

Decoding three-dimensional color codes with boundaries

Friederike Butt[✉],* Lars Esser[✉], and Markus Müller

*Institute for Theoretical Nanoelectronics (PGI-2), Forschungszentrum Jülich, Jülich, Germany
and Institute for Quantum Information, RWTH Aachen University, Aachen, Germany*

 (Received 15 December 2025; revised 6 March 2026; accepted 16 March 2026; published 6 April 2026)

Practical large-scale quantum computation requires both efficient error correction and robust implementation of logical operations. Three-dimensional (3D) color codes are promising candidates for fault-tolerant quantum computation due to their transversal non-Clifford gates, but efficient decoding remains challenging. In this work, we extend previous decoders for two-dimensional color codes [S.-H. Lee *et al.*, *Quantum* **9**, 1609 (2025)], which are based on the restriction of the decoding problem to a subset of the qubit lattice, to three dimensions. Including boundaries of 3D color codes, we demonstrate that the 3D restriction decoder achieves optimal scaling of the logical error rate and a threshold value of 1.55(6)% for code-capacity bit- and phase-flip noise, which is almost a factor of 2 higher than previously reported for this family of codes [N. Delfosse, *Phys. Rev. A* **89**, 012317 (2014); S. Turner *et al.*, *arXiv:2003.11602*]. We furthermore present QCODEPLOT3D, a PYTHON package for visualizing two-dimensional and 3D color codes, error configurations, and decoding paths, which supports the development and analysis of such decoders. These advancements contribute to making 3D color codes a more practical option for exploring fault-tolerant quantum computation.

DOI: [10.1103/7cdb-px1d](https://doi.org/10.1103/7cdb-px1d)

I. INTRODUCTION

Quantum error correction (QEC) encodes information across multiple physical qubits into logical qubits [1–4], such that errors that arise during computation can be detected and corrected. In combination with the capability to perform arbitrary and robust operations on encoded qubits, QEC enables the implementation of reliable large-scale computations. Recent experiments have demonstrated substantial progress in QEC [5–14], as well as fault-tolerant (FT) logical computation on encoded qubits [15–22]. Here, operations are implemented in a way that limits the propagation of errors such that small numbers of errors remain correctable. Color codes (CCs) [23,24] are particularly promising due to their natively transversal gates [25], where certain logical gates can be implemented with only local operations, inherently preventing uncontrolled proliferation of errors. Most notably, color codes in three dimensions support a transversal non-Clifford gate, which is a resource that typically requires costly preparation procedures as for example magic-state distillation [26]. Recently, small instances of three-dimensional (3D) color codes have been realized for the first time [17,19,27] to implement FT logical operations, for example by means of code switching [28–30]. This approach combines the complementary transversal gate sets of 3D and two-dimensional (2D) color codes to implement a universal gate set and has been

entered on trapped ion quantum processors [15,17] as well as neutral atom platforms [19]. However, to fully leverage the advantages of color codes, one requires effective methods for implementing QEC by identifying and correcting errors. *Decoding* refers to the task of interpreting error-symptom measurement outcomes to infer which physical errors have taken place, and to compute a suitable correction that ideally restores the logical state. The accuracy and reliability of this process directly determine the threshold of a code, below which overall error rates can be suppressed by increasing the physical overhead of the code, and long quantum computations become possible [31].

While surface and toric codes benefit from well-developed decoders with high thresholds, color codes are generally considered to be harder to decode, due to their intrinsic structural properties, as their stabilizer structure induces nongraph-like error mechanisms that limit the applicability of efficient matching decoders [32]. Existing decoders for 2D color codes are typically based on the restriction of the problem to a subset of the full qubit lattice [33–42], such that matching-based techniques become applicable. Alternative approaches make use of union-find decoding [34,43], as well as tensor or neural networks [44,45], and other strategies [46–48]. Recently, *vibe* decoding [49] as well as neural-network decoding [50] have achieved 2D color-code performance comparable to that of the surface code in the circuit-level noise setting. However, these strategies have not been generalized to higher-dimensional color codes, and it has been shown that minimum-weight decoding of the color code is NP-hard [51].

Restriction-based decoders [34] make use of minimum-weight perfect matching (MWPM) as a subroutine, which admits polynomial-time decoding with highly optimized implementations and achieves high thresholds for toric and surface codes [52–56]. These restriction decoders as well as

*Contact author: f.butt@fz-juelich.de

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

clustering decoding strategies [57] have been extended to 3D color codes without boundaries, but their performance remains below the theoretically predicted optimal threshold and subthreshold scaling [35,42,58]. In this work, we address this limitation by constructing a restriction-based decoder for color codes with boundaries in three dimensions. Our decoder achieves the optimal subthreshold scaling and improves on previously reported thresholds for this setting by almost a factor of 2 [35,42].

The remainder of this work is structured as follows. In Sec. II, we briefly review the general construction of color codes in three dimensions, including boundaries, and discuss the tetrahedral and cubic color codes. Section III summarizes minimum-weight perfect matching and presents our concatenated MWPM decoder, followed by numerical results in Sec. IV. We present QCODEPLOT3D, which is a PYTHON package that we have developed to visualize 3D codes and decoding graphs, in Sec. V, and conclude in Sec. VI.

II. CONSTRUCTION OF 3D COLOR CODES WITHOUT BOUNDARIES

In this section, we review the formal construction of color codes in three dimensions [24,59–62] without boundaries, i.e., the bulk structure of 3D color codes, before discussing the construction with boundaries in the next section. Three-dimensional color codes can be embedded in a 3D lattice $\mathcal{L} = (V, E, F, C)$ with vertices $v \in V$, edges $e \in E$, faces $f \in F$, and cells $c \in C$. Here, pairs of vertices V form edges E :

$$E \subseteq \{\{v_1, v_2\} \mid v_1, v_2 \in V \wedge v_1 \neq v_2\}. \quad (1)$$

Subsets of vertices form the cells C of the graph, and each cell contains a number of vertices that is a multiple of 4:

$$C \subseteq \{c \mid c \in \mathcal{P}(V) \wedge |c| = 4k \text{ for } k \in \mathbb{N}\}, \quad (2)$$

where c is a connected set of vertices. The interfaces of cells correspond to faces F of the graph, which are sets of three or more vertices:

$$F = \{c_1 \cap c_2 \mid c_1, c_2 \in C \wedge c_1 \neq c_2\}. \quad (3)$$

In the following, we write

$$\begin{aligned} \text{cells}(v) &= \{c \mid v \in c \wedge c \in C\}, \\ \text{cells}(e) &= \{c \mid e \subset c \wedge c \in C\}, \\ \text{cells}(f) &= \{c \mid f \subset c \wedge c \in C\}, \end{aligned} \quad (4)$$

for the set of cells that contain vertex v , edge e , or face f . One can now assign colors to vertices, edges, faces, and cells as a method of bookkeeping [24]. We either assign one *monochrome* color red (R), green (G), blue (B), or yellow (Y) or a combination of two colors, a mixed color, like RG or BY .

There are two conventionally used definitions of color codes [24,53,62]: placing the qubits on either the lowest-dimensional objects (the vertices) or the highest-dimensional objects (the cells). Both definitions are isomorphic to each other but give rise to different kinds of tessellations with different requirements on the lattice structure, which we discuss in the following.

The *primal lattice* $\mathcal{L}$ [24] provides a clear visualization of a code. The following two requirements must be fulfilled for a primal graph of a 3D color code.

(1) Vertices are 4-valent, so each vertex shares an edge with four other vertices.

(2) Cells are 4-colorable, so each cell of $\mathcal{L}$ can be colored with one of the four monochrome colors in such a way that cells sharing a face have different colors.

Additionally, each edge is colored with the monochrome color of the two cells it connects, and each face is colored with the mixed color of the two cells it separates. Now we can use this graph to define a quantum code by placing a qubit at each vertex of $\mathcal{L}$. Each cell c supports an X -type stabilizer generator S_c^X , and each face f supports a Z -type stabilizer generator S_f^Z :

$$\begin{aligned} S_c^X &:= \bigotimes_{v \in c} X_v, \\ S_f^Z &:= \bigotimes_{v \in f} Z_v. \end{aligned} \quad (5)$$

The *dual lattice* $\mathcal{L}^*$ [62] is a useful tool for decoding and fulfills the following two criteria.

(1) Cells are tetrahedra, so each cell has support on four vertices.

(2) Vertices are 4-colorable, so each vertex of $\mathcal{L}^*$ can be colored with one of the four monochrome colors in such a way that vertices sharing an edge have different colors.

Additionally, each face is colored with the monochrome color that is not used by any of its vertices, and each edge is colored with the mixed color of the two vertices it connects. In contrast to the primal lattice, we now place qubits at the cells of $\mathcal{L}^*$. Each vertex v of $\mathcal{L}^*$ supports an X -type stabilizer S_v^X , and each edge e supports a Z -type stabilizer S_e^Z :

$$\begin{aligned} S_v^X &:= \bigotimes_{c \in \text{cells}(v)} X_c, \\ S_e^Z &:= \bigotimes_{c \in \text{cells}(e)} Z_c. \end{aligned} \quad (6)$$

In this construction, a vertex in the primal graph is a cell in the dual graph and a face in the primal graph is an edge in the dual graph, and vice versa. Both definitions Eqs. (5) and (6) give rise to the same stabilizer definitions.

3D color codes with boundaries

The embedding of finite-distance 3D color codes on a 3D lattice requires only locally connected vertices and includes boundaries. Therefore, we need to extend the previous definitions to compact 3D spaces with boundaries.

In the primal picture, there are corner vertices V_{cor} at the topological boundary that are only 3-valent, instead of 4-valent. The number of such 3-valent vertices depends on the tessellation of the lattice and may vary between different codes. Analogously, there are edges at the topological lattice boundary that are only part of two faces, instead of three, and faces that are only part of one cell, instead of two. The set of such edges at the topological boundary that connect two corner vertices is called a border. The sets of faces

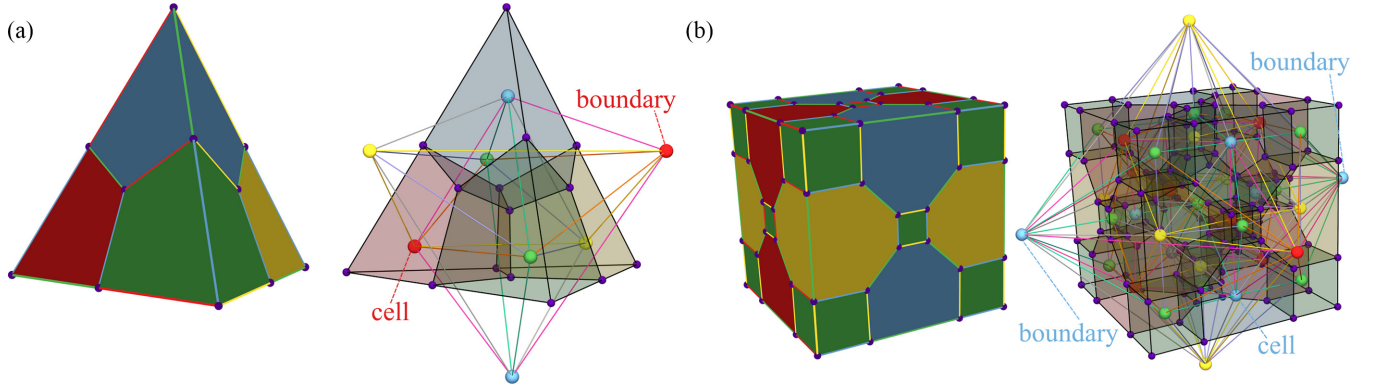


FIG. 1. Primal and dual lattices for tetrahedral and cubic color code: primal (left) and dual (right) lattice for (a) the distance-3 tetrahedral color code and (b) the distance-4 cubic color code. Vertices of the primal lattice are colored in purple. We sketch the dual lattice on top of the primal one with primal edges depicted in black and primal cells shown with transparent colors for better visualization. Dual vertices (i.e., cells of the primal lattice) and edges have bright colors. The mapping of dual edge colors is as follows: RB is pink, RG is brown, RY is golden, BG is olive green, BY is gray, and GY is purple. Primal faces and dual cells are not colored for better readability.

corresponding to the partition of the topological boundary through the borders are called the graph boundary. We modify the requirements on the primal lattice to include boundaries, such that we have the following.

(1) Corner vertices are 3-valent, and all other vertices are 4-valent.

(2) Cells in combination with the boundaries are 4-colorable, so each cell and each boundary can be colored with one of the four monochrome colors in such a way that cells and boundaries sharing a face have different colors, and boundaries sharing a border have different colors.

Additionally, the color of a face or edge connecting a cell and a boundary is determined by the color of the respective cell and boundary. Analogously to faces, borders (i.e., sets of edges) are colored with the two colors of the two boundaries they separate.

We analogously adjust the definition of the dual lattice $\mathcal{L}^*$. For each monochrome boundary of $\mathcal{L}$, we add a new vertex of the same color to $\mathcal{L}^*$. Those vertices are the boundaries of $\mathcal{L}^*$. For each face f between a cell and a boundary, an edge between the respective boundary vertex of $\mathcal{L}^*$ and the vertices of $\mathcal{L}^*$ is added to $\mathcal{L}^*$. An edge is added between two boundary vertices of $\mathcal{L}^*$ if the respective boundaries of $\mathcal{L}$ have a common border. Additionally, all faces and cells that emerge from the two previous steps are added to $\mathcal{L}^*$. In this extension of the dual lattice, the above requirements are automatically fulfilled. The definition of qubits and stabilizers on the graph does not change, except that no X stabilizers are defined at boundaries and no Z stabilizers are defined at borders.

We present two examples of 3D color codes with boundaries in the next subsections to illustrate the construction of the primal and dual lattices as described above. While the tetrahedral 3D color code encodes a single logical qubit and supports a transversal logical T_L gate, the cubic color code hosts three logical qubits and has a transversal implementation of the non-Clifford three-qubit CCZ_L gate.

1. Tetrahedral 3D color codes

Tetrahedral color codes have four boundaries, as illustrated in Fig. 1(a), and they encode $k = 1$ logical qubit. The logical

X operator of minimal support is defined on any boundary; the logical Z operator is defined on a border. Table I summarizes the number of physical qubits, and independent faces and cells for a given distance d . The bulk cells of the primal graph are truncated octahedra with six square faces, eight hexagonal faces and 24 vertices, as shown in Fig. 2. Cells at the boundary are truncated to either cubes; polyhedra with six square faces, two hexagonal faces, and 12 vertices; or polyhedra with six square faces, five hexagonal faces, and 18 vertices. The distance-3 tetrahedral color code has been used to implement fault-tolerant logical operations on a trapped-ion quantum processor [17] as well as on a neutral-atom platform [19].

2. Cubic 3D color codes

Cubic color codes have six boundaries, as illustrated in Fig. 1(b). Opposite boundaries share the same color and the code encodes $k = 3$ logical qubits. The six boundaries of a cubic color code are attributed to three different colors, with two opposing boundaries per color. We assign each logical qubit to a boundary color C . Logical X operators are defined on the C -colored boundaries, and the respective logical Z operators are defined on the borders connecting two C -colored boundaries. The bulk cells of the primal graph are cubes with six square faces and eight vertices, and chamfered cubes [63], i.e., cubes with symmetrically cutoff edges, with six square faces, 12 hexagonal faces, and 32 vertices. Chamfered cubes at the boundary are truncated to either regular cubes; polyhedra with

TABLE I. Color-code metrics. Number of physical qubits n and independent graph faces and cells for a given distance d for tetrahedral color codes and the cubic color code family.

	Tetrahedral ($k = 1$)	Cubic ($k = 3$)
n	$\frac{1}{2}(d^3 + d)$	$5d^3 - 12d^2 + 16$
Faces	$\frac{1}{4}(\frac{5}{3}d^3 - d^2 + \frac{7}{3}d - 3)$	$4d^3 - \frac{21}{2}d^2 + 3d + 8$
Cells	$\frac{1}{4}(\frac{1}{3}d^3 + d^2 - \frac{1}{3}d - 1)$	$d^3 - \frac{3}{2}d^2 - 3d + 5$

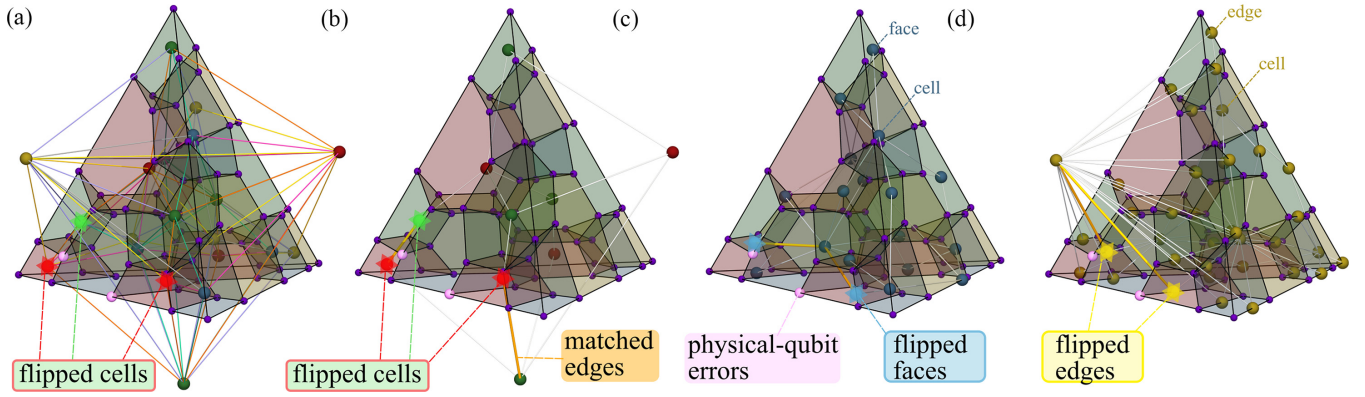


FIG. 2. Example decoding path of the 3D concatenated MWPM decoder on the distance-5 tetrahedral color code. (a) Dual-graph picture of the distance-5 tetrahedral color code with boundaries, where physical qubits of the primal lattice are shown in purple. Errors are placed at physical qubits shown in pink. For this exemplary error configuration, two neighboring red and one green stabilizer are flipped, as indicated by the bright star-shaped large vertices on the dual lattice. (b) The restricted graph $\mathcal{R}_{RG}$ is constructed by removing all blue and yellow vertices of the dual lattice $\mathcal{L}^*$ and all associated edges. MWPM is run on the restricted lattice and we obtain a set of matched edges (thick orange lines). (c) We obtain the first monochrome graph $\mathcal{M}_{RG}^B$ by placing blue vertices at all red-green faces as well as all blue cells of the primal graph. Every vertex that corresponds to an edge of the matching on the restricted graph is marked as flipped (e.g., the two highlighted light-blue vertices), as well as all initially violated blue vertices. We again run MWPM on this instance yielding a set of matched edges. (d) The second monochrome graph $\mathcal{M}_{RG,B}^Y$ is constructed by placing yellow vertices at all yellow edges and yellow cells of the primal graph. We mark every vertex that corresponds to a matched edge of the previous step, as well as all initially flipped yellow cells of the initial graph, and again run MWPM. This final matching is the suggested correction of the decoding path. In the illustrated example, we apply a correction to the physical qubits that reside on the edges connecting the flipped, star-shaped yellow vertices and the yellow boundary, i.e., the pink qubits.

six square faces, seven hexagonal faces, and 22 vertices; or polyhedra with seven square faces, eight hexagonal faces, one octagonal face, and 28 vertices. Table I summarizes the properties of cubic color codes. The smallest instance of a cubic color code, i.e., the $[[8, 3, 2]]$ code, has been encoded on trapped ions to implement a Grover search on three logical qubits [64], and on neutral atoms to realize computationally complex sampling circuits [14].

III. DECODING 3D COLOR CODES WITH BOUNDARIES

In this section, we briefly review MWPM [65] and previous color-code decoders that build on MWPM as a subroutine. We then present a decoder for 3D color codes, the 3D concatenated MWPM decoder, which builds on the concatenated MWPM decoder for 2D color codes introduced in Ref. [33].

A. MWPM

MWPM [65] is a way of matching up a set of vertices in a graph so that each vertex is matched exactly once. This decoding strategy chooses the pairing with the smallest total edge cost, i.e., the minimal total edge weight. MWPM benefits from highly optimized solvers such as blossom-based algorithms and PYMATCHING that enable fast and scalable decoding for large system sizes [56,66]. It has been shown to achieve high thresholds for the toric and surface code [52–56], and to perform efficiently [56,66,67], i.e., with a complexity that scales polynomially with the system size. However, MWPM cannot straightforwardly be used to decode color codes, because single-qubit errors in the color code can flip an odd number of stabilizers, which violates the condition to apply MWPM for a perfect matching [42,56].

One way to overcome this problem is to restrict the color-code syndrome and to consider only the stabilizers of a subset of colors. On the restricted lattice, a single error is guaranteed to yield an even number of violated stabilizers and is thereby decodable with MWPM. The main task of the decoder is then to combine the restricted matchings with a *lifting procedure* to obtain a correction for the original color code [34,35,41,42]. Restriction decoders have been used for decoding 2D color codes [33,34,37], where three restricted graphs can be constructed by considering two out of the three possible colors that are used to construct a 2D code. MWPM is run on all three restricted graphs of two colors, and the lifting procedure combines them into a correction of the initial code. In a similar approach [42], MWPM can be run only on two of the three restricted graphs of two colors and a modified *local* lifting procedure is guaranteed to return a global correction. This decoder was generalized to higher-dimensional color codes [35] by iteratively restricting the lattice until only two colors are left and strategies for 2D color-code decoding [33,42] can be applied. However, previous works [35] do not take boundaries into account, which requires a modified lattice construction as discussed in Sec. II, and their resulting logical failure rate in leading order does not scale optimally with the physical error rate, but rather as $p_L \propto O(\frac{d}{3})$. Reference [33] describes a decoder for 2D color codes with boundaries, which in turn builds upon the restriction approach and employs a lifting scheme that overcomes the issue of uncorrectable errors of weight $O(\frac{d}{3})$. In this work, we generalize this decoder [33] to 3D color codes.

B. 3D concatenated MWPM decoder

In this section, we present our concatenated MWPM decoder for 3D color codes. The general idea is to correct errors

in 3D color codes by successively applying MWPM on a hierarchy of simplified graphs. Each layer captures how errors affect different color subsets of stabilizers. By decoding these layers in sequence, the algorithm efficiently reconstructs the error configuration of lowest weight that generates the observed syndromes. First, we introduce three additional types of graphs: the restricted graph, the first monochrome graph, and the second monochrome graph. For a dual graph $\mathcal{L}^* = (V_D, E_D, F_D, C_D)$ with vertex colors C, D, E , and F , we construct the *restricted graph* of colors C and D , $\mathcal{R}_{CD} = (V_R, E_R)$ by removing all vertices of color E and F and all edges that include vertices of color E or F :

$$\begin{aligned} V_R &= \{v \mid \text{color}(v) \in \{C, D\} \wedge v \in V_D\}, \\ E_R &= \{\{v_1, v_2\} \mid v_1, v_2 \in V_R \wedge v_1 \neq v_2\}. \end{aligned} \quad (7)$$

For example, the graph $\mathcal{R}_{RG}$ contains only red and green vertices and only edges that connect red and green vertices. Next, we construct a *first monochrome graph* $\mathcal{M}_{CD}^E$, which is based on the previous restricted graph $\mathcal{R}_{CD}$, by placing E -colored vertices on all edges of $\mathcal{R}_{CD}$ (i.e., CD -colored faces of the primal graph) as well as on E -colored vertices of $\mathcal{L}^*$. It only contains E -colored vertices. The monochrome graph of color E given the restricted graph $\mathcal{R}_{CD}$ is therefore defined as $\mathcal{M}_{CD}^E = (V_{M_1}, E_{M_1})$:

$$\begin{aligned} V_{M_1} &= E_R \cup \{\{v\} \mid \text{color}(v) = E \wedge v \in V_D\}, \\ E_{M_1} &= \{\{v_1, v_2\} \mid v_1 \cup v_2 \in F_D \wedge v_1, v_2 \in V_{M_1}\}. \end{aligned} \quad (8)$$

For example, $\mathcal{M}_{RG}^B$ contains only blue vertices that are placed on all RG edges and blue vertices of $\mathcal{L}^*$, as shown in Fig. 2(c). Two vertices of the first monochrome graph are connected by an edge if and only if they correspond to an F -colored face of $\mathcal{L}^*$, i.e., one of them is an edge of $\mathcal{R}_{CD}$ and one of them is an E -colored vertex of $\mathcal{L}^*$.

Lastly, we analogously construct the *second monochrome graph* $\mathcal{M}_{CD,E}^F$ of color F by placing F -colored vertices on edges of the first monochrome graph $\mathcal{M}_{CD}^E$ (i.e., F -colored edges of the primal graph) and all F -colored vertices of $\mathcal{L}^*$. We define $\mathcal{M}_{CD,E}^F = (V_{M_2}, E_{M_2})$, based on $\mathcal{M}_{CD}^E$, where

$$\begin{aligned} V_{M_2} &= E_{M_1} \cup \{\{v\} \mid \text{color}(v) = F \wedge v \in V_D\}, \\ E_{M_2} &= \{\{v_1, v_2\} \mid v_1 \cup v_2 \in C_D \wedge v_1, v_2 \in V_{M_2}\}. \end{aligned} \quad (9)$$

For example, $\mathcal{M}_{RG,B}^Y$ contains only yellow vertices that are placed on all yellow edges of $\mathcal{L}$ and yellow vertices of $\mathcal{L}^*$, as illustrated in Fig. 2(d). Two vertices of the second monochrome graph are connected by an edge if and only if one of them is an edge of $\mathcal{M}_{CD}^E$ and one of them is an F -colored vertex of $\mathcal{L}^*$.

We now introduce the *3D concatenated MWPM decoder*, first considering Z -type errors that are detected by cell-type X stabilizers. For a given code, we can construct the respective dual graph $\mathcal{L}^* = (V_D, E_D, F_D, C_D)$ and track the error syndrome $S \subset V_D$, rooting in a set of Z errors on physical qubits. Figure 2(a) shows a distance-5 tetrahedral color code in the dual-lattice picture and an exemplary error configuration. Each error is highlighted in pink and each flipped stabilizer is depicted as a large bright node, while unflipped stabilizers are shown in darker colors.

In a first step, we construct the restricted graph $\mathcal{R}_{RG}$, and mark each vertex that corresponds to a violated red or green stabilizer, as shown in Fig. 2(b). On the restricted graph, single-qubit errors flip by construction at most two stabilizers, one stabilizer per color since the restricted graph includes only two colors. Therefore, we can use MWPM to match the marked vertices on the restricted lattice [thick orange edges in Fig. 2(b)].

Second, we construct the first monochrome graph $\mathcal{M}_{RG}^B$, and mark vertices that correspond to flipped blue stabilizers or to previously matched edges, as shown in Fig. 2(c). Each single-qubit error marks only up to two vertices of the same color: one vertex corresponding to a blue stabilizer, and one vertex corresponding to a previously matched edge. Therefore, we can again run MWPM to match all marked vertices.

Third, we construct the second monochrome graph $\mathcal{M}_{RG,B}^Y$, and mark vertices that correspond to flipped yellow stabilizers or to previously matched edges, as shown in Fig. 2(d). We again run MWPM to obtain a final matching, where each edge of the matching corresponds to a physical qubit of the code. The set of qubits obtained from the final matching is the suggested correction of the decoding path RG, B, Y .

Steps 1, 2, and 3 are repeated for all other color combinations, as for example for the restricted graph $\mathcal{R}_{BY}$, the first monochrome graph $\mathcal{M}_{BY}^R$, and the second monochrome graph $\mathcal{M}_{BY,R}^G$. $\mathcal{L}^*$ has four vertex colors and the restricted graph is created with two colors, so there are $\binom{4}{2} = 6$ restricted graphs. For each restricted graph, there are two possible combinations to construct the first and second monochrome graph, leading to $6 \times 2 = 12$ possible decoding paths. All 12 decoding paths can be evaluated independently and can be run in parallel. The total runtime is therefore determined by the time it takes to perform one decoding path which includes three MWPM sub-routines, as discussed further in Appendix C. After evaluating all 12 decoding paths, we select the suggested correction with the lowest weight.

Our decoder is designed to deal with cell-like stabilizers, and is therefore able to natively correct Z errors. It is also able to correct X errors by combining the facelike stabilizer syndromes to cell-like stabilizer syndromes through multiplying measurement results. This merging of facelike stabilizers into cell-like stabilizers reduces the maximum distance d_x for correcting X errors to match the distance $d_z = d$ for Z errors. For example, the tetrahedral $[[15, 1, 3]]$ code, shown in Fig. 1(a), can correct up to weight-3 X errors by evaluating all ten facelike Z stabilizers and achieve a distance $d_x = 7$ with a look-up-table decoder. By taking into account only the four Z cells, we reduce this distance to $d_x = d_z = 3$, as we can only correct a single X error. More generally, the distance d_z of a 3D color code grows linearly with the length of a border of the 3D lattice which supports the logical operators Z_L for both the tetrahedral and cubic color codes. The distance d_x grows with the size of the X_L operators, which have support on the boundaries, as for example the set of faces on one side of the tetrahedral or cubic lattice. Therefore, $d_x \propto d_z^2$, because the size of a boundary scales quadratically with the length of its borders. In other words, by restricting itself to cell-like stabilizers, the concatenated MWPM decoder induces a square-root reduction in the distance d_x , as observed in previous works on 3D color-code decoding [33–35, 42]. The

TABLE II. Thresholds for the concatenated MWPM decoder. We determine the pseudothreshold and cross threshold for tetrahedral and cubic color codes, as well as the subthreshold scaling and the effective distance. These values are extracted from numerical simulations, shown in Fig. 3.

Code	Pseudothreshold	Cross threshold	Subthreshold scaling	Effective distance
Tetrahedral CC	1.13(2)%	1.48(2)%	$p_{\text{phys}}^{(d+1)/2}$	d
Cubic CC	0.60(6)%	1.55(6)%	$p_{\text{phys}}^{d/2}$	$d - 1$

effective distances of the cubic and tetrahedral color codes are summarized in Table II. However, the overall distance for an arbitrary input state is always limited by the minimum weight of an arbitrary-error configuration and remains the same.

In the next section, we use the presented decoding strategy to decode errors on cubic and tetrahedral color codes up to distance 9.

IV. NUMERICAL RESULTS

We numerically simulate the success rate of the presented concatenated MWPM decoder for color codes considering code-capacity noise. Specifically, we prepare perfect logical states $|0\rangle_L$ and $|+\rangle_L$ followed by a noisy idling channel on all data qubits. Here, we apply only bit-flip errors and only phase-flip errors on logical state $|0\rangle_L$ and $|+\rangle_L$, respectively, on each physical qubit with a probability p_{phys} , and then perform noise-free error correction. In the end, we average the total logical failure rate over both logical input states. The logical error rate is determined by means of Monte Carlo sampling as well as subset sampling [68,69], as discussed in Appendix A. Figure 3 shows the numerically obtained logical error rates for tetrahedral color codes up to distance $d = 9$ and cubic color codes up to a distance $d = 8$.

We determine the *pseudothreshold* [20,70–72], which is the breakeven point of the physical error rate and the logical error rate for the lowest-weight error-correcting code, as well as the *cross threshold* [6,19,33,70–72], which is the crossing point of the logical failure rates for the same codes at different distances. Both of these converge to the *asymptotic threshold* [70–72] for $d \rightarrow \infty$ for an optimal decoder that corrects the theoretically possible maximum number of errors for a given code size. We also analyze the scaling of the logical error rate with the physical error rate below threshold, called the *subthreshold scaling*. This scaling determines the effective distance since $p_L \propto p_{\text{phys}}^{[(d+1)/2]}$ [25]. Table II summarizes pseudothresholds, cross thresholds, and subthreshold scalings for the concatenated MWPM decoder run on 3D tetrahedral and cubic color codes. As discussed above, the overall distance is determined by the minimum-weight noncorrectable error configuration and, therefore, fixed by the number of cell-like stabilizer operators. Importantly, we find a cross threshold of 1.48(2)% for the tetrahedral and 1.55(6)% for cubic color codes for our concatenated MWPM decoder. The ideal asymptotic threshold has been estimated by means of a statistical mechanics mapping to be $p_{3\text{DCC}} \approx 1.9\%$ for stringlike logical operators on a general 3D color code [58]. Previous works have reported a cross threshold of 0.77% for 3D color codes

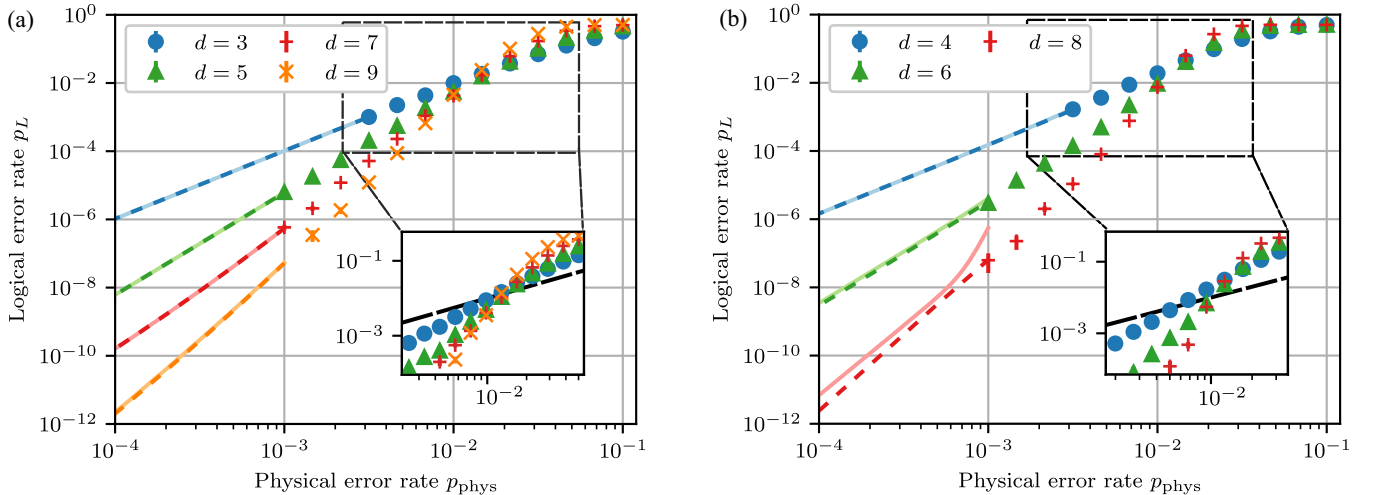


FIG. 3. Logical error rates for decoding with the concatenated MWPM decoder. (a) Decoding of 3D tetrahedral color codes of distance $d = 3, 5, 7, 9$ and (b) $d = 4, 6, 8$ cubic color codes. Data points at physical errors rates $p_{\text{phys}} > 10^{-3}$ are determined by means of direct Monte Carlo sampling. At low physical error rates, we use subset sampling [68,69] (solid lines) to calculate an upper (light, solid color) and a lower bound (dark, dashed color) on the logical error rate. The sampled Monte Carlo values fall within the upper and lower bound estimated from subset sampling for the respective sampled value of p_{phys} . The inset shows the logical error rates close to (a) $p_{\text{phys}} = 0.014$ and (b) $p_{\text{phys}} = 0.015$, below which increasing distance suppresses the logical error rate. (b) Logical error rates for one of the three encoded logical qubits. The logical error rates of the other two logical qubits of the respective cubic color code are extracted simultaneously and show similar performance, as can be seen in Fig. 4.

without boundaries [42], and a cross threshold of 0.7 to 0.8% [35] for 3D color codes with boundaries. Our decoder therefore improves on these previously reported thresholds by almost a factor of 2.

Furthermore, we can identify a color combination of one decoding path for cubic color codes that achieves the scaling of the logical error rate as reported in Table II without the need for evaluating all 12 decoding paths. A single evaluation, and therefore only three instances of MWPM, has to be run, which is discussed further in Appendix E. Here, we effectively localize errors at an early stage in our decoding process by choosing a restricted graph that contains the color green. This choice is motivated by the fact that the set of all green stabilizers has support on every physical qubit. The required instances of MWPM subroutines can therefore be reduced: before, we needed to perform MWPM on all six restricted graphs once, and for each restricted graph there are two combinations of monochrome graphs and for each combination two MWPM routines are executed. In total, we can therefore reduce the number of MWPM subroutines from $(1 + 2 + 2) \times 6 = 30$ to 3 for one evaluation of a single decoding path. While the scaling of the logical failure rate stays the same as when evaluating all 12 decoding paths, the number of correctable higher-weight error configurations decreases, which results in a drop of the pseudothreshold to 0.42(6)% and the cross threshold to 1.02(6)%, as can be seen in Fig. 6. Selecting an intermediate number of decoding paths may improve the threshold relative to evaluating only a single optimal path, while reducing the overall runtime compared to evaluating all 12 paths. A detailed analysis of this tradeoff lies beyond the scope of this work. We did not find an optimal single decoding path for the tetrahedral code, as there is no set of single-color stabilizers that has support on all physical qubits.

V. PYTHON FRAMEWORK FOR VISUALIZATION: QCODEPLOT3D

We have developed a PYTHON framework to visualize 2D and 3D color codes and their decoding processes, utilizing the Visualization Toolkit [73] wrapped by the pyvista PYTHON package [74]. It takes a graph $G = (V, E)$ that describes the color code in the dual-graph picture as an input, and provides an interactive 3D visualization of the color code in the primal-graph picture. Each family of color codes requires specific postprocessing for optimal layout results. These are already implemented for tetrahedral and cubic color codes, and can work as guiding examples for visualizing 3D lattice structures. Additionally, we provide the functionality to create the dual graph of a cubic or tetrahedral color code for any given even or odd code distance. In both the primal and dual-graph picture, our package offers the possibility to place errors and track their decoding paths. For ease of use, we created a graphical user interface that takes the code parameters of interest as input, and directly constructs the respective interactive graph.

We distribute QCODEPLOT3D as a publicly available PYTHON package [75,77] and as a GitHub repository [76,78]. All figures depicting 3D color codes and their graphs were created with QCODEPLOT3D.

VI. DISCUSSION

In this work, we have developed a decoding strategy tailored to 3D color codes with boundaries, building on existing decoders for 2D color codes and for 3D codes without boundaries [33,42]. The main contribution of our work is the extension of these decoders [33,42] to three dimensions and to include boundaries, achieving thresholds almost twice as high as previously reported in this setting [35]. This advancement represents an important step toward practical decoding of 3D color codes, which can natively host FT logical operations required for universal quantum computing. Future work includes the optimization of the presented decoder by exploring tradeoffs between runtime and performance for example by investigating the performance of a subset of decoding paths, instead of all or a single optimal one. Since 3D color codes are inherently capable of correcting more X -type than Z -type errors, adapting the decoder to biased noise models, where one error type occurs more frequently, could further improve the performance. A requirement for this is to use all face-type stabilizers for decoding, not only products of face operators forming cell-like operators as in the presented decoder. Beyond biased noise, incorporating realistic error sources such as measurement errors, erasure errors, and noisy circuit components would bring the decoder closer to practical deployment in experimental settings [15,17,19], as the presented concatenated MWPM decoder is limited to a code-capacity setting. Erroneous syndrome measurements would require additional rounds of stabilizer extraction, which are currently not considered. Circuit-level noise may lead to higher-weight error configurations due to the propagation of physical errors through noisy circuits, and necessitates the careful design of fault-tolerant circuitry for stabilizer measurement, for example by means of flag qubits [75,77] or additional logical resource states [76,78]. In the circuit-level noise setting, additional improvements could be achieved by exploring more sophisticated decoding strategies beyond standard MWPM as a subroutine. One potential approach is to decode individual layers of 2D color codes that collectively form a 3D code instance, in combination with techniques such as vibe decoding [49] or neural network [50] decoding, which have been shown to perform well under this noise model. Furthermore, extending state-of-the-art decoding strategies such as the Tesseract decoder [46] or belief propagation to 3D color codes might improve performance or speed up the decoding process. Overall, our work contributes to the foundation for a range of extensions that could bring 3D color-code decoders closer to practical use in terms of performance and computational efficiency.

ACKNOWLEDGMENTS

We thank Sascha Heußen for helpful discussions on the layout of tetrahedral color codes and Lukas Bödecker for insightful conversations regarding the weighting of matching graph edges. We gratefully acknowledge support by the European Union (EU) Horizon Europe research and innovation program under Grant Agreement No. 101114305 (“MILLENION-SGA1” EU Project); the German Federal Ministry of Research, Technology, and Space (BMFTR)

as part of the Research Program Quantum Systems, Research Project No. 13N17317 (“SQale”); the BMFTR project MUNIQ-ATOMS (Grant No. 13N16070); and the Intelligence Advanced Research Projects Activity (IARPA), under the Entangled Logical Qubits program through Cooperative Agreement No. W911NF-23-2-0216. This research is also part of the Munich Quantum Valley (K-8), which is supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus. We furthermore acknowledge support by the European Research Council Starting Grant QNets through Grant No. 804247 and by the Deutsche Forschungsgemeinschaft (German Research Foundation) under Germany’s Excellence Strategy “Cluster of Excellence Matter and Light for Quantum Computing (ML4Q) EXC 2004/1” Grant No. 390534769. We acknowledge computing time provided at the NHR Center NHR4CES at RWTH Aachen University (Project No. p0020074). This is funded by the Federal Ministry of Education and Research and the state governments participating on the basis of the resolutions of the GWK for national high-performance computing at universities.

F.B. and L.E. developed the presented protocols and L.E. numerically implemented the presented decoders. L.E. developed the PYTHON package QCODEPLOT. L.E. and F.B. performed the numerical simulations and analyzed results. F.B. and L.E. wrote the manuscript, with contributions from all authors. M.M. supervised the project.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of IARPA, the U.S. Army Research Office, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

DATA AVAILABILITY

No data were created or analyzed in this study.

APPENDIX A: NUMERICAL METHODS

We numerically determine logical failure rates to evaluate the performance of the presented decoders. We use *direct Monte Carlo sampling* to simulate logical failure rates for large physical error rates and *subset sampling* [68,69] at small physical error rates, i.e., for rare error events.

1. Direct Monte Carlo

We use the STIM PYTHON package [79], which provides a framework to analyze QEC circuits by means of stabilizer tableau representations [79], to perform Monte Carlo simulations [80,81]. The following protocol is implemented numerically.

(1) Encode $|0\rangle_L/|1\rangle_L$ on each logical qubit k within a QEC code.

(2) Apply an X error to each physical qubit with probability p_{phys} , following the code-capacity noise model [82].

TABLE III. Color-code metrics. Distance d , number of physical qubits n , independent Z generators (defined on faces), and independent X generators (defined on cells) of the first four members of the (a) tetrahedral color code, encoding $k = 1$ logical qubit, and (b) the cubic color-code family, encoding $k = 3$ logical qubits.

(a) d	n	No. of Z generators	No. of X generators
3	15	10	4
5	65	48	16
7	175	134	40
9	369	288	80
(b) d	n	No. of Z generators	No. of X generators
2	8	4	1
4	144	108	33
6	664	512	149
8	1808	1408	397

(3) Projectively measure in the Z basis, and use the measurement results to determine the value of the logical operator $Z_L^{(k)}$ and the Z syndrome.

(4) Use a decoder to decode the syndrome and obtain a Pauli correction.

(5) Postprocess the measurement result of each $Z_L^{(k)}$ by flipping the measurement result for each qubit which is in the support of $Z_L^{(k)}$ and in the correction.

(6) Check if the postprocessed measurement result of each $Z_L^{(k)}$ is $+1$ or -1 . If this is not the case, a logical error occurred on the respective logical qubit k .

For correcting Z errors we use the same protocol for initial states $|+\rangle_L/|-\rangle_L$. By repeating this procedure m times, one can estimate the expectation value of the respective logical operator and logical failure rate for each logical qubit k with

$$p_L^{(k)}(p_{\text{phys}}) = \frac{\text{No. of logical errors on } k}{m}. \quad (\text{A1})$$

The standard deviation of the sampling for large m can be described as [81]

$$\varepsilon_L^{(k)} \sim \sqrt{\frac{p_L^{(k)}(1 - p_L^{(k)})}{m}}. \quad (\text{A2})$$

2. Subset sampling

In principle, direct Monte Carlo simulation can be used to determine the logical failure rate for arbitrarily small p_{phys} . However, this becomes inefficient for small p_{phys} , since actual error events are rare. To achieve results with reasonably small standard deviation, one requires a very large number of repetitions m .

A more efficient way to use computational resources for small physical error rates p_{phys} is subset sampling [68,69], which samples individual error weights separately. In the following, we drop the index of the logical qubits k for readability. For sufficiently small values of p_{phys} , the most probable setting of a direct Monte Carlo simulation contains no physical qubit error at all. Sampling this so-called zero-fault subset yields no insights, if it is already known that the decoder works properly for the trivial error configuration. The next probable setting contains one physical-qubit error,

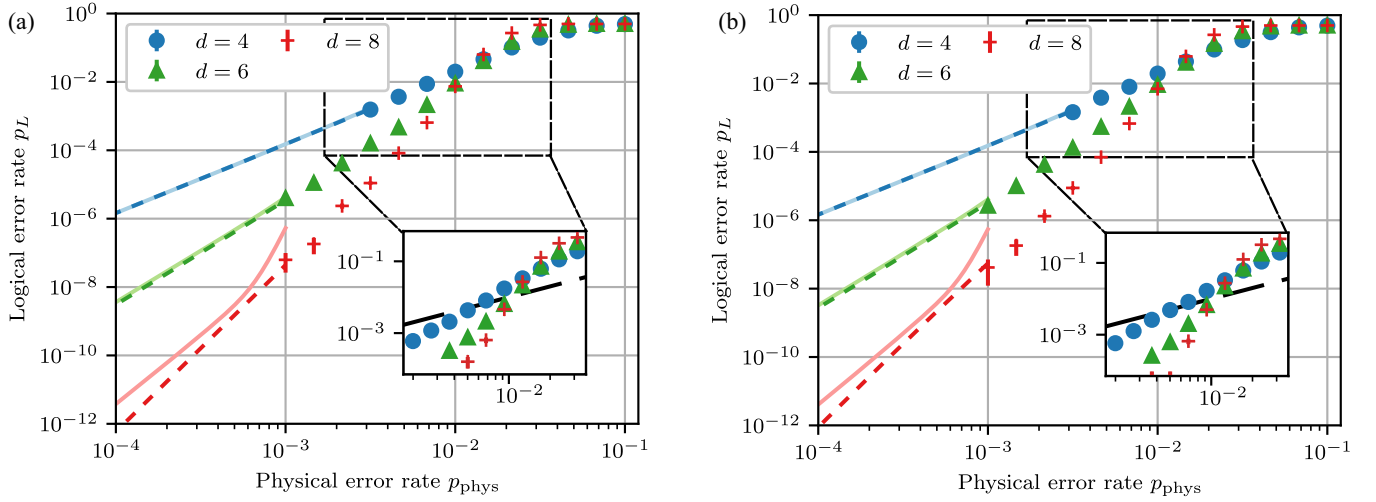


FIG. 4. Logical error rates of logical qubits 2 and 3 for decoding cubic color codes with the concatenated MWPM decoder. (a) Logical error rates for $d = 4, 6, 8$ cubic color codes considering (a) logical qubit 2 and (b) logical qubit 3. Data points at physical error rates $p_{\text{phys}} > 10^{-3}$ are determined by means of direct Monte Carlo sampling. At low physical error rates, we use subset sampling to calculate an upper (light, solid line) and a lower bound (dark, dashed line) on the logical error rate. The inset shows the logical error rates close to $p_{\text{phys}} = 1.5\%$, below which increasing distance suppresses the logical error rate. All three logical qubits show similar performance.

the 1-fault subset, followed by the 2-fault subset with two physical-qubit errors and so on. One can determine the logical failure rate $p_L^{(\omega)}$ of each ω -fault subset individually, and combine them to the logical failure rate

$$p_L(p_{\text{phys}}) = \sum_{\omega=0}^n \binom{n}{\omega} p_{\text{phys}}^{\omega} (1 - p_{\text{phys}})^{n-\omega} p_L^{(\omega)}. \quad (\text{A3})$$

The values of $p_L^{(\omega)}$ can be sampled individually with $m^{(\omega)}$ repetitions. One now chooses the maximal cutoff ω -fault subset ω_{max} . By considering the cases $p_L^{(\omega')} = 0$ and 1 for each $\omega' > \omega_{\text{max}}$, respectively, one can estimate a lower and an upper bound for p_L . The lower bound is the sum of the contributions of each subset with $\omega < \omega_{\text{max}}$, since $p_L^{(\omega')} = 0$ lets all terms with $\omega' > \omega_{\text{max}}$ vanish. The upper bound contains an additional contribution for $\omega' > \omega_{\text{max}}$ of

$$\delta(p_{\text{phys}}) = 1 - \sum_{\omega=0}^{\omega_{\text{max}}} \binom{n}{\omega} p_{\text{phys}}^{\omega} (1 - p_{\text{phys}})^{n-\omega}. \quad (\text{A4})$$

The standard deviation $\varepsilon_L^{(\omega)}$ of each individual sampling of an ω -fault subset $p_L^{(\omega)}$ is described by Eq. (A2). The combined standard deviation of all subset sampling contributions is

$$\varepsilon_L(p_{\text{phys}}) = \sqrt{\sum_{\omega=1}^{\omega_{\text{max}}} \left[\binom{n}{\omega} p_{\text{phys}}^{\omega} (1 - p_{\text{phys}})^{n-\omega} \varepsilon_L^{(\omega)} \right]^2}. \quad (\text{A5})$$

The lower and upper bound of the total logical failure rate are given by a combination of the lower and upper bound of $p_L(p_{\text{phys}})$ with the combined standard deviation of all samplings as

$$\begin{aligned} p_L^{\text{lower}}(p_{\text{phys}}) &= p_L(p_{\text{phys}}) - \varepsilon_L(p_{\text{phys}}), \\ p_L^{\text{upper}}(p_{\text{phys}}) &= p_L(p_{\text{phys}}) + \delta(p_{\text{phys}}) + \varepsilon_L(p_{\text{phys}}). \end{aligned} \quad (\text{A6})$$

We do not use STIM to numerically estimate $p_L^{(\omega)}$, but determine the measurement results of the logical operators and the stabilizer syndrome directly from the set of physical errors. If a stabilizer shares an even or odd number of qubits with the set of errors, we would measure $+1$ or -1 . Given the set

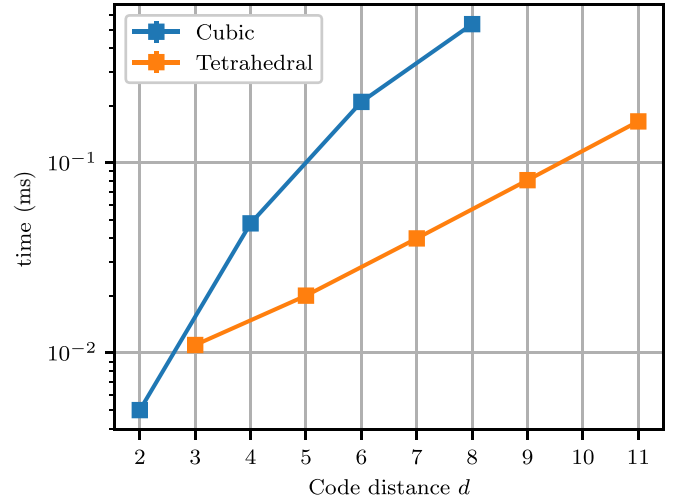


FIG. 5. Runtime of the concatenated decoder. We determine the simulation runtime of one decoding path on a single laptop (Apple M1 Pro) of the concatenated MWPM decoder for the cubic (blue) and the tetrahedral color code (orange). Each simulation is repeated 1000 times in a single-threaded implementation, and the decoding time is determined by averaging over all shots for each distance. The obtained decoding time includes the three MWPM subroutines and the time it takes to generate the syndrome graphs for each subroutine. We observe that the decoding time of the tetrahedral code scales substantially slower with the code distance as compared to the cubic color code due to a smaller number of stabilizer generators at a given distance, as discussed further in Appendix C.

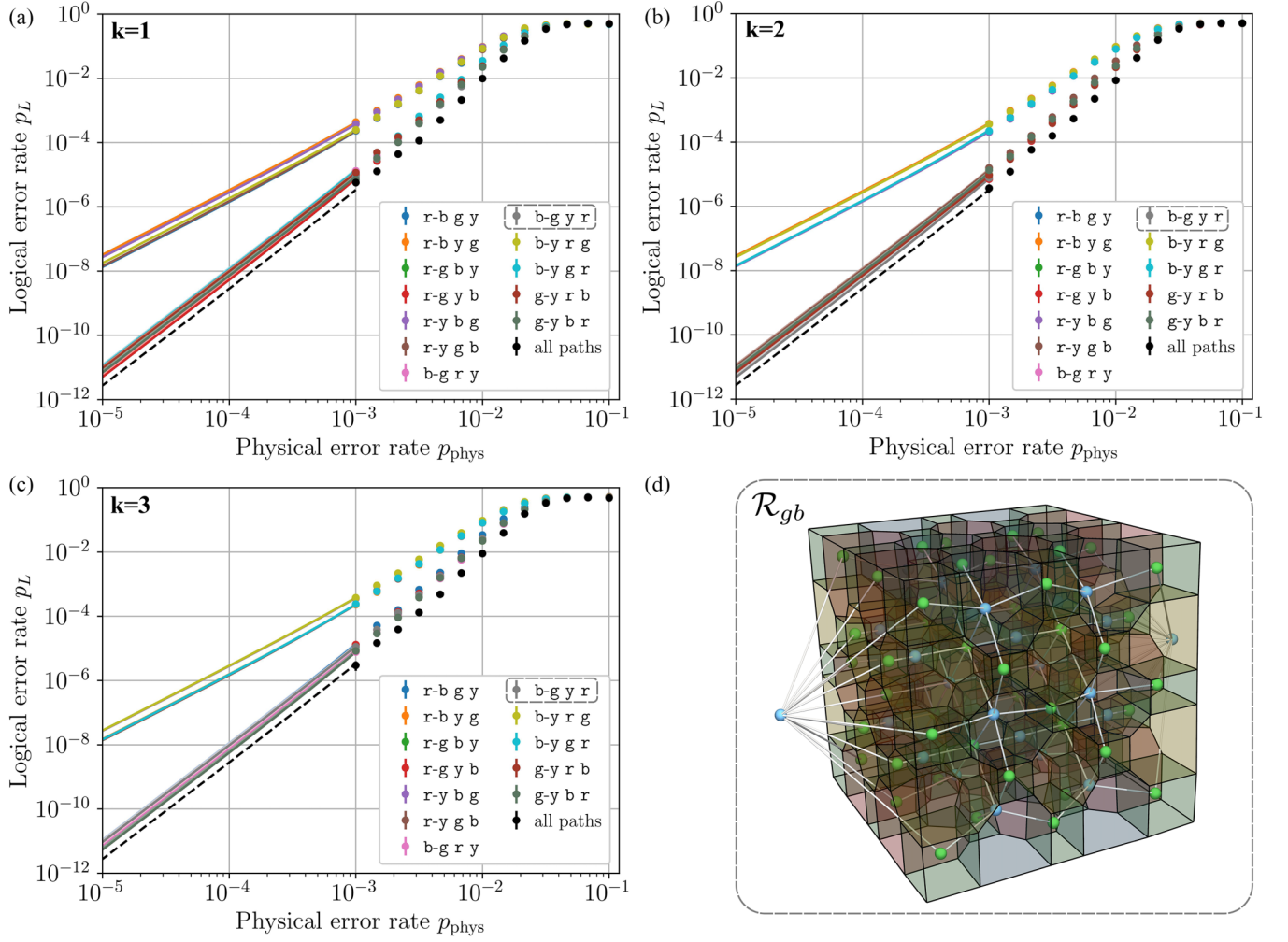


FIG. 6. Individual decoding paths for the $d = 6$ cubic color code. Logical error rates evaluated for all 12 decoding paths for logical qubit (a) 1, (b) 2, and (c) 3. There are single decoding paths, for which the logical error rate scales with $p_L \propto p_{\text{phys}}^3$ with the physical error rate for all three logical qubits, as for example the $B-G$, Y , R path. This means there exists a single decoding path that can be used to correct any weight-2 error and the effective distance is the same as indicated in Table II. (d) $B-G$ restricted lattice. The green cells in combination have support on all physical qubits.

of flipped stabilizers, we can run the concatenated MWPM decoder. Since we know the initially prepared logical state, we can track the initially placed error and determine if the respective logical state has been flipped in the end.

3. Minimum-weight perfect matching

We use PYMATCHING for each MWPM subroutine [56], which is an open-source package in PYTHON [83].

APPENDIX B: EXPLICIT CODE PARAMETERS FOR $d < 10$ COLOR CODES

Table III shows the number of physical qubits, as well as the number of independent faces and cells of the tetrahedral and cubic color code for distances $d < 10$. These were determined by constructing each graph explicitly and then counting the respective quantity. Table I summarizes the analytical

expressions for these quantities as a function of the code distance d .

APPENDIX C: RUNTIME ANALYSIS

Figure 5 shows the scaling of the runtime of the numerical simulation for one decoding path, which contains three MWPM subroutines. Decoding the cubic color code at a given distance d takes longer than decoding the tetrahedral of the next closest distance $d + 1$, because the number of cells, faces, and edges is substantially larger in the cubic color-code lattice. In the restricted and monochrome graphs, cells, faces, as well as edges correspond to vertices in the decoding graph, and the decoding complexity increases with the number of vertices to match. For example, the $d = 4$ cubic code contains 33 cells, 108 faces, and 144 vertices while the $d = 5$ tetrahedral code only includes 16 cells, 48 faces, and 65 vertices, as specified in Tables I and III.

While Fig. 5 shows the numerically determined runtime for evaluating a full decoding path, we can additionally estimate the expected runtime of MWPM within our decoding protocol by neglecting finite-size effects as well as the additional overhead associated with constructing the syndrome graphs. PYMATCHING uses the *blossom* algorithm [56] for decoding on the complete syndrome graph. For large lattices, its runtime scales with $O(|s|^3 \log_2 |s|)$, where s is the syndrome vector and $|s|$ is the length of the syndrome vector. One decoding path of the concatenated MWPM decoder includes three matching subroutines. The first subroutine performs matching on the restricted graph, where the length of the syndrome vector scales with the number of cells $|s_1| \propto \text{No. of cells}$. The second subroutine performs MWPM on the first monochromatic graph, which includes vertices on faces and cells, meaning $|s_2| \propto \text{No. of cells} + \text{No. of faces}$. Analogously, the third subroutine matches vertices placed on edges and cells and therefore $|s_3| \propto \text{No. of cells} + \text{No. of edges}$. Since the number of cells, faces, and edges scales with d^3 , as summarized in Table I, the total runtime of the MWPM matching decoder scales with $O[d^9 \log(d)]$.

APPENDIX D: LOGICAL ERROR RATE OF LOGICAL QUBITS IN A CUBIC COLOR CODE

Figure 3 shows the logical error rates obtained for logical qubit 1 encoded in a $d = 4, 6, 8$ cubic color code. We achieve a similar pseudthreshold and cross threshold for the remaining two encoded logical qubits, as shown in Fig. 4.

APPENDIX E: OPTIMAL DECODING PATH FOR CUBIC COLOR CODES

Our goal is to reduce the number of decoding paths that have to be evaluated to relax the requirements on computational resources. If decoding paths cannot be evaluated in parallel, fewer decoding paths correspond to shorter runtimes since the number of MWPM subroutines can be decreased. We find that a single decoding path suffices to achieve the optimal subthreshold scaling, considering all three logical qubits in a cubic color code. As an example, Fig. 6 shows all 12 individual decoding paths for each logical qubit $k = 1, 2, 3$ for a distance-6 cubic color code. We can identify single paths, as for example (BG, Y, R) or (RG, Y, B) , that lead to the desired scaling of $p_L \propto (p_{\text{phys}})^3$ for all three logical qubits. Here, all optimal individual decoding paths contain the color green in their restricted graph, which can be attributed to the fact that the joint support of all green stabilizers entails all physical qubits, while it does not for the three other colors R, B , and Y . This means that we effectively localize errors on any physical qubit at an early stage of our decoding process by including green in the first restricted graph. We can always choose an optimal decoding path and therefore reduce the number of required MWPM subroutines. Before, we had to consider two combinations of MWPM on two subsequent monochrome graphs for each of the six restricted graphs, so in total $(1 + 2 + 2) \times 6 = 30$ MWPM instances. For a single decoding path, this can be reduced to $1 + 1 + 1 = 3$. However, this decreases the pseudthreshold from 0.60(6) to 0.42(6)% and the cross threshold from 1.55(6) to 1.02(6)%.

-
- [1] D. Gottesman, Theory of fault-tolerant quantum computation, *Phys. Rev. A* **57**, 127 (1998).
 - [2] D. Aharonov and M. Ben-Or, Fault-tolerant quantum computation with constant error, in *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing* (ACM Press, New York, 1997), pp. 176–188.
 - [3] E. Knill, R. Laflamme, and W. H. Zurek, Resilient quantum computation, *Science* **279**, 342 (1998).
 - [4] J. Preskill, Reliable quantum computers, *Proc. R. Soc. A* **454**, 385 (1998).
 - [5] S. Krinner, N. Lacroix, A. Remm, A. Di Paolo, E. Genois, C. Leroux, C. Hellings, S. Lazar, F. Swiadek, J. Herrmann, *et al.*, Realizing repeated quantum error correction in a distance-three surface code, *Nature (London)* **605**, 669 (2022).
 - [6] R. Acharya, D. A. Abanin, L. Aghababaie-Beni, *et al.* (Google Quantum AI and Collaborators), Quantum error correction below the surface code threshold, *Nature (London)* **638**, 920 (2025).
 - [7] C. Ryan-Anderson, N. Brown, C. Baldwin, J. Dreiling, C. Foltz, J. Gaebler, T. Gatterman, N. Hewitt, C. Holliman, C. Horst, *et al.*, High-fidelity teleportation of a logical qubit using transversal gates and lattice surgery, *Science* **385**, 1327 (2024).
 - [8] C. Ryan-Anderson, J. G. Bohnet, K. Lee, D. Gresh, A. Hankin, J. P. Gaebler, D. Francois, A. Chernoguzov, D. Lucchetti, N. C. Brown, *et al.*, Realization of real-time fault-tolerant quantum error correction, *Phys. Rev. X* **11**, 041058 (2021).
 - [9] B. W. Reichardt, D. Aasen, R. Chao, A. Chernoguzov, W. van Dam, J. P. Gaebler, D. Gresh, D. Lucchetti, M. Mills, S. A. Moses, *et al.*, Demonstration of quantum computation and error correction with a tesseract code, [arXiv:2409.04628](https://arxiv.org/abs/2409.04628).
 - [10] S. Huang, K. R. Brown, and M. Cetina, Comparing Shor and Steane error correction using the Bacon-Shor code, *Sci. Adv.* **10**, eadp2008 (2024).
 - [11] L. Postler, F. Butt, I. Pogorelov, C. D. Marciniak, S. Heußen, R. Blatt, P. Schindler, M. Rispler, M. Müller, and T. Monz, Demonstration of fault-tolerant Steane quantum error correction, *PRX Quantum* **5**, 030326 (2024).
 - [12] N. H. Nguyen, M. Li, A. M. Green, C. Huerta Alderete, Y. Zhu, D. Zhu, K. R. Brown, and N. M. Linke, Demonstration of Shor encoding on a trapped-ion quantum computer, *Phys. Rev. Appl.* **16**, 024057 (2021).
 - [13] Y. Zhao, Y. Ye, H.-L. Huang, Y. Zhang, D. Wu, H. Guan, Q. Zhu, Z. Wei, T. He, S. Cao, *et al.*, Realization of an error-correcting surface code with superconducting qubits, *Phys. Rev. Lett.* **129**, 030501 (2022).
 - [14] D. Bluvstein, S. J. Evered, A. A. Geim, S. H. Li, H. Zhou, T. Manovitz, S. Ebadi, M. Cain, M. Kalinowski, D. Hangleiter, *et al.*, Logical quantum processor based on reconfigurable atom arrays, *Nature (London)* **626**, 58 (2024).
 - [15] I. Pogorelov, F. Butt, L. Postler, C. D. Marciniak, P. Schindler, M. Müller, and T. Monz, Experimental fault-tolerant code switching, *Nat. Phys.* **21**, 298 (2025).

- [16] L. Postler, S. Heußen, I. Pogorelov, M. Rispler, T. Feldker, M. Meth, C. D. Marciniak, R. Stricker, M. Ringbauer, R. Blatt, *et al.*, Demonstration of fault-tolerant universal quantum gate operations, *Nature (London)* **605**, 675 (2022).
- [17] L. Daguerre, R. Blume-Kohout, N. C. Brown, D. Hayes, and I. H. Kim, Experimental demonstration of high-fidelity logical magic states from code switching, *Phys. Rev. X* **15**, 041008 (2025).
- [18] N. Lacroix, A. Bourassa, F. J. Heras, L. M. Zhang, J. Bausch, A. W. Senior, T. Edlich, N. Shutty, V. Sivak, A. Bengtsson, *et al.*, Scaling and logic in the color code on a superconducting quantum processor, *Nature (London)* **645**, 614 (2025).
- [19] D. Bluvstein, A. A. Geim, S. H. Li, S. J. Evered, J. P. B. Ataiades, G. Baranes, A. Gu, T. Manovitz, M. Xu, M. Kalinowski, *et al.*, Architectural mechanisms of a universal fault-tolerant quantum computer, [arXiv:2506.20661](#).
- [20] R. S. Gupta, N. Sundaresan, T. Alexander, C. J. Wood, S. T. Merkel, M. B. Healy, M. Hillenbrand, T. Jochym-O'Connor, J. R. Wootton, T. J. Yoder, *et al.*, Encoding a magic state with beyond break-even fidelity, *Nature (London)* **625**, 259 (2024).
- [21] W. C. Chung, D. C. Cole, P. Gokhale, E. B. Jones, K. W. Kuper, D. Mason, V. Omole, A. G. Radnaev, R. Rines, M. H. Teo, *et al.*, Fault-tolerant operation and materials science with neutral atom logical qubits, *npj Quantum Inf.* **11**, 193 (2025).
- [22] S. Dasu, S. Burton, K. Mayer, D. Amaro, J. A. Gerber, K. Gilmore, D. Gresh, D. DelVento, A. C. Potter, and D. Hayes, Breaking even with magic: Demonstration of a high-fidelity logical non-Clifford gate, [arXiv:2506.14688](#).
- [23] H. Bombin and M. A. Martin-Delgado, Topological quantum distillation, *Phys. Rev. Lett.* **97**, 180501 (2006).
- [24] H. Bombin and M. A. Martin-Delgado, Topological computation without braiding, *Phys. Rev. Lett.* **98**, 160502 (2007).
- [25] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, New York, 2010).
- [26] S. Bravyi and J. Haah, Magic-state distillation with low overhead, *Phys. Rev. A* **86**, 052329 (2012).
- [27] D. Honciuc Menendez, A. Ray, and M. Vasmer, Implementing fault-tolerant non-Clifford gates using the $[[8, 3, 2]]$ color code, *Phys. Rev. A* **109**, 062438 (2024).
- [28] F. Butt, S. Heußen, M. Rispler, and M. Müller, Fault-tolerant code-switching protocols for near-term quantum processors, *PRX Quantum* **5**, 020345 (2024).
- [29] H. Bombín, Dimensional jump in quantum error correction, *New J. Phys.* **18**, 043038 (2016).
- [30] J. T. Anderson, G. Duclos-Cianci, and D. Poulin, Fault-tolerant conversion between the Steane and Reed-Muller quantum codes, *Phys. Rev. Lett.* **113**, 080501 (2014).
- [31] P. Aliferis, D. Gottesman, and J. Preskill, Quantum accuracy threshold for concatenated distance-3 codes, [arXiv:quant-ph/0504218](#).
- [32] Y. Takada, Y. Takeuchi, and K. Fujii, Ising model formulation for highly accurate topological color codes decoding, *Phys. Rev. Res.* **6**, 013092 (2024).
- [33] S.-H. Lee, A. Li, and S. D. Bartlett, Color code decoder with improved scaling for correcting circuit-level noise, *Quantum* **9**, 1609 (2025).
- [34] N. Delfosse, Decoding color codes by projection onto surface codes, *Phys. Rev. A* **89**, 012317 (2014).
- [35] S. Turner, J. Hanish, E. Blanchard, N. Davis, and B. La Cour, A decoder for the color code with boundaries, [arXiv:2003.11602](#).
- [36] C. Gidney and C. Jones, New circuits and an open source decoder for the color code, [arXiv:2312.08813](#).
- [37] C. Chamberland, A. Kubica, T. J. Yoder, and G. Zhu, Triangular color codes on trivalent graphs with flag qubits, *New J. Phys.* **22**, 023019 (2020).
- [38] C. Chamberland, G. Zhu, T. J. Yoder, J. B. Hertzberg, and A. W. Cross, Topological and subsystem codes on low-degree graphs with flag qubits, *Phys. Rev. X* **10**, 011022 (2020).
- [39] A. M. Stephens, Efficient fault-tolerant decoding of topological color codes, [arXiv:1402.3037](#).
- [40] D. S. Wang, A. G. Fowler, C. D. Hill, and L. C. L. Hollenberg, Graphical algorithms and threshold error rates for the 2D colour code, [arXiv:0907.1708](#).
- [41] K. Sahay and B. J. Brown, Decoder for the triangular color code by matching on a Möbius strip, *PRX Quantum* **3**, 010310 (2022).
- [42] A. Kubica and N. Delfosse, Efficient color code decoders in $d \geq 2$ dimensions from toric code decoders, *Quantum* **7**, 929 (2023).
- [43] H. Bombín, Gauge color codes: Optimal transversal gates and gauge fixing in topological stabilizer codes, *New J. Phys.* **17**, 083002 (2015).
- [44] P. Baireuther, M. D. Caio, B. Criger, C. W. Beenakker, and T. E. O'Brien, Neural network decoder for topological color codes with circuit level noise, *New J. Phys.* **21**, 013003 (2019).
- [45] C. T. Chubb, General tensor network decoding of 2D Pauli codes, [arXiv:2101.04125](#).
- [46] L. A. Beni, O. Higgott, and N. Shutty, Tesseract: A search-based decoder for quantum error correction, [arXiv:2503.10988](#).
- [47] P. Sarvepalli and R. Raussendorf, Efficient decoding of topological color codes, *Phys. Rev. A* **85**, 022317 (2012).
- [48] H. Bombin, G. Duclos-Cianci, and D. Poulin, Universal topological phase of two-dimensional stabilizer codes, *New J. Phys.* **14**, 073048 (2012).
- [49] S. Koutsoumpas, T. Noszko, H. Sayginel, M. Webster, and J. Roffe, Colour codes reach surface code performance using vib decoding, [arXiv:2508.15743](#).
- [50] A. W. Senior, T. Edlich, F. J. H. Heras, L. M. Zhang, O. Higgott, J. S. Spencer, T. Applebaum, S. Blackwell, J. Ledford, A. Žemgulytė, A. Židek, N. Shutty, A. Cowie, Y. Li, G. Holland, P. Brooks, C. Beattie, M. Newman, A. Davies, C. Jones, *et al.*, A scalable and real-time neural decoder for topological quantum codes, [arXiv:2512.07737](#).
- [51] M. Walters and M. L. Turner, Minimum weight decoding in the colour code is NP-hard, [arXiv:2603.04234](#).
- [52] A. Kitaev, Fault-tolerant quantum computation by anyons, *Ann. Phys. (NY)* **303**, 2 (2003).
- [53] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Topological quantum memory, *J. Math. Phys.* **43**, 4452 (2002).
- [54] A. G. Fowler, A. M. Stephens, and P. Groszkowski, High-threshold universal quantum computation on the surface code, *Phys. Rev. A* **80**, 052312 (2009).
- [55] S. Bravyi, G. Duclos-Cianci, D. Poulin, and M. Suchara, Subsystem surface codes with three-qubit check operators, [arXiv:1207.1443](#).
- [56] O. Higgott, Pymatching: A PYTHON package for decoding quantum codes with minimum-weight perfect matching, *ACM Trans. Quantum Comput.* **3**, 1 (2022).

- [57] B. J. Brown, N. H. Nickerson, and D. E. Browne, Fault-tolerant error correction with the gauge color code, *Nat. Commun.* **7**, 12302 (2016).
- [58] A. Kubica, M. E. Beverland, F. Brandão, J. Preskill, and K. M. Svore, Three-dimensional color code thresholds via statistical-mechanical mapping, *Phys. Rev. Lett.* **120**, 180501 (2018).
- [59] H. Bombin, R. W. Chhajlany, M. Horodecki, and M.-A. Martin-Delgado, Self-correcting quantum computers, *New J. Phys.* **15**, 055023 (2013).
- [60] A. Kubica and M. E. Beverland, Universal transversal gates with color codes: A simplified approach, *Phys. Rev. A* **91**, 032330 (2015).
- [61] H. Bombin, Transversal gates and error propagation in 3D topological codes, [arXiv:1810.09575](https://arxiv.org/abs/1810.09575).
- [62] A. Kubica, B. Yoshida, and F. Pastawski, Unfolding the color code, *New J. Phys.* **17**, 083026 (2015).
- [63] E. W. Weisstein, Chamfered cube, mathWorld—A Wolfram Web Resource (2024), <https://mathworld.wolfram.com/ChamferedCube.html>.
- [64] F. Butt, I. Pogorelov, R. Freund, A. Steiner, M. Meyer, T. Monz, and M. Müller, Demonstration of measurement-free universal logical quantum computation, *Nat. Commun.* **17**, 995 (2026).
- [65] J. Edmonds, Paths, trees, and flowers, *Can. J. Math.* **17**, 449 (1965).
- [66] A. G. Fowler, Optimal complexity correction of correlated errors in the surface code, [arXiv:1310.0863](https://arxiv.org/abs/1310.0863).
- [67] A. Paler and A. G. Fowler, Pipelined correlated minimum weight perfect matching of the surface code, *Quantum* **7**, 1205 (2023).
- [68] M. Li, M. Gutiérrez, S. E. David, A. Hernandez, and K. R. Brown, Fault tolerance with bare ancillary qubits for a $[[7,1,3]]$ code, *Phys. Rev. A* **96**, 032341 (2017).
- [69] S. Heußen, D. Winter, M. Rispler, and M. Müller, Dynamical subset sampling of quantum error-correcting protocols, *Phys. Rev. Res.* **6**, 013177 (2024).
- [70] A. Paetznick and B. W. Reichardt, Fault-tolerant ancilla preparation and noise threshold lower bounds for the 23-qubit Golay code, [arXiv:1106.2190](https://arxiv.org/abs/1106.2190).
- [71] K. M. Svore, A. W. Cross, I. L. Chuang, and A. V. Aho, A flow-map model for analyzing pseudothresholds in fault-tolerant quantum computing, [arXiv:quant-ph/0508176](https://arxiv.org/abs/quant-ph/0508176).
- [72] C. Chamberland, T. Jochym-O'Connor, and R. Laflamme, Thresholds for universal concatenated quantum codes, *Phys. Rev. Lett.* **117**, 010501 (2016).
- [73] G. Wills, Visualization toolkit software, *WIREs Computat. Stats.* **4**, 474 (2012).
- [74] C. B. Sullivan and A. Kaszynski, PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK), *J. Open Source Software* **4**, 1450 (2019).
- [75] <https://pypi.org/project/qCodePlot3D>.
- [76] <https://github.com/larsesser/qcodeplot3d>.
- [77] R. Chao and B. W. Reichardt, Quantum error correction with only two extra qubits, *Phys. Rev. Lett.* **121**, 050502 (2018).
- [78] A. M. Steane, Active stabilization, quantum computation, and quantum state synthesis, *Phys. Rev. Lett.* **78**, 2252 (1997).
- [79] C. Gidney, Stim: A fast stabilizer circuit simulator, *Quantum* **5**, 497 (2021).
- [80] N. Metropolis and S. Ulam, The Monte Carlo method, *J. Am. Stat. Assoc.* **44**, 335 (1949).
- [81] G. Rubino and B. Tuffin, *Rare Event Simulation Using Monte Carlo Methods* (Wiley, New York, 2009).
- [82] A. J. Landahl, J. T. Anderson, and P. R. Rice, Fault-tolerant quantum computing with color codes, [arXiv:1108.5738](https://arxiv.org/abs/1108.5738).
- [83] Oscar Higgott, <https://github.com/oscarhiggott/PyMatching>.