG. 2. An illustration of the weight-reduction technique on the XY code. (a) The XY-code lattice. Qubits are on the vertices. The dark (light) plaquettes are stabilizer generators made of X (Y) operators. This code can be obtained by applying a phase gate and a Hadamard gate to every qubit of the CSS surface code. (b) Linear symmetry of the code at infinite Z bias, assuming periodic boundary conditions. In this regime, the XY code becomes a classical code with the same four-body checks on every square. We can interpret each column as a repetition code (see the dashed line for an example), where the purple edges are the variables and the yellow squares are the checks. Indeed, the product of the yellow checks along the dashed line is effectively identity for pure-Z noise. Decoding this repetition code gives us the parity of the two qubits on each purple edge. (c) The second level of repetition codes. Once matching has been performed on all the columns, we obtain new two-body checks on all the horizontal edges (purple). They themselves form linear symmetries along each row. Matching along these symmetries allows one to decode errors on all the qubits. (d) An example of decoding using the weight-reduction strategy. The errors correspond to red vertices and the face excitations to yellow squares. In the first decoding step, we use MWPM between squares along each vertical line. The resulting "corrections" are the purple edges. This means that one of the two qubits of each purple edge is predicted to have an error. (e) In the second decoding step, the purple edges are reinterpreted as excitations of some horizontal repetition codes. The use of matching on each repetition code gives the desired correction.

checks, surrounding the top and bottom parts of each square, are represented in purple. Note that this is a purely formal manipulation, as the edge checks are not part of the syndrome at the moment.

We now use the linear symmetries consisting of the product of square checks along any vertical line. An example of such a symmetry is highlighted in Fig. 2(b). These linear symmetries give rise to repetition codes, where the data bits are the horizontal edge checks and the parity checks are the squares acting on two neighboring edges. Successful decoding of these repetition codes allows us to obtain the value of all the horizontal edge checks, which therefore becomes part of the syndrome. In other words, assuming a successful matching, we turn the weight-4 checks into weight-2 checks supported on the horizontal edges of the code. This is the core of the weight-reduction technique.

The second decoding step starts by noting that the new horizontal edge checks form a linear symmetry on each horizontal line: the product of edges along any horizontal line is equal to the identity. Each horizontal line can therefore be interpreted as a repetition code, and decoding all the repetition codes allows us to correct errors on all the qubits. Those linear symmetries are illustrated in Fig. 2(c). An example of decoding with this two-step strategy is shown in Figs. 2(d) and 2(e).

We now show that this decoding strategy leads to a threshold error rate of 50%. For a given physical error rate below 50%, the probability that this decoder succeeds is lower bounded by the probability that both the L one-dimensional (1D) matchings of the first step and the L 1D matchings of the second step are successful. More precisely, for a fixed physical error rate below 50%, we have

$$p_{\text{success}} \geq \left(1 - Ae^{-\alpha L}\right)^{2L} \xrightarrow[L\to\infty]{} 1, \tag{7}$$

where A and α are positive constants. This shows that the threshold error rate of the XY code under infinite Z bias is 50%.

III. 3D CLIFFORD-DEFORMED TOPOLOGICAL CODES

In this section, we present Clifford deformations of 3D codes with a macroscopic number of materialized symmetries at infinite bias. Having a macroscopic number of such symmetries leads to a macroscopic number of conservation laws obeyed by the syndromes and this can lead to a decoder with a high threshold error rate.

Our general strategy for showing that a Clifford-deformed code has a threshold error rate of 50% at infinite Z bias is the following. We start by identifying the linear symmetries of the code when the Z part of the stabilizers is ignored. Each linear symmetry gives rise to a repetition-code decoding problem. We can therefore construct a decoder that starts with a round of MWPM decoding [43,48] on the 1D submanifolds supporting the symmetries. This results in a new model with parity-check operators of reduced weight, as demonstrated for the XY code in Sec. II D. We then identify the linear symmetries of this new model with reduced-weight stabilizers, decode the corresponding repetition codes using MWPM, and repeat the process until a correction operator has been assigned to all the qubits. Due to the fact that each step of this decoding strategy consists of decoding repetition codes, the existence of such a decoder for any Clifford-deformed code shows that the overall threshold error rate of the code is 50%.

We now present our Clifford-deformed codes and their explicit decoders one by one below.

A. 3D surface code

The conventional form of the 3D surface code is defined on a cubic lattice with qubits sitting on edges. The stabilizer generators consist of the vertex operators $A_v = \prod_{e\in v} Z_e$, made of Pauli-Z operators on each of the six edges e adjacent to a vertex v and the face operators $B_f = \prod_{e\in f} X_e$, made of Pauli-X operators on each of the four edges adjacent to a face f, as illustrated in Fig. 3(a). The syndromes associated with violations of vertex-stabilizer generators are pointlike and created in pairs at the boundaries of strings of X errors. The syndromes associated with violations of face stabilizers are looplike syndromes and created at the boundaries of membranes of Z errors, as shown in Fig. 4(a). The logical X operators are topologically nontrivial string operators and the logical Z operators are topologically nontrivial membranes. In particular, on a 3D torus, there are three pairs of inequivalent minimum-weight logical operators $\overline{X}_u, \overline{Z}_u$, one for each axis $u \in \{\hat{x}, \hat{y}, \hat{z}\}$, where $\overline{X}_u$ is a string of X operators oriented along u and $\overline{Z}_u$ is a membrane orthogonal to u. Examples are shown in Fig. 4(b). Thus, the 3D surface code encodes three logical qubits and has code distance $\min(L_x, L_y, L_z)$, specified by the minimum weight of the string operators. One can define a open-boundary version of the code on a cubic lattice with rough boundaries on a pair of opposite faces and smooth boundaries on remaining four sides. The logical string operator connects the rough boundaries, while the logical membrane operator connects the smooth boundaries. Hence, the code encodes one logical qubit.

We consider a Clifford deformation of the 3D surface code where we apply a Hadamard operator on all the qubits on edges oriented along the z direction, which we call *vertical qubits*. On the contrary, *horizontal qubits*, which reside on edges oriented along the x or y directions, remain untouched by the Clifford deformation. The resulting stabilizer generators are shown in Fig. 3(b).

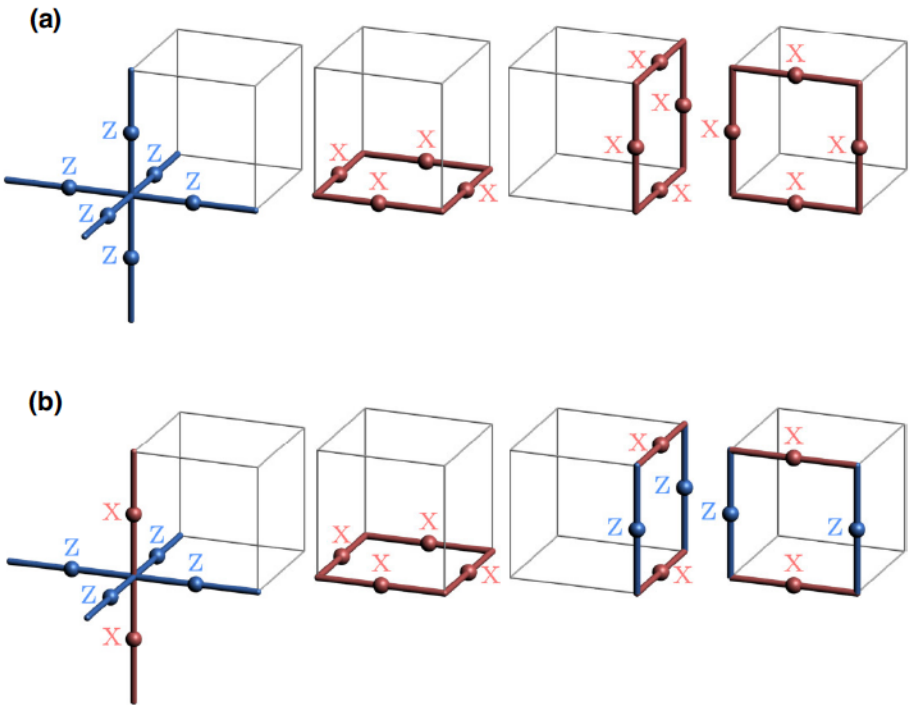

FIG. 3. (a) Stabilizer generators in the CSS 3D surface code are vertex operators on each vertex and face operators for each face, for which there are three orientations. (b) Stabilizer generators in the Clifford-deformed surface code obtained via Hadamard operations on qubits along vertical edges.

The code has certain linear materialized symmetries for infinitely biased noise. Following the techniques that have utilized materialized symmetries at biased noise to define decoding strategies for 2D topological codes [7,28,48,52], we prove the following theorem.

Theorem 1.—The Clifford-deformed 3D surface code has a threshold error rate of 50% under pure-Z noise.

Proof.—This code has linear materialized symmetries as shown in Fig. 5. The first set of symmetry operators consists of products of vertex stabilizers along a 1D closed cycle in the z direction. These products consist solely of Pauli-Z operators and hence effectively act as identity at infinite Z bias. The other symmetry consists of products of XZZX face stabilizers in the x-z and y-z planes along vertical lines. Due to the conservation laws obeyed by the syndrome along these symmetry lines, we can independently match excitations along these lines as explained below.

At infinite bias, we have only Pauli-Z errors, which anticommute with only Pauli-X operators in the stabilizer generators. Hence one can ignore the Z terms and consider only anticommutation between the Z errors and the X-stabilizer generators. This is equivalent to considering a classical parity-check code where Z errors are detected by a parity-check matrix that denotes the location of X terms in the stabilizer generators. Thus the Clifford-deformed

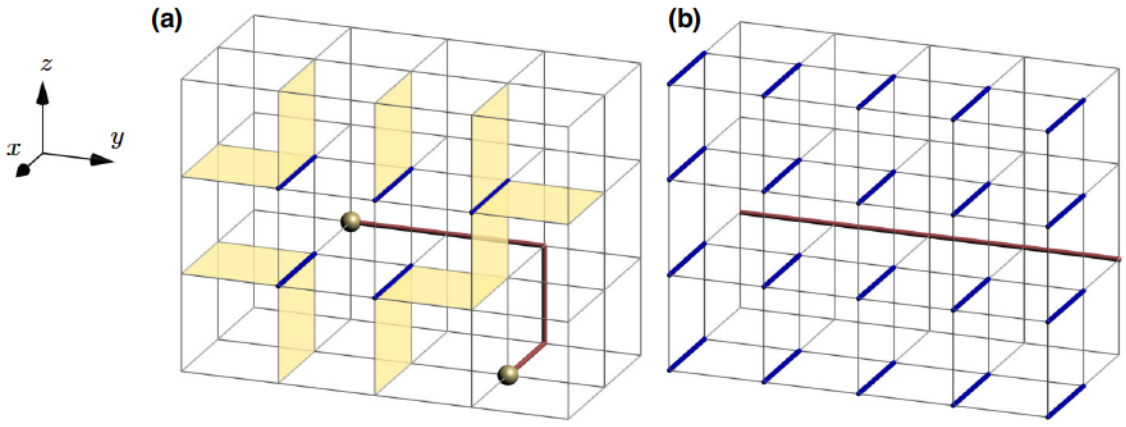

FIG. 4. The errors and logical operators of the CSS 3D surface code. (a) Errors in the 3D surface code create two types of syndromes: pointlike syndromes at the boundary of chains of X errors and looplike syndromes around membranes of Z errors. (b) Examples of logical X and Z operators of the 3D surface code.

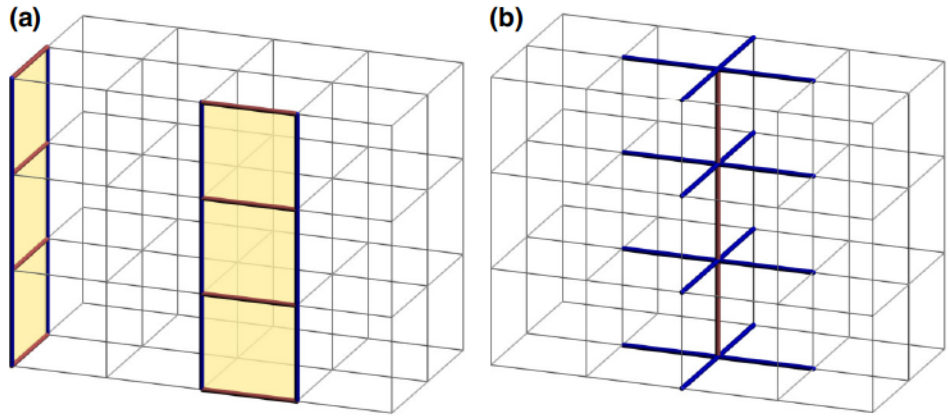

FIG. 5. The materialized linear symmetries of the Clifford-deformed 3D surface code. Products of plaquette or vertex operators along vertical lines are made only of Z operators and are therefore effectively equivalent to the identity in the purely Z infinite-bias noise regime. (a) The product of vertical plaquette operators (yellow) along two vertical lines. (b) The product of vertex operators along a vertical line.

3D surface code becomes a classical code, with weight-2 checks on vertices, x-z faces, and y-z faces. These checks form the linear symmetries discussed earlier and illustrated in Fig. 5. Errors on the qubits oriented in the z direction can be decoded by performing matching on the vertices along the corresponding symmetry lines. An example of decoding of the errors that create syndromes of vertex operators is illustrated in Fig. 6(a). Errors on qubits oriented in the x and y directions can be decoded by performing matching on the x-z and y-z faces along each vertical line, respectively. An example of decoding of face syndromes is illustrated in Fig. 6(b). If all these $3L^2$ matchings succeed, by correctly identifying the position of the errors we have succeeded in decoding all the qubits. Therefore, the probability of success is lower bounded by the probability of succeeding in all these $3L^2$ matchings. Hence, for a fixed physical error rate below 50%, we have

$$p_{\text{success}} \geq \left(1 - Ae^{-\alpha L}\right)^{3L^2} \xrightarrow[L\to\infty]{} 1 \tag{8}$$

where α and A are positive constants. Therefore, the code has a threshold error rate of 50% at infinite Z bias. ■

While the decoder described in the proof is sufficient to obtain a 50% threshold error rate, note that it can be further improved by taking into account the x-y face syndrome information as well. Let us consider a failed matching during the face decoding step. By definition, it results in a line of errors on the horizontal qubits along the vertical direction, as illustrated in Fig. 6(c). This means that all the x-y face syndromes are identical on every x-y plane. Since we have a 2D materialized symmetry on x-y planes, we can

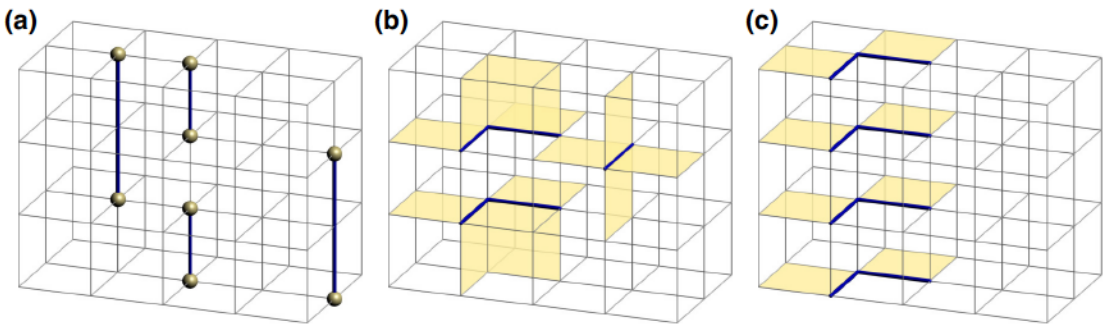

FIG. 6. The decoder used to prove that the Clifford-deformed 3D surface code has a threshold error rate of 50% at infinite Z bias. (a) The decoding of syndromes on vertices. Under local Pauli errors, vertex syndromes (yellow spheres) can only move in the vertical direction due to the linear symmetries of Fig. 5(b). We can therefore decode errors on vertical qubits by matching vertex syndromes along this axis. (b) The decoding of syndromes on faces. Under local Pauli errors, x-z and y-z face syndromes (shown as yellow squares) can only move in the vertical direction due to the linear symmetries of Fig. 5(a). We can therefore decode errors on the horizontal qubits by matching the face syndromes along the vertical axis. (c) The decoding of residual face syndromes. The decoder can be further improved by decoding residual errors coming from failed matchings in (b). A failed matching results in an identical x-y face syndrome on every x-y plane. These can be decoded using a 2D MWPM algorithm.

decode them using a 2D MWPM algorithm to return to the code space.

We note that since the 3D toric code can be written as a hypergraph product of three repetition codes [21], our above-mentioned Clifford-deformed variant can be obtained by applying a Hadamard gate to all the qubits of one of the constituent repetition codes. Hence, it is an instance of the general Clifford deformation of hypergraph product codes outlined in Ref. [58].

1. 3D surface code on the checkerboard lattice

Topological codes can be defined on various lattices or triangulations of a manifold for the same topological order. The 3D surface code on a checkerboard lattice is shown in Fig. 7(b). The code is defined using Z-cube stabilizers on one sublattice and X-triangle stabilizers on the other sublattice.

Such a variant of the 3D surface code has been used in the construction of 3D surface codes with a transversal control-control-Z CCZ gate [16]. Moreover, these checkerboard-lattice codes are instrumental in the construction of the CCZ gate for the 2D surface code [59] and the 3D-subsystem surface code [25]. This surface-code variant is defined on a cubic lattice of even dimensions with qubits sitting on edges. The cubic cells of the checkerboard lattice come in two colors. Half of these cells (i.e., of one color) have a 12-body Z-cube stabilizer supported on them, i.e., $A_c = \prod_{e\in c} Z_e$ is product of Z operators over the 12 edges of the cube. The other half of the cubic cells each have eight triangle-shaped stabilizer operators, associated with the eight vertices of the cell. A triangle stabilizer on a vertex v of a cubic cell c is defined as the product of three X operators adjacent to v and contained in c, $B_{c,v} = \prod_{e\in c\cap v} X_e$. The stabilizer generators are illustrated in Fig. 7. Since the topological order is independent of the lattice details, the syndromes of this code also come in pointlike and looplike flavors. The syndromes of the cube

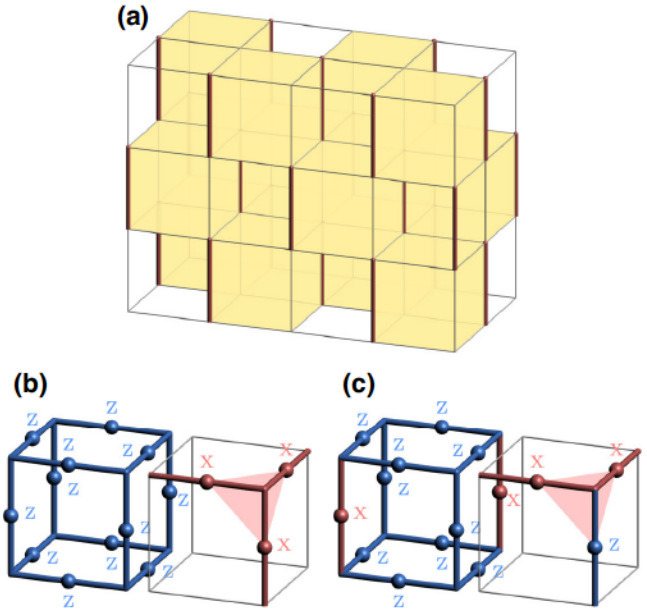

FIG. 7. The 3D surface code on the checkerboard lattice. (a) The checkerboard lattice. The yellow cubes represent the cube stabilizers and the empty cubes represent where triangle stabilizers would go. They have been omitted for clarity but note that there are eight of them in each empty cube. (b) The original CSS stabilizers. (c) The stabilizers of the Clifford-deformed code. The red edges highlighted in (a) are the edges where the $X \leftrightarrow Z$ Clifford deformation is applied.

stabilizers are pointlike and created at the ends of a string of Pauli-X errors. The syndromes associated with the triangle stabilizers form a loop around membranes of Z errors, as shown in Fig. 8(a).

The checkerboard-lattice surface code also encodes three logical qubits on an *even* $\times$ *even* $\times$ *even* torus with the logical operator pairs consisting of nontrivial X strings and Z membranes along and orthogonal to three lattice directions, respectively [see Fig. 8(b)]. We consider

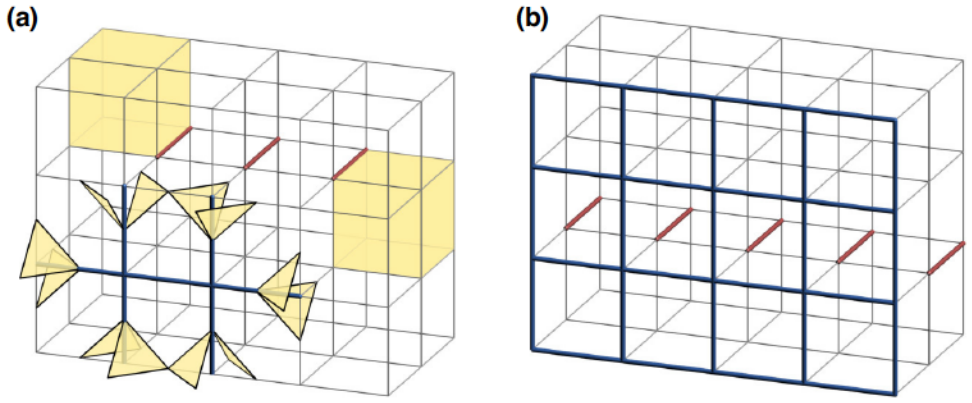

FIG. 8. The errors and logical operators of the surface code on a checkerboard lattice. (a) The errors in the surface code on a checkerboard lattice with nontrivial syndromes (yellow). (b) Examples of X and Z logical operators of the surface code on a checkerboard lattice.

a Clifford deformation of the checkerboard-lattice surface code that consists of applying a Hadamard operation on half of the vertical qubits, in a 3D-checkerboard manner [see Figs. 7 and 7(c)]. This Clifford-deformed checkerboard-lattice surface code has a 50% threshold error rate under pure-Z noise. The proof is presented in Appendix A.

B. 3D color code

The 3D color code can be defined on any 4-valent 3D lattice the cells of which are 4-colorable, i.e., one should be able to assign one of four colors to each of the cells such that any two cells sharing a face have different colors. Here, we study the 3D color code defined on the truncated octahedral lattice with periodic boundary conditions, as shown in Fig. 9(a). In this lattice, each cell is a truncated octahedron, made of 24 vertices, six square faces, and eight hexagonal faces. X-stabilizer generators are defined on every cell, and Z stabilizers on every face, as shown in Fig. 9(b). Coloring the cells using yellow, red, blue, and green, we can describe the lattice as the interlacing of a red-yellow and a blue-green sublattice, where cells of each sublattice are connected via square faces. Cells belonging to different sublattices are connected via hexagonal faces. Here, we use the convention of describing faces by the two colors of their adjacent cells. For instance, a face at the intersection of a yellow cell and a red cell is called a yellow-red face.

There exists a mapping between color codes and surface codes in any spatial dimension. A string of X errors also produces a pair of pointlike syndromes on 3-cells and a membrane of Z errors also produces a looplike syndromes on 2-cells (faces). If we impose periodic boundary conditions on the lattice defined above, the code encodes nine logical qubits, with three X-string logical operators on each direction and three Z-membrane logical operators on each plane. The similarities between the two codes can be understood using a folding procedure, which maps three copies of the 3D surface code to the 3D color code [15,60]. However, the 3D color code has some unique properties, such its transversal T gate and its flexible subsystem variant, making it a competitive candidate for a practical 3D code.

To tailor the 3D color code to biased noise, we select all yellow-red squares normal to the x direction and apply a Hadamard to diagonally opposite qubits of each square, as illustrated in Fig. 9(a). We now show that the resulting code has a 50% threshold error rate at infinite Z bias.

Theorem 2.—The Clifford-deformed 3D color code has a threshold error rate of 50% under pure-Z noise

Proof.—We start by decoding the syndromes on the 3-cells, effectively supported on four qubits at infinite Z bias [the purple qubits in Fig. 9(a)]. For this, we exploit two materialized linear symmetries, represented in Figs. 10(a) and 10(b), along the x and z directions, respectively. By the weight-reduction technique, matching cell syndromes along these two directions results in new weight-2 checks, which form linear symmetries along the y axis, as shown in Fig. 10(c). Matching along these resulting linear symmetries completes the decoding of the syndromes on the 3-cells. Since all steps consist of decoding repetition codes, we have a 50% threshold error rate on the cell sector.

The decoding of the face syndromes is done in three steps, each of which decodes errors on a different subset of the qubits. In the first step, illustrated in Fig. 11, we note the existence of a linear symmetry along the y direction. The main unit of this symmetry is a four-body check constructed by taking the product of four adjacent hexagons in a red or yellow cell, as shown in Fig. 11(a). Matching

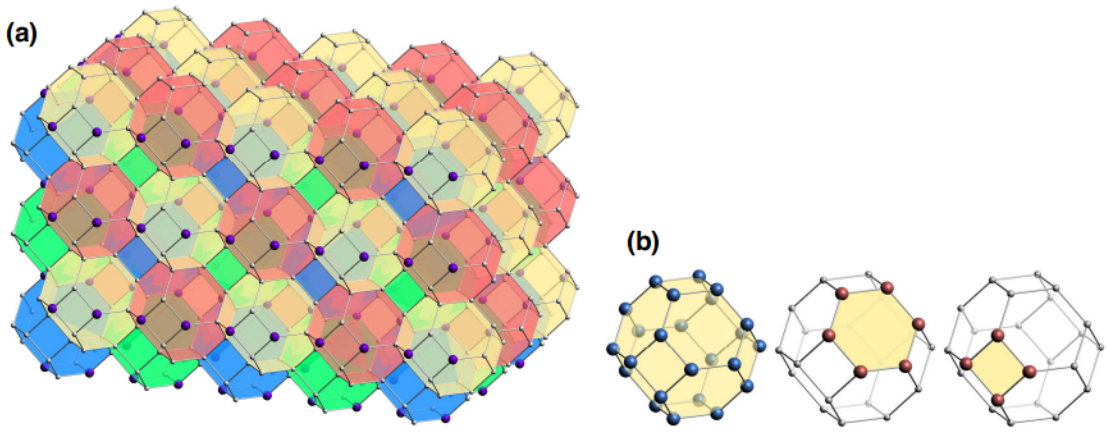

FIG. 9. The 3D color code on a truncated octahedral lattice with periodic boundary conditions. (a) The truncated octahedral lattice, where qubits live on vertices and stabilizers live on cells and faces. The purple vertices correspond to qubits that we choose to Clifford deform with a Hadamard operation. (b) The original stabilizers. Z-stabilizer generators are supported on the 24 qubits (blue vertices) of every cell. X-stabilizer generators are supported on every face, both hexagonal and square.

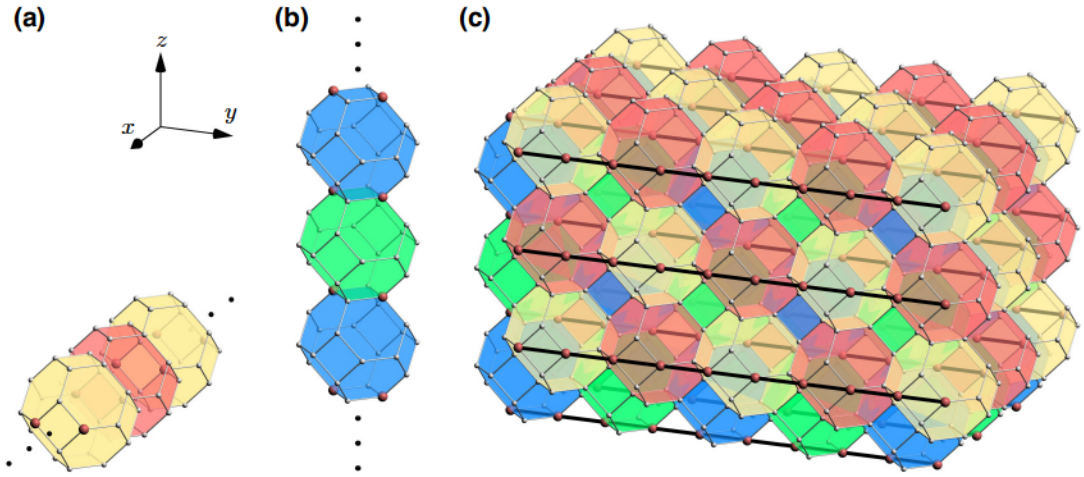

FIG. 10. The linear symmetries of the cells. The red qubits are the Clifford-deformed ones, which effectively become the support of the cell stabilizers in the infinite-Z-bias regime. (a) The symmetry of the yellow and red cells along the x axis. (b) The symmetry of the blue and green cells along the z axis. (c) Matching along the symmetries (a) and (b) gives rise to 2-body checks that form new linear symmetries along the y axis (black lines).

along this symmetry results in new weight-2 checks on the yellow-red faces of the y-z plane, which can be combined with effectively weight-2 checks on the blue-green faces of the x-y plane in an alternating fashion to form a linear symmetry along the x axis, as shown in Fig. 11(b). Matching along such symmetries completes the decoding of errors on the purple qubits in Fig. 11(b).

In the second step, illustrated in Fig. 12, we note that the hexagons are now effectively weight-4 and form linear symmetries in the y direction when combined with square faces in a square-hexagon-hexagon repeating manner. Matching along these symmetries, as shown in Fig. 12(a), gives us new weight-2 checks, supported on the ends of either a hexagon-hexagon intersection edge, or a square-hexagon edge. Combining the weight-2 checks on hexagon-hexagon edges with the weight-2 checks supported on the red-yellow squares of the y-z plane, we obtain new linear symmetries in the z direction, as shown in Fig. 12(b). Matching along these completes the decoding of errors on the purple qubits of Fig. 12(b).

In the final step, illustrated in Fig. 13, we observe that the remaining weight-2 checks on square-hexagon edges obtained in the second step combine with effectively weight-2 checks on yellow-red squares of the x-y

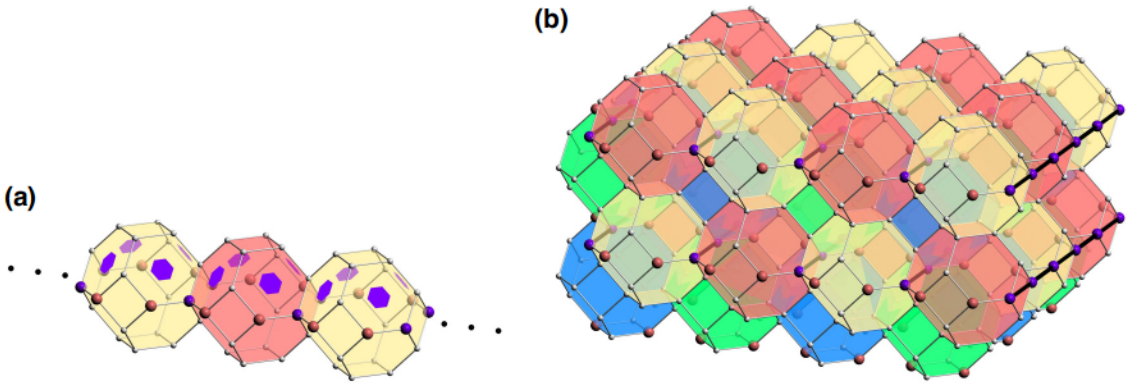

FIG. 11. Step 1 of the decoding of the face syndromes. (a) The linear symmetry of the hexagonal faces: multiplying four hexagonal faces (marked by purple hexagons) on any red or yellow cell gives an eight-body check (red and purple vertices). Since the red vertices represent Clifford-deformed qubits that can effectively be removed from the stabilizers at infinite Z bias, these eight-body checks effectively become four-body (purple vertices). The product of these checks along the y axis forms a linear symmetry. (b) Successful matching along the linear symmetry in (a) gives rise to two-body checks supported on the yellow-red squares of the x-z plane. Combining them with the checks lying on the green-blue squares of the x-y plane, which are effectively two-body due to the Clifford deformation, we obtain a linear symmetry along the x direction, represented by the black lines. Matching along these black lines allows us to decode errors on all purple qubits.

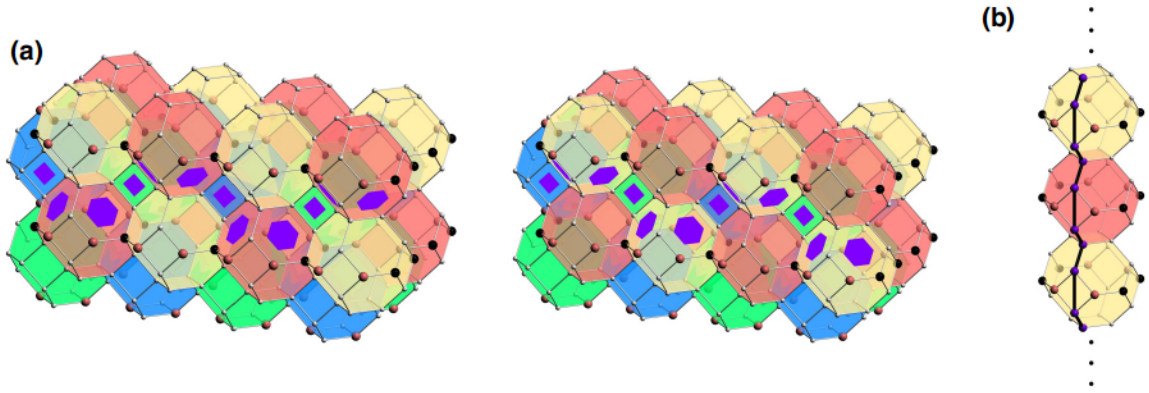

FIG. 12. Step 2 of the decoding of the face syndromes. The red vertices represent the Clifford-deformed qubits (effectively excluded from the face stabilizers at infinite Z bias), while the black vertices represent the qubits on which errors have already been decoded in the previous step. (a) Linear symmetries in the y direction, involving weight-4 checks sitting on both squares and hexagons. (b) Matching along the symmetries represented in (a) results in new weight-2 checks, supported either on ends of hexagon-square edges or on ends of hexagon-hexagon edges. Combining the new checks supported on ends of hexagon-hexagon edges with the checks supported on the red-yellow squares of the y-z plane (which are also weight-2 due to the Clifford deformation), we obtain a linear symmetry along the z axis, represented by the black line. The remaining new weight-2 checks on ends of hexagon-square edges on the blue-green squares of the y-z plane are used in the next step. Note also that the yellow-red squares of the x-y plane effectively become weight-2 checks after this step.

plane to form new linear symmetries along the y axis. Matching along these symmetries decodes the errors on the remaining qubits.

Since errors on all the qubits have been decoded by performing matching on a polynomial number of repetition codes, our decoder has a threshold error rate of 50% for the Clifford-deformed 3D color code. ■

C. Fracton codes

Fracton models offer an interesting set of models to study under biased noise, because the models have intrinsically rigid logical operators. This means that under multiplication by stabilizer generators, the logical operators do not deform in a topological sense. For instance, under stabilizer multiplication, a rigid-string-like logical operator may not deform into a stringlike logical operator of the same width.

By choosing an appropriate Clifford deformation of the stabilizers, fracton models can have materialized subsystem symmetries with respect to infinite-bias noise. The combination of the intrinsic conservation laws in addition to the conservation laws associated with the materialized subsystem symmetries with respect to the noise can lead to decoders with high threshold error rates. Below, we discuss a few canonical examples of fracton models, the X-cube model (type-I fracton model), the Sierpiński fracton model

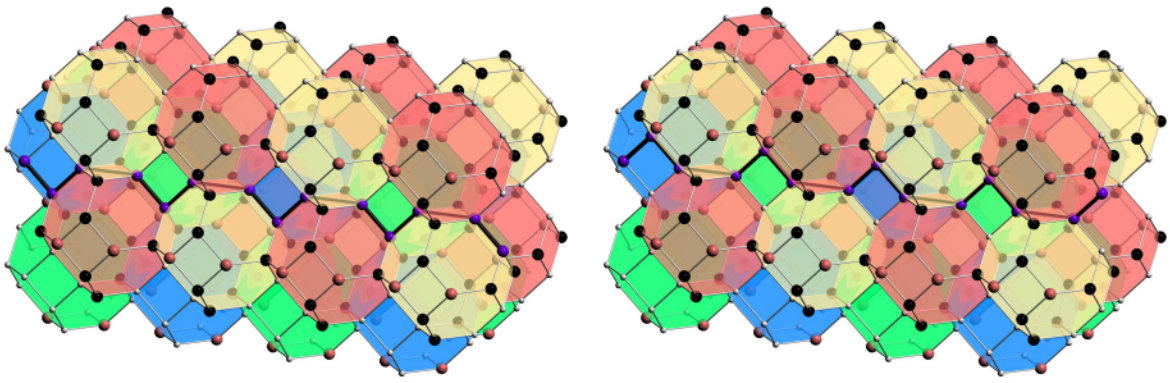

FIG. 13. Step 3 of the decoding of the face syndromes. Using both the weight-2 checks supported on the blue-green squares of the y-z plane, obtained in the second decoding step, and the checks sitting on the yellow-red squares of the x-y plane (which are weight-2 due to the second step as well), we obtain new linear symmetries along the y axis, shown with black lines. Matching along these black lines allows us to decode errors on all the remaining qubits shown in purple.

(fractal type I), and the Haah code (type II), along with their Clifford-deformed codes [61].

1. X-cube model

The X-cube fracton model is the canonical example of a (foliated) type-I fracton topological order that is defined by the presence of topological excitations with restricted mobility. It is characterized by a subextensive ground-space degeneracy and rigid-string logical operators. A foliated topological stabilizer model is defined by a foliation structure [62], which implies that the model can be grown by stacking with a 2D topological state and applying a local unitary. The X-cube model is 3-foliated, which implies that stacks of surface codes can be extracted under a local unitary along all three lattice directions.

The X-cube fracton model [63] is defined on a cubic lattice with qubits on edges. The stabilizer generators come in two types: the *cube stabilizers*, defined on each cubic cell of the lattice as the product of Z operators over the 12 edges of the cube, $A_c = \prod_{e\in c} Z_e$, and the *vertex stabilizers*, defined for each vertex v and orientation $u \in \{\hat{x},\hat{y},\hat{z}\}$ as the product of the four X operators adjacent to v and orthogonal to u, $B_{v,u} = \prod_{e\in v:e\perp u} X_e$ [see Fig. 14(a)]. Considering the X-cube model on an $L_x \times L_y \times L_z$ cuboid with periodic boundary conditions, the logical-operator basis has independent rigid logical string operators that cannot be deformed into each other, i.e., are inequivalent under stabilizer multiplication. This leads to a macroscopic number of independent logical-operator pairs and a linear growth of the number of encoded qubits. These logical operators can be expressed as $\bar{X}^{\hat{i}}_{\hat{k},\ell}, \bar{Z}^{\hat{j}}_{\hat{k},\ell}$ on pairs of noncontractible loops, where $\hat{i}\neq\hat{j}\neq\hat{k}$ run over $\{\hat{x},\hat{y},\hat{z}\}$ and $\ell = 0,\ldots,L_k-1$. They are defined as

$$\bar{X}^{\hat{x}}_{\hat{z},\ell} = \prod_x X_{x,0,\ell,\hat{z}}, \qquad \bar{Z}^{\hat{y}}_{\hat{z},\ell} = \prod_y Z_{0,y,\ell,\hat{z}}, \tag{9}$$

and in a similar fashion for other permutations of x, y, and z, where $X_{x,y,z,\hat{k}}$ ($Z_{x,y,z,\hat{k}}$) denotes a Pauli-X (Pauli-Z) operator on the edge adjacent to the vertex at coordinates (x,y,z) pointing in the $+\hat{k}$ direction for $\hat{k}\in\{\hat{x},\hat{y},\hat{z}\}$. These string operators are not independent due to the three relations given by $\prod_\ell \bar{X}^{\hat{i}}_{\hat{k},\ell} = \prod_\ell \bar{X}^{\hat{k}}_{\hat{i},\ell}$ and $\bar{Z}^{\hat{i}}_{\hat{j},0} = \bar{Z}^{\hat{i}}_{\hat{k},0}$. Thus, there are, overall, $2(L_x+L_y+L_z)-3$ logical-operator pairs. These string operators are rigid in nature, as is characteristic of type-I models. The rigidity of the string operators directly corresponds to the restricted mobility of excitations. For example, particles that are pair created by a completely rigid undeformable string operator are restricted to move in one dimension and are therefore *lineons*. Truncations of logical string operators of X errors on a lattice plane create syndromes of cube stabilizers at their end points, as shown in Fig. 15(a). These syndromes cannot freely move to another plane (under arbitrary noise) without creating other syndromes. Hence, the cube syndromes are referred to as *planons*. Similarly, the vertex syndromes are created at the ends of rigid strings of Z errors. Note that two of the vertex-stabilizer generators are violated at each end of the string. This composite syndrome at each end is referred to as a lineon, since it cannot move (under arbitrary noise) to another line away from the rigid string, without creating more syndromes.

We consider a Clifford deformation of the X-cube model where a Hadamard is applied on all vertical edges, similar to the Clifford deformation of the 3D surface code on a

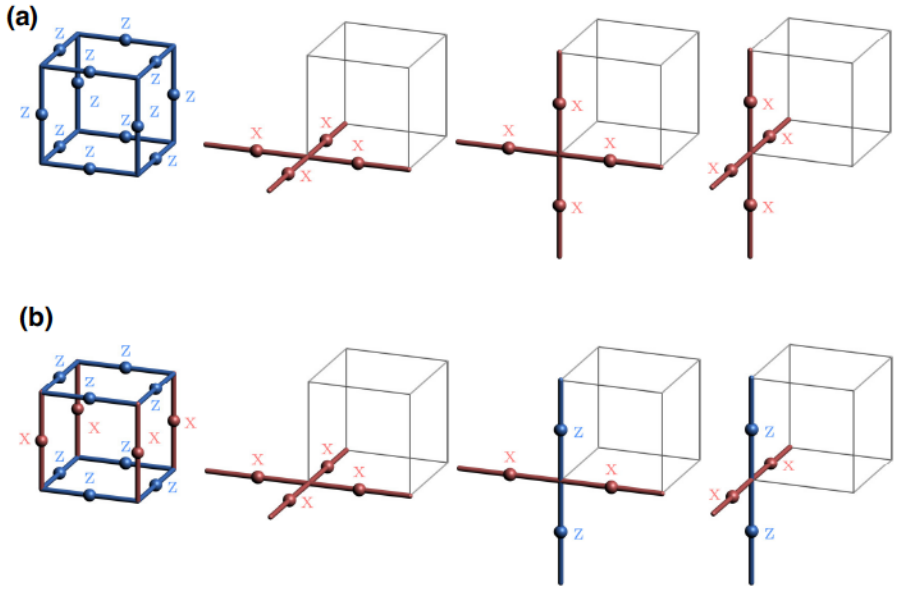

FIG. 14. The X-cube model. (a) The original stabilizers. (b) The Clifford-deformed stabilizers.

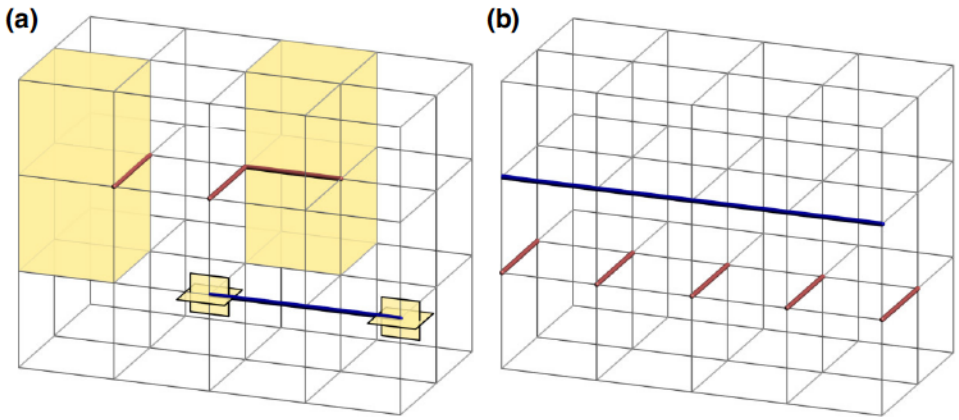

FIG. 15. (a) Errors in the X-cube model create two types of syndromes: planons (at cells) and lineons (at vertices with orientation). (b) Examples of the X and Z logical operators of the X-cube model in red and blue, respectively.

cubic lattice. The Clifford-deformed stabilizer generators are represented in Fig. 14(b). At infinite Z bias, lineons on the Clifford-deformed X-cube model can only be created by Z errors on z edges, while planons can only be created by Z errors on x and y edges. As a result, we have the following materialized symmetries: the product of vertical planons along a vertical line is effectively the identity and the product of cubes along a horizontal line is effectively the identity. These symmetries are represented in Fig. 16. Using the conservation laws associated with these symmetries, we prove that Clifford-deformed X-cube model has a threshold error rate of 50% at infinite Z bias. Note that using statistical-mechanical simulations, the optimal infinite bias thresholds for the CSS X-cube model with the cube (fracton) term made of Pauli-X operators have been found to be 15.2% and 7.5% at infinite X and Z biases, respectively [64].

Theorem 3.—The Clifford-deformed X-cube model has a 50% threshold error rate under pure-Z noise.

Proof.—Let us first consider the cube-stabilizer generators. As illustrated in Fig. 16(a), at infinite bias, these stabilizer generators are now effectively supported on four vertical qubits and form independent sheets of infinite-bias XY surface codes on each layer. We have shown in Sec. II D that this code has linear symmetries on all its rows and columns,and by using the weight-reduction technique, we have proved that it has a threshold error rate of 50%. Since decoding the cube stabilizers at infinite bias is equivalent to decoding L different XY codes, we can infer that the cube sector has a threshold error rate of 50%.

Let us now consider the lineon sector. As illustrated in Fig. 16(b), at infinite bias, vertex stabilizers in the y-z and x-z planes effectively become two-body checks between qubits on horizontal edges that form linear symmetries along the x and y axes, respectively. The problem of decoding the lineon sector therefore becomes equivalent to decoding $O(L^2)$ repetition codes, which also has a 50% threshold error rate. Therefore, the overall code has a threshold error rate of 50%. ■

2. Sierpiński fractal model

The Sierpiński fractal model, due to Castelnovo, Chamon, and Yoshida [65,66], is the simplest example of fractal type-I topological order. Fractal type-I topological order is defined as type-I fracton topological order that is characterized by the presence of fractal-shaped logical operators and hence does not admit a foliation structure.

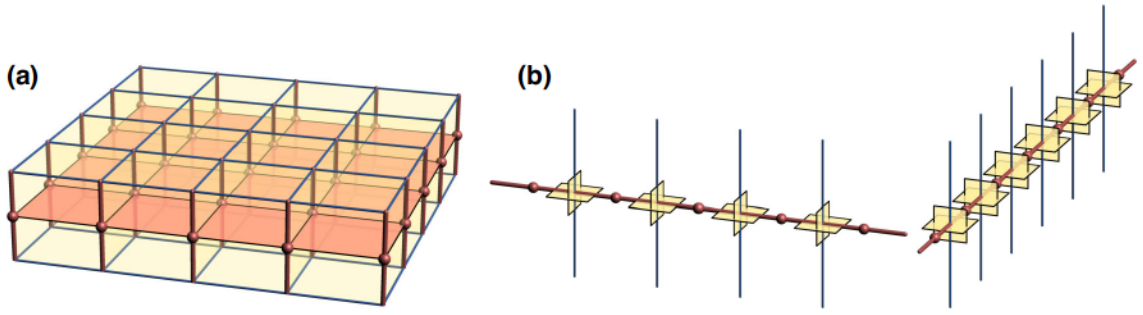

FIG. 16. The symmetries of the Clifford-deformed X-cube model at infinite Z bias. (a) The cube stabilizers reduce to four-body checks (red squares) between vertical qubits, forming an independent infinite-bias XY code on each layer. They therefore inherit the linear symmetries and the 50%-threshold error rate of the XY code (see Sec. II D). (b) The vertex stabilizers in the y-z and x-z planes effectively become two-body checks (red edges) between qubits on horizontal edges that form linear symmetries along the x and y axes, respectively.

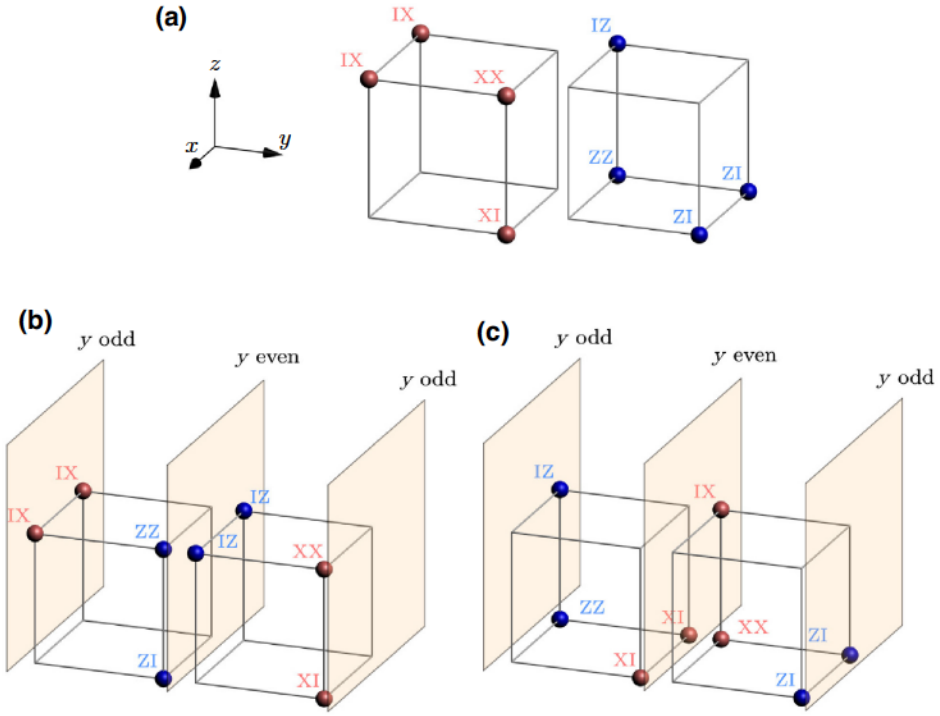

FIG. 17. The Sierpiński code, original and Clifford deformed. (a) The original CSS stabilizers, defined on every cube of a cubic lattice with two qubits per vertex. There are two types: X stabilizers (left) and Z stabilizers (right). (b) The Clifford-deformed X stabilizers. The Clifford deformation consists of applying a Hadamard on all qubits of vertices on x-z planes of even y. Therefore, stabilizers alternate between the left and the right versions depending on the parity of y (c) The Clifford-deformed Z stabilizers, using the Clifford deformation described in (b).

The model is defined on a cubic lattice, where each vertex has two qubits. The stabilizer generators are shown in Fig. 17(a). This model supports rigid-string operators (corresponding to 1D particles or lineons) in the $\hat{z}$ direction and a Sierpinski triangle fractal operator that moves topological excitations apart in 2D. Hence this model provides an example with no planons, which is consistent with it not having a foliation structure [67,68].

We present the Clifford deformation of this model, where a Hadamard is applied to all qubits of alternating planes, such as on all x-z planes with an *even* y coordinate, as shown in Figs. 17(b) and 17(c). The model has materialized symmetries that lead to a threshold error rate of 50% at infinite bias as stated below. Note that for the original CSS model, one can use the relation from Ref. [69] involving the entropy function $h(p)$ for infinite bias threshold error rates p_X (p_Z) at infinite X bias (Z bias), respectively, as follows:

$$h(p_X) + h(p_Z) \approx 1, \tag{10}$$

where $h(p) = -p\log_2(p) - (1-p)\log_2(1-p)$ is the binary entropy. Due to the invariance of the model stabilizers under $X \leftrightarrow Z$ permutation, permutation of two qubits on the sites, and inversion, we have $h(p_X) = h(p_Z)$. Together, we obtain $h(p_Z) \approx 1/2$, which yields an optimal threshold estimate of $p_z \approx 0.11$ at infinite bias.

Theorem 4.—The Clifford-deformed Sierpiński code has a threshold error rate of 50% under pure-Z noise.

Proof.—We first study the Clifford-deformed X stabilizers in Fig. 17(b). At infinite Z bias, these effectively become two-vertex checks supported on planes of odd y, oriented either along the x direction or the z direction. The checks oriented along the x direction have a term IX on each vertex and form a linear symmetry, as shown in Fig. 18(a). Matching along it allows us to decode the errors on the second qubits of the vertices living on planes of odd y. Once the errors on these qubits have been decoded, we can use them to simplify the checks oriented in the z direction. More precisely, all the terms sitting on the second qubit of a vertex can now be removed from the check, turning XX into XI. Those updated checks form a new linear symmetry, shown in Fig. 18(b). Matching along this symmetry allows us to decode the first qubit of every vertex living on planes of odd y. The proof for the Clifford-deformed Z stabilizers follows a similar pattern and is illustrated in Figs. 18(c) and 18(d). Overall, this decoding strategy is equivalent to decoding a polynomial number of repetition codes (in the lattice size L), showing that it has a threshold error rate of 50%. ■

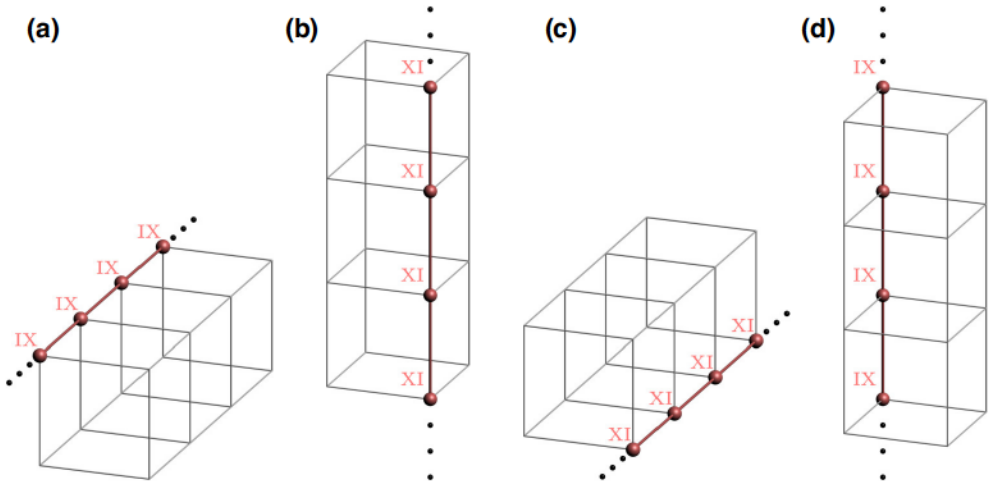

FIG. 18. The effective symmetries of the Clifford-deformed Sierpiński code under infinite bias. (a) The linear symmetry of the Clifford-deformed X stabilizers, which allows us to decode errors on the second qubits of vertices on planes of odd y. (b) The new linear symmetry of the Clifford-deformed X stabilizers obtained after decoding the errors on qubits in (a). Since errors on the second qubits of the red vertices have all been decoded, the XX terms become XI terms, allowing us to decode the errors on the first qubits of vertices on planes of odd y. (c) The linear symmetry of the Clifford-deformed Z stabilizers that allows us to decode errors on the first qubits of vertices on planes of even y. (d) The new linear symmetry of the Clifford-deformed Z stabilizers, obtained after decoding the errors on qubits in (c) using the same argument as in (b). This allows us to decode errors on the second qubits of vertices on planes of even y.

3. The Haah code

The Haah code is the canonical example of type-II fracton topological order, which is characterized by the absence of string logical operators, the presence of fractal logical operators, and a subextensive ground-space degeneracy that can fluctuate with the system size. The stabilizer generators of the Haah code (CSS model) are shown in Fig. 19(a).

We present the Clifford-deformed Haah code in Figs. 19(b) and 19(c), which we prove below to have a 50% threshold error rate at infinite bias. Note that, similar to the CSS Sierpiński model, for the CSS Haah code, one can also use the relation from Ref. [69] involving the entropy function $h(p)$ for infinite bias threshold error rates p_X (p_Z) at infinite X bias (Z bias), respectively, as follows:

$$h(p_X) + h(p_Z) \approx 1, \tag{11}$$

where $h(p) = -p\log_2(p) - (1-p)\log_2(1-p)$. And again, due to the invariance of the model stabilizers under $X \leftrightarrow Z$ permutation, permutation of two qubits on the sites, and inversion, we have $h(p_X) = h(p_Z)$. Together, we obtain

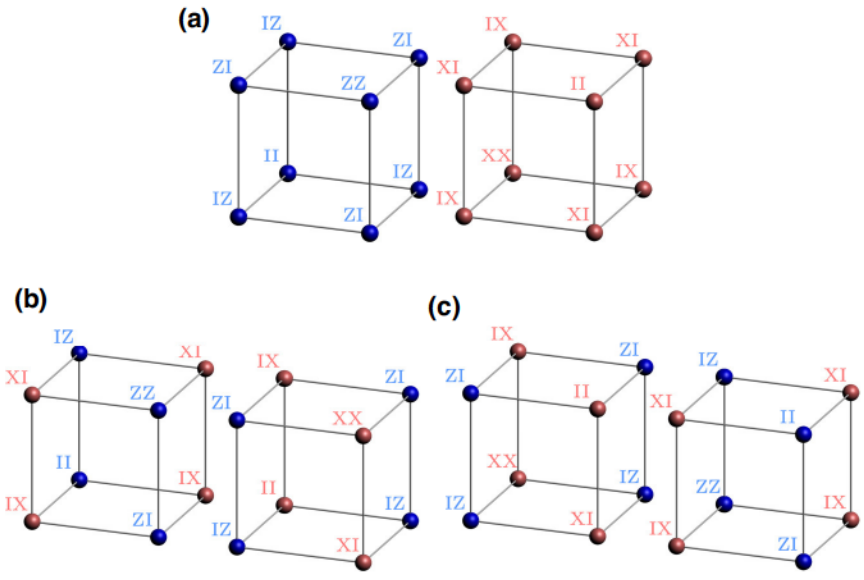

FIG. 19. The Haah code. (a) The original stabilizers. The code is defined on a cubic lattice, with two qubits per vertex. Each cube of the lattice contains both an X and a Z stabilizer. (b) The Clifford-deformed Z stabilizer. A Hadamard is applied on the two qubits of half of the vertices, in a checkerboard manner on each layer. As a result, half of the cells contain the stabilizer on the left and half of them contain the stabilizer on the right. (c) The Clifford-deformed X stabilizer, when applying the transformation described in (b).

$h(p_Z) \approx 1/2$, which yields an optimal threshold estimate of $p_Z \approx 0.11$ at infinite Z bias.

We now state the theorem about the threshold and its proof.

Theorem 5.—The Clifford-deformed Haah code on a periodic lattice with dimensions (L_x, L_y, L_z), such that $L_z = 2^k$, $L_x = L_y$, and $\gcd(L_x, L_z) = 2$, has a threshold error rate of 50% under pure-Z noise.

Such constraints ensure that the horizontal dimensions are even, which is required for our checkerboardlike Clifford deformation to be well defined, and that $\gcd(L_x, L_z)$ does not grow with the lattice size, which is needed for the linear symmetries considered in our proof. Because the number of encoded qubits in a fractal type-II fracton model such as the Haah code can fluctuate wildly with the system size, it is subtle to extract the threshold error rate from an arbitrary family of increasing system sizes. Nevertheless, there are families of codes satisfying the above constraints, the number of encoded qubits of which is constant. We have checked this numerically for codes of size $(2^k + 2, 2^k + 2, 2^k)$ for $k \geq 2$, which have six encoded qubits, and $(2 \cdot 3^k, 2 \cdot 3^k, 2^k)$ for $k \geq 1$, which have six encoded qubits.

Proof.—Let us call the Clifford-deformed qubits *type-A qubits* and the vertices on which they live *type-A vertices*. We refer to the other half of the qubits (vertices) as *type-B qubits* (*type-B vertices*). Those two types of qubits alternate in a 2D-checkerboard manner on each horizontal layer of the lattice. In this language, the Clifford deformation of the Z-type stabilizers gives stabilizers supported on type-A qubits only. We call these stabilizers *type-A stabilizers*. Similarly, *type-B stabilizers* are those resulting from the Clifford deformation of the X-type stabilizers and are supported on type-B qubits only.

Let us start by considering only type-A stabilizers, which are shown in Fig. 19(b). The decoding strategy is the same when tackling type-B stabilizers. In the infinite-bias regime, the Clifford-deformed code becomes a classical code, with two types of parity-check operators alternating in a checkerboard manner. Those two types of checks have weight 4 but one is supported on four vertices and the other on three. We can observe the presence of a linear symmetry for the ones supported on four vertices, as shown in Fig. 20(a). Matching along this symmetry results in the appearance of new weight-2 checks, with the term "XI" on one vertex and "IX" on the other vertex. We call them "XI-IX" checks.

We then consider the other type of check supported on three vertices. Multiplying it by two "XI-IX" checks, we obtain a new L-shaped weight-4 check made only of "IX" terms, as represented in Fig. 20(c). We call them "L-checks."

We now use a technique introduced to study the classical Fibonacci codes [70] and the XYZ color code [11]. We first note that applying four L-checks as an L results in a new L-check where the spacing between the nonzero qubits has doubled but the weight is still 4. Applying the same process recursively results in a fractal of original checks, forming an L-check the size of which can be an arbitrary power of two. This process is shown in Fig. 21(a). Note that this family of L-checks always lives on a 2D diagonal slice of our lattice, which has dimensions (L_x, L_z). This is due to the choice of periodic boundary conditions and equal horizontal dimensions.

We now use the fact that $L_z = 2^k$. Applying the fractal process described previously, we can create an L-check where two terms are separated by a distance of exactly 2^k. Due to the periodic boundary conditions, these two terms

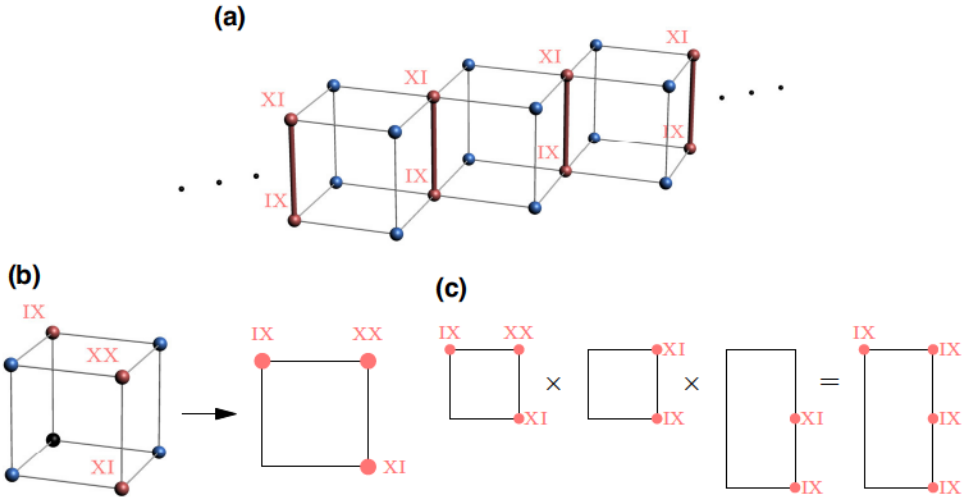

FIG. 20. Step 1 in the proof of the deformed-Haah-code 50% threshold error rate. (a) The linear symmetry. Matching along it allows a weight reduction to weight-2 checks (red edges). (b) A visualization of the second stabilizer type on a 2D plane. (c) A new L-shaped pure "IX" stabilizer, obtained by multiplying the stabilizer in (b) by the weight-2 checks obtained in (a). We call this an L-check.

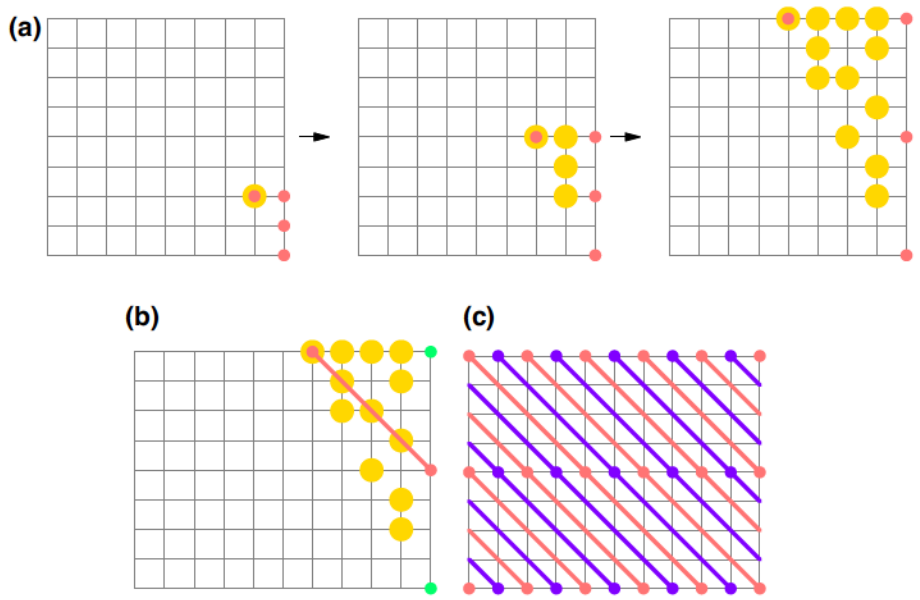

FIG. 21. Step 2 in the decoding of the Clifford-deformed Haah code at infinite bias. Yellow dots on vertices represent the application of an L-check, as shown on the left of (a), whereas red dots represent the qubits in its support. (a) By multiplying the checks in a fractal manner, we can obtain a similar weight-4 check with any power-of-2 spacing between the qubits in its support. (b) Taking a periodic lattice with vertical dimension $L_z = 2^k$, we can make two of the qubits cancel out (green) to obtain a new weight-2 check (red). (c) On any periodic lattice with dimensions (L_x, L_y) such that $\gcd(L_x, L_y) = 2$, diagonal lines wrapping around the torus cover half of the vertices (red or purple). Therefore, the product of these new weight-2 checks on a line forms a linear symmetry (either red or purple depending on the starting point).

cancel out, leaving only two qubits in the support of the check. This new weight-2 check is shown in Fig. 21(b).

As $\gcd(L_x, L_z) = 2$, we can multiply these weight-2 checks on a line to cover exactly L_x qubits, before the line comes back to itself, as shown in Fig. 21(c).

This is due to the fact that on a periodic lattice with $\gcd(L_x, L_y) = g$, there are g distinct diagonal lines covering each a fraction $1/g$ of the vertices [52]. Since the dimensions of our diagonal slice are (L_x, L_z), it contains $L_x L_z$ vertices and the diagonal line formed by the weight-2 checks covers $L_x L_z/2$ vertices. Dividing this by the size of the weight-2 check, $L_z/2$, we obtain that the product of weight-2 checks on the line covers exactly L_x qubits.

Therefore, this product is equal to the identity operator and we obtain a linear symmetry. By translating the large L-check, we can include any arbitrary qubit of the 2D diagonal slice in the support of this linear symmetry. Matching along them on all the 2D slices therefore results in decoding the errors on second qubits of all the type-A vertices. Using the "XI-IX" check allows us to decode the errors on the first qubits of these vertices as well. Finally, applying the same decoder to the type-B stabilizer, we can decode errors on all type-B qubits.

Since this decoder has only involved matching on a polynomial number of repetition codes, we can deduce that our Clifford-deformed Haah code has a 50% threshold error rate at infinite bias. ■

Note that the choices of $L_x = L_y$ and $\gcd(L_x, L_z) = 2$, while simplifying the proof, are not strictly necessary to obtain the desired linear symmetries and the 50% threshold error rate. Relaxing these constraints has the effect of changing the size of the 2D diagonal slice, as it can wrap around the torus several times when the horizontal dimensions are not equal. The new size of the slice can be shown to be $\ell = L_x L_y / \gcd(L_x, L_y) = \mathrm{lcm}(L_x, L_y)$. As a result, the condition to obtain a linear symmetry on the weight-2 checks is modified: we want the number of terms in the linear symmetry to grow polynomially with the system size. As the number of terms is given by the size of the diagonal line that supports the symmetry, $\mathrm{lcm}(\ell, L_z) = \mathrm{lcm}(L_x, L_y, L_z)$ [71], divided by the size of the weight-2 check, $L_z/2$, the condition can be reformulated as

$$\frac{\mathrm{lcm}(L_x, L_y, L_z)}{L_z} = \Omega(\mathrm{poly}(L_z)). \tag{12}$$

The imposition of Eq. (12) with $L_z = 2^k$, and L_x, L_y even (to guarantee that the Clifford deformation is well defined) is enough to have a 50% threshold error rate for the Haah code.

IV. THRESHOLD ERROR RATES AT FINITE BIAS

Using the BP OSD [21,49,50], described in Appendix B 1, we evaluate the threshold error rates of both

TABLE I. Estimates and uncertainties of the threshold error rate p_{th} for CSS and Clifford-deformed surface codes on a cubic lattice using different decoders under finite and infinite bias as plotted in Fig. 23(a).

	CSS		Deformed	
Bias η_Z	BP OSD (%)	Sweep matching (%)	BP OSD (%)	Sweep matching (%)
0.5	5.95 ± 0.03	$4.0^{+0.3}_{-0.5}$	$5.99^{+0.08}_{-0.1}$	$4.42^{+0.12}_{-0.32}$
1	...	5.4 ± 0.4	...	$5.06^{+0.06}_{-0.24}$
3	$12.25^{+0.05}_{-0.06}$	$11.48^{+0.11}_{-0.19}$	7.94 ± 0.03	6.7 ± 0.6
10	$22.3^{+0.04}_{-0.05}$	$14.9^{+0.5}_{-1.0}$	$12.12^{+0.12}_{-0.11}$	$11.6^{+0.3}_{-0.2}$
30	21.7 ± 0.04	$14.6^{+0.3}_{-0.7}$	$17.76^{+0.12}_{-0.1}$	19.3 ± 0.3
100	$21.46^{+0.02}_{-0.04}$	$14.3^{+0.4}_{-0.7}$	$23.0^{+0.6}_{-0.5}$	$20.6^{+0.5}_{-0.7}$
∞	$21.37^{+0.04}_{-0.03}$	$14.59^{+0.11}_{-0.18}$	50	$20.7^{+0.2}_{-0.3}$

the CSS and Clifford-deformed 3D surface code defined on cubic and checkerboard lattices as well as the X-cube model at different bias ratios over several orders of magnitude and at infinite bias. We also estimate threshold error rates for the surface code on the cubic lattice using the sweep-matching decoder described in Appendix B 2. We plot the threshold-error-rate estimates for different values of bias in Fig. 23. The numerical values of the threshold-error-rate estimates and uncertainties are listed in Table I for the surface code on a cubic lattice, in Table II for the surface code on a checkerboard lattice, and in Table III for the X-cube model. The estimates are best-fit parameters to a finite-size scaling ansatz and uncertainties are bootstrapped 1σ credible intervals that account for the finite number of trials and choice of parameters. An in-depth overview of how these are obtained can be found in Appendix C. Unless a lower numerical threshold error rate can be resolved by finite-size scaling, the theoretically proved 50% threshold error rate is tabulated and plotted for Clifford-deformed codes at infinite bias.

We note that at moderate biases, 3D surface codes such as the Clifford-deformed surface code and the CSS surface code on a 3D-checkerboard lattice can have threshold error rates close to the hashing bound and to those of 2D codes such as XZZX and XY. This offers a noise regime in which one could consider a dimensional jump for implementation of non-Clifford gates and be able to maintain at least the code-capacity threshold error rates. For high bias, $\eta_Z \gtrsim 100$, the Clifford-deformed surface code on a cubic lattice beats the CSS surface code on a cubic lattice in threshold-error-rate performance, with threshold error rates of 50% and 21.37(4)%, respectively, at infinite bias. The Clifford-deformed code on the checkerboard lattice at infinite bias also boasts an advantage at infinite bias. Above modest values of bias, $\eta_Z \gtrsim 30$, the Clifford-deformed X-cube model outperforms its CSS counterpart. This is due to the rigid noise symmetries, which allow decoding in rigid submanifolds.

The relative difference in the finite-bias thresholds for the three codes (with 50% infinite-bias threshold) in Fig. 23 can be attributed to the differences in the weights of logical-operator representations with more Pauli-Zs than Pauli-Xs and Pauli-Ys. These differences in turn arise due to the fact that the minimum-weight logical-operator representations of the 3D surface codes are membranelike, while the minimum-weight logical-operator representations of the X-cube model are stringlike. Among the two types of 3D surface codes, there is a difference in the weights of logical-operator representations due to the differences in average stabilizer weights corresponding to the pointlike and looplike syndromes, respectively.

TABLE II. Estimates and uncertainties of the threshold error rate p_{th} for CSS and Clifford-deformed surface codes on a checkerboard lattice using a BP-OSD decoder under finite and infinite bias as plotted in Fig. 23(b).

Bias η_Z	CSS (%)	Deformed (%)
0.5	1.35 ± 0.04	$1.25^{+0.03}_{-0.07}$
3	3.74 ± 0.03	$2.67^{+0.06}_{-0.1}$
10	10.24 ± 0.09	$5.2^{+0.14}_{-0.19}$
15	15.1 ± 0.08	...
20	$19.72^{+0.09}_{-0.1}$	...
30	$29.09^{+0.13}_{-0.16}$	10.3 ± 0.2
100	$28.79^{+0.12}_{-0.11}$	$19.3^{+0.3}_{-0.2}$
∞	28.55 ± 0.13	50.0

TABLE III. Estimates and uncertainties of the threshold error rate p_{th} for the CSS X-cube model and Clifford-deformed X-cube model using a BP-OSD decoder under finite and infinite bias as plotted in Fig. 23(c).

Bias η_Z	CSS (%)	Deformed (%)
0.5	$4.54^{+0.05}_{-0.08}$	$4.6^{+0.3}_{-0.7}$
3	$10.64^{+0.06}_{-0.07}$	$5.9^{+0.4}_{-0.8}$
10	$9.3^{+0.2}_{-0.4}$	$7.5^{+0.4}_{-1.0}$
30	$8.9^{+0.3}_{-0.6}$	10.7 ± 0.4
100	$8.9^{+0.2}_{-0.3}$	$17.4^{+0.3}_{-0.2}$
∞	9.25 ± 0.08	$17.2^{+0.17}_{-0.12}$

TABLE IV. The girth and split-belief numbers for four of the 3D codes studied in this work. The girth is the size of the shortest cycle of the Tanner graph, while the split-belief number is the weight of the smallest error that causes a degenerate syndrome. We consider here the X and Z parts of the Tanner graph separately, as we are using independent BP-OSD decoders for each error type. Here, the X-girth (Z-girth) corresponds to the girth when considering only the X (Z) stabilizers in the Tanner graph, and similarly for the X- and Z-split-belief numbers.

Code	X-girth	Z-girth	X-split-belief number	Z-split-belief number
3D surface code (cubic)	8	8	2	3
3D surface code (checkerboard)	8	6	2	6
X-cube model	8	4	2	6
3D color code	4	6	2	12

A. Limitations of the BP OSD

As discussed in Appendix B 1, the performance of the BP OSD depends greatly on the characteristics of the code, particularly its *girth* (the size of the shortest cycle on the Tanner graph) and its *split-belief number* (the weight of the smallest error that produces a degenerate syndrome). The latter can be calculated by taking the weight of the smallest even-weight stabilizer and dividing it by two. We show these two numbers for different 3D codes in Table IV.

A common phenomenon that appears for codes with low girth or a low split-belief number is the receding threshold-error-rate problem: the apparent threshold error rate decreases with increasing system sizes [22]. We have observed this finite-size effect for the 3D surface code on a checkerboard lattice and for the X-cube model, while the 3D surface code on a cubic lattice did not have this issue for sizes up to 22. An illustration of this phenomenon is shown in Fig. 22 for the X-cube model at bias $\eta = 10$. Consequently, on codes where the BP OSD suffers this limitation, there is greater uncertainty in the threshold-error-rate estimates upon using the bootstrapped finite-size scaling method described in Appendix C, as can be seen in, e.g., Table III.

We also perform preliminary experiments on the 3D color code but we observe a strong receding threshold effect with an apparent threshold error rate orders of magnitude below its optimal value. This can be explained by the particularly low girth of the 3D color code and it suggests that we should not pursue these experiments.

Note that while we have provided decoders with a 50% threshold error rate at infinite bias for all the Clifford-deformed codes studied in this paper, the BP OSD does not always achieve that threshold; e.g., in the X-cube model as seen in Fig. 23(c) and Table III, where the threshold error rate is only 17.2(3)%. Nevertheless, the Clifford-deformed X-cube model still outperforms the CSS X-cube model, which has a lower threshold error rate of 9.25(10)% when decoding with the BP OSD.

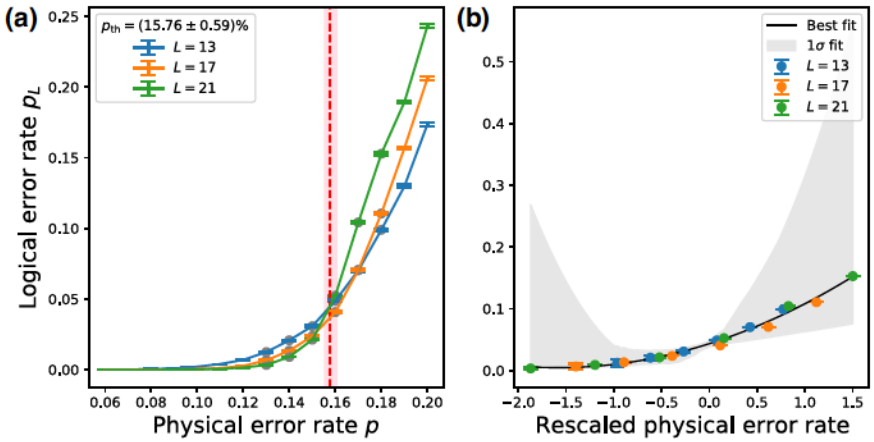

FIG. 22. The size-dependent reduction in the apparent threshold error rate using a BP-OSD decoder for increasing Clifford-deformed X-cube model lattice sizes under Z-biased noise with $\eta_Z = 10$. (a) The rate of logical Z errors versus the physical error rate p. Note that the apparent intersection of the curves for $L = 13$ is somewhat different from that of those for $L = 21$. Despite the apparent threshold-error-rate shift, we still provide a threshold-error-rate estimate by regression with the finite-size scaling ansatz. (b) The logical Z error rate versus the rescaled physical error rate $x = (p - p_{\text{th}})^{1/\nu}$, where the threshold error rate p_{th} and the critical exponent ν have been estimated by fitting to a finite-sized scaling ansatz, for which the best-fit curve and 1σ fits are plotted against data points colored by the code-lattice size, as detailed in Appendix C.

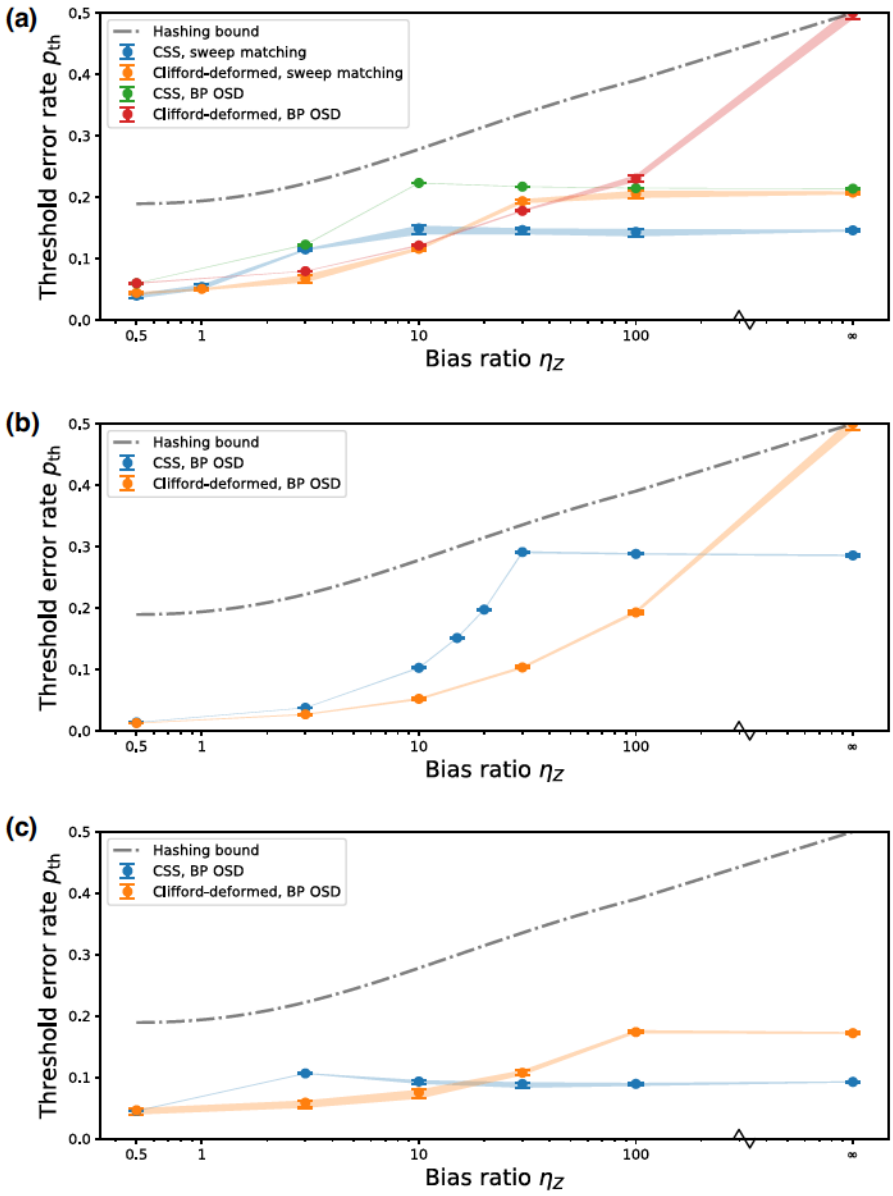

FIG. 23. The threshold error rate p_{th} versus the dephasing bias η_Z for some CSS and Clifford-deformed 3D topological codes. (a) The BP OSD and sweep-matching threshold error rates of the 3D CSS surface code and the Clifford-deformed surface code on a cubic lattice. (b) The BP-OSD threshold error rates of the CSS and Clifford-deformed surface codes on a checkerboard lattice. (c) The BP-OSD threshold error rates of the CSS and Clifford-deformed X-cube model.

V. ROTATED LAYOUT AND SUBTHRESHOLD SCALING

A. Rotated layout for the 3D surface code

We now define a rotated layout for the 3D surface code. The new lattice is obtained by rotating the coordinates about the vertical z axis by 45° such that the horizontal qubits formerly living on x and y edges now live on vertices on horizontal x-y planes, while vertical qubits formerly living on z edges now live on vertices floating between horizontal x-y planes. Nevertheless, we continue to refer to qubits living on x-y planes as *horizontal* qubits and qubits floating in between x-y planes as *vertical* qubits. Former vertex operators become octahedron stabilizers, former vertical face stabilizers become diamond stabilizers, and former horizontal face stabilizers become square stabilizers. Such a rotated layout preserves the distances (d_X, d_Z) for both Pauli-X and Pauli-Z logical operators, while using roughly half the number of physical qubits compared to the regular layout. This new lattice is illustrated in Figs. 24(a) and 24(b) for open and periodic boundary conditions, respectively.

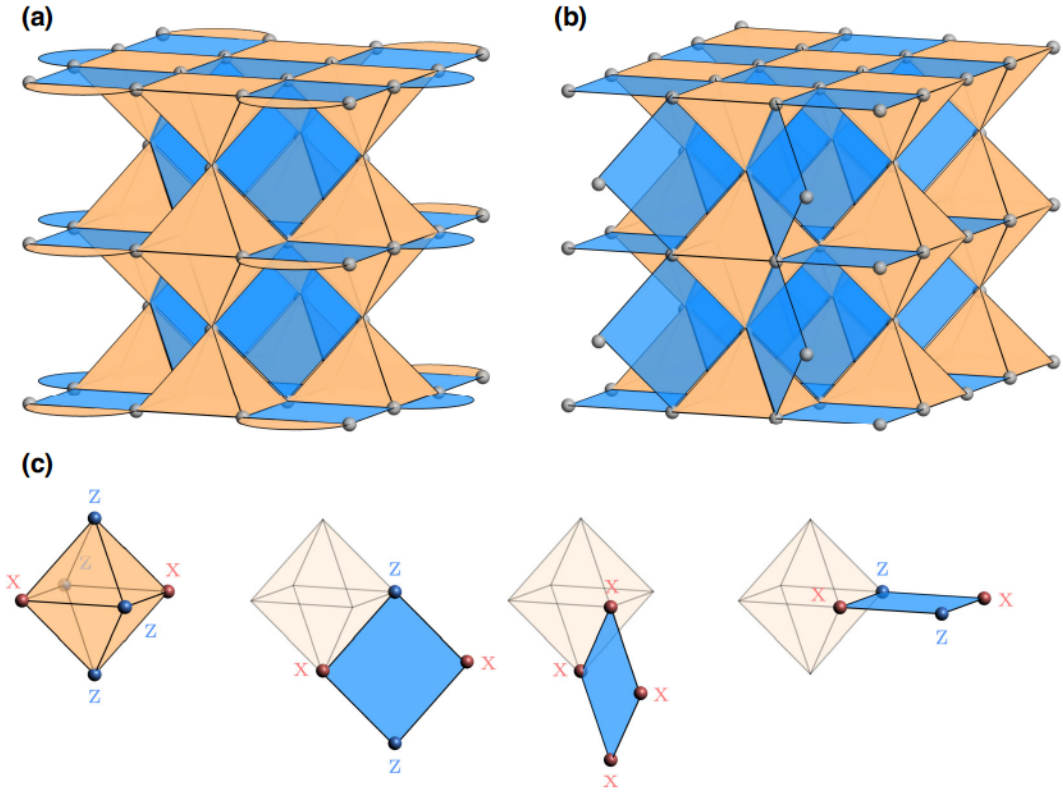

FIG. 24. The Clifford-deformed 3D surface code on a rotated layout with smooth boundaries at the top and bottom. In this representation, qubits live on vertices, while stabilizers live on octahedra (orange) and faces (blue), which are either diamonds between x-y planes or squares on x-y planes. (a) A $4 \times 4 \times 3$ rotated lattice with open boundaries. (b) A $4 \times 3 \times 3$ rotated lattice with periodic boundaries. Note that although the horizontal x-y planes are always periodic, if one of the dimensions along x or y is odd, then the diamonds in between the x-y planes are not periodic along that direction, as illustrated here (the blue diamonds are not periodic in the length-3 direction). (c) Clifford-deformed stabilizer generators: (left to right) an octahedron acting as XZZX on horizontal qubits and as Z on vertical qubits above and below, a diamond acting as Z on horizontal qubits and as X on vertical qubits, a diamond acting as XXXX, and a square acting as XZZX on horizontal qubits.

Note that for periodic boundary conditions with one odd horizontal dimension, such as in Fig. 24(b), the horizontal planes are periodic in both directions, but the vertical diamonds are not periodic in the odd-length direction, resulting in a *seam* across which the checkerboard pattern of the horizontal planes of octahedron and horizontal square stabilizers are incompatible, and vertical qubits are not connected by diamond stabilizers. To ensure that the octahedron and square stabilizers commute across this seam, stabilizers touching the seam on only *one* chosen side are modified by introducing "defects" at every horizontal qubit on the seam, where a Hadamard is applied to modify these stabilizer definitions with $X \leftrightarrow Z$ at these defects, as illustrated in Fig. 25(a).

To Clifford deform the code, we apply a Hadamard only on every second horizontal qubit on each x-y plane in a checkerboard manner, while leaving vertical qubits untouched, such that the resulting Clifford-deformed stabilizers are as shown in Fig. 24(c). In the case of an odd-length lattice, even the stabilizers with defects on the seam will be of this form after Clifford deformation. Consequently, all the octahedron and horizontal square stabilizers act as XZZX when restricted to horizontal planes, as shown in Fig. 25(b), forming L_z coupled layers of 2D XZZX codes.

B. Pure-Z logical-operator representative

The 2D XZZX code on a rotated layout with periodic boundaries and coprime dimensions has pure-Z logical operators supported on $O(L^2) = O(n)$ physical qubits [7]. The intuition behind this fact is that syndromes propagate on the diagonals and when the dimensions of the lattice are coprime, strings of errors need to wrap around the whole torus in order to form a nontrivial loop. As a consequence, the logical error rate $\bar{p}$ for purely Z-biased noise scales as

$$\bar{p} \propto e^{-\alpha(p) d_Z} = e^{-\alpha(p) n} \tag{13}$$

when the physical error rate p goes to zero, where $\alpha(p)$ is a polynomial in p and d_Z is the Z-distance of the code.

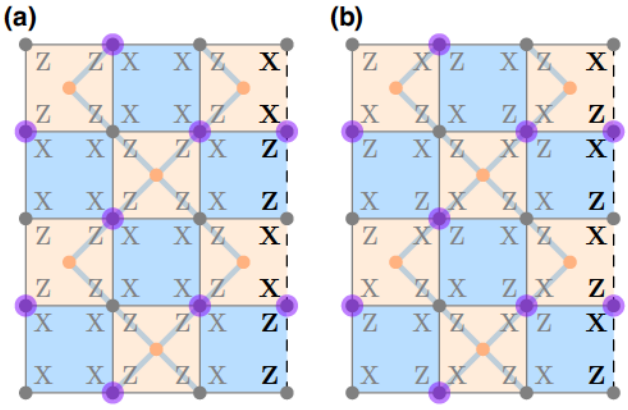

FIG. 25. The horizontal layer of a 3D surface code on a rotated layout with *odd* × *even* horizontal dimensions and periodic boundary conditions with a seam (dashed), like that of Fig. 24(b). The restricted action of the octahedron stabilizers (orange) and the horizontal square stabilizers (blue) on horizontal qubits (vertices) in the layer are labeled. The diamond stabilizers between adjacent layers, the shadows of which are drawn as faint blue lines, are disconnected across the seam. (a) The original stabilizers. Stabilizer terms modified by defects on the seam are labeled in bold. Qubits where the Clifford deformation is applied as a Hadamard are highlighted in purple. (b) The Clifford-deformed stabilizers. Note that all stabilizers restricted on the plane are of the form XZZX, including those on the seam.

We establish a similar result for the Clifford-deformed 3D surface code, summarized in the following theorem.

Theorem 6.—(lowest-weight Z-only logical). Consider an $L \times (L+1) \times L_z$ Clifford-deformed 3D-rotated surface code with periodic boundary conditions. If $L \equiv 1$ or 2 mod 4, then the lowest-weight logical operator that consists of only I and Pauli-Z operators acts with Z on all horizontal qubits.

This means that the Z-distance d_Z scales as $O\left(L^3\right)$ or, in other words, the logical error rate for pure-Z-biased noise scales as

$$\bar{p} \propto e^{-\alpha(p)n}. \tag{14}$$

We show here that under pure-Z-biased noise, the minimum distance of a rotated Clifford-deformed 3D surface code (with periodic boundary conditions) can scale as $O(L^3)$. Recall that our code has two types of qubits—*horizontal qubits* that live on horizontal planes and *vertical qubits* that live on vertical edges. We can establish the following theorem.

Proof.—As we only consider Z errors, it suffices to work with classical parity-check operators that detect Z errors rather than the full quantum stabilizers, using the maps $I \mapsto 0$, $X \mapsto 1$, $Y \mapsto 1$, and $Z \mapsto 0$. The horizontal parity-check operators are all horizontal squares of the form 0110, while there are two types of vertical parity checks—1111 and 0110 (see Fig. 26). We divide our proof into three distinct parts. In (1), we show that if a Z-logical has support in a horizontal qubit, then it has support in the whole horizontal layer containing this qubit. In (2), we show that if a Z-logical has support in a horizontal layer and $L \equiv 1$ or 2 mod 4, then it has support in all horizontal layers. Finally, we show in (3) that a Z-logical cannot have support uniquely in vertical qubits. It shows that if $L \equiv 1$ or 2 mod 4, the minimum-weight logical is the one with all horizontal qubits activated.

(1) Let $\mathcal{L}$ be a pure-Z logical with a horizontal qubit in its support. We show that $\mathcal{L}$ is supported on the whole horizontal layer containing this qubit. This problem reduces to showing that there is always a weight-2 parity check between every pair of qubits in a chosen layer. Indeed, if that is the case, it means that the parity of every pair of qubits must be 0, which eliminates the possibility of the layer not being entirely composed of 0 or 1. To prove this, we note that every diagonal line in the 2D coprime lattice forms a parity check with a 1 at its boundary, as illustrated in Fig. 27(a). Since the lattice has coprime dimensions, there exists a diagonal line that goes through all the qubits before looping to itself. In particular, this line goes through every pair of qubits, which achieves our proof.

(2) If $L \equiv 1$ or 2 mod 4, this means that one of the directions contains $4k+2$ cells, for some integer $k \geq 0$. Consider a parity-check operator that consists of a product of parity-check operators in

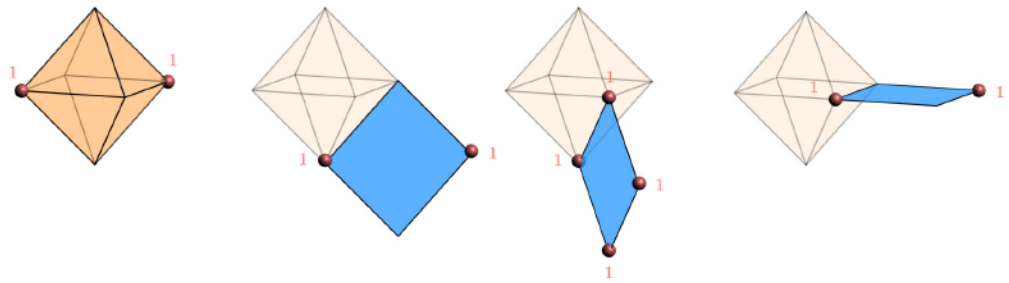

FIG. 26. Stabilizers can be converted into binary parity checks for Z errors.

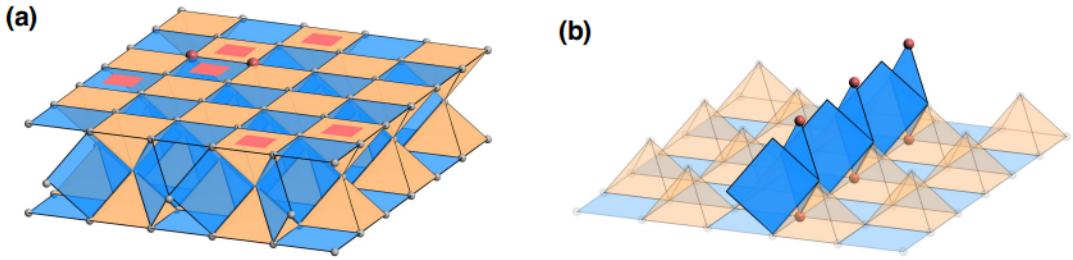

FIG. 27. Proof that the pure-Z logical of our tailored 3D-rotated surface code is supported on all horizontal qubits of the code. (a) A parity check between any two horizontal qubits (red dots) can be constructed by wrapping around the coprime plane along a diagonal (red squares). (b) The product of zigzag parity-check operators along the $4k+2$ periodic direction, which consists of $2k+1$ 0110-type checks and $2k+1$ 1111-type checks.

a zigzag fashion that wraps through the periodic boundaries of the vertical qubit lattice, as shown in Fig. 27(b). Such a chain would be made up of $2k+1$ checks of type 1111 and $2k+1$ checks of type 0110. Their product is an operator that has support on $2k+1$ horizontal qubits on the horizontal layer below and $2k+1$ horizontal qubits on the horizontal layer above, in an identical manner. Recall from (1) that there exists a parity operator that acts on any two horizontal qubits on the same layer. By composing this pairwise parity check between every pair of qubits on a layer, the layers can be removed pairwise, leaving only one remaining qubit on that layer. The exact procedure can be applied to the other layer, resulting in a parity-check operator that acts only on two qubits, one in each layer. This provides a constraint that the Z-only logical must be the same between layers of horizontal qubits; i.e., either all horizontal qubits are I or all horizontal qubits are Z.

(3) We show here that there cannot be a logical operator that has support only on vertical edges. We first note that there is only one logical qubit encoded in the Clifford-deformed 3D-rotated surface code. One pair of anticommuting logical operators has a string of horizontal qubits of the form $\mathrm{XZXZ}\cdots$ and a membrane of horizontal qubits made of Y operators. Since both the string and the membrane logical operators do not act on vertical qubits, they would commute with an only-vertical operator. And since there is only one logical qubit, such an operator cannot be a logical operator.

The operator that acts with Z on all horizontal qubits is a valid logical since it anticommutes with the Y-membrane logical and it commutes with all the stabilizers. We have shown that there cannot be a pure-Z logical of lower weight that acts nontrivially on a horizontal qubit or that is supported only on vertical qubits. Therefore, the minimum-weight Z-only logical is one that acts as Z on all horizontal qubits. ■

Note that the proof fails if the even dimension is not $4k+2$ for some integer k, as the constraint between layers [see (2)] does not apply, in which case the scaling becomes $O(L^2)$ instead.

C. Robustness of the Z-weight scaling

We note that, as in the 2D case [72], the above statement is not robust, in the sense that allowing for a single X in our logical operator drops the effective scaling down from $O(L^3)$ to $O(L^2)$. This can be seen in Fig. 28, where we present an example of a logical operator that has $O(1)$ Pauli-Xs and $\Theta(L^2)$ Pauli-Zs.

However, we show that there is no string logical operator that contains $O(L)$ Zs and $O(1)$ Xs as follows.

Theorem 7.—In a 3D-rotated surface code the dimensions of which satisfy the assumptions of Theorem 6, any logical operator containing $O(1)$ Xs also contains $\Theta(L^2)$ Zs.

Proof.—To prove this theorem, we adopt the following strategy. We first show that if such a logical operator exists, it must belong to the same coset as the logical string operator that comes from the Clifford deformation of an X-string logical operator. This logical operator, which we call the XZ string, by virtue of it having alternating Xs and Zs along its length, is represented in Fig. 28(a). We then derive all the transformations of this string, through the application of stabilizers, which results in $O(1)$ Xs on horizontal qubits. For that, we show that this is equivalent to finding all the solutions of a matching problem on a 2D lattice and we prove that all such solutions necessarily result in creating $\Theta(L^2)$ Zs on horizontal qubits.

Let us start by showing that our string logical operator must be logically equivalent to the XZ string. Since the XZ string is free to move in 3D by application of stabilizer generators, any other string logical operator must commute with at least one of its instances by avoidance. Moreover, the 3D-rotated surface code with the dimensions of Theorem 6 only encodes one qubit. Therefore, any other string logical operator must either be trivial or logically equivalent to the XZ string.

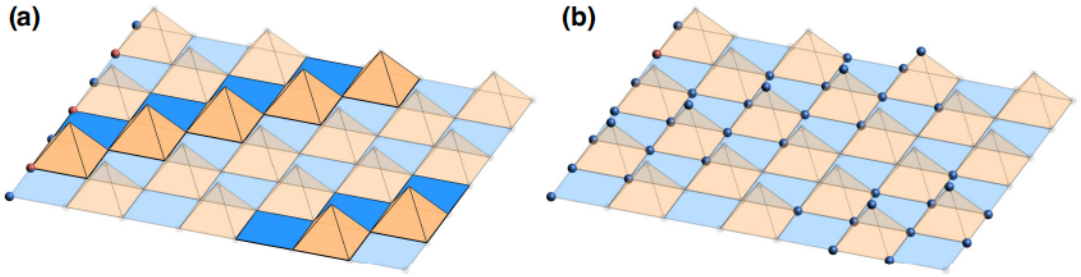

FIG. 28. An example of a membrane logical operator made of $O(1)$ Xs and $\Theta(L^2)$ Zs. (a) The membrane can be obtained by starting from a string logical made of alternating Xs (red vertices) and Zs (blue vertices) and applying stabilizers (highlighted in the figure) along the diagonal lines connecting pairs of Xs. This has the effect of annihilating all the connected pairs of Xs, leaving only one unpaired X in the logical operator and a trail of $\Theta(L^2)$ Zs. (b) The resulting membrane logical operator.

We now show that any logical operator equivalent to the XZ string with $O(1)$ Xs on the horizontal qubits also has $\Theta(L^2)$ Zs on the horizontal qubits. This statement, while restricted to the horizontal qubits, implies the stronger result that requiring $O(1)$ Xs on both the horizontal and vertical qubits leads to the presence of $\Theta(L^2)$ Zs in the operator. Therefore, we ignore the vertical qubits in the rest of the proof.

The goal is now to show that, by applying stabilizers on the XZ string, we can eliminate all Xs except for $O(1)$ of them. To see how Xs can be moved and eliminated, let us focus on the X part of the stabilizers. This corresponds to the parity-check operators of Fig. 26. Since we choose to ignore the vertical qubits, we consider the restriction of these stabilizers to the horizontal qubits. The final restricted operators are all weight-2. We can therefore represent all the horizontal qubits and stabilizers on a 2D lattice, constructed by taking a diagonal slice of the 3D-rotated surface-code lattice that is vertical and runs parallel to the line connecting a pair of Xs on a horizontal square stabilizer. Since the code has coprime dimensions, there is only a single such diagonal slice on which all the horizontal qubits sit. This 2D lattice is represented in Fig. 29(a).

The problem can now be formulated as a matching problem on this 2D lattice. Indeed, since the stabilizers have X-weight 2 restricted on horizontal qubits, the Xs can only be annihilated in pairs, by applying a chain of stabilizers that connects the pair. Since we allow $O(1)$ Xs to remain in the logical operator, the more precise problem is to match all but $O(1)$ Xs. Any logical operators with $O(1)$ Xs can then be seen as a different solution to this matching problem.

The next step is to count how many Zs are created for each matching solution. When applying a horizontal stabilizer, which is either an octahedron stabilizer or a horizontal square stabilizer, two Zs are introduced on the horizontal qubits, located $L_y + 1$ vertices to the left and to the right of the stabilizer as viewed on the 2D diagonal slice. Note that additional Zs are also introduced on the vertical qubits for octahedron stabilizers. An example of a matching solution with its introduced Zs is shown in Fig. 29(b). To count them, we note that any matching solution has an alternation of even- and odd-parity sections, where a section is defined as the space between two original Xs and its parity is defined as the number of horizontal stabilizers applied on each column of the section, modulo two. Since we can choose $O(1)$ Xs that are not matched, this alternating parity pattern breaks at these unmatched Xs, where either two even-parity sections or two odd-parity sections follow one another. This can be seen in the rightmost sections of Fig. 29(b). However, since there are only $O(1)$ unmatched Xs, the number of such breaks in alternation is also $O(1)$.

We can then use this last fact to prove that the number of Zs is $\Theta(L^2)$. Indeed, the number of Zs on a given column of horizontal qubits is, by construction, equal to the number of stabilizers applied $L_y + 1$ columns to the left and to the right. Those two columns of stabilizers are $2L_y - 1$ edges apart and since the size of a section is $2L_y$ edges, they belong in different sections as long the column of Zs is not precisely in the middle of a section. Excluding these $O(L)$ columns, as well as the $O(L)$ columns located L_y edges to the left and to the right of a parity-alternation breaking point, of which there are only $O(1)$, we can see that the number of Zs applied to a given column is equal to the sum of the number of Zs in an odd-parity and in an even-parity section. Therefore, in $\Theta(L^2)$ columns, there is an odd number of Zs, where the number of Zs must be at least 1. The logical operator therefore contains $\Theta(L^2)$ Zs. ■

This theorem shows that under high dephasing bias, with a low number of X errors, the subthreshold error rate scales as $O(e^{-\alpha L^2})$, similarly to the original 3D-rotated surface code. However, the number of logical operators with a low number of Xs is expected to be lower in the Clifford-deformed code and we therefore expect the coefficient α to be higher. This can be seen through the following heuristic argument. In any transformation of the XZ string considered in the proof of Theorem 7, applying a vertical stabilizer creates some Xs on the vertical qubits. Those

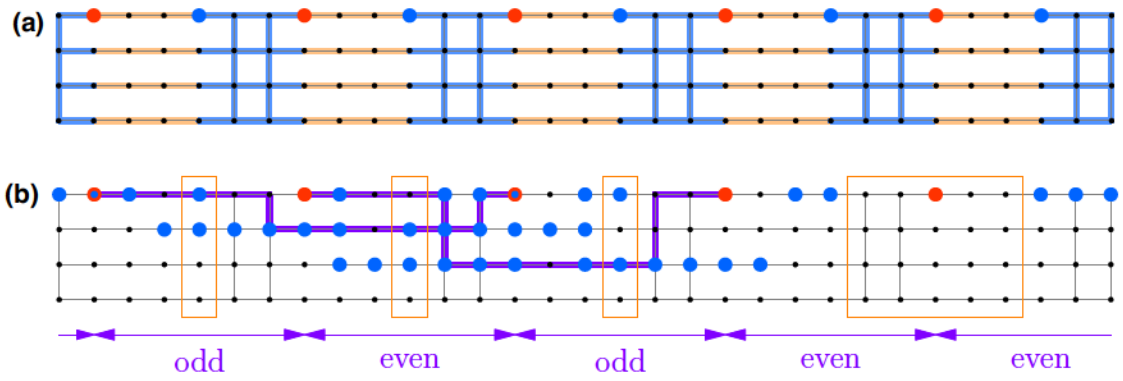

FIG. 29. Proof that there is no string logical made of $O(1)$ Xs and $O(L)$ Zs. (a) All the horizontal qubits can be placed on a long diagonal slice of size $(L_x \cdot L_y, L_z)$. Considering only Xs acting on horizontal qubits, all stabilizers become 2-body terms. The lattice shown is the diagonal slice of a 3D-rotated surface code with dimensions $(10, 3, 4)$. Horizontal qubits are represented as black vertices, octahedron stabilizers as orange edges, and face stabilizers as blue edges. The leftmost and rightmost vertices are identified to represent periodic boundary conditions. The support of the XZ-string logical operator is represented by red (X) and blue (Z) vertices. (b) An example of a matching solution for two pairs of Xs belonging to the XZ-string logical operator. The purple edges represent the stabilizers used in the matching solution. This results in eliminating all but one of the Xs (red), while introducing new Zs (blue) on horizontal qubits located $L_y + 1$ vertices to the left and to the right of every horizontal stabilizer used in the matching solution. Every column of horizontal stabilizers has either an odd or an even number used in the matching solution, which is its parity. All $2L_y - 1$ columns of horizontal stabilizers between any two neighboring Xs have the same parity, as annotated in purple. Adjacent sections have different parities, except for the sections around unmatched Xs, which are instead of equal parity (the two rightmost sections). As a result of this parity alternation, there are an odd number of Zs introduced in every column of horizontal qubits, except in columns of horizontal qubits at the midpoint between neighboring Xs and in the columns of horizontal qubits around unmatched Xs. Since there are only $O(L)$ such exceptions (boxed in orange), we deduce that $\Theta(L^2)$ Zs have been introduced in the process.

Xs cannot be eliminated through the application of other stabilizers, so vertical stabilizers necessarily increase the number of Xs in the operator. In order to keep only one X in our operator, the matching must therefore be performed on the horizontal plane. But for a fixed choice of which X to keep, there are only two possible matching solutions confined to the plane. By moving the remaining X on the plane or choosing different planes, we can deduce that the number of membrane operators with a single X scales as $\Theta(L^3)$. This can be compared to the original 3D-rotated surface code, where the number of such logical operators scales exponentially in the system size. This reasoning can be generalized to logical operators containing $O(1)$ Xs. When more than one X is present, each logical operator is characterized by the planes in which the Xs are supported and the positions of these Xs within these planes. This is due to the confinement property discussed for the case of a single X. The number of such choices still scales polynomially with the system size and hence remains an exponential improvement compared to the original code.

VI. DISCUSSION AND CONCLUSIONS

In this work, we have presented Clifford deformations of many 3D topological codes with high quantum memory threshold error rates for biased Pauli noise. One important question following our study is whether it is always possible to design a Clifford deformation of a topological stabilizer code such that there exists a decoding strategy with a 50% threshold error rate at infinitely biased noise. On the basis of the wide range of examples we have presented in this work, we conjecture that this is true. We have also presented a rotated layout of the surface code for which choosing appropriate dimensions and boundary conditions leads to a subthreshold scaling of $\exp -O(n)$, for infinitely biased noise. We have shown that in the regime of large finite bias, which we model as the presence of $O(1)$ X errors, this subthreshold scaling becomes $\exp -O(L^2)$. It would be interesting to consider how such geometrical optimizations can improve the code performance for other 3D codes such as the 3D color code.

Families of random Clifford-deformed surface codes in two dimensions have been shown to exhibit high threshold error rates and subthreshold scaling better than the XZZX and XY surface codes [8]. The performance of the random codes at infinite bias can be intuitively explained via a mapping to percolation problems. One could consider random Clifford-deformed 3D surface codes and color codes, for which we expect a similar mapping to percolation problems and a phase diagram containing a phase of 50% threshold error rate analogous to the random Clifford-deformed surface codes in 2D. It would be also interesting to study how random Clifford deformations affect the memory performance of fracton codes at infinite bias, which have intrinsically rigid logical operators irrespective of the bias.

A natural next step is to extend the code-capacity results in our work to the phenomenological fault-tolerant scenario as well as the more realistic circuit-level scenario. In the circuit-level scenario, it becomes important to use bias-preserving gates to maintain the performance advantage found for biased noise.

In three dimensions, surface codes have been defined on fractal lattices with Hausdorff dimension $D_H = 2 + \epsilon$ [29,30]. Our Clifford deformation of the 3D surface code naturally applies to such fractal surface codes that can be created by punching holes in the 3D surface code.

The source code for the numerical simulations of the quantum error-correcting codes, noise models, and decoders described in this paper is freely available on a GitHub repository via Ref. [73]. This is the source-code repository for the PanQEC PYTHON package (pronounced "pancake"). The vision for PanQEC is to be a collection of quantum error-correcting codes, noise models, and decoders, which are amenable to numerical simulation and interactive 3D visualization. The documentation for PanQEC is available via Ref. [74], which also includes tutorials on usage. Furthermore, an online demonstration of its 3D-visualization capabilities is available via Ref. [75].

ACKNOWLEDGMENTS

We thank Benjamin Brown for showing how to prove a nontrivial part of Theorem 1, i.e., the decoding of the plaquette syndromes. We thank Steve Flammia for comments, especially for asking whether our rotated layout has the robustness discussed in Sec. V C. We thank Dan Browne, Michael Gullans, Oscar Higgott, Armanda Quintavalle, Joschka Roffe, and George Umbrarescu for comments on the manuscript. A.P. is supported by the Engineering and Physical Sciences Research Council (EPSRC) (EP/S021582/1). E.H. was supported by the Perimeter Scholars International scholarship and the Fulbright Future Scholarship. E.H. acknowledges support from the National Science Foundation (NSF) through Quantum Leap Challenge Institutes (QLCI) Grant No. OMA-2120757. 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. CTC acknowledges support from the Swiss National Science Foundation through the Sinergia grant (CRSII5-186364), the National Centres for Competence in Research in Quantum Science and Technology (QSIT) and The Mathematics of Physics (SwissMAP), and the ETH Zurich Quantum Center. A.D. is supported by the Simons Foundation through the collaboration on Ultra-Quantum Matter (651438, AD) and by the Institute for Quantum Information and Matter, an NSF Physics Frontiers Center (PHY-1733907).

APPENDIX A: PROOF OF 50% THRESHOLD FOR THE 3D SURFACE CODE ON THE CHECKERBOARD LATTICE

We have presented a Clifford deformation of the checkerboard-lattice surface code that consists of applying a Hadamard operation on half of the vertical qubits, in a 3D-checkerboard manner [see Fig. 7(a)]. For this Clifford-deformed code, cube-stabilizer generators are violated under Z errors on the Clifford-deformed edges, while the triangle-stabilizer generators are violated by Z errors on the remaining edges. Using this fact, one can show that the products of cubes and the products of triangles along the diagonals of the code are effectively identity for pure-Z errors. The presence of these linear symmetries gives rise to the following theorem.

Theorem 8.—The Clifford-deformed checkerboard-lattice surface code has a 50% threshold error rate under pure-Z noise.

Proof.—The Clifford deformation we consider for the checkerboard-lattice surface code consists of applying a Hadamard on half of the vertical qubits, in a 3D-checkerboard fashion [see Fig. 7(a)]. In this new code, under pure-Z noise, cube stabilizers can only be excited by errors acting on the Clifford-deformed qubits, while triangle stabilizers can be excited by any of the remaining qubits. We now show that this code has a 50% threshold error rate under pure-Z noise.

We start by decoding the cube stabilizers. Due to the Clifford deformation, these stabilizers are now effectively weight-2 at infinite bias and are involved in some linear symmetries represented in Fig. 30(a). We can therefore decode the cubes by performing matching along each symmetry line.

We then tackle the triangle stabilizers. To decode them, we can perform matching along the two linear symmetries represented in Figs. 30(b) and 30(c). One symmetry allows us to decode all the remaining vertical qubits and the other all the horizontal qubits.

Since all the steps involve decoding a polynomial number of repetition codes (in the lattice size L) and the probability of success is lower bounded by the probability of correctly decoding all these repetition codes, it shows that this decoding strategy leads to 50% threshold error rate. ■

APPENDIX B: DECODERS

1. The BP OSD

The BP OSD is a generic decoder for quantum LDPC codes. Based on a classical technique to improve the iterative decoding of linear codes [76,77], it was introduced to the quantum domain by Panteleev and Kalachev [49] and has been shown to have high performance on a large class of LDPC codes, including topological codes [21,50].

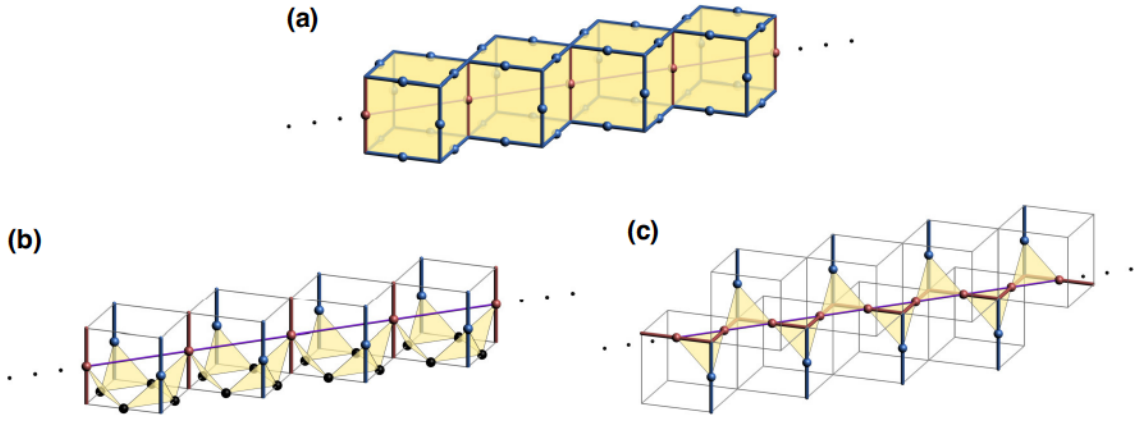

FIG. 30. The linear symmetries of the checkerboard-lattice surface code. (a) The symmetry of the cube stabilizers. Due to the Clifford deformation, the cube stabilizers become weight-2 and are supported on two diagonally opposite edges (red). Multiplying cubes diagonally effectively gives the identity. Matching along this line allows us to decode all the Clifford-deformed vertical qubits. (b) The symmetry of the triangle stabilizers involving vertical qubits only. The product of four triangle stabilizers within a cube gives a four-body stabilizer supported on the four vertical qubits of the cube (blue and red). Due to the Clifford deformation, this becomes a two-body term, supported on two diagonally opposite qubits (blue). Multiplying these weight-2 stabilizers on a diagonal line (purple) gives the identity. Matching along all such lines allows us to decode all the undeformed vertical qubits. (c) The symmetry of the triangle stabilizers involving horizontal qubits only. The triangles containing a Clifford-deformed qubit become two-body terms after applying the Clifford deformation. Multiplying them along a line (purple) gives the identity. All the other horizontal qubits of the cube are involved in a similar symmetry. Matching along all these symmetries allows us to decode all the horizontal qubits of the code.

The BP OSD is built from two components: the belief-propagation (BP) algorithm, which estimates the probability for each qubit to have an error, and the ordered statistics decoder (OSD), which takes these probabilities as input and proposes a correction that fits the syndrome. It is particularly well adapted to the decoding of Clifford-deformed codes under biased noise, as it naturally takes into account the nonuniform probability of errors along the different axes in the Clifford-deformed noise model.

a. Belief propagation

The belief-propagation decoder is one of the most commonly used decoders for classical LDPC codes [78]. It is an inference algorithm that computes an approximation of the probabilities $P(e_i|\mathbf{s})$ that an error has occurred on each bit i given a syndrome $\mathbf{s}$. A correction operator is then applied to all the bits i such that $P(e_i|\mathbf{s}) > 0.5$. While computing this marginal probability involves, in principle, summing over an exponential number of terms, belief propagation exploits the fact that for LDPC codes, this sum can be factored into a small number of terms. It then uses an algorithm called the *product-sum algorithm* (or its variant, the *min-sum algorithm*) to calculate this sum, by iteratively passing messages between parity checks and data bits. Belief propagation can be shown to converge to the exact marginal distribution when the Tanner graph is a tree. For more general Tanner graphs that can contain loops, it is used as a heuristic algorithm to approximate the distribution and it is sometimes called *loopy belief propagation*. While the approximation is often acceptable when the *girth* [79] of the graph is large, the presence of short cycles tends to be detrimental to the performance of the BP [78,80].

Several methods have been proposed in the literature to generalize belief propagation to quantum codes [50,81–86]. For instance, one can decode X and Z errors separately using the classical version of the BP. The potential correlations between X and Z errors can be taken into account by first decoding X errors, adjusting the channel probabilities based on the correction, and decoding Z errors with this adjusted probability, as proposed in Ref. [84]. It has also been proposed to send vector instead of scalar messages, to compute the probability $P(e_i = W|\mathbf{s})$ that a Pauli error $W \in \{I, X, Y, Z\}$ has occurred on each qubit i [82]. However, this results in an increase in complexity compared to the original BP algorithm and sometimes reduced performance due to the presence of shorter cycles in the whole Tanner graph compared to the X and Z ones. A simplified message-passing rule has been proposed in Ref. [87] to reduce this complexity while guaranteeing the same output as the original version but the presence of short cycles is still hindering its performance. In this work, we have chosen to decode X and Z errors separately.

Apart from the presence of short cycles in quantum Tanner graphs, a major problem with BP decoding of quantum codes is the *degeneracy problem*, also called the *split-belief* phenomenon [82]. Indeed, in quantum codes, a syndrome can often be generated by several equally likely combinations of errors. By symmetry, the BP algorithm

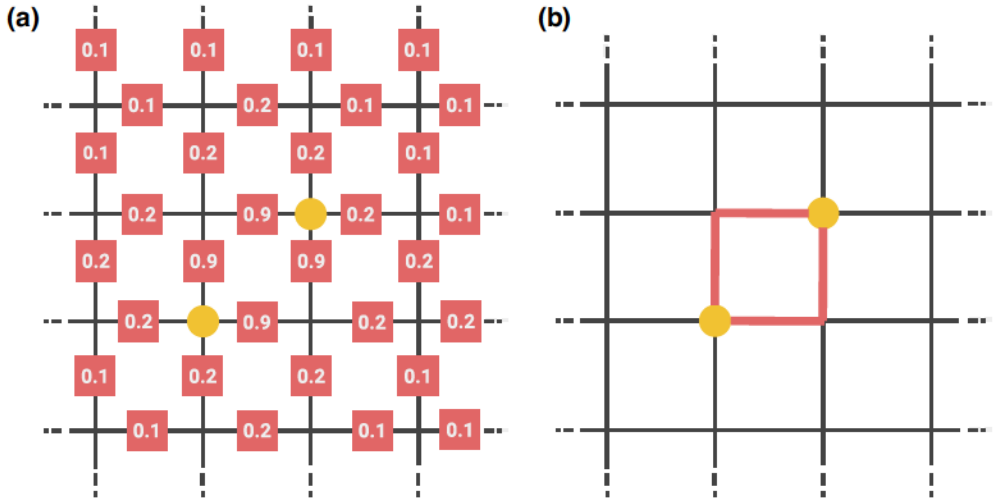

FIG. 31. An illustration of the BP-OSD decoder on a 2D surface code. (a) The belief-propagation algorithm takes as input a syndrome **s** (yellow dots) and an error model and computes a probability $P(e_i|\mathbf{s})$ of error on each individual qubit (red squares). When the solution is degenerate (e.g., two chains of errors have minimal weight), a high probability (0.9 shown in the figure) is assigned to all the solutions. (b) The split-belief phenomenon. In the basic version of belief propagation, a correction operator (red edges) is applied to all qubits i having $P(e_i|\mathbf{s}) > 0.5$. In degenerate cases, it can produce an invalid correction, with some defects remaining. The OSD algorithm consists of solving the parity-check equation $\mathbf{He} = \mathbf{s}$ for the most probable set of errors, as found by belief propagation, guaranteeing that the final corrected state lives in the code space.

outputs the same probability for all these errors and if they are all higher than 0.5, it applies a correction operator to all these equally likely errors, resulting in an invalid correction that does not fit the syndrome. An illustration of a split-belief problem is shown in Fig. 31.

To mitigate the degeneracy problem, several solutions have been proposed in the literature, such as: breaking the degeneracy with random noise [82]; adjusting the error probabilities when the decoder fails [86]; using a neural network to learn the BP procedure, with a loss function tailored to avoid degeneracies [88]; using previous messages in the message-passing update rule [85]; or complementing the BP decoder with a second decoder, such as the ordered statistics decoder (OSD) [50]. Since the OSD has recently been shown to outperform other methods for many different codes [49], we are using this solution in our work.

b. Ordered statistics decoding

For any classical linear code with a parity-check matrix **H**, the following equation, called the *syndrome equation*, holds:

$$\mathbf{He} = \mathbf{s}. \tag{B1}$$

Since many errors can correspond to a given syndrome, **H** is not directly invertible. The idea of the OSD is to only solve the system for the most likely errors, as given by the BP algorithm. More precisely, we sort the columns of **H** by increasing probability of error and eliminate them one by one in that order until the system is full rank. We then solve the reduced system to find a set of errors that respect the syndrome equation. The remaining qubits can either be set to have no error or be searched over for a better correction using some heuristics [49,50].

In the quantum setting, a similar syndrome equation holds, where the parity-check matrix and the error vector can either be written over the field $GF(4)$ or in a symplectic form. For CSS codes, we can also consider X and Z errors separately and write a syndrome equation for each of them:

$$\mathbf{H_Z e_X} = \mathbf{s_Z}, \tag{B2a}$$

$$\mathbf{H_X e_Z} = \mathbf{s_X}. \tag{B2b}$$

A classical OSD algorithm can then be applied separately for each equation.

In this work, we use the BP OSD through the PYTHON library BPOSD developed by Roffe [89]. In particular, we use the min-sum algorithm for BP and the *combination sweep strategy* (to order 50) described in Ref. [50] to search over the remaining errors in the OSD.

c. Limitations of the BP OSD

While the OSD turns the output of the BP into a valid correction, the algorithm still suffers from the main drawbacks of loopy belief propagation as discussed, such as short cycles and error degeneracies. For instance, the effect of short cycles can be seen when decoding long strings on the 2D surface code. When the size of a string is higher

than 8 (the girth of the surface code when considering X and Z errors separately), short cycles tend to deteriorate the messages passed between the two distant defects. This phenomenon, called *bounded information spread*, has been documented in the literature for the 2D surface and color codes [22]. As a result, decoding topological codes of large sizes is often harder for the BP OSD than for small sizes. This can be observed in some threshold-error-rate plots where the apparent threshold error rate seems to shrink when increasing the system size. Therefore, finite-size approximations of the BP-OSD threshold error rate might not reflect the true threshold error rate, obtained when taking the system size to infinity. Examples of this finite-size effect on 3D codes are given in Sec. IV.

Apart from short cycles, error degeneracies can also have a negative effect on the BP OSD. In general, split-belief problems appear around stabilizers with even weight. Indeed, any error supported on half of an even-weight stabilizer gives the same syndrome after the application of the stabilizer, while the new error has the same weight. This argument has been used to explain why the BP OSD performs poorly on the 2D surface code while performing well on the 3D surface code, observing that the smallest split-belief appears for weight-2 errors on the 2D surface code but on weight-3 errors on the loop sector of the 3D surface code [22]. We call the size of the smallest error that causes a split-belief the *split-belief number* of the code. It can be calculated by taking the smallest even-weight stabilizer and dividing by two. Examples of split-belief numbers for different 3D codes are given in Table IV.

2. Sweep-matching decoder

The sweep-matching decoder on the CSS 3D surface code uses MWPM [43,90,91] between vertices to correct for pointlike syndromes and the sweep decoder [19,20] over face syndromes to correct for stringlike syndromes. The two sectors are decoded independently and the procedure can be generalized for the Clifford-deformed code by using the Clifford-deformed stabilizer generators instead.

Implementations of MWPM are easily applicable in 3D for the Clifford-deformed code and for biased noise by adjusting the graph weights of the match to match the known noise parameters. In this work, the PYTHON library PyMatching [92,93] is used for fast MWPM.

The sweep decoder is a local cellular-automaton decoder, meaning that it is an iterative algorithm where, at each step, a correction operator is computed locally according to the current syndrome using a cellular-automaton rule. To be of use, such a decoder should be able to eliminate every syndrome after a number of steps that is polynomial in the size of the code.

The sweep decoder is based on a cellular automaton called the sweep rule [19]. We now briefly review the sweep rule in the special case of the simple cubic lattice. We start by choosing a spatial direction, defined by a 3D vector $\vec{v}$, called the sweep direction, with the only condition that it is not parallel to an edge of the lattice. In practice, we choose the sweep direction from one of eight possibilities $(\pm 1, \pm 1, \pm 1)$. As illustrated in Fig. 32, we then apply the following rule at each iteration, simultaneously for all the vertices:

1. Find the three oriented lattice edges, $\vec{e}_1$, $\vec{e}_2$, and $\vec{e}_3$, pointing away from the vertex and in the same direction as $\vec{v}$, i.e., such that $\vec{e}_i \cdot \vec{v} > 0$. Each pair of edges corresponds to a face of the lattice.
2. If two of these faces are excited, then apply a Z operator to the intersecting edge. If all three faces are excited, then apply a Z operator to a random edge among $\vec{e}_1$, $\vec{e}_2$, and $\vec{e}_3$. Otherwise, do nothing.

In the sweep decoder, we apply the sweep rule $T_{\max} = O(L)$ times, where L is the linear lattice size. For lattices with boundaries, we run the decoder multiple times using different sweep directions, as described in Ref. [20]. The sweep decoder can fail in two ways: either if the product of the original error and the operators applied by the sweep rule is a nontrivial logical operator or if the syndrome is nontrivial after $T_{\max}$ applications of the rule.

We have implemented the sweep-matching decoder and simulated its performance for 3D surface codes defined on cubic lattices, with and without boundaries. The code is available on line [94].

APPENDIX C: NUMERICAL-SIMULATION DETAILS

To compute the threshold error rate of the different codes, we have simulated up to $n_{\text{trials}} = 10\,000$ for each physical error rate p and for each lattice size L. Values of p were taken in intervals between 0 and 0.5, with a maximum step size of 0.01, while lattice sizes were chosen to be greater than $L = 9$ and up to $L = 21$, using at least three values of L for each p. We have extracted the threshold error rate from crossover plots using a common finite-size scaling regression analysis [43,95,96]. The logical error simulation data is fitted to the following ansatz for the logical error rate $p_L(p, L)$ as a function of the physical error rate p and system size L:

$$p_L = A + Bx + Cx^2, \tag{C1}$$

$$x = (p - p_{\text{th}})L^{1/\nu}, \tag{C2}$$

where p_{th} is the threshold error rate that we seek to evaluate, ν is a critical exponent, and A, B, and C are coefficients of the quadratic ansatz, all of which are free parameters to be determined by fitting to the data. Here, x is termed the *rescaled physical error rate*, which is zero at the phase

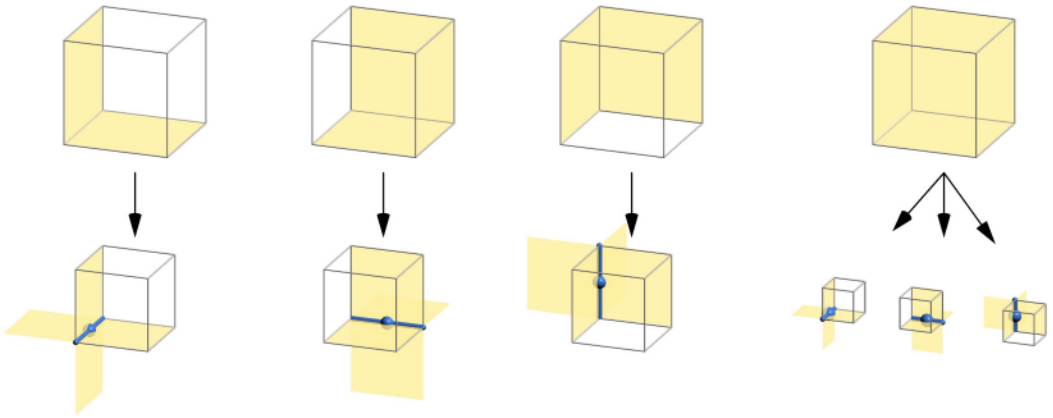

FIG. 32. The greedy sweep rule as applied to a single cell. The top figures enumerate the four possible nontrivial initial syndrome configurations on the three faces of a cell adjoining the vertex that is furthest from the sweep direction, where faces with nontrivial syndromes are shaded yellow. The corresponding corrections are shown in the figures below, where the Z edge operator to be applied as a correction is shown in blue and the corresponding syndromes to be flipped and updated are shaded yellow. In the rightmost initial configuration, where nontrivial syndromes are on all three faces, the correction to apply is chosen randomly out of the three possible corrections. This rule is greedily applied to all cells at once and repeated $T_{\max}$ times.

transition $p = p_{\text{th}}$. That p_L is a quadratic function of x is only expected to be a valid approximation near this phase transition for x, so only data points with physical error rates close to the phase transition were used for the fitting.

For each given physical error rate p and system size L, suppose that n_{fail} trials out of n_{trials} trials result in a logical error after running the numerical simulations of sampling the noise model, syndrome extraction, and decoding. The logical error rate can then be estimated by $p_L = n_{\text{fail}}/n_{\text{trials}}$.

Using these estimated logical error rates, we have run an optimization procedure to obtain the set of free parameters $(p_{\text{th}}, \nu, A, B, C)$ that fits the data the best, as measured by minimizing the mean-squared error.

Uncertainties for the threshold-error-rate estimate p_{th} have been calculated using the following bootstrap resampling method.

The first step is to obtain a distribution to sample for estimates of the logical error rate p_L for each (p, L), to account for the finite number of trials n_{trials}. Starting from a uniform prior distribution before taking into account the number of trials and failures, the posterior distribution for p_L is a beta distribution with

$$p_L \sim \text{Beta}\left(n_{\text{trials}} - n_{\text{fail}} + 1, n_{\text{fail}} + 1\right), \tag{C3}$$

where $\text{Beta}(a, b)$ is a probability distribution with support over the interval $[0, 1]$ and probability density function,

$$f\left(x; a, b\right) = \frac{\Gamma\left(a + b\right) x^{a-1} (1 - x)^{b-1}}{\Gamma\left(a\right) \Gamma\left(b\right)} \tag{C4}$$

for real parameters a and b. Here, Γ is the gamma function, with the property that $\Gamma(n) = (n - 1)!$. Thus the logical error rates $p_L(p, L)$ for each p and L can be sampled independently from these posterior distributions to produce a resampled set of p_L values to use for fitting. The second

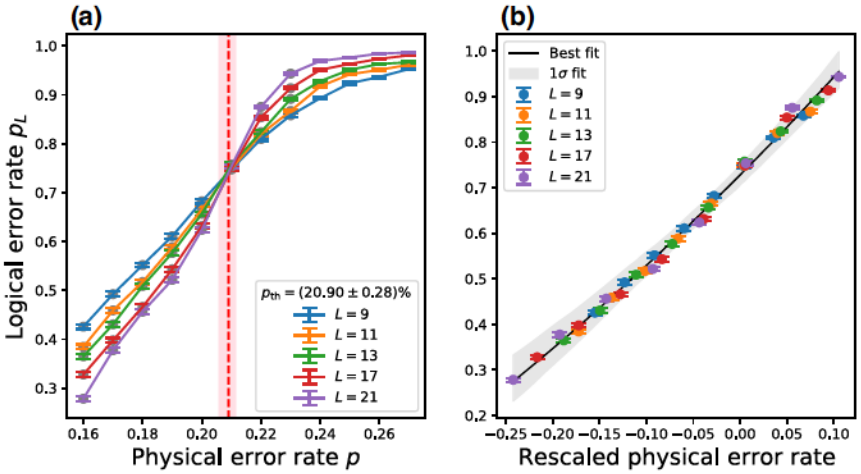

FIG. 33. (a) The total logical error rate p_L versus the physical error rate p for Z-biased noise with $\eta = 30$ decoded with the sweep-matching decoder. (b) Data collapse onto the quadratic finite-size-scaling ansatz with bootstrapped error bars and 1σ fitted bounds. Clifford-deformed surface code on cubic lattice, $\eta_Z = 30$, sweep matching, all errors.

step is to reflect the uncertainty that arises from the choice of data points, which may be done by resampling with replacement the set of (p, L) pairs to use.

With these two sources of uncertainty accounted for by resampling $n_{\text{bs}} = 100$ times, the least-squares fit of $p_L(p, L)$ can be done repeatedly on the resampled data to produce a set of n_{bs} best-fit parameters $\{(p_{\text{th}}, \nu, A, B, C)\}$ for each resampling. This collection of best-fit parameters can be used to produce error bars on the threshold error rate p_{th} by taking the 1σ bounds of the resampled threshold-error-rate estimates to be interpreted as a credible interval. Note that these uncertainty bounds need not be symmetrical in upper and lower directions, as seen in Tables I–III. An example of this fitting is shown in Fig. 33, with the bootstrapped uncertainty estimates shown in pink and the best-fit value of p_{th} marked with the red dashed vertical line. The validity of the ansatz may be vindicated by visual inspection of the so-called data-collapse plot of the logical error rate p_L over the rescaled physical error rate $x = (p - p_{\text{th}})L^{1/\nu}$, where all data points collapse onto the fit line per the quadratic ansatz in Eq. (C1) to within reasonable bounds, as quantified by the 1σ envelope above and below the fit line. This ensures that the data points chosen for the finite-size scaling have been chosen sufficiently close to the critical point such that the ansatz is valid.

To verify the reliability of the estimated threshold error rates, the average X and Z logical failure rates over every logical qubit have been used as the logical error rates and subjected to the same analysis to extract corresponding threshold error rates. This is important since, due to finite-size effects, the apparent threshold error rate as determined by the total logical error rate may be higher than the threshold error rates determined by the logical X and Z error rates. For the case of the X-cube model, where the number of logical operators increases with the code distance, the logical X error rate is determined by taking the average logical X error rate over all logical qubits. The logical Z error rate has been calculated analogously.

We have used this procedure to compute the threshold error rate of both the CSS and Clifford-deformed codes, for bias ratios $\eta_Z \in \{0.5, 1, 3, 10, 30, 100, \infty\}$, sampling more where interesting features are to be elucidated. Representative examples of crossover plots of the logical error rate over the physical error rate along with the ansatz fitting for both X and Z logical errors are given separately in Fig. 34.

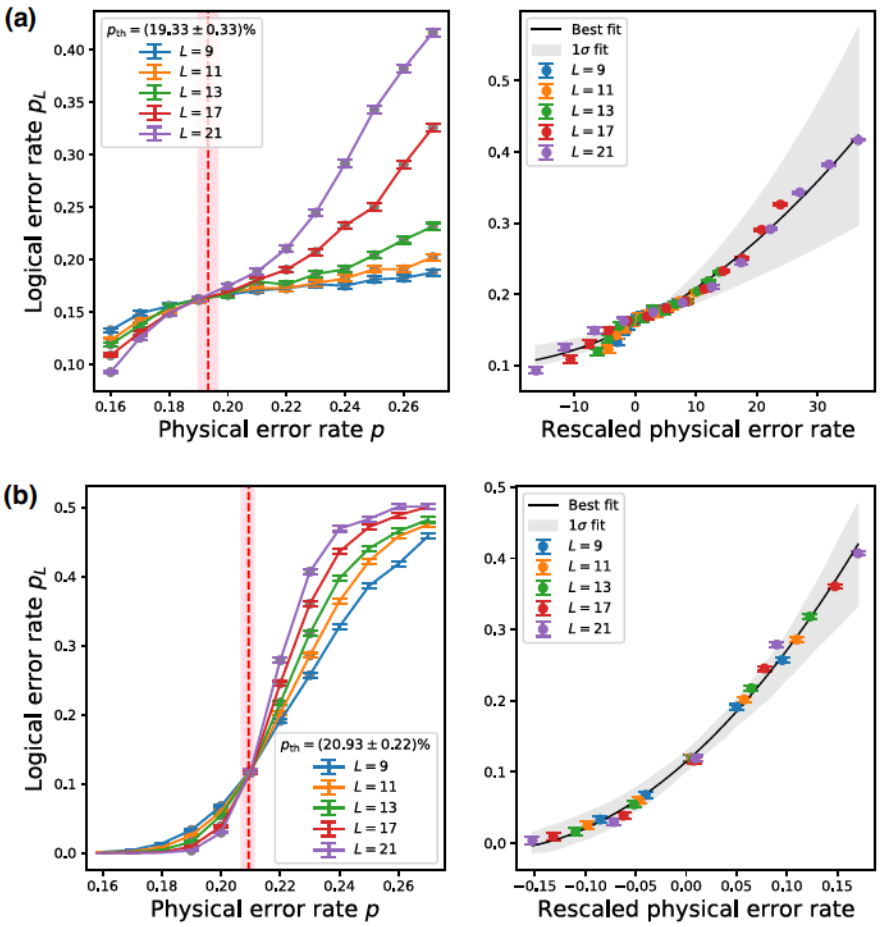

FIG. 34. Examples of data-collapse plots in terms of logical (a) X and (b) Z errors for the 3D surface code under Z-biased noise, with $\eta_Z = 30$ and decoding by sweep matching. Note that these are the same parameters as in Fig. 33 but that the minimum threshold error rate is slightly lower than that estimated using the total error rate in Fig. 33 and is thus a more conservative estimate.

The above procedure produces best-fit estimates and credible intervals for the threshold error rate p_{th} with respect to the total logical error rate, logical X errors, and logical Z errors, which may differ significantly. To be conservative, the reported threshold error rate is the minimum of these estimates, as determined by which 1σ (68% equal-tail) credible interval has the lowest lower bound.

[1] P. Aliferis and J. Preskill, Fault-tolerant quantum computation against biased noise, Phys. Rev. A **78**, 052331 (2008).

[2] D. K. Tuckett, S. D. Bartlett, and S. T. Flammia, Ultrahigh Error Threshold for Surface Codes with Biased Noise, Phys. Rev. Lett. **120**, 050505 (2018).

[3] 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).

[4] D. Nigg, M. Muller, 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).

[5] G. Burkard, T. D. Ladd, J. M. Nichol, A. Pan, and J. R. Petta, Semiconductor spin qubits (2021). ArXiv:2112.08863.

[6] S. Puri, L. St-Jean, J. A. Gross, A. Grimm, N. E. Frattini, P. S. Iyer, A. Krishna, S. Touzard, L. Jiang, and A. Blais, *et al.*, Bias-preserving gates with stabilized cat qubits, Sci. Adv. **6**, eaay5901 (2020).

[7] J. Pablo Bonilla Ataides, D. K. Tuckett, S. D. Bartlett, S. T. Flammia, and B. J. Brown, The XZZX surface code, Nat. Commun. **12**, 2172 (2021).

[8] A. Dua, A. Kubica, L. Jiang, S. T. Flammia, and M. J. Gullans, Clifford-deformed surface codes (2022).

[9] K. Tiurev, P.-J. H. S. Derks, J. Roffe, J. Eisert, and J.-M. Reiner, Correcting non-independent and non-identically distributed errors with surface codes (2022). ArXiv:2208.02191.

[10] B. Srivastava, A. Frisk Kockum, and M. Granath, The XYZ^2 hexagonal stabilizer code (2021). ArXiv:2112.06036.

[11] J. F. San Miguel, D. J. Williamson, and B. J. Brown, A cellular automaton decoder for a noise-bias tailored color code (2022).

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

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

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

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

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

[17] N. P. Breuckmann, K. Duivenvoorden, D. Michels, and B. M. Terhal, Local decoders for the 2D and 4D toric code, Quantum Inf. Comput. **17**, 181 (2017).

[18] K. Duivenvoorden, N. P. Breuckmann, and B. M. Terhal, Renormalization group decoder for a four-dimensional toric code, IEEE Trans. Inform. Theory **65**, 2545 (2019).

[19] A. Kubica and J. Preskill, Cellular-Automaton Decoders with Provable Thresholds for Topological Codes, Phys. Rev. Lett. **123**, 020501 (2019).

[20] M. Vasmer, D. E. Browne, and A. Kubica, Cellular automaton decoders for topological quantum codes with noisy measurements and beyond, Sci. Rep. **11**, 2027 (2021).

[21] 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).

[22] O. Higgott and N. P. Breuckmann, Improved single-shot decoding of higher dimensional hypergraph product codes (2022). ArXiv:2206.03122.

[23] H. Bombín, Single-Shot Fault-Tolerant Quantum Error Correction, Phys. Rev. X **5**, 031043 (2015).

[24] B. J. Brown, N. H. Nickerson, and D. E. Browne, Fault-tolerant error correction with the gauge color code, Nat. Commun. **7**, 12302 (2016).

[25] A. Kubica and M. Vasmer, Single-shot quantum error correction with the three-dimensional subsystem toric code (2021). ArXiv:2106.02621.

[26] This should be contrasted with 2D topological codes, where, in order to ensure fault tolerance, the noisy error syndrome must be measured d times for a distance-d code.

[27] H. Bombín, Resilience to Time-Correlated Noise in Quantum Computation, Phys. Rev. X **6**, 041034 (2016).

[28] B. J. Brown and D. J. Williamson, Parallelized quantum error correction with fracton topological codes, Phys. Rev. Res. **2**, 013303 (2020).

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

[30] A. Dua, T. Jochym-O'Connor, and G. Zhu, Quantum error correction with fractal topological codes (2022) ArXiv:2201.03568.

[31] Z. Cai, A. Siegel, and S. Benjamin, Looped pipelines enabling effective 3d qubit lattices in a strictly 2D device (2022). ArXiv:2203.13123.

[32] B. Buonacorsi, Z. Cai, E. B. Ramirez, K. S. Willick, S. M. Walker, J. Li, B. D. Shaw, X. Xu, S. C. Benjamin, and J. Baugh, Network architecture for a topological quantum computer in silicon, Quantum Sci. Technol. **4**, 025003 (2019).

[33] M. Akhtar, F. Bonus, F. R. Lebrun-Gallagher, N. I. Johnson, M. Siegele-Brown, S. Hong, S. J. Hile, S. A. Kulmiya, S. Weidt, and W. K. Hensinger, A high-fidelity quantum matter-link between ion-trap microchip modules (2022). ArXiv:2203.14062.

[34] D. Bluvstein, H. Levine, G. Semeghini, T. T. Wang, S. Ebadi, M. Kalinowski, A. Keesling, N. Maskara, H. Pichler, M. Greiner, V. Vuletić, and M. D. Lukin, A quantum processor based on coherent transport of entangled atom arrays, Nature **604**, 451 (2022).

[35] J. L. Mallek, D.-R. W. Yost, D. Rosenberg, J. L. Yoder, G. Calusine, M. Cook, R. Das, A. Day, E. Golden, D. K. Kim,

J. Knecht, B. M. Niedzielski, M. Schwartz, A. Sevi, C. Stull, W. Woods, A. J. Kerman, and W. D. Oliver, Fabrication of superconducting through-silicon vias (2021). ArXiv:2103.08536.

[36] D. Rosenberg, D. Kim, R. Das, D. Yost, S. Gustavsson, D. Hover, P. Krantz, A. Melville, L. Racz, and G. O. Samach, *et al.*, 3D integrated superconducting qubits, npj Quantum Inf. **3**, 42 (2017).

[37] J. Chow, O. Dial, and J. Gambetta, IBM Quantum breaks the 100-qubit processor barrier. https://research.ibm.com/blog/127-qubit-quantum-processor-eagle (2021).

[38] S. Bartolucci, P. Birchall, H. Bombín, H. Cable, C. Dawson, M. Gimeno-Segovia, E. Johnston, K. Kieling, N. Nickerson, M. Pant, F. Pastawski, T. Rudolph, and C. Sparrow, Fusion-based quantum computation (2021). ArXiv:2101.09310.

[39] H. Bombín, I. H. Kim, D. Litinski, N. Nickerson, M. Pant, F. Pastawski, S. Roberts, and T. Rudolph, Interleaving: Modular architectures for fault-tolerant photonic quantum computing (2021). ArXiv:2103.08612.

[40] J. Eli Bourassa, R. N. Alexander, M. Vasmer, A. Patil, I. Tzitrin, T. Matsuura, D. Su, B. Q. Baragiola, S. Guha, G. Dauphinais, K. K. Sabapathy, N. C. Menicucci, and I. Dhand, Blueprint for a Scalable Photonic Fault-Tolerant Quantum Computer, Quantum **5**, 392 (2021).

[41] I. Tzitrin, T. Matsuura, R. N. Alexander, G. Dauphinais, J. Eli Bourassa, K. K. Sabapathy, N. C. Menicucci, and I. Dhand, Fault-Tolerant Quantum Computation with Static Linear Optics, PRX Quantum **2**, 040353 (2021).

[42] H. Bombín, 2D quantum computation with 3D topological codes (2018).

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

[44] H. Bombín and M. A. Martin-Delgado, Exact topological quantum order in $D = 3$ and beyond: Branyons and brane-net condensates, Phys. Rev. B **75**, 075103 (2007).

[45] S. Vijay, J. Haah, and L. Fu, Fracton topological order, generalized lattice gauge theory, and duality, Phys. Rev. B **94**, 235157 (2016).

[46] B. Yoshida, Exotic topological order in fractal spin liquids, Phys. Rev. B **88**, 125122 (2013).

[47] J. Haah, Local stabilizer codes in three dimensions without string logical operators, Phys. Rev. A **83**, 042330 (2011).

[48] B. J. Brown, Conservation laws and quantum error correction: Towards a generalised matching decoder (2022). ArXiv:2207.06428.

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

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

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

[52] D. K. Tuckett, A. S. Darmawan, C. T. Chubb, S. Bravyi, S. D. Bartlett, and S. T. Flammia, Tailoring Surface Codes for Highly Biased Noise, Phys. Rev. X **9**, 041031 (2019).

[53] 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).

[54] J. Fern, J. Kempe, S. N. Simic, and S. Sastry, Generalized performance of concatenated quantum codes—a dynamical systems approach, IEEE Trans. Automat. Contr. **51**, 448 (2006).

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

[56] E. Huang, A. C. Doherty, and S. Flammia, Performance of quantum error correction with coherent errors, Phys. Rev. A **99**, 022313 (2019).

[57] F. Venn, J. Behrends, and B. Béri, Coherent error threshold for surface codes from majorana delocalization (2022).

[58] J. Roffe, L. Z. Cohen, A. O. Quintivalle, D. Chandra, and E. T. Campbell, Bias-tailored quantum LDPC codes (2022).

[59] B. J. Brown, A fault-tolerant non-Clifford gate for the surface code in two dimensions, Sci. Adv. **6**, eaay4929 (2020).

[60] M. Vasmer and A. Kubica, Morphing Quantum Codes, PRX Quantum **3**, 030319 (2022).

[61] Type-I fracton models have string logical operators while type-II do not. Fractal type-I fracton models have fractal-shaped rigid logical operators.

[62] W. Shirley, K. Slagle, Z. Wang, and X. Chen, Fracton Models on General Three-Dimensional Manifolds, Phys. Rev. X **8**, 031051 (2018).

[63] S. Vijay, J. Haah, and L. Fu, Fracton topological order, generalized lattice gauge theory, and duality, Phys. Rev. B **94**, 235157 (2016).

[64] H. Song, J. Schönmeier-Kromer, K. Liu, O. Viyuela, L. Pollet, and M. A. Martin-Delgado, Optimal thresholds for fracton codes and random spin models with subsystem symmetry (2021).

[65] C. Castelnovo and C. Chamon, Topological quantum glassiness, Philos. Mag. **92**, 304 (2012).

[66] B. Yoshida, Exotic topological order in fractal spin liquids, Phys. Rev. B **88**, 125122 (2013).

[67] A. Dua, I. H. Kim, M. Cheng, and D. J. Williamson, Sorting topological stabilizer models in three dimensions, Phys. Rev. B **100**, 155137 (2019).

[68] A. Dua, P. Sarkar, D. J. Williamson, and M. Cheng, Bifurcating entanglement-renormalization group flows of fracton stabilizer models, Phys. Rev. Res. **2**, 033021 (2020).

[69] K. Takeda, T. Sasamoto, and H. Nishimori, Exact location of the multicritical point for finite-dimensional spin glasses: A conjecture, J. Phys. A: Math. Gen. **38**, 3751 (2005).

[70] G. M. Nixon and B. J. Brown, Correcting spanning errors with a fractal code, IEEE Trans. Inf. Theory **67**, 4504 (2021).

[71] Here, we use the fact that $\mathrm{lcm}(\mathrm{lcm}(a,b),c) = \mathrm{lcm}(a,b,c)$.

[72] O. Higgott, T. C. Bohdanowicz, A. Kubica, S. T. Flammia, and E. T. Campbell, Fragile boundaries of tailored surface codes and improved decoding of circuit-level noise (2022). ArXiv:2203.04948.

[73] https://github.com/panqec/panqec

[74] https://panqec.readthedocs.io

[75] https://gui.quantumcodes.io

[76] M. P. C. Fossorier and S. Lin, Soft-decision decoding of linear block codes based on ordered statistics, IEEE Trans. Inf. Theory **41**, 1379 (1995).

[77] M. P. C. Fossorier, Iterative reliability-based decoding of low-density parity check codes, IEEE J. Sel. Areas Commun. **19**, 908 (2001).

[78] D. J. C. MacKay, *Information Theory, Inference and Learning Algorithms* (Cambridge University Press, 2003).
[79] The girth of a graph is the size of its shortest cycle.
[80] N. Raveendran and B. Vasić, Trapping Sets of Quantum LDPC Codes, Quantum **5**, 562 (2021).
[81] D. Poulin, Optimal and efficient decoding of concatenated quantum block codes, Phys. Rev. A **74**, 052333 (2006).
[82] D. Poulin and Y. Chung, On the iterative decoding of sparse quantum codes, Quantum Inf. Comput. **8**, 987 (2008).
[83] Z. Babar, P. Botsinis, D. Alanis, S. Xin Ng, and L. Hanzo, Fifteen years of quantum LDPC coding and improved decoding strategies, IEEE Access **3**, 2492 (2015).
[84] A. Rigby, J. C. Olivier, and P. Jarvis, Modified belief propagation decoders for quantum low-density parity-check codes, Phys. Rev. A **100**, 012330 (2019).
[85] K.-Y. Kuo and C.-Y. Lai, Exploiting degeneracy in belief propagation decoding of quantum codes (2021). ArXiv:2104.13659.
[86] Y.-J. Wang, B. C. Sanders, B.-M. Bai, and X.-M. Wang, Enhanced feedback iterative decoding of sparse quantum codes, IEEE Trans. Inf. Theory **58**, 1231 (2012).
[87] K.-Y. Kuo and C.-Y. Lai, Refined belief-propagation decoding of quantum codes with scalar messages. In 2020 IEEE Globecom Workshops. P. 1. (2020). ArXiv:2102.07122.
[88] Y.-H. Liu and D. Poulin, Neural Belief-Propagation Decoders for Quantum Error-Correcting Codes, Phys. Rev. Lett. **122**, 200501 (2019).
[89] J. Roffe, BP+OSD: A decoder for quantum LDPC codes. https://pypi.org/project/bposd/ (2021).
[90] D. S. Wang, A. G. Fowler, A. M. Stephens, and L. C. L. Hollenberg, Threshold error rates for the toric and planar codes, Quantum Inf. Comput. **10**, 456 (2010).
[91] A. G. Fowler, Minimum weight perfect matching of fault-tolerant topological quantum error correction in average O(1) parallel time, Quantum Inf. Comput. **15**, 145 (2015).
[92] O. Higgott, PyMatching: A PYTHON package for decoding quantum codes with minimum-weight perfect matching, ACM Trans. Quantum Comput. **3**, (2022).
[93] B. Dezs, A. Jüttner, and P. Kovács, LEMON—an open source C++ graph template library, Electron. Notes Theor. Comput. Sci. **264**, 23 (2011).
[94] See the repository at https://github.com/panqec/panqec for the implementation of the sweep-matching decoder.
[95] C. T. Chubb and S. T. Flammia, Statistical mechanical models for quantum codes with correlated noise, Ann. de l'Inst. Henri Poincaré D **8**, 269 (2018).
[96] C. T. Chubb, General tensor network decoding of 2D Pauli codes (2021), ArXiv:2101.04125.