

Low-Overhead Magic State Distillation with Color Codes

Seok-Hyung Lee^{1,*}, Felix Thomsen¹, Nicholas Fazio¹, Benjamin J. Brown^{2,3} and Stephen D. Bartlett^{1,†}

¹Centre for Engineered Quantum Systems, School of Physics, *The University of Sydney*, Sydney, New South Wales 2006, Australia

²T. J. Watson Research Center, *IBM Quantum*, Yorktown Heights, New York 10598, USA

³*IBM Denmark*, Sundkrogsgade 11, Copenhagen 2100, Denmark

 (Received 30 September 2024; revised 21 March 2025; accepted 30 May 2025; published 30 July 2025)

Fault-tolerant implementation of non-Clifford gates is a major challenge for achieving universal fault-tolerant quantum computing with quantum error-correcting codes. Magic state distillation is the most well-studied method for this but requires significant resources. Hence, it is crucial to tailor and optimize magic state distillation for specific codes from both logical- and physical-level perspectives. In this work, we perform such optimization for two-dimensional color codes, which are promising due to their higher encoding rates compared to surface codes, transversal implementation of Clifford gates, and efficient lattice surgery. We propose two carefully designed distillation schemes based on the 15-to-1 distillation circuit and lattice surgery, differing in their methods for handling faulty rotations. Our first scheme employs faulty T measurement, achieving infidelities of $O(p^3)$ for physical noise strength p . To achieve lower infidelities, our second scheme integrates distillation with “cultivation” (a distillation-free approach to fault tolerantly prepare magic states through transversal Clifford measurements). Our second scheme achieves significantly lower infidelities (e.g., approximately 2×10^{-16} at $p = 10^{-3}$), surpassing the capabilities of both cultivation and single-level distillation. Notably, to reach a given target infidelity, our schemes require approximately 2 orders of magnitude fewer resources than the previous best magic-state-distillation schemes for color codes.

DOI: [10.1103/ch5r-cnfg](https://doi.org/10.1103/ch5r-cnfg)

I. INTRODUCTION

A large-scale quantum computer should possess two properties: *universality* and *fault tolerance*. Universality refers to the capability of applying any unitary operations on qubits with arbitrary accuracy. This is achievable if the computer can implement a universal set of gates, which commonly includes the Hadamard gate (H), phase gate ($S = Z^{1/2}$), controlled-NOT (CNOT) gate, and T gate ($T = Z^{1/4}$) [1]. While there is flexibility in this choice of gate set, for universality it must contain at least one non-Clifford gate, such as the T gate in this example. Fault tolerance is essential due to the noisy physical environments that may corrupt quantum information stored in qubits. Stabilizer quantum error-correcting (QEC) codes can be

employed for fault-tolerant quantum memories (i.e., idling operations), provided that the physical noise strength is sufficiently low. However, implementing every gate in a universal set of gates fault tolerantly remains a far more challenging task.

Two-dimensional (2D) color codes [2,3] are a family of stabilizer QEC codes defined on a trivalent and three-colorable lattice of qubits, which can be constructed using only local interactions. Compared with surface codes [4,5], color codes use fewer physical qubits per logical qubit at a given code distance [6] and have richer topological structures [7]. This allows an arbitrary pair of multiqubit Pauli operators to be measured in parallel via lattice surgery [8]. Additionally, the seven-qubit Steane code, a small example of a color code with distance 3, has been recently demonstrated experimentally [9–12], exhibiting not only memory experiments but also the implementation of nontrivial gates. Furthermore, a recent experiment [13] successfully demonstrated that the distance-5 color code provides better logical error suppression than the distance-3 color code.

With color codes, logical Clifford gates can be fault tolerantly implemented efficiently using transversal gates [2,14] or lattice surgery [7,8,15]. However, the

*Contact author: seokhyung.lee@sydney.edu.au

†Contact author: stephen.bartlett@sydney.edu.au

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.

fault-tolerant implementation of logical non-Clifford gates is rather challenging. One well-studied method to achieve this is by employing magic state distillation (MSD) [16–22], a scheme that distills high-quality logical magic states (specific nonstabilizer states) from multiple faulty magic states. The distilled magic states are then consumed to implement desired non-Clifford gates on logical qubits. For instance, the state $|\overline{A}\rangle := |\overline{0}\rangle + e^{i\pi/4}|\overline{1}\rangle$, where $|\overline{0}\rangle$ and $|\overline{1}\rangle$ are the logical basis states, can be used to execute the logical T gate or, more generally, any $\pi/8$ -rotation gate $\overline{P}_{\pi/8} := e^{-i(\pi/8)\overline{P}}$, where $\overline{P}$ is a logical Pauli operator on multiple logical qubits. (Throughout the paper, we omit normalization factors of quantum states unless necessary and use overlines to denote logical states or operators.)

MSD is generally resource intensive due to its demand for a significant number of logical Clifford gates between multiple logical qubits. Therefore, when applying it to a specific quantum error-correcting code, the procedure should be tailored and optimized for that code at both the logical and physical levels, in order to minimize its resource cost. Such adaptations have been well studied for surface codes, notably by Litinski [23], reducing its cost by orders of magnitude compared to previous proposals. The scheme in Ref. [23] is based on two key ideas: (i) distillation circuits can be designed in a way that many faulty $\pi/8$ -rotations are executed on a small number of logical qubits (e.g., 15 $\pi/8$ -rotations on five logical qubits in the 15-to-1 MSD protocol), and (ii) ancillary logical qubits for the scheme can have smaller code distances than the code distance d_{out} of the logical qubit outputting the distilled magic state. More specifically, the code distance d_Z with respect to $\overline{Z}$ errors on ancillary logical qubits can be chosen to be smaller than d_{out} , since such $\overline{Z}$ errors are detectable by the final $\overline{X}$ measurements of the circuit.

In this work, we propose two resource-efficient MSD schemes for two-dimensional (2D) color codes based on the 15-to-1 MSD protocol, by adapting and developing Litinski’s ideas applied to surface codes. Our first scheme employs lattice surgery and faulty T measurement as its main components. The distillation circuit consists of 15 $\pi/8$ -rotations on five logical qubits (initialized to $|\overline{+}\rangle^{\otimes 5}$), followed by $\overline{X}$ measurements on four of them (referred to as *validation qubits*). Provided that all the $\overline{X}$ measurements yield +1, the distilled magic state is obtained from the unmeasured logical qubit. To execute each $\pi/8$ -rotation, we use an auxiliary logical qubit, which undergoes lattice surgery with other logical qubits, followed by a non-fault-tolerant measurement in the basis of $|\overline{0}\rangle \pm e^{-i\pi/4}|\overline{1}\rangle$, referred to as a faulty T measurement. We carefully design a layout for the scheme from both macroscopic and microscopic perspectives and verify that it is fault tolerant, meaning it preserves code distances during lattice surgery.

We optimize the resource cost of the scheme using various approaches. Notably, we show that, not only are $\overline{Z}$

errors on validation qubits tolerable (cf. Ref. [23]), but any single-location $\overline{X}$ error on a validation qubit is also tolerable, although it may incur three or more rotation errors. Consequently, both of the distances d_X and d_Z (for $\overline{X}$ and $\overline{Z}$ errors, respectively) can be set lower than d_{out} . Additionally, we leverage the rich structure of color codes for optimization. For instance, color codes allow the measurement of an arbitrary pair of commuting Pauli operators in parallel [8], making it sufficient to use a single ancillary region surrounded by all the logical patches, including two auxiliary patches. Furthermore, domain walls for lattice surgery can be shortened by using rectangular patches instead of standard triangular patches.

Our first scheme using faulty T measurement has an inherent limitation, as its output logical error rate cannot be lower than approximately $35p_{\text{FT}}^3$, where $p_{\text{FT}} = O(p)$ denotes the error rate of the faulty T measurement and p is the physical error rate. To address this issue, we develop our second MSD scheme that employs a two-level approach. In this scheme, we integrate our first scheme with recent distillation-free magic state preparation protocols, such as those described in Refs. [24–26], referred to as *magic state cultivation* [26]. These protocols leverage the transversality of logical Clifford gates in color codes for directly preparing magic states fault tolerantly. They are highly resource efficient since they avoid logical operations on multiple logical qubits. However, they have a fundamental scalability issue due to their heavy reliance on postselection, which imposes a practical lower bound on achievable infidelity. To reach lower infidelities, our approach is to use magic states from cultivation as inputs for MSD, with appropriate optimization. The growing operation on cultivated magic states can be a bottleneck that significantly hinders fault tolerance, but we demonstrate that this issue can be mitigated by incorporating postselection during decoding with the concatenated minimum-weight perfect matching (MWPM) decoder [27] (the circuit-level matching-based decoder for color codes with the best known subthreshold scaling).

We assess the performance of our schemes based on their output infidelities and resource costs, assuming that errors are corrected using the recently proposed concatenated MWPM decoder. Compared to previous MSD schemes for color codes, such as the one in Ref. [22], our schemes achieve a spacetime cost that is approximately 2 orders of magnitude lower for a given target infidelity. For instance, at $p = 10^{-3}$, a magic state with an infidelity of 10^{-9} can be prepared using around 15 000 qubits over 1400 time steps, with a failure rate of approximately 0.2%, resulting in an effective spacetime cost of about 2.1×10^7 (in contrast to 6.6×10^9 in Ref. [22]). Compared to the latest cultivation scheme [26], although our schemes do not surpass its high resource efficiency, they can achieve significantly lower infidelities (e.g., approximately $2 \times$

10^{-16} at $p = 10^{-3}$) than those attainable with cultivation ($\gtrsim 10^{-9}$).

This paper is structured as follows: In Sec. II, we present preliminaries on 2D color codes, including their definitions, error-detecting properties, anyon model (with descriptions of domain walls), and methods for executing logical operations. In Sec. III, we describe our MSD scheme using faulty T measurement and lattice surgery, along with our methodology for optimizing its resource cost. In Sec. IV, we modify the scheme to achieve lower infidelities by integrating cultivation. In Sec. V, we analyze the performance of our schemes based on their output infidelities and resource costs, and compare them with other approaches. We conclude with final remarks in Sec. VI.

II. TWO-DIMENSIONAL COLOR CODES

In this section, we briefly review the basics of the 2D color codes including definitions, stabilizer structures, and anyon model.

A. Definition and stabilizer structure

A 2D color-code lattice [2] is defined as a trivalent and three-colorable lattice; that is, each vertex of the lattice is connected with three edges such that one of the three colors (red, green, or blue) can be assigned to each face in a way that adjacent faces do not have the same color. Each edge of the lattice can be given the same color as that of the two faces that are connected by the edge. The hexagonal (6-6-6) lattice in Fig. 1 is one of its representative examples and we will consider it throughout this paper.

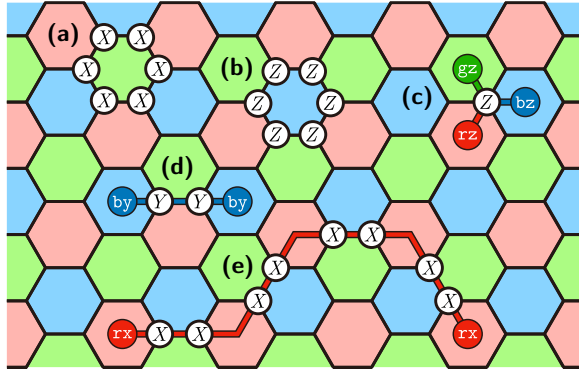


FIG. 1. Hexagonal color code lattice with X - and Z -type checks on each face, exemplified in (a) and (b), respectively. (c) A single-qubit Pauli error flips surrounding three checks, creating a triplet of bosons with different colors. (d) A pair of Pauli errors with the same type on an edge flips two checks on the faces that it connects, creating a pair of bosons with the same type. (e) A string operator of a specific type (“red Pauli X ” in the example) flips a pair of checks located at its ends, creating a pair of bosons with that type (rx).

For a given 2D color code lattice, a 2D color code is defined with a qubit placed on each vertex of the lattice. It has two types of checks (or stabilizer generators) S_f^X and S_f^Z for each face f , defined as

$$S_f^X := \prod_{v \in f} X_v, \quad S_f^Z := \prod_{v \in f} Z_v,$$

where X_v and Z_v are the Pauli- X and Z operators, respectively, on the qubit placed at v , and “ $v \in f$ ” denotes that the vertex v is included in f . In other words, the code space is the common $+1$ eigenspace of these operators. We refer to S_f^X , S_f^Z , and $S_f^X S_f^Z$ as the X -, Z -, and Y -type checks on face f , respectively. Examples of X - and Z -type checks are shown in Figs. 1(a) and 1(b), respectively.

B. Detection of Pauli errors

Pauli errors flip the eigenvalues of checks. For instance, a Pauli error on a qubit flips the checks anticommuting with the error on the surrounding three faces (with different colors), as exemplified in Fig. 1(c). If two qubits connected by an edge simultaneously undergo Pauli errors of the same type, the corresponding checks on the two faces (with the same color) connected by the edge are flipped, as shown in Fig. 1(d). Errors on multiple consecutive edges of the same color can form a string operator, which flips the two checks located at its two endpoints. For example, as shown in Fig. 1(e), a red Pauli- X string operator flips two Z -type checks on the red faces at which the string operator terminates. A general string-net operator is composed of a combination of string operators (with different types) connected by single-qubit errors, which may flip multiple checks located at its ends. Note that Pauli- Y string operators can be interpreted as pairs of overlapped Pauli- X and Z string operators; thus, a string-net operator may also contain string operators of different Pauli types as well as different colors.

Flips of checks are commonly described using the language of anyon theory. The color-code phase contains nine nontrivial types (termed *charge labels*) of bosons: rx, ry, rz, gx, gy, gz, bx, by, and bz. Each boson has a color label (r, g, or b) and a Pauli label (x, y, or z). These nine bosons generate the entire group of anyons. A set of flipped checks on a face is associated with a boson that has a color label corresponding to the color of the face and a Pauli label corresponding to the operator needed to flip the checks. For example, if only the X -type check on a red face is flipped, an rz boson is created. If both the X - and Z -type checks on a green face are flipped, a gy boson is created. Figure 1 visualizes some examples of bosons created by Pauli errors. For instance, a red Pauli- X string operator creates two rx bosons at its ends, or equivalently, it transfers an rx boson from one end to the other. Throughout

the paper, we refer to a red Pauli- X string operator simply as an rx -string operator, and similarly for the other types of string operators. The interactions between bosons via string(-net) operators can be expressed in terms of the fusion rules:

$$\begin{aligned} rx \times rx &= ry \times ry = \dots = 1 \quad (\text{holds for all bosons}), \\ rx \times gx \times bx &= ry \times gy \times by = rz \times gz \times bz = 1, \\ rx \times ry \times rz &= gx \times gy \times gz = bx \times by \times bz = 1. \end{aligned}$$

By using these properties, errors can be predicted from check measurement outcomes.

The above discussion is limited to the case when there are no errors during syndrome extraction. If such errors are considered, we need to measure checks repeatedly and the product of two consecutive check measurement outcomes (called *detectors*) replace the roles of checks. That is, an X or Z error on a data qubit flips three spatially adjacent detectors and a measurement error of a check flips two temporarily adjacent detectors. The anyon theory can be natively extended to this spacetime picture; for example, consecutive measurement errors on a red Z -type check moves an rx boson in the temporal direction.

See Ref. [7] for more details on the anyon theory of color codes.

C. Logical qubits encoded in patches

Logical qubits can be encoded in a color code in various ways [2,28,29] and one representative method is to use a “patch” surrounded by boundaries [2]. A boundary of a specific color (red, green, or blue) is created by cutting the lattice, ensuring that the boundary is incident to only edges of that color (namely, it touches only faces of the other two colors). By definition, a string operator of a specific color can terminate at a boundary of that color, or equivalently, in terms of the anyon theory, bosons of that color label can be absorbed by the boundary (this is also known as “condensation” [7]). We consider two of the simplest types of logical patches, triangular and rectangular patches, and refer to the corresponding codes as triangular and rectangular color codes, respectively.

A triangular logical patch is surrounded by three boundaries of different colors, as exemplified in Fig. 2(a). It encodes a single logical qubit, whose logical Pauli- X and Z operators $\bar{X}$, $\bar{Z}$ are, respectively, Pauli- X and Z string-net operators terminating at the three boundaries. These logical Pauli operators are also schematically drawn in Fig. 2(b) for a general triangular logical patch. Here, solid and dotted lines represent Pauli- Z and X string-net operators, respectively, and we will use these notations throughout this work. Note that the string-net operators can be moved to one of the three boundaries, meaning that we can define $\bar{X}$ and $\bar{Z}$ to be supported on qubits placed along

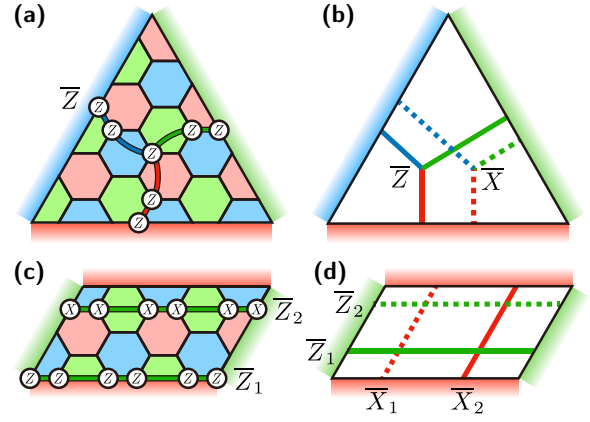


FIG. 2. (a) Triangular logical patch with distance 7, which encodes a single logical qubit. Its logical Pauli- Z (X) operator $\bar{Z}$ ($\bar{X}$) is the Pauli- Z (X) string-net operator terminating at the three boundaries. (b) Schematic diagram of a general triangular color code. Solid (dotted) lines represent Pauli- Z (X) string-net operators. (c) Rectangular logical patch with code distances of $d_X = 4$ and $d_Z = 6$ (respectively, for logical Pauli- X and Z errors), which encodes two logical qubits. Its two logical Pauli- Z (X) operators $\bar{Z}_1$, $\bar{Z}_2$ ($\bar{X}_1$, $\bar{X}_2$) are, respectively, gz - and gx -string (rx - and rz -string) operators terminating at the two opposite green (red) boundaries. (d) Schematic diagram of a general rectangular color code with red and green boundaries. Similarly, solid (dotted) lines represent Pauli- Z (X) string operators.

the boundary. The code distance d of the code (i.e., minimum weight of $\bar{X}$ and $\bar{Z}$) is odd since $\bar{X}$ and $\bar{Z}$ may have the same support and should anticommute with each other.

A rectangular logical patch is surrounded by two pairs of parallel single-colored boundaries, where the two pairs have different colors, as exemplified in Fig. 2(c). It encodes two logical qubits with logical Pauli operators $\bar{X}_1$, $\bar{Z}_1$ and $\bar{X}_2$, $\bar{Z}_2$. $\bar{Z}_1$ and $\bar{Z}_2$ are, respectively, defined as Pauli- Z and X string operators terminating at a specific pair of opposite boundaries (green in the example), while $\bar{X}_1$ and $\bar{X}_2$ are Pauli- X and Z string operators terminating at the other pair of boundaries (red in the example), which are drawn schematically in Fig. 2(d). The two logical qubits may have different code distances d_X and d_Z (which are even) for $\bar{X}$ and $\bar{Z}$ errors, determined by the lengths of the boundaries. We emphasize that $\bar{X}_2$ and $\bar{Z}_2$ are defined with opposite labels from their physical Pauli types (i.e., Z for $\bar{X}_2$ and X for $\bar{Z}_2$), which is to ensure that the two logical qubits have the same pair of code distances (d_X , d_Z) for logical X and Z errors. If this is not desirable (due to biased physical noise, for example), one can define the patch using Pauli boundaries [8,30] instead of color boundaries. We will not consider this in our schemes, but modifying them to use rectangular patches with Pauli boundaries would not be difficult and would not significantly change their resource costs or performance.

While triangular patches are sufficient in many cases, rectangular patches can be particularly useful when logical noise is biased (i.e., a specific type of logical Pauli error is dominant or more harmful), such as in MSD (see Sec. III A) or magic state injection [31], where independent adjustment of d_X and d_Z is beneficial. Moreover, using rectangular patches may help reduce resource costs. When performing lattice surgery, the region between a logical patch and the ancillary region for lattice surgery can be reduced by half by using a rectangular patch instead of a triangular patch, though this comes at the cost of sacrificing the ability to perform any Pauli measurements. See Sec. III D for more details.

D. Logical operations

We first consider initializing or measuring a logical qubit fault tolerantly in a Pauli basis. To initialize a triangular color-code qubit to $|\bar{0}\rangle$, we just need to initialize every physical qubit to $|0\rangle$ and measure the checks. Assuming no errors, Z -type checks deterministically give +1, while the values of X -type checks are randomly determined, which does not matter since products of two consecutive check outcomes are used for decoding. To measure the logical qubit in the $\bar{Z}$ basis, we need to measure every physical qubit in the Z basis, which gives the measurement outcome as the product of the outcomes on qubits along one of the three boundaries. For each face, the product of the Z -measurement outcomes of qubits belonging can be used to infer the value of the final Z -type check outcome. In the case that measurements are noisy, the single-qubit measurements form a detector together with the previous Z -type check outcome. Note that the above processes correspond to placing Z -type temporal boundaries, which condense rz, gz, and bz bosons [7]. Initialization and measurement to other Pauli bases can be done analogously.

For the rectangular color-code qubit in Fig. 2(c), we can measure $\bar{Z}_1$ ($\bar{Z}_2$) by measuring $\bar{Z} \otimes \bar{Z}$ ($\bar{X} \otimes \bar{X}$) on every green edge. Operators $\bar{Z}_1$ and $\bar{Z}_2$ can be measured at the same time by measuring every green edge in the Bell basis, which corresponds to placing a green temporal boundary condensing gx, gy, and gz bosons. Note that we can obtain the values of red and blue checks that commute with these Bell bases. $\bar{X}_1$ and $\bar{X}_2$ can be measured similarly by measuring red edges.

Next, we consider logical Clifford operations. For triangular color codes, the logical Hadamard, phase, and CNOT gates (which generate the Clifford group) can be implemented transversally. For example, the logical Hadamard gate $\bar{H}$ can be done just by applying physical Hadamard gates on all the physical qubits in the patch. The logical phase gate $\bar{S}$ is similar but a little more tricky since some qubits undergo S while the other qubits undergo $S^\dagger$. For the logical CNOT gate, we just need to apply a physical CNOT gate on every pair of corresponding physical

qubits in the two logical patches, although the two patches need to be stacked in three-dimensional space if only local interactions are available. See Ref. [14] for more details.

The above methods (except for the CNOT gate) may not generally work for other types of logical patches including rectangular ones. Alternatively, we can use nondestructive multiqubit Pauli measurements instead of Clifford gates as ingredients for universal quantum computing. In other words, given a quantum circuit composed of Clifford and T gates, we can commute all the Clifford gates to the end of the circuit (while transforming T gates appropriately) and merge them with final measurements, which yields a series of Pauli measurements. In color-code logical patches, Pauli measurements can be performed by using lattice surgery [6–8], which is a process to merge the boundaries of multiple checks with additional check operators. We will elaborate on detailed schemes for this in Sec. III D.

We lastly need non-Clifford gates. In particular, the $\pi/8$ -rotation gate $\bar{P}_{\pi/8}$ should be executable for any logical Pauli operator $\bar{P}$. (Throughout this work, we denote $P_\theta := e^{-i\theta P}$ for a Pauli operator P and a real number θ .) Two circuits for this operation [23,32] are presented in Figs. 3(a) and 3(b), which consist of Pauli measurements

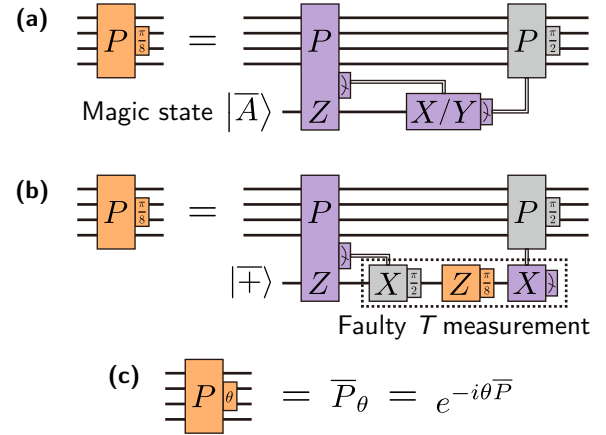


FIG. 3. Two circuits for implementing the $\pi/8$ -rotation gate $\bar{P}_{\pi/8} := e^{-i(\pi/8)\bar{P}}$ for a logical multiqubit Pauli operator $\bar{P}$, respectively, employing (a) a logical magic state $|\bar{A}\rangle := |\bar{0}\rangle + e^{i\pi/4}|\bar{1}\rangle$ and (b) faulty T measurement, which are expressed using Pauli measurements (purple boxes) and Pauli rotation gates (orange or gray boxes) defined in (c). In (a), $|\bar{A}\rangle$ is first prepared on an auxiliary logical qubit. $\bar{P} \otimes \bar{Z}$ is then measured jointly on the input and auxiliary qubits, which determines the basis (X if +1 and Y if -1) for the following measurement of the auxiliary qubit. If this measurement returns -1, a Pauli correction of $\bar{P}$ should be applied. In (b), $|\bar{+}\rangle := |\bar{0}\rangle + |\bar{1}\rangle$ is prepared on an auxiliary logical qubit, followed by a $\bar{P} \otimes \bar{Z}$ measurement on the input and auxiliary qubits, which returns $\lambda = \pm 1$. The auxiliary logical qubit is then measured in the basis of $\{|\bar{0}\rangle \pm e^{-i\lambda\pi/4}|\bar{1}\rangle\}$ via a faulty T measurement. If this measurement returns -1, a Pauli correction of $\bar{P}$ should be applied as well.

(represented by purple boxes) and Pauli rotation gates (represented by orange or gray boxes) defined in Fig. 3(c). Here, $\pi/2$ -rotation gates, which are simply Pauli gates with a global phase change, are shown as gray boxes to emphasize that they do not need to be applied explicitly but only change the Pauli frame [33]. The circuit in Fig. 3(a) consumes a magic state

$$|\overline{A}\rangle := |\overline{0}\rangle + e^{i\pi/4} |\overline{1}\rangle \quad (1)$$

with a joint Pauli measurement $\overline{P} \otimes \overline{Z}$ (returning λ), followed by a measurement in the basis of $\overline{X}$ (for $\lambda = 1$) or $\overline{Y}$ (for $\lambda = -1$). If the second measurement returns -1 , a Pauli correction $\overline{P}$ should be applied to the remaining qubits. By reversing the time order of this circuit, we obtain the second circuit in Fig. 3(b). Here, the auxiliary qubit is prepared to $|\overline{+}\rangle$, undergoes a $\overline{P} \otimes \overline{Z}$ measurement jointly with the input qubits (which returns λ), and then measured in the basis of nonstabilizer states $\{|\overline{0}\rangle \pm e^{-i\lambda\pi/4} |\overline{1}\rangle\}$, making a Pauli correction on the remaining qubits. Note that the circuit in Fig. 3(a) can be rewritten in a way that the measurement basis for the auxiliary qubit is fixed to $\overline{X}$ and a Clifford correction is applied to the other qubits. However, we avoid using this construction since a Clifford correction may flip the angles of the following rotation gates in the MSD circuit, which is not desirable.

These two circuits take an approach of concentrating the “non-Clifford part” into the initialization or measurement of the auxiliary logical qubit, which is thus the core of the circuits that determine their fault tolerance. A logical magic state can be prepared from physical magic states by using state injection. In a color-code logical patch, this can be done by preparing the physical state on a specific physical qubit in the patch and measuring other qubits appropriately, followed by ordinary check measurements [22]. Faulty T measurement is the reverse process: reducing a logical patch into a single physical qubit and then measuring it in the desired basis. This is one of the key ingredients of our MSD scheme, thus we will describe this in detail in Sec. III B. Both state injection and faulty T measurement are not fault tolerant, namely, a single physical error during the protocols may cause a logical failure. However, MSD protocols can be employed to extract high-quality magic states from multiple trials of non-fault-tolerant $\pi/8$ -rotations [16], which can be inputted in the circuit of Fig. 3(a) for executing fault-tolerant $\pi/8$ -rotations. Alternatively, transversal logical Clifford gates on color codes can be utilized to construct distillation-free magic state preparation protocols [24,25,34,35]. We will discuss these methods in more details in Sec. IV

E. Domain walls

We now briefly review domain walls, which are key ingredients for lattice surgery of color codes. Domain walls

are one-dimensional (1D) subregions along which two topological phases interface. In simple terms, a domain wall of a color code is a thin region in which checks are deformed from ordinary color-code checks under a specific rule so that bosons approaching the domain wall are affected in a particular way. Such effects are determined by the charge labels of the bosons ($rx, ry, \dots, bz$) and the directions in which they approach. Bosons located in one of the two regions separated by the domain wall can be classified as follows:

- (1) **Condensed:** The boson is condensed on the domain wall; that is, the domain wall acts as a boundary of the corresponding type for the boson (such as a red boundary for an rx boson) so that the boson can be absorbed at the domain wall.
- (2) **Deconfined:** The boson can freely move across the domain wall to the other side, but its charge label can be changed.
- (3) **Confined:** The boson is confined to the region; that is, it cannot pass through or be condensed on the domain wall.

Note that bosons can be classified differently for either side of the domain wall. Based on these features of domain walls, they can be categorized into three types: *opaque*, *transparent*, and *semitransparent* domain walls [7].

1. Opaque domain wall

An opaque domain wall is a trivial one that does not have any deconfined bosons. In other words, it is just an empty region between two color codes with boundaries. Each side of the domain wall is a color or Pauli boundary, which condenses bosons that have the corresponding color or Pauli label (e.g., if it is a red boundary, it condenses rx , ry , and rz bosons).

2. Transparent domain wall

A transparent (or invertible) domain wall [36] allows every boson to be deconfined. The charge labels of bosons are permuted according to certain rules when they move across the domain wall. The rule is always symmetric; namely, if a boson a is transformed into a' in one direction, a' transforms into a in the opposite direction. Hence, the charge-changing rule can be characterized by a permutation of nine charge labels of bosons. Note that only 72 permutations are allowed considering the fusion and braiding rules of bosons [30]. Each of them can be expressed as a combination of a color permutation (between r , g , and b), a Pauli permutation (between x , y , and z), and an exchange between the color and Pauli labels ($r \leftrightarrow x$, $g \leftrightarrow y$, $b \leftrightarrow z$), which leads to a total $6 \times 6 \times 2 = 72$ permutations.

3. Semitransparent domain wall

A semitransparent domain wall allows only some bosons to be deconfined. Each side of the domain wall condenses bosons with one particular charge label, which can be different for the two sides. Bosons that have the same color (Pauli) label as the condensed boson are deconfined in the region and referred to as *electric (magnetic)* charges for that side of the domain wall. Importantly, the charge-changing rule of deconfined bosons applied when they move across the domain wall is either “em preserving” or “em exchanging”; namely, if it is em preserving (em exchanging), an electric charge of one side is transformed into an electric (magnetic) charge of the other side, and vice versa for a magnetic charge. Note that it is not a one-to-one correspondence and the transformed charge can be any of two types of electric (magnetic) charges.

For example, let us suppose that a semitransparent domain wall condenses rx on side A and gz on side B and its charge-changing rule is em exchanging. Then ry , rz , gx , and bx (gx , gy , rz , and bz) bosons are deconfined in side A (B), where the first two are electric charges and the latter two are magnetic charges. Since it is em exchanging, an electric charge (ry or rz) moving across the domain wall from side A is transformed into any of magnetic charges (rz and bz) in the other side, and vice versa for a magnetic charge in side A. The other bosons gy , gz , by , and bz (rx , ry , bx , and by) are confined to the region that side A (B) belongs to.

The effects of a semitransparent domain wall on bosons can be intuitively described by using a boson table [30]

rx	gx	bx
ry	gy	by
rz	gz	bz

where the nine charge labels of bosons are aligned in a 3×3 table. On this table, we mark condensed bosons with a bullet ($\bullet$), confined bosons with a cross ($\times$), electric charges with a square ($\blacksquare$), and magnetic charges with a triangle ($\blacktriangle$). With these notations, the domain wall in the above example can be expressed as a pair of boson tables:

$\bullet$	$\blacktriangle$	$\blacktriangle$
$\blacksquare$	$\times$	$\times$
$\blacksquare$	$\times$	$\times$

 $\xleftrightarrow{A \rightarrow B}$

$\times$	$\blacksquare$	$\times$
$\times$	$\blacksquare$	$\times$
$\blacktriangle$	$\bullet$	$\blacktriangle$

where the colors (orange or sky blue) of the squares and triangles indicate the charge-changing rule of the domain wall; namely, bosons with the same symbol color on both sides can be mapped.

III. SINGLE-LEVEL 15-TO-1 MAGIC-STATE-DISTILLATION SCHEME

In this section, we introduce our 15-to-1 MSD scheme for color codes, which use lattice surgery and faulty T

measurement as its basic ingredients. We first present the distillation circuit based on the 15-qubit Reed-Muller code [16,19], which is modified to contain 15 $\pi/8$ -rotations on five logical qubits [23,32]. We then describe its color-code implementation from both macroscopic and microscopic perspectives.

We consider the circuit-level noise model with strength p , which is defined as follows:

- Every measurement outcome is flipped with probability p .
- Every preparation of a qubit produces an orthogonal state with probability p .
- Every single- or two-qubit unitary gate (including the idle gate I) is followed by a single- or two-qubit depolarizing noise channel of strength p . We here regard that, for every time step of the circuit, idle gates I are acted on all the qubits that are not involved in any nontrivial unitary gates or measurements.

Here, the single- and two-qubit depolarizing channels of strength p are, respectively, defined as

$$\begin{aligned} \mathcal{E}_p^{(1)} : \rho^{(1)} &\mapsto (1-p)\rho^{(1)} + \frac{p}{3} \sum_{P \in \{X, Y, Z\}} P \rho^{(1)} P, \\ \mathcal{E}_p^{(2)} : \rho^{(2)} &\mapsto (1-p)\rho^{(2)} \\ &+ \frac{p}{15} \sum_{\substack{P_1, P_2 \in \{I, X, Y, Z\} \\ P_1 \otimes P_2 \neq I \otimes I}} (P_1 \otimes P_2) \rho^{(2)} (P_1 \otimes P_2), \end{aligned}$$

where $\rho^{(1)}$ and $\rho^{(2)}$ are arbitrary single- and two-qubit density matrices, respectively.

A. Distillation circuit

We consider the 15-to-1 MSD circuit in Fig. 4 that distills one high-fidelity logical magic state $|\bar{0}\rangle + e^{-i\pi/4} |\bar{1}\rangle = e^{-i\pi/4} \bar{X} |\bar{A}\rangle$ from 15 faulty $\pi/8$ -rotations. Five logical qubits, which consist of one *output qubit* and four *validation qubits* (respectively, A–D), are initialized to $|\bar{+}\rangle$ and undergo 15 faulty $\pi/8$ -rotations. Each of the rotations is performed via one of the two circuits in Figs. 3(a) and 3(b). Although this choice does not make significant difference, we select using the circuit in Fig. 3(b) that contains a faulty T measurement, as the $\bar{Y}$ measurement in Fig. 3(a) makes decoding slightly difficult. ($\bar{Y}$ measurement outcomes involve both decoding graphs, respectively, for X and Z errors, thus the two graphs need to be connected.) Note that all the $\pi/8$ -rotations contain only $\bar{Z}$ in their rotation bases, thus they all commute with each other. After that, the four validation qubits are measured in the $\bar{X}$ basis. If all of these outcomes are +1, we conclude that the protocol succeeds and use the marginal state on the output

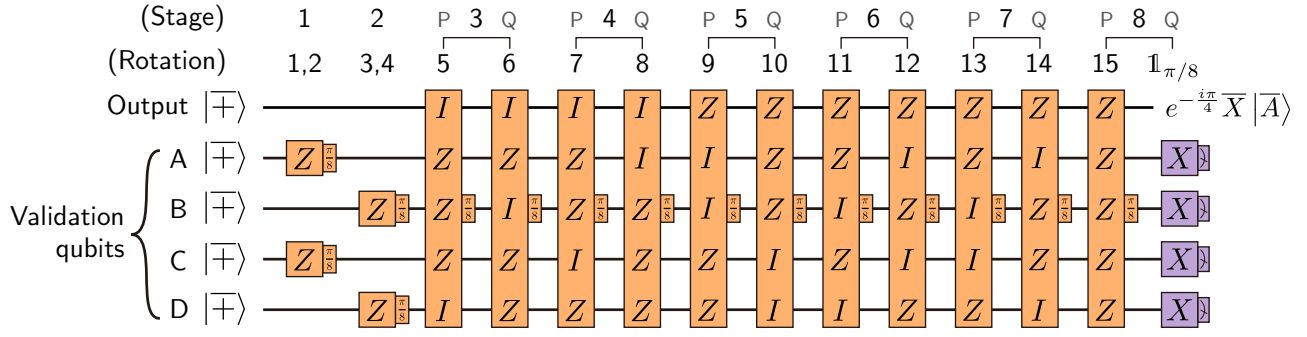


FIG. 4. 15-to-1 magic-state-distillation circuit. Five logical qubits including one output qubit and four validation qubits (A–D) are initialized to $|\bar{+}\rangle := |\bar{0}\rangle + |\bar{1}\rangle$ and undergo (non-fault-tolerant) 15 $\pi/8$ -rotations via the circuit in Fig. 3(b). All the validation qubits are then measured in the $\bar{X}$ basis. If all of these measurements give the outcome of +1, the distillation succeeds, and the output qubit has a distilled magic state, which should be $e^{-i\pi/4}\bar{X}|\bar{A}\rangle = |\bar{0}\rangle + e^{-i\pi/4}|\bar{1}\rangle$ when there are no errors. In our default scheme, the 15 rotations, labeled by 1–15, are paired up as $\{(\bar{P}^{(k)}, \bar{Q}^{(k)})\}_{k=1}^8$ in order (including a placeholder rotation $1_{\pi/8}$) and executed, respectively, in stages 1–8.

qubit as a distilled magic state $e^{-i\pi/4}\bar{X}|\bar{A}\rangle$. This is an input to the circuit of Fig. 3(a) for implementing a $\pi/8$ -rotation gate, but the $\bar{P} \otimes \bar{Z}$ measurement outcome should be interpreted reversely due to the extra $\bar{X}$. If the protocol fails, we discard the output state and retry the protocol.

The MSD circuit works since the combination of the 15 $\pi/8$ -rotations is mathematically identical to the unitary gate $(\bar{Z}_{-\pi/8})_{\text{out}} \otimes \mathbb{1}_{\text{ABCD}}$ in the ideal case [23]. Not only that, various types of errors in the circuit can be detected by the final $\bar{X}$ measurements on the validation qubits. Let us explore this fault-tolerance property in more detail. Errors in the circuit can be categorized into two groups: (i) errors from faulty T measurements and (ii) memory errors of the five logical qubits. Note that other types of errors are equivalent to errors in one of these groups. For example, the $\bar{P} \otimes \bar{Z}$ measurement in the circuit of Fig. 3(b) may give a flipped outcome, which is equivalent to a $\bar{X}$ error just before the faulty T measurement. To spoil the conclusion first, the MSD circuit can tolerate up to two rotation errors (from faulty T measurements), any $\bar{Z}$ errors on validation qubits, and up to one $\bar{X}$ error on a validation qubit.

For the first group of errors, if we model the faulty T measurement as random Pauli noise followed by the perfect T measurement, the corresponding $\bar{X}$, $\bar{Y}$, and $\bar{Z}$ errors are, respectively, converted to $\bar{P}_{-\pi/4}$, $\bar{P}_{\pi/4}$, and $\bar{P}_{\pi/2}(=\bar{P})$ errors after the circuit is executed [23]. Importantly, every combination of at most two $\bar{P}_{\pi/2}$ errors can be detected by the final $\bar{X}$ measurements, while some combinations of three $\bar{P}_{\pi/2}$ errors are not detectable, such as rotations 5, 7, and 14 in Fig. 4. $\bar{P}_{\pm\pi/4}$ errors are even less detrimental since $\bar{P}_{\pm\pi/4} = (\mathbb{1} \mp i\bar{P})/\sqrt{2}$. Therefore, the output error rate q_{dist} scales like $q_{\text{dist}} \sim p_T^3$, where p_T is the noise strength of that random Pauli noise.

Let us now consider the second group of errors. A $\bar{Z}$ error on any part of the MSD circuit can be commuted to the end of the circuit without changing anything. If it is

on the output qubit, it incurs a logical error on the distilled magic state, whereas it is always detectable if it is on one of the validation qubits. In other words, the MSD circuit is tolerant to $\bar{Z}$ errors on the validation qubits. For $\bar{X}$ errors, situations are more complicated because they may anticommute with some rotation bases. An $\bar{X}$ error on a qubit q after the i th rotation can be commuted to the beginning of the circuit and absorbed into the initial $|\bar{+}\rangle$ state, which changes the angles of several rotations [that are placed before the $(i+1)$ th rotation and nontrivially involves q] to $-\pi/8$. This can be regarded as correlated $(-\pi/4)$ -rotation errors after all the $\pi/8$ -rotations are perfectly executed. Since more than two rotation errors can be correlated, one might expect the errors to be detrimental. However, this is not the case when q is one of the validation qubits; namely, the errors cannot damage the output state without being detected. See Appendix A for the proof. (To sketch the proof, supposing that the errors make an output logical error, at least one subset $\tilde{\mathcal{P}}$ of the set of the correlated rotation errors affects the output qubit an odd number of times and affects each validation qubit an even number of times. Then the weight sum of the elements of $\tilde{\mathcal{P}}$ is odd, implying that $|\tilde{\mathcal{P}}|$ is also odd. However, since each element of $\tilde{\mathcal{P}}$ involves q , $\tilde{\mathcal{P}}$ affects q an odd number of times, which is a contradiction.) To summarize, the MSD circuit is tolerant to a single-location $\bar{X}$ error on one of the validation qubits, while two or more $\bar{X}$ errors may not be tolerable. On the other hand, a $\bar{X}$ error on the output qubit is always detrimental.

To implement the MSD circuit with color codes, we will investigate the following questions throughout the next three subsections: how to perform the faulty T measurement (Sec. III B), how to arrange logical patches encoding the output qubit, four validation qubits, and auxiliary qubits (Sec. III C), and how to perform lattice surgery for measuring $\bar{P} \otimes \bar{Z}$ in the circuit (Sec. III D). We will then

describe our scheme comprehensively in Sec. III E and calculate its resource costs in Sec. III F.

B. Faulty T measurement

Faulty T measurement is a process to measure a logical qubit non-fault-tolerantly in the basis of $\{|\bar{0}\rangle \pm e^{-i\lambda\pi/4} |\bar{1}\rangle\}$ for $\lambda \in \{0, 1\}$, as illustrated in Fig. 3(b). For a triangular logical patch, it is implemented by shrinking the patch into a single physical qubit and measuring it in the corresponding basis of the physical qubit. In detail, we first measure $\{X \otimes X, Z \otimes Z\}$ on every red edge of the patch, leaving one physical qubit q_{corner} (located at the corner where the blue and green boundaries meet) unmeasured, as exemplified in Fig. 5 for $d = 7$. Let $m_{XX}^{(e)}, m_{ZZ}^{(e)} \in \{\pm 1\}$ denote the measurement outcomes of $X \otimes X$ and $Z \otimes Z$, respectively, on a red edge e . We also define

$$\begin{aligned}\lambda_X &:= \prod_{e \in \{\text{green bdry}\}} m_{XX}^{(e)}, \\ \lambda_Z &:= \prod_{e \in \{\text{green bdry}\}} m_{ZZ}^{(e)},\end{aligned}\quad (2)$$

where $\{\text{greenbdry}\}$ is the set of red edges located along the green boundary (e.g., $\{e_1, e_2, e_3\}$ in Fig. 5). We then measure q_{corner} in the basis of $\{Z_{-\lambda\lambda_Z\pi/8} |\pm\rangle\}$. Given that the measurement outcome corresponds to $Z_{-\lambda\lambda_Z\pi/8} |\pm\rangle$, the final outcome of the faulty T measurement is $\pm\lambda_X$. The scheme works since the qubits on the green boundary support $\bar{X}$ and $\bar{Z}$, implying that the state of q_{corner} after measuring the red edges is

$$X^{(1-\lambda_Z)/2} Z^{(1-\lambda_X)/2} (|0\rangle\langle 0| + |1\rangle\langle 1|) |\bar{\psi}\rangle$$

for an initial logical state $|\bar{\psi}\rangle$.

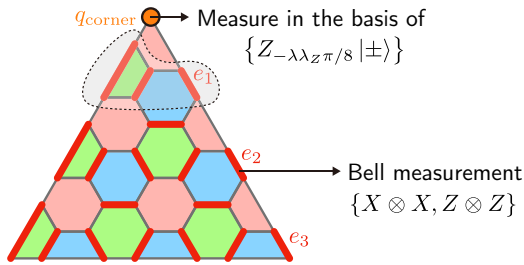


FIG. 5. Implementation of a faulty T measurement on a triangular logical patch for measuring the logical qubit in the basis of $\{|\bar{0}\rangle \pm e^{-i\lambda\pi/4} |\bar{1}\rangle\}$ for $\lambda \in \{0, 1\}$. Each red edge (e) is measured in the Bell basis, outputting two outcomes $m_{XX}^{(e)}$ and $m_{ZZ}^{(e)}$ for $X \otimes X$ and $Z \otimes Z$, respectively. The remaining single qubit q_{corner} (orange dot) is then measured in the basis of $\{Z_{-\lambda\lambda_Z\pi/8} |\pm\rangle\}$, where $\lambda_Z := m_{ZZ}^{(e_1)} m_{ZZ}^{(e_2)} m_{ZZ}^{(e_3)}$. The corresponding outcome of the faulty T measurement is $\pm\lambda_X$, where $\lambda_X := m_{XX}^{(e_1)} m_{XX}^{(e_2)} m_{XX}^{(e_3)}$. Red edges involved in the least-weight undetectable nontrivial combination of errors are highlighted as an area surrounded by a dashed line.

The aforementioned faulty T measurement scheme has a logical error rate of $O(p)$, originated from physical-level errors on q_{corner} , which are equivalent to logical errors on the patch before applying the faulty T measurement. We suppose that q_{corner} idles during two time steps for Bell measurements and then is measured in a single time step. Then, the leading-order terms of $\bar{X}$, $\bar{Y}$, and $\bar{Z}$ error rates are $(2/3)p$, $(2/3)p$, and $(5/3)p$, respectively, under the circuit-level noise model. (Here, the final measurement only contributes to the $\bar{Z}$ error rate, as our noise model assumes a probabilistic flip of the measurement outcome. This assumption is stricter than modeling the measurement noise as a depolarizing channel, as $\bar{Z}$ errors are more harmful than $\bar{X}$ and $\bar{Y}$ errors.) Note that Bell measurements may cause logical errors as well, but they can be suppressed by using X - and Z -type checks on green and blue faces (whose values are obtained from the Bell measurement outcomes on red edges). They can be decoded straightforwardly by MWPM, as each error during a Bell measurement affects at most two checks for each Pauli type. Since undetectable nontrivial combinations of such errors have weights of at least three as illustrated in Fig. 5, their contribution on the logical error rates of the faulty T measurement is $O(p^2)$, which we will ignore in our analysis.

C. Layout

We now discuss how to design a layout for implementing the 15-to-1 MSD circuit in Fig. 4. We encode the output qubit in a triangular patch (denoted as TRI_{out}) and the four validation qubits in two rectangular patches (denoted as REC_{AB} and REC_{CD}) that, respectively, encode qubits A, B and qubits C, D. We suppose that the $\bar{Z}$ operators of qubits A and C consist of physical Z operators, while those of qubits B and D consist of physical X operators; see Fig. 2(d). In addition, we need auxiliary logical qubits for faulty T measurements. These qubits are involved in the $\bar{P} \otimes \bar{Z}$ measurement of the circuit in Fig. 3(a) for each rotation $\bar{P}_{\pi/8}$, which is executed via lattice surgery. Importantly, a single-lattice surgery process can measure two commuting Pauli operators at the same time [8], thus we employ two triangular patches TRI_{α} and TRI_{β} that, respectively, encode auxiliary qubits α and β . Lastly, lattice surgery requires an ancillary region with single-color boundaries, which is surrounded by the patches. (See Sec. III D for more details on lattice surgery.) In summary, the layout consists of three triangular patches (TRI_{out} , TRI_{α} , TRI_{β}), two rectangular patches (REC_{AB} , REC_{CD}), and an ancillary region surrounded by the patches.

We use different code distances for the five patches. Namely, TRI_{out} has a code distance of d_{out} and both TRI_{α} and TRI_{β} have a code distance of d_{m} . Both REC_{AB} and REC_{CD} have code distances of d_X and d_Z , which are the smallest weights of undetectable Pauli errors equivalent to

$\bar{X}$ and $\bar{Z}$, respectively. In addition, the temporal code distance is set to be d_m , which is the number of syndrome extraction rounds required for each merging operation of lattice surgery. We suppose $d_m, d_Z < d_{\text{out}}$ throughout the discussion.

We have two reasons for using rectangular patches to encode the validation qubits. First, $\bar{X}$ and $\bar{Z}$ errors on the validation qubits have asymmetric effects: $\bar{Z}$ errors are always detectable, while $\bar{X}$ errors can be detrimental if two of them occur at the same time, as discussed in Sec. III A. Note that it is not essential to set $d_X \approx d_{\text{out}}$ unlike in Ref. [23] since a single $\bar{X}$ error on a validation qubit is tolerable. Secondly, considering that all the $\pi/8$ -rotations involve only $\bar{Z}$ operators, using rectangular patches can contribute to lowering the resource cost. Namely, the ancillary region only needs to be adjacent to the boundaries supporting $\bar{Z}$ operators of the validation qubits, thus it is more efficient to use a rectangular patch where two logical qubits share the same boundary for their respective $\bar{Z}$ operators.

We denote the seven logical qubits as $\bar{q}_{\text{out}}$, $\bar{q}_A$, $\bar{q}_B$, $\bar{q}_C$, $\bar{q}_D$, $\bar{q}_\alpha$, and $\bar{q}_\beta$, respectively, and the corresponding Pauli operators by using the same subscripts (e.g., $\bar{Z}_A$ for the $\bar{Z}$ operator of $\bar{q}_A$). Additionally, we use the shorthand notation for a tensor product of these operators such as $\bar{Z}_{\{\text{out}, A, D, \alpha\}} = \bar{Z}_{\{\text{OAD}\alpha\}} := \bar{Z}_{\text{out}} \otimes \bar{Z}_A \otimes \bar{I}_B \otimes \bar{I}_C \otimes \bar{Z}_D \otimes \bar{Z}_\alpha \otimes \bar{I}_\beta$.

In Figs. 6(a) and 6(b), we present the macroscopic pictures of possible layouts, respectively, for $2d_Z > d_{\text{out}}$ and $2d_Z < d_{\text{out}}$, where TRI_{out} , TRI_α , REC_{AB} , REC_{CD} , and TRI_β are placed clockwise in order and surrounds an ancillary region with green boundaries. The $\bar{X}$ ($\bar{Z}$) operators of

REC_{AB} and REC_{CD} terminate at their red (green) boundaries. The ancillary region consists of the ancillary patch, which is a rectangular patch with only green boundaries (that does not encode logical qubits), and the interface regions between the ancillary patch and the logical patches, which contain domain walls for lattice surgery. The ancillary patch is fixed throughout the scheme, whereas the interface regions vary depending on the operators we measure. For example, interface regions corresponding to patches that are not involved in the measurement are turned off. The horizontal and vertical dimensions of the ancillary patch are $d_H := \max(d_{\text{out}} + 1, 2d_Z)$ and $d_V := \max(d_m + 1, d_Z)$, respectively, in terms of the weights of the shortest green string operators terminating at the corresponding pairs of boundaries. We require $d_{\text{out}} - d_Z \leq d_m < 2d_Z$ for the layout to be distance preserving; see Condition 1 in Sec. III E for more details.

Exceptionally, single-qubit $\pi/8$ -rotations (rotations 1–4 in Fig. 4) can be executed more efficiently by attaching auxiliary patches (with distances $d_Z - 1$) directly to the rectangular patches through a thin ancillary region, as shown in Fig. 6(c). These auxiliary patches can be placed inside the ancillary region of the regular layout in Fig. 6(a) or Fig. 6(b), thus the space cost does not increase.

D. Lattice surgery

We now describe lattice surgery for measuring Pauli operators on the logical patches. Color codes allow measurement of a pair of commuting Pauli operators at the same time [8]. Given a pair of Pauli operators $(\bar{P}, \bar{Q})$ to measure, the structure of the ancillary region between the

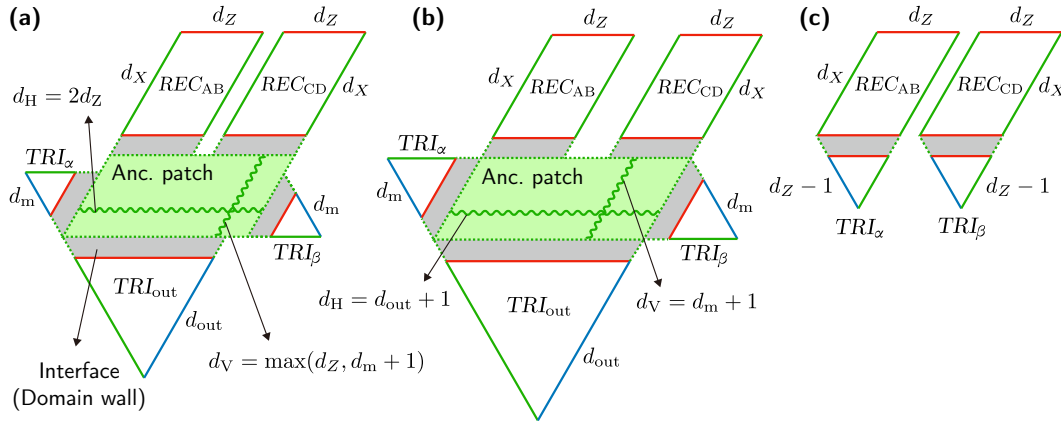


FIG. 6. Macroscopic layouts of our 15-to-1 MSD scheme for (a) $2d_Z > d_{\text{out}}$ and (b) $2d_Z < d_{\text{out}}$, which requires $d_{\text{out}} - d_Z \leq d_m < 2d_Z$ to be fault tolerant. Exceptionally, single-qubit rotations (1–4 in Fig. 4) are executed using the layout in (c). The layouts in (a),(b) consist of five patches (TRI_{out} , REC_{AB} , REC_{CD} , TRI_α , and TRI_β), an ancillary patch with green boundaries, and interface regions (gray areas) that contain domain walls for lattice surgery. The ancillary patch and interface regions are collectively called the ancillary region. The colors of the solid lines indicate the colors of the corresponding boundaries. The horizontal and vertical dimensions of the ancillary patch (in terms of the weights of the shortest green string operators terminating at the respective pairs of opposite boundaries) are, respectively, $d_H := \max(d_{\text{out}} + 1, 2d_Z)$ and $d_V := \max(d_Z, d_m + 1)$ for both (a),(b).

patches is determined appropriately based on certain rules. Hereafter we only consider the cases where $\bar{P}$ and $\bar{Q}$ contain only $\bar{I}$'s and $\bar{Z}$'s (which are sufficient for performing the MSD circuit in Fig. 4) and the layouts in Fig. 6 are used. See Appendix B for more general cases.

A lattice-surgery operation consists of three steps: the initialization of the ancillary region, merging operation, and splitting operation. Supposing that the red boundary of each patch is in contact with the ancillary region (as depicted in Fig. 6), we first initialize the ancillary region with a red temporal boundary; that is, we prepare the Bell state $|\Phi_+\rangle := (|00\rangle + |11\rangle)/\sqrt{2}$ on every red edge in the region. We then merge the patches by measuring checks in the ancillary region, which is repeated d_m times to correct temporal error chains. Here, checks in the interface regions should be chosen appropriately, ensuring that each of $\bar{P}$ and $\bar{Q}$ can be expressed as $R_r \prod_{S \in G_{\text{anc}}} S$, where R_r is a rx-string (rz-string) operator for $\bar{P}$ ($\bar{Q}$) and G_{anc} is a certain set of checks in the ancillary region. (Note that $R_r = \mathbb{1}$ in every case that we will consider for MSD, but it can be nontrivial in general lattice surgery.) After that, we split the patches by measuring the ancillary region with a red temporal boundary (i.e., measuring every red edge in the Bell basis), thereby obtaining the measurement outcomes of $\bar{P}$ and $\bar{Q}$. In addition, the Pauli frames of the logical qubits may need to be updated depending on the measurement outcomes of red edges.

We first consider a simple example of measuring $\bar{Z}_{\{A\alpha\}}$ by using the layout in Fig. 6(c). For this, in the interface region between REC_{AB} and TRI_α , we place an em-preserving semitransparent domain wall that condenses rz bosons on its both sides, which is expressed in terms of boson tables as

$$\begin{array}{|c|c|c|} \hline \blacksquare & \times & \times \\ \hline \blacksquare & \times & \times \\ \hline \bullet & \blacktriangle & \blacktriangle \\ \hline \end{array} \xleftarrow{REC_{AB}} \xrightarrow{TRI_\alpha} \begin{array}{|c|c|c|} \hline \blacksquare & \times & \times \\ \hline \blacksquare & \times & \times \\ \hline \bullet & \blacktriangle & \blacktriangle \\ \hline \end{array}.$$

Thereby, gz and bz ($\blacktriangle$) in TRI_α are deconfined and can be mapped to gz ($\blacktriangle$) in REC_{AB} , while rz is condensed in TRI_α . Therefore, a Pauli-Z string-net operator in TRI_α that represents $\bar{Z}_\alpha$ can be transformed into a gz-string operator in REC_{AB} that represents $\bar{Z}_A$ by multiplying checks. The microscopic structure of the domain wall is depicted in Fig. 7(a) for $d_Z = 6$. In the interface region, the red faces only have Z-type checks and the red edges support two-body checks $Z \otimes Z$, together determining the value of $\bar{Z}_{\{A\alpha\}}$. Note that the domain wall contains $d_Z - 2$ additional data qubits. Using weight-8 checks instead of weight-6 checks for the domain wall does not require additional data qubits [7,8], but it may significantly increase the time cost for syndrome extraction.

Similarly, $\bar{Z}_{\{B\alpha\}}$ can be measured using an em-preserving domain wall that condenses rx on the side of REC_{AB} and

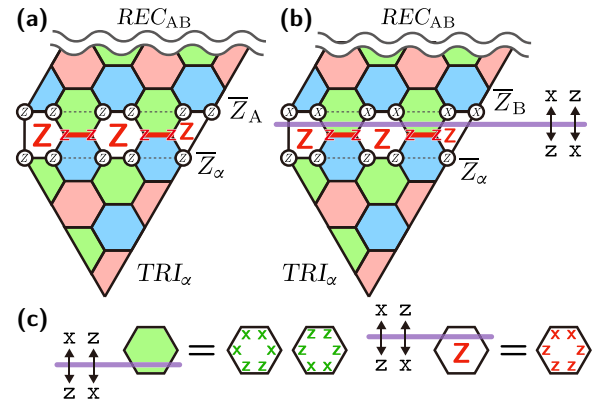


FIG. 7. Lattice-surgery schemes for measuring (a) $\bar{Z}_{\{A\alpha\}}$ and (b) $\bar{Z}_{\{B\alpha\}}$ for $d_Z = 6$. Additional qubits and checks are placed between the two patches and serve as a semitransparent domain wall, thereby measuring $\bar{Z}_{\{A\alpha\}}$ or $\bar{Z}_{\{B\alpha\}}$. Each face with a big “Z” written on it has only the Z-type check. Red edges indicated as thick red lines support two-body checks $Z \otimes Z$. In (b), the purple thick line indicate a Pauli-permuting domain wall that swaps x and z , which deforms checks on the line as exemplified in (c).

rz on the side of TRI_α , which is expressed as

$$\begin{array}{|c|c|c|} \hline \bullet & \blacktriangle & \blacktriangle \\ \hline \blacksquare & \times & \times \\ \hline \blacksquare & \times & \times \\ \hline \end{array} \xleftarrow{REC_{AB}} \xrightarrow{TRI_\alpha} \begin{array}{|c|c|c|} \hline \blacksquare & \times & \times \\ \hline \blacksquare & \times & \times \\ \hline \bullet & \blacktriangle & \blacktriangle \\ \hline \end{array}.$$

The microscopic structure of the domain wall is depicted in Fig. 7(b), where the purple line indicates a Pauli-permuting ($x \leftrightarrow z$) transparent domain wall, which deforms checks on the line as exemplified in Fig. 7(c)

Let us now consider more complicated cases that use the full layout in Fig. 6(a) or Fig. 6(b). Our goal is to measure a pair of Pauli operators ($\bar{P}, \bar{Q}$) in parallel, which both consist of only $\bar{Z}$ operators. For this, we adapt the method in Ref. [8] to be applicable to rectangular patches as well as triangular patches. That is, as displayed in Fig. 8(a) for the case of $\bar{P} = \bar{Z}_{\{OAD\alpha\}}$ and $\bar{Q} = \bar{Z}_{\{OBC\beta\}}$ (where solid and dotted lines are, respectively, Pauli-Z and X string operators), we place appropriate domain walls between the patches and the ancillary region, ensuring that each nontrivial factor of $\bar{P}$ ($\bar{Q}$) can be transformed into a gx-string (gz-string) operator “jumping over” the corresponding logical patch (i.e., connecting the two green boundaries of the ancillary region adjacent to the logical patch) by multiplying checks. In addition, if a logical patch is not involved in $\bar{P}$ ($\bar{Q}$), we require any gx-string (gz-string) operator jumping over the patch to be trivial, i.e., to be a stabilizer. By doing so, $\bar{P}$ and $\bar{Q}$ can be transformed into certain trivial green string operators in the ancillary patch, implying that their values can be determined from check measurement outcomes.

We now discuss the way to determine the types of the domain walls. If a patch is not involved in both $\bar{P}$ and

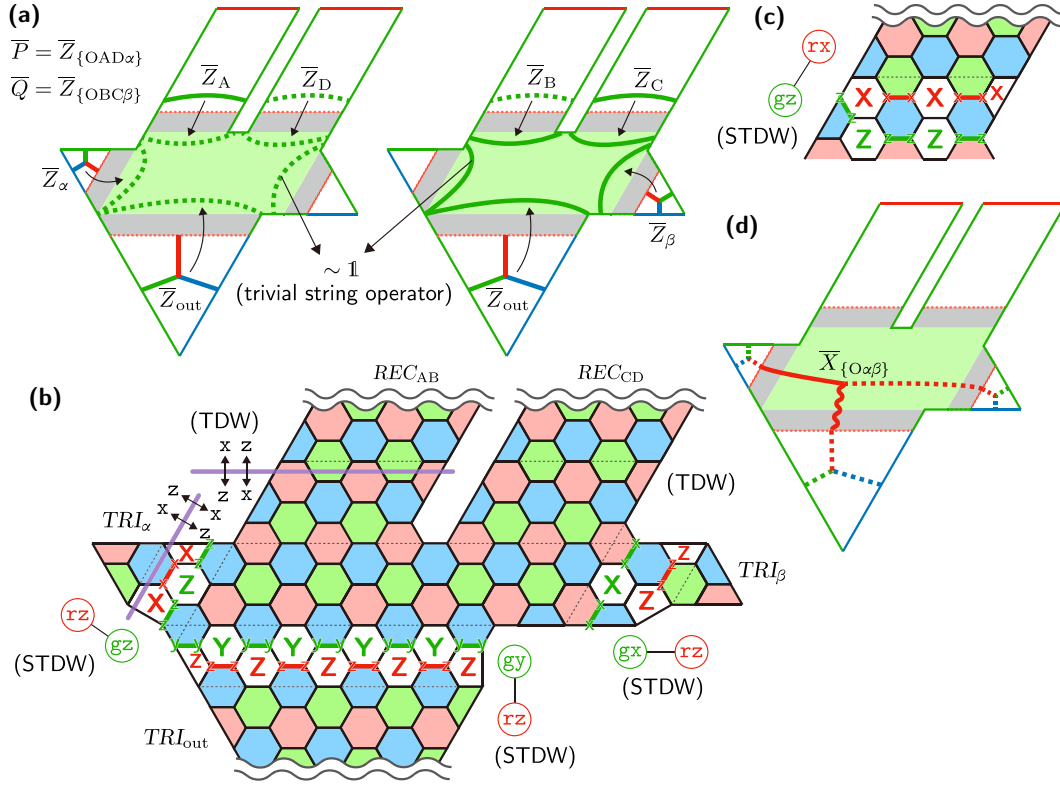


FIG. 8. (a) Lattice surgery for measuring $\bar{P} = \bar{Z}_{\{OAD\alpha\}}$ and $\bar{Q} = \bar{Z}_{\{OBC\beta\}}$ in parallel. Pauli- X (Z) string operators are visualized as thick dotted (solid) lines. Domain walls are placed between the logical patches and ancillary region so that $\bar{P}$ and $\bar{Q}$ are, respectively, equivalent to certain gx - and gz -string operators in the ancillary patch. Note that the domain wall adjoining TRI_{α} (TRI_{β}) condenses gz (gx) on the side of the ancillary region, implying that any gz -string (gx -string) operator connecting its both sides is trivial. (b) Microscopic structure of the layout of the above process for $d_{out} = 9$, $d_z = 6$, and $d_m = 3$ irregular checks in the domain walls are explicitly presented (see Fig. 7 for the notations). TDW and STDW stand for transparent and semitransparent domain walls, respectively. For each STDW, bosons condensing on its two sides are specified by a pair of circles connected by a line; e.g., the STDW adjoining TRI_{out} condenses rz and gy on the sides of TRI_{out} and the ancillary region, respectively. Domain walls adjoining rectangular patches are transparent in this example, but other measurements may require STDWs for them as shown in (c). (d) String-net operator representing a new logical operator $\bar{X}_{\{O\alpha\beta\}}$ that replaces $\bar{X}_{out}$ during the merging operation, where the wavy line indicates a Pauli- Y string operator. After the splitting operation, the value of its portion belonging to the ancillary region is determined. If it is -1 , a Pauli correction $\bar{Z}_{out}$ is applied.

$\bar{Q}$, the corresponding domain wall is opaque; namely, the patch is completely separated from the ancillary patch that has a green boundary. Other nontrivial cases are as follows: First, the domain wall adjoining each triangular patch (TRI_{out} , TRI_{α} , or TRI_{β}) is an em-exchanging semitransparent one that condenses rz and gw on the sides of the patch and the ancillary region, respectively, where

$$w = \begin{cases} x & \text{if the qubit is involved in } \bar{Q} \text{ but not in } \bar{P}, \\ z & \text{if the qubit is involved in } \bar{P} \text{ but not in } \bar{Q}, \\ y & \text{if the qubit is involved in both } \bar{P} \text{ and } \bar{Q}. \end{cases}$$

For $w = x$, the domain wall is expressed in terms of boson tables as

$$\begin{array}{|c|c|c|} \hline \blacksquare & \times & \times \\ \hline \blacksquare & \times & \times \\ \hline \bullet & \blacktriangle & \blacktriangle \\ \hline \end{array} \xleftarrow{q \text{ Anc. region}} \begin{array}{|c|c|c|} \hline \blacktriangle & \bullet & \blacktriangle \\ \hline \times & \blacksquare & \times \\ \hline \times & \blacksquare & \times \\ \hline \end{array},$$

thus a Pauli- Z string-net operator in the patch (representing $\bar{Z}_q$) can be transformed into a gz -string operator in the ancillary region and any gx -string operator jumping over the patch is trivial. The case of $w = z$ can be interpreted analogously. For $w = y$, $\bar{Z}_q$ can be transformed into any of gx - and gz -string operators in the ancillary region, as the domain wall is expressed as

$$\begin{array}{|c|c|c|} \hline \blacksquare & \times & \times \\ \hline \blacksquare & \times & \times \\ \hline \bullet & \blacktriangle & \blacktriangle \\ \hline \end{array} \xleftarrow{q \text{ Anc. region}} \begin{array}{|c|c|c|} \hline \times & \blacksquare & \times \\ \hline \blacktriangle & \bullet & \blacktriangle \\ \hline \times & \blacksquare & \times \\ \hline \end{array}. \quad (3)$$

Domain walls for rectangular patches are more diverse. Let us consider REC_{AB} as an example. We denote the restrictions of $\bar{P}$ and $\bar{Q}$ on qubits A and B as $\bar{P}_{AB}, \bar{Q}_{AB} \in \{1, \bar{Z}_A \otimes \bar{I}_B, \bar{I}_A \otimes \bar{Z}_B, \bar{Z}_A \otimes \bar{Z}_B\}$, respectively. We also

define a Pauli label

$$p_{AB} := \begin{cases} z & \text{if } \bar{P}_{AB} = \bar{Z}_A \otimes \bar{I}_B, \\ x & \text{if } \bar{P}_{AB} = \bar{I}_A \otimes \bar{Z}_B, \\ y & \text{otherwise,} \end{cases}$$

and similarly q_{AB} by replacing $\bar{P}_{AB}$ in the definition with $\bar{Q}_{AB}$. If $\mathbb{1} \neq \bar{P}_{AB} \neq \bar{Q}_{AB} \neq \mathbb{1}$ (implying $p_{AB} \neq q_{AB}$), the domain wall is a Pauli-permuting transparent one that maps p_{AB} and q_{AB} in REC_{AB} to x and z in the ancillary region, respectively. Otherwise, it is an em-exchanging semitransparent domain wall that condenses bosons a and b on the sides of REC_{AB} and the ancillary region, respectively, where

$$(a, b) = \begin{cases} (rp_{AB}, gy) & \text{if } \bar{P}_{AB} = \bar{Q}_{AB} \neq \mathbb{1}, \\ (rp_{AB}, gz) & \text{if } \bar{P}_{AB} \neq \bar{Q}_{AB} = \mathbb{1}, \\ (rq_{AB}, gx) & \text{if } \bar{Q}_{AB} \neq \bar{P}_{AB} = \mathbb{1}. \end{cases} \quad (4)$$

It is straightforward to show that the above configuration works properly from the fact that $\bar{P}_{AB}$ and $\bar{Q}_{AB}$ can be, respectively, represented by gp_{AB} - and gq_{AB} -string operators connecting the two green boundaries of REC_{AB} .

As an example, in Fig. 8(b), we describe the microscopic structure of the layout to measure $\bar{P} = \bar{Z}_{\{OAD\alpha\}}$ and $\bar{Q} = \bar{Z}_{\{OBC\beta\}}$ for $d_{\text{out}} = 9$, $d_Z = 6$, and $d_m = 3$. Checks are deformed appropriately to form necessary domain walls described above [7]. The domain walls adjoining the rectangular patches are transparent in this example, thus in Fig. 8(c) we additionally show how they can be structured when they are semitransparent.

Lastly, it is important to note that the Pauli frames of the logical qubits need to be updated appropriately after the lattice surgery finishes. This is necessary only when using the full layout as Fig. 8, not when using the simple layout as Fig. 7. Let us again consider the example of measuring $\bar{P} = \bar{Z}_{\{OAD\alpha\}}$ and $\bar{Q} = \bar{Z}_{\{OBC\beta\}}$. During the merging operation, original $\bar{X}$'s of the logical qubits are no longer logical operators as they anticommute with $\bar{P}$ or $\bar{Q}$, thus we replace them with new logical operators $\bar{X}_{\{O\alpha\beta\}}$, $\bar{X}_{\{A\alpha\}}$, $\bar{X}_{\{B\beta\}}$, $\bar{X}_{\{C\beta\}}$, and $\bar{X}_{\{D\alpha\}}$, which anticommute with $\bar{Z}_{\{O\}}$, $\bar{Z}_{\{A\}}$, $\bar{Z}_{\{B\}}$, $\bar{Z}_{\{C\}}$, and $\bar{Z}_{\{D\}}$, respectively. (Note that $\bar{Z}_{\{\alpha\}}$ and $\bar{Z}_{\{\beta\}}$ are no longer independent from other logical operators, thus considering these five pairs of logical operators is sufficient.) To identify a Pauli correction applied on the output qubit, we consider $\bar{X}_{\{O\alpha\beta\}}$, which can be represented by a string-net operator connecting the original representations of $\bar{X}_{\text{out}}$, $\bar{X}_\alpha$, and $\bar{X}_\beta$, as visualized in Fig. 8(d). Here, the red wavy line indicates an ry -string operator. Note that we should be careful about the behavior of the string net near each domain wall. For example, its rx -string part in TRI_{out} can be connected with its ry -string part in the ancillary region through the domain wall; see Eq. (3). The

value of its portion belonging to the ancillary region can be determined by measuring red edges fault tolerantly in the splitting operation. If it is -1 , we apply a Pauli correction $\bar{Z}_{\text{out}}$.

In general, for each logical qubit q (encoded in a patch PAT_q) that is not qubit α or β , we determine a Pauli correction on q by the following method: if both $\bar{P}$ and $\bar{Q}$ involve q , we choose a red string-net operator R belonging to the ancillary region that consists of rx -, rx -, and ry -string parts, which are connected with TRI_α , TRI_β , and PAT_q , respectively. If only $\bar{P}$ ($\bar{Q}$) involves q , we choose an rx -string (ry -string) operator R belonging to the ancillary region that connects TRI_α (TRI_β) and PAT_q . If the value of R is -1 as a result of measuring the red edges, we apply a Pauli correction $\bar{Z}$ on q .

E. Scheme

We finally describe the overall procedure of our MSD scheme by combining the above ingredients. The 15 $\pi/8$ -rotations in the MSD circuit of Fig. 4 are executed pairwise through eight *stages* by adding one placeholder rotation $\mathbb{1}_{\pi/8} = \mathbb{1}$. We denote the pair of rotations for the k th stage as $(\bar{P}_{\pi/8}^{(k)}, \bar{Q}_{\pi/8}^{(k)})$, where $\bar{P}^{(k)}$ and $\bar{Q}^{(k)}$ are logical Pauli operators. The first two stages are allocated for four single-qubit rotations on validation qubits A, B, C, and D, which are rotations 1–4 in Fig. 4, namely, $\bar{P}^{(1)} = \bar{Z}_{\{A\}}$, $\bar{Q}^{(1)} = \bar{Z}_{\{C\}}$, $\bar{P}^{(2)} = \bar{Z}_{\{B\}}$, and $\bar{Q}^{(2)} = \bar{Z}_{\{D\}}$. These are executed by using the simple layout in Fig. 6(c) instead of the regular one in Fig. 6(a) or Fig. 6(b). For example, to perform $(\bar{Z}_{\{A\}})_{\pi/8}$, we initialize qubit α to $|\bar{+}\rangle$, measure $\bar{Z}_{\{A\alpha\}}$ via lattice surgery (which takes d_m rounds), and perform a faulty T measurement on qubit α . In the k th stage where $k \geq 3$, we initialize qubits α and β to $|\bar{+}\rangle$ and simultaneously measure

$$\begin{aligned} \bar{P}_{\text{LS}}^{(k)} &:= \bar{P}_{\text{out},A,B,C,D}^{(k)} \otimes \bar{Z}_\alpha \otimes \bar{I}_\beta, \\ \bar{Q}_{\text{LS}}^{(k)} &:= \bar{Q}_{\text{out},A,B,C,D}^{(k)} \otimes \bar{I}_\alpha \otimes \bar{Z}_\beta \end{aligned} \quad (5)$$

via lattice surgery (which takes d_m rounds), followed by faulty T measurements on TRI_α and TRI_β . We should carefully track the Pauli corrections made by the lattice surgery and faulty T measurements. The former may make corrections of $\bar{Z}$ operators depending on the measurement outcomes of red edges in the ancillary region. The latter may make corrections of $\bar{P}^{(k)}$, $\bar{Q}^{(k)}$, or $\bar{P}^{(k)}\bar{Q}^{(k)}$ depending on the faulty T -measurement outcomes. After performing all eight stages, we check whether the X -measurement outcomes after applying the accumulated Pauli corrections are all equal to $+1$. If then, we conclude that the distillation succeeds and a distilled logical magic state $\bar{X}|\bar{A}\rangle$ (or $\bar{Y}|\bar{A}\rangle$) if there is an odd number of corrections on the output qubit

is produced from TRI_{out} . Otherwise, the entire process is retried.

We cannot arbitrarily determine the six pairs of 11 rotations $\left\{ \left(\bar{P}_{\pi/8}^{(k)}, \bar{Q}_{\pi/8}^{(k)} \right) \right\}_{k=3}^8$, due to the following two factors: (i) The number of errors that cause logical errors on the output state varies depending on this configuration. (ii) Some configurations damage the fault tolerance of the layout in Fig. 6(a) or Fig. 6(b); namely, they may result in the existence of undetectable string operators equivalent to a logical operator of a patch but shorter than the corresponding code distance. To spoil the conclusion first, the configuration presented in Fig. 4 is one of the optimal ones considering these two.

Elaborating on the first factor, memory errors on the output qubit can be reduced by preparing the output qubit as late as possible. This can be achieved by executing rotations 1–8 (that do not involve the output qubit) first, followed by the remaining rotations. Note that this configuration has an additional advantage of allowing enough time to consume the output magic state even if the next MSD starts immediately. Furthermore, among four possible locations of $\bar{X}_{\text{out}}$ errors (right after stages 5, 6, 7, and 8), we can make two of them (right after stages 5 and 6) not harmful. A $\bar{X}_{\text{out}}$ error right after stage 5 is always unarmful since it makes only two correlated rotation errors, which are detectable. A $\bar{X}_{\text{out}}$ error right after stage 6 can be made unarmful by setting stages 5 and 6 to include four rotations in two among three pairs $(\bar{Z}_{\{\text{OAB}\}}, \bar{Z}_{\{\text{OCD}\}})$, $(\bar{Z}_{\{\text{OAC}\}}, \bar{Z}_{\{\text{OBD}\}})$, and $(\bar{Z}_{\{\text{OAD}\}}, \bar{Z}_{\{\text{OBC}\}})$, which are pairs of weight-3 rotations acting on disjoint sets of validation qubits. By doing so, no subset of these four rotations acts on the output qubit an odd number of times and acts on each validation qubit an even number of times, meaning that a $\bar{X}_{\text{out}}$ error right after stage 6 is unarmful (see Appendix A for details on why this argument works).

Let us now consider the second factor on the fault tolerance of the layout. During lattice surgery, each logical Pauli operator of a patch that commutes with the operators to measure becomes equivalent (under stabilizer multiplication) with certain string operators in the ancillary region. We demand these string operators to have weights not smaller than the corresponding code distance of the patch; namely, the layout should be distance preserving. The following conditions should be satisfied for this:

Condition 1. (i) $d_{\text{out}} - d_Z \leq d_m < 2d_Z$, (ii) $\bar{Z}_{\{\text{OCD}\}} \in \left\{ \bar{P}^{(k)} \right\}_{i=1}^6$, and (iii) if $d_Z > 2d_m + 2$, $(\bar{Z}_{\{s\}})_{\pi/8}$ and $(\bar{Z}_{\{t\}})_{\pi/8}$ are not paired for each (s, t) in (OAC, OBC) , (OAD, OBD) , (OAC, OAD) , (OBC, OBD) , $(\text{OCD}, \text{OABCD})$, and $(\text{OAB}, \text{OABCD})$.

The first condition pertains to the layout itself, rather than the configuration of rotations, but we mention it here since it relates to this second factor. In Appendix C, we

verify that the layout is distance preserving if and only if these conditions are met. For example, $d_{\text{out}} - d_Z \leq d_m$ is required because, when measuring $\bar{Z}_{\{\text{OAB}\alpha\}}$ or $\bar{Z}_{\{\text{OAB}\beta\}}$, $\bar{Z}_{\text{out}}$ is equivalent to a green string operator in the ancillary region with weight $d_Z + d_m$. Other conditions can be derived similarly.

The configuration presented in Fig. 4 satisfies the above requirements, thus we set this as our default configuration.

F. Resource costs

We lastly evaluate the space and time costs of our MSD scheme. The space cost of the scheme is quantified by the maximal number of physical qubits simultaneously required while executing the scheme. Here, physical qubits include not only data qubits but also syndrome qubits, which are used to extract check measurement outcomes. We suppose that each check has one syndrome qubit, which enables the simultaneous measurements of all checks during a single syndrome extraction round. The time cost is quantified by the number of time steps required for each attempt of the scheme. Note that a single syndrome extraction round can be done in eight time steps by selecting the entangling gate schedule appropriately [22].

A triangular patch with distance d contains $(3d^2 + 1)/4$ data qubits and $(3d^2 - 3)/4$ syndrome qubits, with a total of

$$n_{\text{tri}}(d) := \frac{3d^2 - 1}{2}$$

qubits. A rectangular patch with distances d_1 and d_2 contains $3d_1d_2/2 - d_1 - d_2 + 2$ data qubits and $3d_1d_2/2 - d_1 - d_2$ syndrome qubits, with a total of

$$n_{\text{rec}}(d_1, d_2) := 3d_1d_2 - 2d_1 - 2d_2 + 2$$

qubits. The ancillary patch (surrounded by green boundaries) contains

$$n_{\text{anc,patch}} := 3d_Hd_V - 2d_H - 2d_V + 2$$

qubits, where $d_H = \max(d_{\text{out}} + 1, 2d_Z)$ and $d_V = \max(d_Z, d_m + 1)$. Lastly, the interface regions covering domain walls contain at most

$$\begin{aligned} n_{\text{int}} &:= 2 \cdot [2d_{\text{out}} + 2(3d_Z - 2) + 2d_m + (3d_m + 1)] \\ &= 4d_{\text{out}} + 12d_Z + 10d_m - 6 \end{aligned}$$

qubits, excluding those that already belong to the ancillary patch. Therefore, the space cost of the scheme is

$$\begin{aligned} n_{d_{\text{out}}, d_X, d_Z, d_m}^{(\text{org})} &:= n_{\text{tri}}(d_{\text{out}}) + 2n_{\text{rec}}(d_X, d_Z) + 2n_{\text{tri}}(d_m) \\ &\quad + n_{\text{anc,patch}} + n_{\text{int}}. \end{aligned} \quad (6)$$

To evaluate the time cost, we consider the following: (i) Each round consists of $t_{\text{rnd}} = 8$ time steps. (ii) For each

stage, the merging operation takes d_m rounds. (iii) After each merging operation, the measurement and reinitialization of the ancillary region together takes one round. Note that it actually takes four time steps (as we need two time steps to make Bell measurements), but we use a full round for it to synchronize syndrome extraction. If the merging operation is for the final stage, the reinitialization of the ancillary region is for the next MSD. (iv) Two time steps are required to initialize logical qubits, but it can be performed in parallel with initializing the ancillary region for the first stage. Similarly, logical qubits can be measured in parallel with measuring the ancillary region for the final stage. (v) Each faulty T measurement takes three time steps (i.e., two for Bell measurements and one for measuring q_{corner}), but it can be performed in parallel during the extra single round in (iii) before starting the next stage. Therefore, the time cost of the scheme is

$$t_{d_m}^{(\text{org})} := 8t_{\text{rnd}}(d_m + 1), \quad (7)$$

time steps.

IV. PRODUCTION OF HIGHER-QUALITY MAGIC STATES

The MSD scheme in Sec. III cannot produce magic states with logical error rates lower than approximately $35p_{\text{FT}}^3$ even if there are no memory errors, where $p_{\text{FT}} = O(p)$ is the error rate of the faulty T measurement. A conventional method to reach lower logical error rates is to concatenate the scheme with itself or another MSD scheme, namely, to input magic states produced from the scheme into MSD again. However, here we take an alternative approach building on several recent results, combining distillation with recently proposed distillation-free magic state preparation protocols, which we will refer to as “*magic state cultivation*” following the terminology introduced in Ref. [26].

Magic state cultivation is an alternative approach to distillation for generating logical magic states by leveraging the transversality of Clifford operations in color codes. This method relies on the fact that the eigenstates of certain Clifford operators can serve as magic states; for example, $|A\rangle$ is a $+1$ eigenstate of $(X + Y)/\sqrt{2}$. By measuring a logical qubit multiple times in the basis of such eigenstates (through noisy transversal controlled-Clifford gates) and postselecting based on these measurement outcomes as well as the check outcomes, the final state is highly likely to be projected onto the desired magic state. Notably, cultivation does not involve any multiqubit logical operations, leading to its high resource efficiency compared to distillation. For example, the scheme in Ref. [26] achieves an output infidelity of 10^{-9} at a spacetime cost of approximately 10^6 for $p = 10^{-3}$ (see Fig. 12 for more detailed comparison).

Several cultivation schemes have been proposed [24–26], differing in their detailed methods, such as circuits for measuring Clifford logical operators and color-code checks. The scheme in Ref. [24] employs flag qubits to detect all sets of faults at up to $(d - 1)/2$ locations. In this scheme, the logical Clifford operator and check measurement circuits are designed such that the flag and ancillary qubits switch roles depending on the operator to measure. In Ref. [25], the authors propose a cultivation scheme that uses only nearest-neighbor two-qubit gates on a square grid and enables efficient teleportation into surface codes. The most recent work, Ref. [26], refines these ideas by introducing a gradual increase in code size and advanced techniques for color codes (such as the superdense syndrome extraction circuit [37]), significantly improving resource efficiency and fault tolerance. Importantly, unlike the two earlier works, the authors carefully address the process of growing the output magic state to a large code distance, showing that this step has a substantial impact on the fault tolerance and can be the most challenging part in the entire process.

Despite the high resource efficiency of cultivation, it has a fundamental limitation in scalability due to the extensive use of postselection. Specifically, the retry cost grows exponentially with the code distance, making it difficult to achieve output infidelities below a certain bound. For instance, the scheme in Ref. [26] has been explored only for patches with $d \leq 5$, where the output infidelity is lower bounded at approximately 10^{-9} when $p = 10^{-3}$ (noting that the discard rate reaches 99% for $d = 5$). A promising approach to achieving lower output infidelities is combining cultivation and distillation. That is, cultivation can be used to inject high-quality magic states into distillation, replacing non-fault-tolerant injection or the faulty T measurement. Notably, our MSD scheme, based on color codes, can be seamlessly integrated with cultivation by adjusting the single-level scheme described in Sec. III. We elaborate on this cultivation-MSD scheme through the following subsections.

A. Magic state cultivation

Cultivation, the primary ingredient in our construction, is treated as a black box. This process can be implemented using any of the schemes in Refs. [24–26] or with an improved scheme that may be proposed in the future. It is performed on a triangular color-code patch of distance d_{cult} , directly preparing a logical magic state $|\bar{A}\rangle$ with infidelity q_{cult} and a success rate of $q_{\text{cult}}^{\text{succ}}$. (Note that the schemes in Refs. [24,25] prepare an eigenstate of the Hadamard gate rather than $|\bar{A}\rangle$; this difference can be addressed by adjusting the measurement bases in the rotation gate circuits in Fig. 3.) The space and time costs for a single successful cultivation attempt (without considering retrying) are denoted as n_{cult} and t_{cult} , representing the

number of physical qubits and the number of time steps, respectively. For simplicity, we assume that the success of cultivation is determined immediately upon the completion of the process, although, in practice, it is more efficient to retry immediately after detecting a failure in the middle of the circuit.

B. Growing operation

The distance d_{cult} must be set small enough to minimize the retry cost; however, it is typically insufficient to store the generated magic state. For instance, the scheme described in Ref. [26] can prepare a magic state with an infidelity of 6×10^{-7} on a distance-3 patch when $p = 10^{-3}$, which corresponds to the error rate of a color-code patch with a distance of around 19 [27]. Thus, it is crucial to grow the patch to a sufficiently large code distance, d_m , immediately after completing cultivation. As illustrated in Fig. 9(a), this can be performed by preparing Bell states, $|00\rangle + |11\rangle$, on additional red edges (simultaneously with the completion of cultivation) and performing regular check measurements thereafter. In the spacetime picture, this operation can be interpreted as extending the red spatial boundary of the patch to a red temporal boundary, as shown in Fig. 9(b).

However, a critical obstacle is that the fault tolerance of the growing operation depends on d_{cult} , rather than d_m , due to error strings terminating at the red temporal boundary, as shown in Fig. 9(b). Thus, even just a single round of the growing operation may significantly damage the cultivated magic state. To address this issue, we employ postselection during decoding for the growing operation. Specifically, we consider performing the growing operation and subsequent d_m rounds of syndrome extraction, decoded jointly. If the confidence of the prediction exceeds a preset threshold, we accept the final state; otherwise, we abort it and restart cultivation from the beginning.

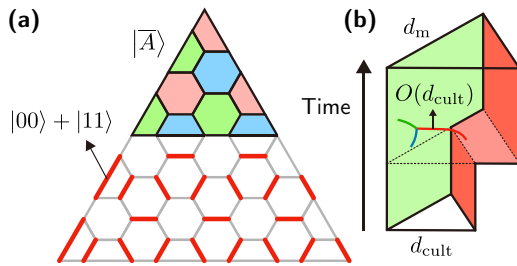


FIG. 9. (a) Example of a growing operation of a triangular patch from $d_{\text{cult}} = 5$ to $d_m = 9$, which is done by preparing Bell states $|00\rangle + |11\rangle$ on additional red edges. (b) Spacetime picture of the operation, which can be interpreted as extending the red spatial boundary of the patch to a red temporal boundary. The blue spatial boundary (front face) is not colored for visibility. An example of a nontrivial string-net operator with weight $O(d_{\text{cult}})$ is placed inside the diagram.

One way to quantify confidence is by using the *logical gap* (or *complementary gap*) [38–40], defined as the minimum log-likelihood weight difference between the correction and an alternative correction in a different logical class. A larger logical gap indicates greater confidence in the prediction, thus we abort a trial when the logical gap is below a certain threshold c_{gap} . For surface codes, the logical gap can be computed by running the minimum-weight perfect matching (MWPM) decoder [5] multiple times with different preassigned logical values [38–40], potentially raising the noise threshold to at most 50% [40]. For color codes, we calculate the logical gap using the *concatenated MWPM decoder* [27] (one of the best-performing circuit-level decoders for color codes in subthreshold scaling) in a similar way by varying the preassigned logical values. Note that, unlike surface codes, the concatenated MWPM decoder does not guarantee the least-weight correction, so the calculated logical gap is an approximation.

To analyze the effect of postselection, we run circuit-level simulations of the growing operation followed by d_m rounds of syndrome extraction. In Fig. 10, the logical failure rates (i.e., the summations of $\bar{X}$ and $\bar{Z}$ failure rates) and the acceptance rates obtained by varying the logical gap threshold are presented for $p = 10^{-3}$ and $d_{\text{cult}} \in \{3, 5\}$. (We say, e.g., $\bar{X}$ fails if a $\bar{Y}$ or $\bar{Z}$ error occurs.) These results demonstrate that the postselection method works very well with the concatenated MWPM decoder. For instance, for $d_{\text{cult}} = 3$ and $d_m \geq 7$, the logical error rate can be reduced from 10^{-2} to 10^{-4} (3×10^{-5}) by aborting only about 10%

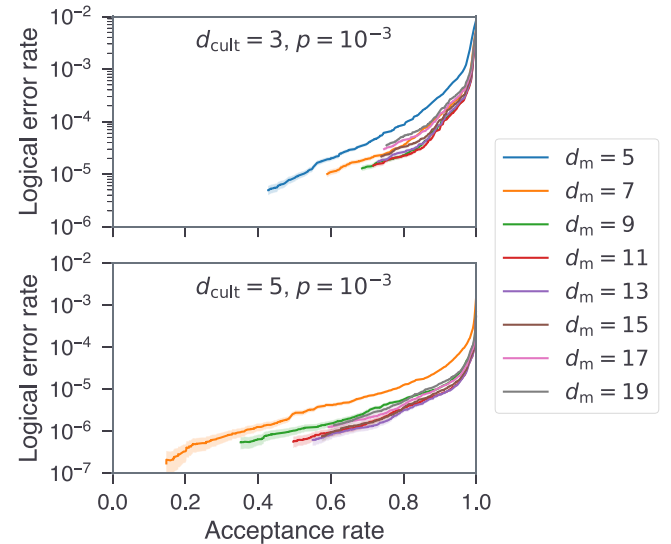


FIG. 10. Simulations of the growing operation from distance d_{cult} to d_m with postselection. The summation of $\bar{X}$ and $\bar{Z}$ failure rates are plotted against the acceptance rates for $d_{\text{cult}} \in \{3, 5\}$, $d_m > d_{\text{cult}}$, and $p = 10^{-3}$, obtained by varying the logical gap threshold c_{gap} . The shaded regions indicate the 99% confidence intervals.

(20%) of trials. The details of the simulation method and additional numerical results are provided in Appendix D.

It is worth noting that the logical gap approach does not seem to work well with the Möbius decoder [37,41] (another matching-based color-code decoder with performance comparable to the concatenated MWPM decoder), as investigated in Ref. [26]. The reason for this discrepancy remains unclear, and it is uncertain whether this is a fundamental limitation of the Möbius decoder or an issue that could be overcome.

C. Cultivation-MSD scheme

After preparing magic states via cultivation (followed by the growing operation), we input them into MSD. The layout for the cultivation-MSD scheme is presented in Fig. 11, which is a modified version of the original layout in Fig. 6(a) or Fig. 6(b) to include $2N_m$ auxiliary triangular patches with distance d_m . To minimize the ancillary region, we place

$$N_{m,\text{side}} := \min \left(N_m, \left\lceil \frac{d_z}{d_m + 1} \right\rceil \right)$$

auxiliary patches on each of the left and right sides of the ancillary region, while the others are placed next to REC_{AB} or TRI_{out} . Note that $N_m = 3$ and $N_{m,\text{side}} = 2$ in the example of Fig. 11. We group them into “left” and “right” auxiliary patches, which are, respectively, labeled as (a), (b), (c) and (a’), (b’), (c’) in Fig. 11. The horizontal and vertical

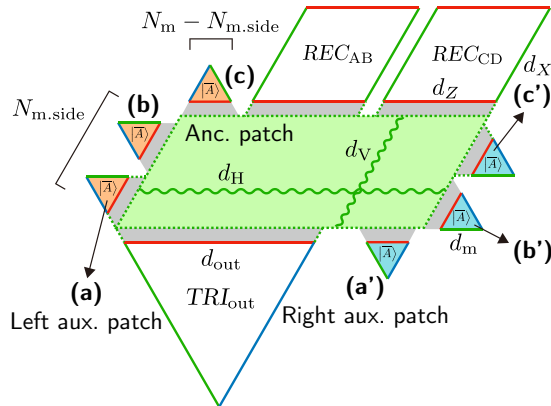


FIG. 11. Layout for the cultivation-MSD scheme. The layout in Fig. 6(a) or Fig. 6(b) is modified to contain $2N_m$ triangular patches ($N_m = 3$ in the figure), where magic states $|\bar{A}\rangle$ are prepared via cultivation and the growing operation. Among them, $N_{m,\text{side}} := \min(N_m, \lceil (d_z/(d_m + 1)) \rceil)$ patches are placed on each of the left and right sides of the ancillary region ($N_{m,\text{side}} = 2$ in the figure), while the other patches are placed next to TRI_{out} or REC_{AB} . They are divided into two groups, left and right auxiliary patches, which are labeled as (a)–(c) and (a’)–(c’), respectively. The horizontal and vertical dimensions of the ancillary patch are respectively d_H and d_V defined in Eq. (8).

dimensions of the ancillary patch are, respectively,

$$\begin{aligned} d_H &:= \max(2d_z, d_{\text{out}} + 1) + (N_m - N_{m,\text{side}})(d_m + 1), \\ d_V &:= \max[d_z, N_{m,\text{side}}(d_m + 1)]. \end{aligned} \quad (8)$$

Cultivation is executed in each auxiliary patch as frequently as possible, ensuring that successive executions within each of the left and right groups are separated by at least T_m rounds, where

$$T_m := \left\lceil \frac{t_{\text{cult}}}{t_{\text{md}} N_m} \right\rceil.$$

For instance, in Fig. 11, after initiating cultivation in patch (a), T_m rounds must pass before cultivation can begin in patch (b) or (c). Once cultivation is completed successfully, the patch undergoes the growing operation, followed by d_m rounds of syndrome extraction. (Exceptionally, if $d_m = d_{\text{cult}}$, these additional steps are omitted.) If the logical gap obtained from decoding exceeds a preset threshold, the created magic state can then be used for distillation. If either the cultivation or the growing operation fails, the resulting state in the patch is discarded, and cultivation is retried (subject to the aforementioned condition on successive executions). Additionally, if lattice surgery involving other auxiliary patches is in progress when the magic state is created, the state is also discarded instead of waiting until the lattice surgery ends. (This is to prevent additional logical errors on the magic state caused by idling, though it is not strictly necessary.)

We wait until two magic states, one from each of the left and right groups, are prepared and grown successfully. The earlier-generated magic state should idle while waiting for the second state. (If a new magic state is generated while another within the same group is idling, the earlier one is discarded.) These magic states are then consumed to perform a pair of $\pi/8$ -rotations using the circuit shown in Fig. 3(a). This process is repeated throughout all eight stages of the MSD. Note that, unlike the single-level scheme, single-qubit rotations in stages 1 and 2 of Fig. 4 are implemented by using the same layout as other rotations.

The space cost of the scheme is

$$\begin{aligned} n_{d_{\text{out}}, d_X, d_Z, d_m, d_{\text{cult}}, N_m}^{(\text{comb})} &:= n_{\text{tri}}(d_{\text{out}}) + 2n_{\text{rec}}(d_X, d_Z) + n_{\text{anc. patch}} + n_{\text{int}} \\ &\quad + 2N_m [n_{\text{cult}} + (1 - \delta_{d_m, d_{\text{cult}}}) \{n_{\text{tri}}(d_m) - n_{\text{tri}}(d_{\text{cult}})\}], \end{aligned} \quad (9)$$

where δ_{ij} is the Kronecker delta defined to be 1 if $i = j$ and 0 otherwise, and

$$n_{\text{anc,patch}} := 3d_{\text{H}}d_{\text{V}} - 2d_{\text{H}} - 2d_{\text{V}} + 2,$$

$$n_{\text{int}} := 4d_{\text{out}} + 12d_{\text{Z}} + 10N_{\text{m}}d_{\text{m}} + 2N_{\text{m}} - 8$$

are the space costs of the ancillary patch and the interface regions for domain walls, respectively. Denoting the expected number of rounds between the initiation of successive stages as T_{intv} , the expected time cost of the scheme is

$$t_{d_{\text{m}}, N_{\text{m}}, p}^{(\text{comb})} := 8t_{\text{rnd}}T_{\text{intv}}. \quad (10)$$

In our numerical analysis that will be described in the next section, we estimate T_{intv} by simulating the aforementioned procedure of the scheme for 1000 stages. Note that, through the simulation, we additionally estimate the average number of rounds T_{idle} that the auxiliary patches idle before they are consumed, which is used for analyzing errors. The code we used for this simulation is available on GitHub [42]. See Sec. VB and Table I for explicit examples on the estimated values of T_{intv} and T_{idle} .

V. PERFORMANCE ANALYSIS

In this section, we numerically analyze the performance of our MSD schemes, including both single-level and cultivation-MSD schemes, in terms of their output infidelities and success probabilities. For cultivation, we assume that the state-of-the-art scheme in Ref. [26] is employed.

Instead of directly simulating the entire MSD circuit at once, we first calculate the probabilities of logical errors in individual patches (including timelike error strings) via Monte Carlo simulations and then carefully track their effects on the final magic state. Although this approach may be less reliable than fully simulating MSD, we adopt this for the following reasons: (i) The target output infidelity is extremely low (e.g., $< 10^{-9}$), and the system size is very large, making direct Monte Carlo simulations computationally infeasible. (ii) Existing color-code decoders [22,27,41,43–52] are primarily designed for idling gates and may require modifications to handle logical operations (particularly irregular checks in domain walls). While such modifications are essential for practical MSD implementation, this work focuses on estimating achievable performance rather than detailing decoder adjustments. We assume that irregular checks for lattice surgery do not significantly degrade decoding performance (see Sec. VA for its partial justification). (iii) This modular approach allows different decoders or cultivation protocols to be tested by simply adjusting a few parameters in the logical error tracking step. For this, we provide analytical expressions for infidelities and success probabilities in terms of logical error rates, which may assist other researchers in evaluating their decoders or cultivation protocols.

A. Method

Our analysis method is described in Appendix E in detail. We outline this here as follows.

We first simulate $4d$ rounds of syndrome extraction of triangular and rectangular patches with code distances $d \leq 21$ under circuit-level noise of $p \in [10^{-4}, 10^{-3}]$, decoded via the concatenated MWPM decoder. The syndrome extraction circuit is carefully chosen to minimize the logical failure rate (see Appendix F for more details on the selection method and the resulting circuit). For rectangular patches, we assume that they have $\bar{Z}(\bar{X})$ failure rates proportional to $d_{\text{Z}}(d_{\text{X}})$, which makes it sufficient to simulate only the cases of $d_{\text{X}} = d_{\text{Z}} = d$. From these simulations, we estimate per-round logical failure rates of logical patches. In addition, we simulate stability experiments [53] by performing $T(\leq 14)$ rounds of syndrome extraction of a patch encoding no logical qubits, which give “per-area” logical failure rates caused by timelike errors.

For generalizing the outcomes to other regimes of p and d (or T), the computed values for the per-round or per-area logical failure rates are fitted into the ansatz

$$p_{\text{fail}}(p, d) = \alpha \left(\frac{p}{p_{\text{th}}} \right)^{\beta d + \eta} \left[1 + \epsilon \left(\frac{p}{p_{\text{th}}} \right)^{\zeta d^{\lambda}} \right] \quad (11)$$

with seven parameters $p_{\text{th}}, \alpha, \beta, \eta, \epsilon, \zeta, \lambda$, where $p_{\text{th}}, \alpha, \beta$, and ζ are positive. (For stability experiments, d is replaced with T .) Note that, for a more precise prediction, the ansatz contains a subleading order term with respect to p/p_{th} , selected from several candidates via cross-validation to prevent overfitting.

In Appendix G, we detail the simulation method and the ansatz selection process, with the numerical results and the corresponding parameter estimates.

We denote the logical Pauli error rates of triangular and rectangular patches as p_{tri}^P and p_{rec}^P , respectively, for a Pauli operator P . To determine $p_{\text{tri}}^{X/Y/Z}$ from logical failure rates, we assume that $p_{\text{tri}}^X = p_{\text{tri}}^Z$ and $p_{\text{tri}}^Y = r_{\text{y}} p_{\text{tri}}^X$, where r_{y} is a small non-negative number quantifying the contribution of Pauli- Y string errors. Similarly, for a rectangular patch, we assume $p_{\text{rec}}^{X_1} = p_{\text{rec}}^{X_2}, p_{\text{rec}}^{X_1 X_2} = r_{\text{y}} p_{\text{rec}}^{X_1}, p_{\text{rec}}^{Z_1} = p_{\text{rec}}^{Z_2}$, and $p_{\text{rec}}^{Z_1 Z_2} = r_{\text{y}} p_{\text{rec}}^{Z_1}$. The same assumption apply to the growing operation ($p_{\text{grow}}^X = p_{\text{grow}}^Z$ and $p_{\text{grow}}^Y = r_{\text{y}} p_{\text{grow}}^X$) and to cultivation, where only the output infidelity is known ($p_{\text{cult}}^X = p_{\text{cult}}^Z = q_{\text{cult}}/(2 + r_{\text{y}})$ and $p_{\text{cult}}^Y = r_{\text{y}} p_{\text{cult}}^X$). Although r_{y} may appear to be an artificial coefficient, we will later show that it has a negligible impact on the MSD performance.

The next step is to map every possible logical Pauli error (caused by an error string in a logical patch or the ancillary region) during MSD to an equivalent noise channel acted on the output and validation qubits immediately before the final $\bar{X}$ measurements of the validation qubits. The noise

channel is of the form

$$\Lambda_{\bar{U}, p_{\text{err}}} : \bar{\rho} \mapsto (1 - p_{\text{err}})\bar{\rho} + p_{\text{err}}\bar{U}\bar{\rho}\bar{U}^\dagger,$$

where $\bar{\rho}$ is the logical state of the output and validation qubits, p_{err} is the logical error rate (expressed in terms of the above notations on logical Pauli error rates), and $\bar{U}$ is a product of $\pi/2$ - or $(\pm\pi/4)$ -rotations. See Appendix H for a exhaustive list of the possible error sources and the corresponding noise channels.

Denoting the set of noise channels as $\{\Lambda_{\bar{U}_i, p_i}\}_{i=1}^m$, the unnormalized output state after the final measurements is

$$\bar{\rho}_{\text{out}} = (\mathbb{1}_{\text{out}} \otimes \langle +++++ |_{\text{ABCD}}) \Lambda_{\text{noise}} (|\bar{\psi}_{\text{init}}\rangle \langle \bar{\psi}_{\text{init}}|),$$

where

$$\begin{aligned} \Lambda_{\text{noise}} &:= \Lambda_{\bar{U}_1, p_1} \circ \cdots \circ \Lambda_{\bar{U}_m, p_m}, \\ |\bar{\psi}_{\text{init}}\rangle &:= |\bar{A}_-\rangle_{\text{out}} \otimes |++++\rangle_{\text{ABCD}}, \\ |\bar{A}_-\rangle &:= \frac{1}{\sqrt{2}} (|\bar{0}\rangle + e^{-i\pi/4} |\bar{1}\rangle). \end{aligned}$$

The success probability q_{succ} of the scheme and the output infidelity q_{dist} are then, respectively, given as

$$q_{\text{succ}} = \text{Tr}(\bar{\rho}_{\text{out}}), \quad q_{\text{dist}} = 1 - \frac{1}{q_{\text{succ}}} \langle \bar{A}_- | \bar{\rho}_{\text{out}} | \bar{A}_- \rangle.$$

In Appendix I, we present analytic expressions of q_{succ} and q_{dist} as functions of the physical error rate p , the code distances, and the logical error rates of several patches.

We now specify several assumptions underlying the above method:

- (1) We assume that the presence of irregular checks (e.g., two-body checks) within domain walls for lattice surgery has a negligible effect on the decoder's performance (that is, we treat irregular checks as regular checks when estimating logical error rates). Although this has not been rigorously verified, we anticipate that this assumption may hold to a sufficient degree, as domain walls preserve the code distance and do not increase the required connectivity of data and ancillary qubits. In particular, irregular checks can reduce the number of entangling gates acting on each data qubit from six to five [see Fig. 8(b)], and since this number is a key factor in determining the “effective” error rate of the data qubit, fault tolerance might be even better within domain walls than in the bulk. Furthermore, a recent study [54] indicates that noise along a 1D region within a 2D code does not substantially affect performance, providing additional support for this assumption.

- (2) We assume that, during lattice surgery, logical qubits that are being measured do not have logical errors anticommuting with any operators to measure. For example, when measuring $\bar{P}^{(k)} = \bar{Z}_{\{\text{OAB}\}}$ and $\bar{Q}^{(k)} = \bar{Z}_{\{\text{OCD}\}}$, we ignore $\bar{X}$ and $\bar{Y}$ errors on the output and validation qubits. This assumption is reasonable because error strings that lead to such logical errors have larger weights than the original code distances of the patches, as illustrated in Fig. 8(d).
- (3) Due to the previous item, error strings within the ancillary region can incur only $\bar{Z}_q$ errors on each logical qubit q being measured. We estimate the corresponding $\bar{Z}_q$ error rate by considering a hypothetical rectangular patch, with one boundary aligned along the boundary of the logical patch encoding q and the other extending across the entire ancillary region (see Appendix H 2 for more details). In this way, we can reasonably expect to avoid underestimating the $\bar{Z}_q$ error rate, as the hypothetical patch always contains at least as many minimum-weight error strings equivalent to $\bar{Z}_q$ than the ancillary region does, and their weights remain the same in both patches.
- (4) For the cultivation-MSD scheme, we assume that stages begin at fixed intervals of T_{intv} rounds. As defined in Sec. IV C, T_{intv} is actually the expected value of these intervals.

B. Results

In Fig. 12, we plot the effective spacetime costs (i.e., the spacetime costs divided by the success probabilities) and output infidelities of the schemes for various parameter combinations when $p \in \{10^{-4}, 5 \times 10^{-4}, 10^{-3}\}$ and $r_y = 1/10$. Note that the single-level scheme has four parameters ($d_{\text{out}}, d_X, d_Z, d_m$) and the cultivation-MSD scheme has seven parameters ($d_{\text{out}}, d_X, d_Z, d_m, d_{\text{cult}}, N_m, c_{\text{gap}}$), where $d_{\text{out}} - d_Z \leq d_m < 2d_Z$, $d_{\text{cult}} \leq d_m$, and c_{gap} is the logical gap threshold. To highlight the effect of postselection during the growing operation, we also plot the cost of a variant of the cultivation-MSD scheme where growing is performed for a single round without postselection. The cultivation-MSD scheme is not presented for $p = 10^{-4}$ due to the high computational cost of simulating the growing operation (see Appendix D). However, based on the other two cases, it is reasonable to expect that its cost would exhibit similar behavior to the green line (cultivation-MSD with postselection-free growing) for $q_{\text{dist}} \gtrsim 10^{-14}$ and eventually achieve very low infidelities under 10^{-22} .

Figure 12 shows that, if the target infidelity is within the range that the single-level scheme can reach (i.e., $q_{\text{dist}} \gtrsim 35(7p/3)^3$), it is generally more preferable than the cultivation-MSD scheme. If the target infidelity is lower

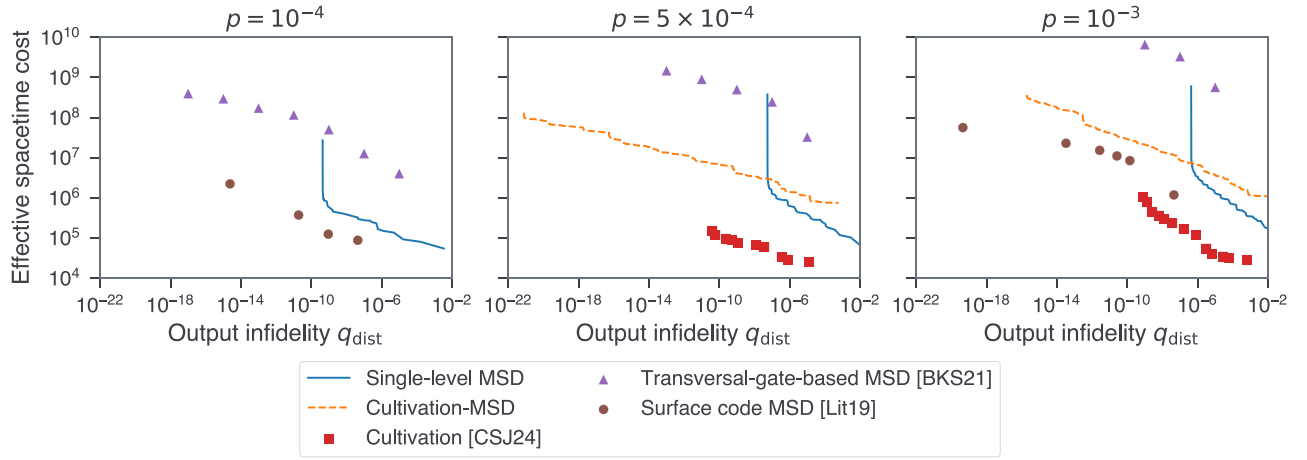


FIG. 12. Cost comparison of various magic state preparation schemes. The effective spacetime costs (i.e., spacetime costs divided by success probabilities q_{succ}) are plotted against the output infidelities q_{dist} for the single-level and cultivation-MSD schemes for various combinations of parameters, when $p \in \{10^{-4}, 5 \times 10^{-4}, 10^{-3}\}$ and $r_y = 1/10$. For comparison, the costs of the following schemes are additionally displayed: (i) a variant of the cultivation-MSD scheme in which the growing operation is performed for a single round without postselection, (ii) the cultivation protocol without MSD integration in Ref. [26], (iii) the transversal-gate-based color-code MSD scheme in Ref. [22], and (iv) the surface code MSD schemes in Ref. [23]. We suppose that a single round of syndrome extraction takes eight (six) time steps for color (surface) codes.

than this, the cultivation-MSD scheme should be used, which can achieve infidelities higher than 7.7×10^{-22} at $p = 5 \times 10^{-4}$ and 2.0×10^{-16} at $p = 10^{-3}$. We additionally note that the output infidelities do not significantly depend on r_y , as demonstrated in Appendix J. Specifically, the difference in infidelity for the two extreme cases, $r_y = 0$ and $r_y = 1$, is at most only a factor of about 1.5.

In Table I, we list several data points in Fig. 12 with more details on their failure probabilities and individual resource costs, where single-level and cultivation-MSD schemes are labeled as “sng- $(d_{\text{out}}, d_X, d_Z, d_m)$ ” and “cmb- $(d_{\text{out}}, d_X, d_Z, d_m, d_{\text{cult}}, N_m, c_{\text{gap}})$ ”, respectively.

For comparison, Fig. 12 additionally presents the costs of other schemes: the original cultivation scheme in Ref. [26], the color-code MSD scheme based on transversal gates in Ref. [22], and the surface-code schemes in Ref. [23]. The cultivation protocol is notably resource efficient as it does not require encoded operations; however, it is challenging to reach very low infidelities (e.g., $q_{\text{dist}} \lesssim 10^{-10}$ when $p = 10^{-3}$ and $q_{\text{dist}} \lesssim 10^{-11}$ when $p = 5 \times 10^{-4}$) since its success probability exponentially decreases as the fault distance grows. In contrast, our cultivation-MSD scheme offers a clear advantage in this regard, enabling the achievement of much lower infidelities suitable for implementing a wide range of quantum algorithms. Compared to the MSD scheme described in Ref. [22], which employs transversal operations between stacked patches, our schemes show a significant reduction in spacetime cost, improving by about 2 orders of magnitude. In comparison to surface-code schemes, our

color-code schemes exhibit higher spacetime costs for a given target infidelity. This difference is estimated to be less than one order of magnitude.

The main reason that MSD is more costly with color codes than with surface codes (despite their lower encoding rate and efficient lattice surgery capabilities) is their weaker fault tolerance under circuit-level noise. Specifically, the *scaling threshold* [the parameter p_{th} in the subthreshold ansatz given by Eq. (11)] is estimated as 0.2%–0.6% for color codes under the concatenated MWPM decoder (see Appendix G 4), whereas surface codes achieve a threshold of nearly 1% [56–59]. Nevertheless, circuit-level decoders for color codes continue to improve [22,27,37], motivating an analysis of how much further decoding methods need to improve for color codes to become preferable over surface codes in MSD applications.

In Fig. 13, we roughly estimate the spacetime costs of our MSD schemes at $p = 10^{-3}$, assuming that the scaling threshold (p_{th}) for each setting used in our decoder simulations (see Appendix G) improves to $(1 - \lambda)p_{\text{th}} + \lambda(1\%)$ with a tuning parameter λ , while all other parameters in the ansatz of Eq. (11) are unchanged. For the growing operation, we assume that the error rate at any c_{gap} is reduced by the same proportion as at $c_{\text{gap}} = 0$, while the acceptance rate remains invariant. The figure indicates that at least $\lambda \approx 0.4$ is necessary for the cultivation-MSD scheme to be more resource efficient than the surface-code scheme for certain target infidelities. This corresponds to thresholds of $p_{\text{th}} = 0.54\%$ for triangular patches, 0.65%

TABLE I. Output infidelities, failure rates, and resource costs of the single-level and cultivation-MSD schemes for various combinations of parameters at (a) $p = 5 \times 10^{-4}$, and (b) $p = 10^{-3}$. Each variant of the single-level and cultivation-MSD schemes is labeled as “sng- $(d_{\text{out}}, d_X, d_Z, d_m)$ ” or “cmb- $(d_{\text{out}}, d_X, d_Z, d_m, d_{\text{cult}}, N_m, c_{\text{gap}})$ ”. For the cultivation-MSD scheme, the values of T_m (minimum number of rounds between successive executions of the CN protocol), T_{intv} (average number of rounds between successive stages), and T_{idle} (average number of rounds that auxiliary patches idle before being consumed) are additionally specified. The values of T_{intv} and T_{idle} are estimated by simulating the procedure of the cultivation-MSD scheme described in Sec. IV C for 1000 stages.

Scheme	Output infidelity q_{dist}	Failure rate $1 - q_{\text{succ}}$	Space cost n (Qubits)	Time cost t (Time steps)	Effective spacetime cost (nt/q_{succ})	T_m	T_{intv}	T_{idle}
(a) $p = 5 \times 10^{-4}$								
sng-(11, 8, 6, 5)	1.52×10^{-5}	4.71×10^{-2}	833	384	3.36×10^5			
sng-(19, 10, 12, 7)	1.02×10^{-7}	1.99×10^{-2}	2401	512	1.25×10^6			
cmb-(23, 14, 16, 7, 3, 4, 5.03)	1.13×10^{-9}	3.69×10^{-3}	5347	759	4.08×10^6	2	11.9	0.6
cmb-(31, 18, 20, 11, 3, 3, 10.05)	1.11×10^{-12}	1.00×10^{-4}	8825	1298	1.15×10^7	2	20.3	1.2
cmb-(41, 22, 28, 13, 3, 4, 13.41)	1.09×10^{-15}	3.02×10^{-5}	1.59×10^4	1348	2.14×10^7	2	21.1	1.1
cmb-(49, 30, 34, 15, 5, 4, 18.12)	1.24×10^{-18}	4.09×10^{-6}	2.52×10^4	2272	5.73×10^7	3	35.5	3.6
cmb-(59, 34, 40, 19, 5, 6, 23.23)	1.06×10^{-21}	4.19×10^{-7}	4.02×10^4	2378	9.57×10^7	2	37.2	3.0
cmb-(63, 40, 44, 19, 5, 8, 23.23)	7.75×10^{-22}	4.21×10^{-7}	6.05×10^4	2073	1.25×10^8	2	32.4	2.2
(b) $p = 10^{-3}$								
sng-(19, 8, 12, 7)	1.21×10^{-5}	7.17×10^{-2}	2265	512	1.25×10^6			
sng-(25, 12, 16, 11)	1.03×10^{-6}	3.77×10^{-2}	4181	768	3.34×10^6			
cmb-(29, 16, 20, 9, 3, 5, 6.09)	1.04×10^{-7}	1.34×10^{-2}	9081	925	8.52×10^6	1	14.5	0.7
cmb-(39, 22, 26, 13, 3, 4, 7.60)	1.03×10^{-9}	1.96×10^{-3}	1.50×10^4	1391	2.10×10^7	2	21.7	1.4
cmb-(51, 28, 36, 15, 3, 4, 10.66)	1.00×10^{-11}	8.37×10^{-4}	2.60×10^4	1595	4.16×10^7	2	24.9	1.5
cmb-(63, 36, 46, 17, 5, 6, 16.75)	1.02×10^{-13}	1.95×10^{-4}	4.58×10^4	3020	1.38×10^8	2	47.2	5.1
cmb-(71, 36, 48, 23, 5, 8, 20.62)	1.01×10^{-15}	2.60×10^{-5}	6.70×10^4	3513	2.35×10^8	2	54.9	5.4
cmb-(81, 44, 58, 23, 5, 8, 20.62)	2.00×10^{-16}	2.65×10^{-5}	9.07×10^4	3513	3.19×10^8	2	54.9	5.4

($\bar{Z}$ failure) and 0.58% ($\bar{X}$ failure) for rectangular patches, and 0.77% for stability experiments (improved from the current thresholds of 0.24%, 0.42%, 0.31%, and 0.62%, respectively) [60].

Finally, we note that the performance analysis presented above relies on treating the cultivation and growing operations as separate modules. While this modular approach allows for flexibility in substituting different cultivation protocols, the interface between these two stages could potentially impact the overall performance. We investigate this effect in Appendix K by performing integrated simulations of cultivation followed immediately by the growing operation. These simulations account for detectors spanning the interface and their use in cultivation postselection. The results indicate a moderate increase in the spacetime cost of the cultivation-MSD scheme (by up to a factor of approximately 1.7 for the case $p = 10^{-3}$, $d_{\text{cult}} = 5$) compared to the separate analysis. This increase, while not entirely negligible, remains relatively minor considering that the overall spacetime costs shown in Fig. 12 span several orders of magnitude (from 10^4 to 10^{10}). In addition, we estimate that the minimum achievable output infidelity could increase by roughly 3–30 times, which is again not negligible but still allows the scheme to remain sufficiently competitive, given that the original infidelity value was already extremely low at about 2×10^{-16} . See Appendix K

for a detailed discussion and the corresponding simulation results.

VI. REMARKS

In this work, we proposed two magic-state-distillation (MSD) schemes for 2D color codes based on the 15-to-1 MSD circuit in Fig. 4. We presented the end-to-end descriptions of the schemes, from the definitions of logical qubits to the arrangement of logical-level operations, providing a comprehensive guide for their implementation. The *single-level MSD scheme*, covered in Sec. III, implements each $\pi/8$ -rotation by employing the faulty T measurement, i.e., measuring logical qubits non-fault-tolerantly in the basis of $\{|\bar{0}\rangle \pm e^{-i\pi/4} |\bar{1}\rangle\}$. The *cultivation-MSD scheme*, covered in Sec. IV, first fault tolerantly prepares magic states via cultivation [24–26], grows the patches to accommodate the low error rate of the cultivated magic states, and finally consumes them to implement the $\pi/8$ -rotations. Both schemes can be performed via multiple stages of lattice surgery, which measures Pauli operators of multiple logical qubits by extending the lattice and connecting the patch boundaries. The single-level scheme has a limited output infidelity of approximately $35(7p/3)^3$ when p is the physical error rate, while the

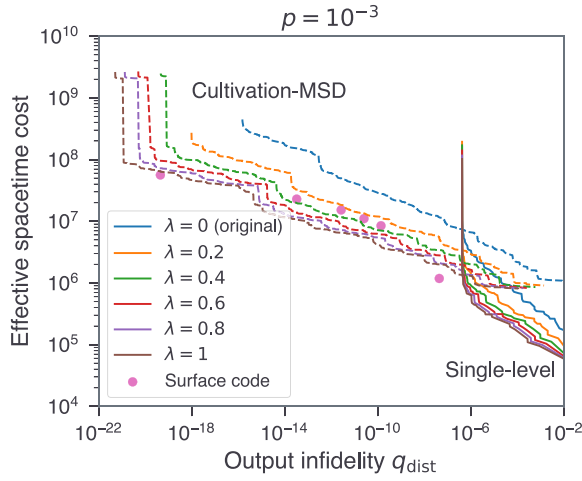


FIG. 13. Costs of our MSD schemes under improved thresholds at $p = 10^{-3}$, compared with those of the surface-code schemes from Ref. [55]. We assume that the threshold (p_{th}) for each setting (including triangular or rectangular patch memory experiments and stability experiments) is improved to $(1 - \lambda)p_{\text{th}} + \lambda(1\%)$, where λ is a tuning parameter, while keeping all other parameters in the ansatz unchanged. Solid and dashed lines represent the single-level and cultivation-MSD schemes, respectively.

cultivation-MSD scheme can achieve much lower infidelities (e.g., $\gtrsim 2 \times 10^{-16}$ when $p = 10^{-3}$) at the cost of higher resource overheads.

We optimized the resource costs of our MSD schemes through various approaches: First, each lattice surgery operation measures two commuting Pauli operators in parallel, which is possible thanks to the rich topological structure of the color codes [8]. Thus, the 15 rotations can be conducted pairwise through eight stages. Note that the way of pairing the rotations needs to be carefully chosen, as it affects the fault tolerance of MSD (see Sec. III E). In addition, logical qubits measured during MSD (such as the validation qubits A–D in Fig. 4) can be encoded in patches with smaller code distances than the patch for outputting a magic state, which is an approach used in Ref. [23] for surface codes. This is possible since memory errors in those qubits can be detected by the MSD circuit, while a logical error in the output patch directly affects the distilled magic state. Moreover, the validation qubits are encoded in rectangular patches in Fig. 2(b) instead of ordinary triangular patches in Fig. 2(a). Each rectangular patch encodes two logical qubits with their logical Z ($\bar{Z}$) operators supported on the same boundary (called its Z boundary), while their $\bar{X}$ operators are supported on another boundary (X boundary). Since lattice surgery for the MSD circuit involves only $\bar{Z}$ operators, we can let the ancillary region adjacent to only the Z boundaries of the two rectangular patches, which is more resource efficient than using four triangular patches. Another advantage of using rectangular patches

is that code distances (d_X, d_Z) for $\bar{X}$ and $\bar{Z}$ errors can be optimized separately, considering that they have different effects during MSD. However, contrary to common belief, we revealed that the MSD circuit is tolerant to single-location $\bar{X}$ errors as well as $\bar{Z}$ errors on validation qubits, thus d_X and d_Z do not have to be significantly different as exemplified in Table I.

For the cultivation-MSD scheme, a crucial problem was that the growing operation acted as a bottleneck, degrading a magic state generated through cultivation, as its fault tolerance depends on the code distance before growing. We resolved this issue by employing postselection based on the logical gap computed by the concatenated MWPM decoder [27]. The impact of postselection is noteworthy: by aborting only 20% of trials, the logical error rate can be suppressed by nearly three orders of magnitude when growing from $d_{\text{cult}} = 3$ to $d_m \geq 7$, as shown in Fig. 10.

By thoroughly investigating logical errors during MSD, we assessed the performance of our schemes based on their output infidelities and resource costs for various parameter combinations, as illustrated in Fig. 12 and Table I. Notably, our schemes have about 2 orders of magnitude lower spacetime costs compared to the previous color-code MSD scheme [22]. Furthermore, we verified that the cultivation-MSD scheme can reach sufficiently low output infidelities of about 2×10^{-16} when $p = 10^{-3}$ and about 7.8×10^{-22} when $p = 5 \times 10^{-4}$, which are challenging to achieve with either single-level MSD or cultivation alone.

Despite this improvement, our color-code schemes still do not surpass the performance of surface-code MSD, exhibiting less than an order of magnitude higher spacetime overhead. In order for color-code schemes to reduce these overheads, we require improvements in color-code decoders to achieve a higher circuit-level threshold (currently around 0.2%–0.6%), aiming closer to that of surface codes (approximately 1%). Specifically, by parametrizing the improvement in the *scaling threshold* [the parameter p_{th} in the subthreshold logical failure rate ansatz given by Eq. (11)] as $p_{\text{th}} \rightarrow (1 - \lambda)p_{\text{th}} + \lambda(1\%)$, we estimated that achieving at least $\lambda = 0.4$ is necessary for our schemes to outperform the surface-code scheme in Ref. [23]. In practice, this requires a decoder achieving a scaling threshold of $\gtrsim 0.54\%$ in memory experiments with triangular patches (improved from the current threshold of 0.24%). Although this remains challenging, we emphasize that it may not be strictly necessary to reach the 1% threshold to realize practical benefits. The current *cross thresholds* (the intersection points of logical failure rate curves at different code distances) of color-code decoders are already around 0.5% [22,27,37,61], thus reducing the gap between the scaling and cross thresholds would be an important next step. Note that the surface-code scheme could also benefit from integration with cultivation, potentially widening the performance gap between surface and color codes. However, converting a color code into a surface code (e.g.,

via the “grafting” technique proposed in Ref. [26]) introduces additional technical complexity and may require further simplification. Exploring this would be a promising direction for future work as well.

In addition, our analysis assumes that irregular checks for lattice surgery have a negligible effect on decoding performance. Although this assumption was partially justified in Sec. V A, developing refined color-code decoders that explicitly account for domain walls would be a valuable next step. Such decoders would enable simulations of the complete end-to-end MSD circuit rather than relying on the modular approach used in our analysis, leading to a more accurate evaluation of MSD performance.

Moreover, utilizing transversal Clifford gates for MSD is another intriguing avenue for investigation. While lattice surgery typically requires $O(d)$ rounds, transversal gates can be executed in $O(1)$ rounds [62], which enables a significant reduction in resource costs if Clifford gates between multiple logical qubits are implemented transversally rather than via lattice surgery. However, two major challenges arise: (i) Transversal gates between multiple logical patches inherently require long-range connectivity. Although stacking logical patches can partially alleviate this issue, gates involving distant logical patches may still demand a large number of swap operations unless hardware directly supports long-range connections. (ii) Logical patches must have identical sizes to facilitate transversal gates, imposing constraints on the optimization of the MSD layout. Nevertheless, leveraging transversal gates for MSD remains a promising direction for future research. For instance, one might consider applying transversal gates to implement two-qubit Clifford operations between ancillary logical qubits of equal size, while using lattice surgery for gates connecting output and ancillary qubits with different code distances. Furthermore, developing MSD schemes that fully exploit transversal gates could be particularly advantageous if hardware supports long-range connections, as explored in Refs. [62,63].

ACKNOWLEDGMENTS

We thank Samuel C. Smith, Andrew Li, Lucas English, Sam Roberts, and Craig Gidney for helpful discussions and comments. This work is supported by the Australian Research Council via the Centre of Excellence in Engineered Quantum Systems (EQUS) Project No. CE170100009. This article is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001122C0063. Any opinions, findings and conclusions or recommendations expressed in this article are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA).

DATA AVAILABILITY

The data that support the findings of this article are openly available [64].

APPENDIX A: PROOF OF FAULT TOLERANCE FOR LOGICAL X ERRORS IN THE MSD CIRCUIT

In this Appendix, we prove that any single $\bar{X}$ error on a validation qubit in middle of the 15-to-1 MSD circuit in Fig. 4 does not make a logical error on the output qubit, assuming that there are no other logical errors. Throughout this Appendix, all the states and operators are in the logical level and overlines are omitted for readability. We denote the set of qubits as $\mathcal{Q} := \{q_{\text{out}}, q_A, q_B, q_C, q_D\}$.

Denoting the i th $\pi/8$ -rotation of the MSD circuit as $P_{\pi/8}^{(i)}$ for each $i \in \{1, \dots, 15\}$, an X error on a qubit q after applying $P_{\pi/8}^{(n)}$ can be commuted to the beginning of the circuit and absorbed into the initial $|+\rangle$ state, which makes an error

$$E_{q,n} := \prod_{P \in \mathcal{P}_{q,n}} P_{-\pi/4},$$

where

$$\mathcal{P}_{q,n} := \{P^{(j)} \mid j \leq n \wedge q \in \text{supp } P^{(j)}\}. \quad (\text{A1})$$

We prove the following claim:

Claim 1. For an arbitrary qubit state $|\psi\rangle \in \mathcal{H}_2$, a validation qubit $q \in \{q_A, q_B, q_C, q_D\}$, and an integer $n \in \{1, \dots, 15\}$,

$$(\mathbb{1}_{\text{out}} \otimes \langle++++|_{\text{ABCD}}) E_{q,n} (|\psi\rangle_{\text{out}} \otimes |++++\rangle_{\text{ABCD}}) \propto |\psi\rangle_{\text{out}}. \quad (\text{A2})$$

Proof. Since $P_{-\pi/4} = (\mathbb{1} + iP)/\sqrt{2}$ for any Pauli operator P and every $P^{(j)}$ contains only Z operators, we have

$$\begin{aligned} E_{q,n} &\propto \prod_{P \in \mathcal{P}_{q,n}} (\mathbb{1} + iP) = \sum_{\tilde{\mathcal{P}} \subseteq \mathcal{P}_{q,n}} \left[i^{|\tilde{\mathcal{P}}|} \prod_{P \in \tilde{\mathcal{P}}} P \right] \\ &= \sum_{\tilde{\mathcal{P}} \subseteq \mathcal{P}_{q,n}} \left[i^{|\tilde{\mathcal{P}}|} \prod_{\substack{q' \in \mathcal{Q} \\ |\tilde{\mathcal{P}}_{q'}|: \text{odd}}} Z_{q'} \right], \end{aligned} \quad (\text{A3})$$

where an empty product $\prod_{P \in \emptyset} P$ is defined as $\mathbb{1}$ and

$$\tilde{\mathcal{P}}_{q'} := \{P \in \tilde{\mathcal{P}} \mid q' \in \text{supp } P\}. \quad (\text{A4})$$

It is straightforward to see that

$$\begin{aligned}\tilde{\mathcal{P}}_q &= \tilde{\mathcal{P}}, \\ \sum_{q' \in Q} |\tilde{\mathcal{P}}_{q'}| &= \sum_{P \in \tilde{\mathcal{P}}} |\text{supp } P|.\end{aligned}$$

Assume for contradiction that the claim does not hold. Then the summation in Eq. (A3) contains at least one term involving only Z_{out} (and a constant factor), since all other terms except $\mathbb{1}$ vanish when computing the left-hand side of Eq. (A2). Therefore, there exists a subset $\tilde{\mathcal{P}} \subseteq \mathcal{P}_{q,n}$ such that $|\tilde{\mathcal{P}}_{q_{\text{out}}}|$ is odd and $|\tilde{\mathcal{P}}_{q'}|$ is even for every $q' \in \{q_A, q_B, q_C, q_D\}$, which entails that $\sum_{q' \in Q} |\tilde{\mathcal{P}}_{q'}| = \sum_{P \in \tilde{\mathcal{P}}} |\text{supp } P|$ is odd. Since $|\text{supp } P^{(j)}|$ is odd for every j , $|\tilde{\mathcal{P}}| = |\tilde{\mathcal{P}}_q|$ is also odd, implying that q should be the output qubit q_{out} , which contradicts the precondition that q is a validation qubit. This contradiction shows that the assumption is false, and hence the claim holds. ■

APPENDIX B: GENERAL LATTICE-SURGERY SCHEME

In this Appendix, we describe the macroscopic process of lattice surgery for measuring an arbitrary pair of commuting logical Pauli operators of triangular color codes.

We consider N triangular logical patches surrounding an ancillary region adjacent to their red boundaries of the patches, as exemplified in Fig. 14(a) for $N = 8$. In each patch, the red, green, and blue boundaries are placed clockwise. The N logical qubits encoded in the patches are labeled as $\bar{q}_1, \dots, \bar{q}_N$ in a clockwise order starting from an arbitrary patch. We aim to measure two commuting Pauli operators $\bar{P} = \bigotimes_{i=1}^N \bar{P}_i$ and $\bar{Q} = \bigotimes_{i=1}^N \bar{Q}_i$ on these logical

qubits, where $\bar{P}_i, \bar{Q}_i \in \{\bar{I}, \bar{X}, \bar{Y}, \bar{Z}\}$ are logical Pauli operators on $\bar{q}_i$. The procedure of the lattice surgery operation is as follows:

- (1) Initialize the ancillary region with a red temporal boundary; that is, prepare the Bell state $|\Phi_+\rangle := (|00\rangle + |11\rangle)/\sqrt{2}$ on every red edge in the region.
- (2) **(Merging operation)** Merge the triangular patches with the ancillary region by measuring appropriate checks, as shown in Fig. 14(b). The interior of the ancillary region has ordinary color-code checks. Domain walls are placed between this interior and the triangular patches, which are visualized as orange regions, where checks are deformed from ordinary color-code ones. Exceptionally, if $\bar{P}_i = \bar{Q}_i = \bar{I}$, the patch for $\bar{q}_i$ is not merged with the ancillary region, such as $\bar{q}_3$ and $\bar{q}_6$ in Fig. 14(b). The colors of the boundaries (purple lines) between the triangular patches should be carefully chosen. Denoting them as $c_1, \dots, c_N \in \{g, b\}$ in a cyclic way from the boundary between $\bar{q}_N$ and $\bar{q}_1$, we determine them by the following rule: for every $i \in \{1, \dots, N\}$,

$$\begin{cases} c_{i+1} = c_i & \text{if } [\bar{P}_i, \bar{Q}_i] = 0, \\ c_{i+1} \neq c_i & \text{otherwise,} \end{cases} \quad (\text{B1})$$

where $c_{N+1} := c_1$.

- (3) Wait for d code cycles to correct timelike error strings in the ancillary region.
- (4) **(Splitting operation)** Separate the triangular patches from the ancillary region by measuring it with a red temporal boundary. The measurement outcomes of $\bar{P}$ and $\bar{Q}$ and the updated Pauli frames of the logical qubits are determined from the measurement outcomes of checks and red edges in the ancillary region.

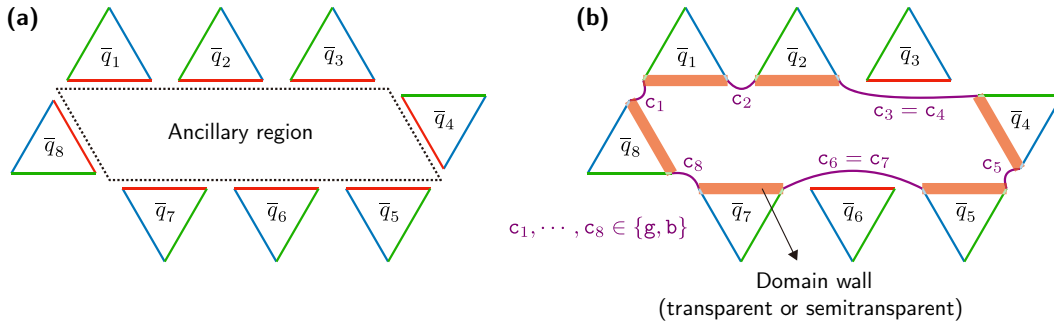


FIG. 14. (a) Layout for lattice surgery, where $N = 8$ triangular patches surround an ancillary region adjacent to their red boundaries. In each face, its red, green, and blue boundaries are placed clockwise. The eight logical qubits encoded in the patches are labeled as $\bar{q}_1, \bar{q}_2, \dots, \bar{q}_8$. (b) Layout during the merging operation of lattice surgery, where $\bar{q}_3$ and $\bar{q}_6$ are not involved in the operation. The triangular patches are connected via the ancillary region with domain walls placed between the patches and the ancillary region. The ancillary region has boundaries (purple lines) between adjacent patches that are nontrivially involved in the operation. The colors of these boundaries are labeled as $c_1, \dots, c_8 \in \{g, b\}$, where $c_3 = c_4$ and $c_6 = c_7$ in this case.

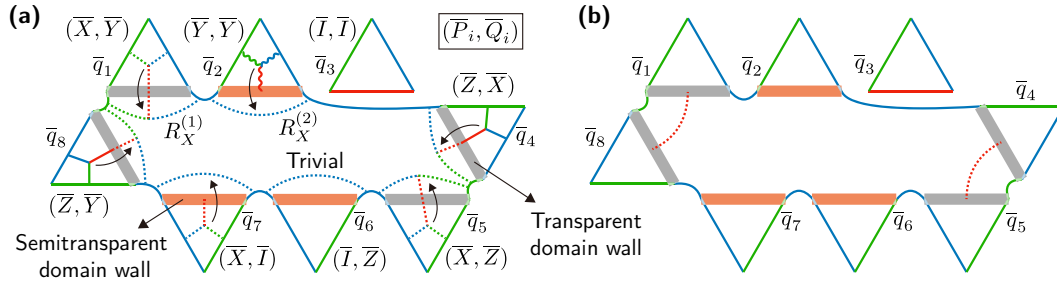


FIG. 15. Example of a lattice surgery operation for measuring $\bar{P} = \bar{X}_1 \bar{Y}_2 \bar{I}_3 \bar{Z}_4 \bar{X}_5 \bar{I}_6 \bar{X}_7 \bar{Z}_8$ and $\bar{Q} = \bar{Y}_1 \bar{Y}_2 \bar{I}_3 \bar{X}_4 \bar{Z}_5 \bar{Z}_6 \bar{I}_7 \bar{Y}_8$. Solid, dotted, and wavy lines indicate Pauli-Z, X, and Y string operators, respectively. (a) The string-net operator in the patch of $\bar{q}_i$, representing $\bar{P}_i$, is transformed into a string-net operator $R_X^{(i)}$ inside the ancillary region via a transparent (gray) or semitransparent (orange) domain wall, determined by Table II. Exceptionally, $\bar{q}_3$ and $\bar{q}_6$ are not involved in $\bar{P}$, thus $R_X^{(3)}$ and $R_X^{(6)}$ are trivial (i.e., equivalent to identity). (b) The product of $R_X^{(i)}$'s is equivalent to a product of rx-string operators connecting transparent domain walls within the ancillary region, meaning that the value of $\bar{P}$ can be determined from the measurement outcomes of checks and red edges. The value of $\bar{Q}$ can be determined similarly.

Note that, since $\bar{P}$ and $\bar{Q}$ commute, there are even flips of colors between neighboring boundaries connecting triangular patches, thus we can always find $\{c_i\}$ that satisfies the rule in Eq. (B1).

The domain wall adjacent to a triangular patch $\bar{q}_i$ (nontrivially involved in the lattice surgery operation) is selected so that $\bar{P}_i (\bar{Q}_i)$ is equivalent to a Pauli-X (Z) string-net operator $R_X^{(i)} (R_Z^{(i)})$ in the ancillary region, which terminates at the adjacent boundaries of the ancillary region and may also terminate at the domain wall. In other words, $\bar{P}_i (\bar{Q}_i)$ can be transformed into $R_X^{(i)} (R_Z^{(i)})$ by multiplying checks in the domain wall. See Fig. 15(a) for an example showing the transformation of $\bar{P}$ when $\bar{P} = \bar{X}_1 \bar{Y}_2 \bar{I}_3 \bar{Z}_4 \bar{X}_5 \bar{I}_6 \bar{X}_7 \bar{Z}_8$ and $\bar{Q} = \bar{Y}_1 \bar{Y}_2 \bar{I}_3 \bar{X}_4 \bar{Z}_5 \bar{Z}_6 \bar{I}_7 \bar{Y}_8$. As a result, $\bar{P} (\bar{Q})$ is equivalent to $R_X := \prod_{i=1}^N R_X^{(i)}$ ($R_Z := \prod_{i=1}^N R_Z^{(i)}$) under stabilizer multiplication, which can be transformed further into a product of rx-string (rz-string) operators connecting domain walls in the ancillary region, as shown in Fig. 15(b). Hence, we can determine the value of $\bar{P}$ and $\bar{Q}$ from the measurement outcomes of checks and red edges in the ancillary region including the domain

walls. The microscopic structure of the layout should be considered to specify exactly which checks and red edges are involved in this, which will not be covered in this Appendix.

For the above argument to be valid, we should appropriately choose the type of each domain wall (for $\bar{q}_i$), which depends on $\bar{P}_i$ and $\bar{Q}_i$. We present this determination rule in Table II for five different cases: (i) $[\bar{P}_i \bar{Q}_i] \neq 0$, (ii) $\bar{P}_i = \bar{Q}_i \neq \bar{I}$, (iii) $\bar{P}_i \neq \bar{Q}_i = \bar{I}$, (iv) $\bar{Q}_i \neq \bar{P}_i = \bar{I}$, and (v) $\bar{P}_i = \bar{Q}_i = \bar{I}$. Here, we denote the Pauli charge label corresponding to $\bar{P}_i$ and $\bar{Q}_i$ as p_i and q_i , respectively (e.g., $p_i = x$ for $\bar{P}_i = \bar{X}$). Let us examine these cases one by one, except the trivial case (v) where $\bar{q}_i$ and the ancillary region are disconnected (namely, an opaque domain wall is placed).

1. $[\bar{P}_i \bar{Q}_i] \neq 0$ ($c_i \neq c_{i+1}$)

In this case, we place a transparent domain wall such that a boson crossing it from $\bar{q}_i$ to the ancillary region

TABLE II. Determination rule of the domain wall placed between the i th triangular patch ($\bar{q}_i$) and the ancillary region (Anc.) during a lattice surgery operation for measuring $\bar{P} = \bigotimes_{j=1}^N \bar{P}_j$ and $\bar{Q} = \bigotimes_{j=1}^N \bar{Q}_j$. We say that a semitransparent domain wall condenses (cw, c'w') for two charge labels cw and c'w' if it condenses cw and c'w' on its sides facing $\bar{q}_i$ and the ancillary region, respectively.

$(\bar{P}_i, \bar{Q}_i)$	Boundary colors ($c_i, c_{i+1} \in \{g, b\}$)	Domain-wall type	Effects on bosons (p_i, q_i : Pauli charge labels for $\bar{P}_i, \bar{Q}_i$)
$[\bar{P}_i, \bar{Q}_i] \neq 0$	$c_i \neq c_{i+1}$	Transparent	Permute charge labels with rules ($\bar{q}_i$ on left/Anc. on right): $\{g \leftrightarrow c_i, b \leftrightarrow c_{i+1}, p_i \leftrightarrow x, q_i \leftrightarrow z\}$.
$\bar{P}_i = \bar{Q}_i \neq \bar{I}$	$c_i = c_{i+1}$	Semitransparent	Condense (rp _i , c _i y); em exchanging.
$\bar{P}_i \neq \bar{Q}_i = \bar{I}$	$c_i = c_{i+1}$	Semitransparent	Condense (rp _i , c _i z); em exchanging.
$\bar{Q}_i \neq \bar{P}_i = \bar{I}$	$c_i = c_{i+1}$	Semitransparent	Condense (rq _i , c _i x); em exchanging.
$\bar{P}_i = \bar{Q}_i = \bar{I}$	$c_i = c_{i+1}$	Opaque	Condense rx, ry, rz on $\bar{q}_i$ side and c _i x, c _i y, c _i z on Anc. side.

undergoes a charge permutation $\sigma(cw) = c'w'$, where

$$c' = \begin{cases} c_i & \text{for } c = g, \\ c_{i+1} & \text{for } c = b, \\ r & \text{otherwise,} \end{cases} \quad w' = \begin{cases} x & \text{for } w = p_i, \\ z & \text{for } w = q_i, \\ y & \text{otherwise.} \end{cases}$$

For instance, when $(\bar{P}_i, \bar{Q}_i) = (\bar{Y}, \bar{X})$ and $(c_i, c_{i+1}) = (b, g)$, $\bar{P}_i = \bar{Y}$ can be represented as a Pauli-Y string-net operator connecting the three boundaries of the patch. This operator can be transformed by multiplying stabilizers into a Pauli-X string-net operator $R_X^{(i)}$ belonging to the ancillary region, as $\sigma(ry) = rx$, $\sigma(gy) = bx$, and $\sigma(by) = gx$. Similarly, we can find a Pauli-Z string-net operator $R_Z^{(i)} \sim \bar{Q}_i = \bar{X}$, where “ $U_1 \sim U_2$ ” for operators U_1 and U_2 denotes that U_1 and U_2 are equivalent under stabilizer multiplication.

2. $\bar{P}_i = \bar{Q}_i \neq \bar{I}$ ($c_i = c_{i+1}$)

In this case, we place a semitransparent domain wall between $\bar{q}_i$ and the ancillary region. This domain wall condenses rp_i and $c_i y$ on the sides facing $\bar{q}_i$ and the ancillary region, respectively, and its charge-changing rule is em exchanging. For example, when $\bar{P}_i = \bar{Q}_i = \bar{Z}$ and $c_i = c_{i+1} = b$, the domain wall is characterized by the boson tables as

$$\begin{array}{|c|c|c|} \hline \blacksquare & \times & \times \\ \hline \blacksquare & \times & \times \\ \hline \bullet & \blacktriangle & \blacktriangle \\ \hline \end{array} \xleftrightarrow[\bar{q}_i]{\text{Anc.}} \begin{array}{|c|c|c|} \hline \times & \times & \blacksquare \\ \hline \blacktriangle & \blacktriangle & \bullet \\ \hline \times & \times & \blacksquare \\ \hline \end{array},$$

which is a notation defined in Sec. II E 3. Here, gz and bz ($\blacktriangle$) in $\bar{q}_i$ are deconfined and can be mapped to any of bx and bz ($\blacksquare$) in the ancillary region, while rz is condensed in $\bar{q}_i$. Therefore, if we move a Pauli-Z string-net operator that represents $\bar{P}_i = \bar{Q}_i = \bar{Z}$ from the triangular patch to the ancillary region, we can obtain any of bx - and bz -string operators ($R_X^{(i)}, R_Z^{(i)}$) connecting the adjacent blue boundaries. (Note that $R_X^{(i)} \sim R_Z^{(i)}$.) Additionally, we can ensure that $\bar{X}$ and $\bar{Y}$ cannot be moved outside the patch in a similar way since gx , bx , gy , and by are confined in the patch.

3. $\bar{P}_i \neq \bar{Q}_i = \bar{I}$ or $\bar{Q}_i \neq \bar{P}_i = \bar{I}$ ($c_i = c_{i+1}$)

If $\bar{Q}_i$ is identity but $\bar{P}_i$ is not, we place an em -exchanging semitransparent domain wall that condenses rp_i and $c_i z$ on its sides facing $\bar{q}_i$ and the ancillary region, respectively. For example, when $(\bar{P}_i, \bar{Q}_i) = (\bar{Y}, \bar{I})$ and $c_i = c_{i+1} = g$, the domain wall is characterized by

$$\begin{array}{|c|c|c|} \hline \blacksquare & \times & \times \\ \hline \bullet & \blacktriangle & \blacktriangle \\ \hline \blacksquare & \times & \times \\ \hline \end{array} \xleftrightarrow[\bar{q}_i]{\text{Anc.}} \begin{array}{|c|c|c|} \hline \times & \blacksquare & \times \\ \hline \times & \blacksquare & \times \\ \hline \blacktriangle & \bullet & \blacktriangle \\ \hline \end{array}.$$

Here, gy and by ($\blacktriangle$) are deconfined from $\bar{q}_i$ and can be mapped to any of gx and gy ($\blacksquare$), while ry is condensed

on the $\bar{q}_i$ side of the domain wall. Hence, we can find a gx -string operator $R_X^{(i)}$ connecting the adjacent green boundaries that is equivalent to $\bar{Y}$ of $\bar{q}_i$. On the other hand, any gz -string operator $R_Z^{(i)}$ connecting the adjacent green boundaries is trivial, as gz is condensed on the ancillary region side of the domain wall. Similar to the previous case, $\bar{X}$ and $\bar{Z}$ are confined inside the patch. For the opposite case where $\bar{P}_i$ is identity but $\bar{Q}_i$ is not, we can do similarly with a domain wall that condenses rq_i and $c_i x$ on its two sides.

APPENDIX C: FAULT TOLERANCE OF THE MSD LAYOUT

In this Appendix, we prove that our MSD layouts in Figs. 6(a) and (b) are distance preserving if and only if the conditions in Condition 1 are met.

1. Notations

We first define notations. We schematize the ancillary region as a pentagon with edges labeled by e_0, e_1, e_2, e_3 , and e_4 (in clockwise order), each of which is adjacent to one of the five patches ($TRI_{\text{out}}, REC_{AB}, REC_{CD}, TRI_\alpha$, and TRI_β), as shown in Fig. 16. We denote the set of these five edges as $E := \{e_0, e_1, e_2, e_3, e_4\}$. We extend the series cyclically as $e_{i+5l} = e_i$ for every integer l . The patch adjacent to e_i and the corresponding domain wall is referred to as PAT_i and DW_i , respectively. The vertex between e_i and e_j (where $j = i + 5l \pm 1$ for an integer l) is denoted as v_{ij} and a set of vertices $V := \{v_{0,1}, v_{1,2}, v_{2,3}, v_{3,4}, v_{4,0}\}$ is defined. (We also sometimes omit commas in v_{ij} , e.g., $v_{01} := v_{0,1}$.) Every vertex or edge represents a specific 1D subregion in the ancillary region: An edge e_i represents the ancillary region side of DW_i and a vertex v_{ij} represents the green boundary connecting DW_i and DW_j . (Hereafter we identify each edge or vertex with the subregion represented

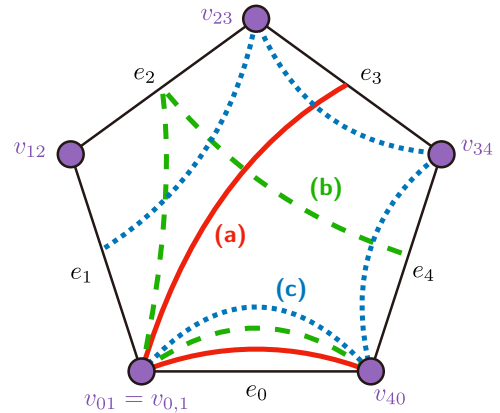


FIG. 16. Schematics of the ancillary region and (a)–(c) three examples of cyclic piecewise green string (CPGS) operators.

by it.) We also define the i th code distance

$$d_i := \begin{cases} d_{\text{out}} & \text{if } PAT_i \text{ is } TRI_{\text{out}}, \\ d_Z & \text{if } PAT_i \text{ is } REC_{AB} \text{ or } REC_{CD}, \\ d_m & \text{otherwise.} \end{cases}$$

We now consider a specific pair of Pauli operators $(\bar{P}, \bar{Q})$ to measure. Since every vertex $v_{i,i+1}$ is a green boundary, the set of bosons condensed by the vertex is

$$\mathcal{C}(v_{i,i+1}) := \{gx, gy, gz\}.$$

On the other hand, each edge condenses all or some of the green bosons depending on the Pauli operator measured on the corresponding patch. Namely, the set of bosons condensed by e_i is given as

$$\mathcal{C}(e_i) = \mathcal{C}_i := \begin{cases} \{gx, gy, gz\} & \text{if } \bar{P}_i = \bar{Q}_i = \mathbb{1}, \\ \{gx\} & \text{if } \bar{Q}_i \neq \bar{P}_i = \mathbb{1}, \\ \{gz\} & \text{if } \bar{P}_i \neq \bar{Q}_i = \mathbb{1}, \\ \{gy\} & \text{if } \bar{P}_i = \bar{Q}_i \neq \mathbb{1}, \\ \emptyset & \text{otherwise,} \end{cases} \quad (C1)$$

where $\bar{P}_i, \bar{Q}_i \in \{\bar{I}, \bar{Z}\}^{\otimes m_i}$ are, respectively, the restrictions of $\bar{P}$ and $\bar{Q}$ on the $m_i \in \{1, 2\}$ logical qubit(s) encoded in PAT_i ; see Fig. 8(b) for an example. For instance, if $PAT_0 = TRI_{\text{out}}$, $PAT_2 = REC_{AB}$, and $\bar{P} = \bar{Z}_{\{OAC\alpha\}}$, we get $\bar{P}_0 = \bar{P}_{\text{out}} = \bar{Z}$ and $\bar{P}_2 = \bar{P}_{AB} = \bar{Z} \otimes \bar{I}$. Note that, if PAT_i is TRI_{α} (TRI_{β}), $\mathcal{C}_i = \{gz\}$ ($\{gx\}$). For an index set $I = \{i_1, i_2, \dots, i_l\} \subset \mathbb{Z}$, we also define

$$\mathcal{C}_I = \mathcal{C}_{i_1, i_2, \dots, i_l} := \begin{cases} \{gx, gy, gz\} & \text{if } I = \emptyset, \\ \bigcap_{i \in I} \mathcal{C}_i & \text{otherwise.} \end{cases}$$

We define a cyclic order of the set $V \cup E$ as

$$\begin{aligned} v_{4,0} < e_0 < v_{0,1} < e_1 < v_{1,2} < e_2 \\ < v_{2,3} < e_3 < v_{3,4} < e_4 < v_{4,0}. \end{aligned}$$

Unlike a normal linear ordering, a cyclic order is only well-defined between three or more elements, such as $v_{4,0} < e_2 < v_{3,4}$. For $o, o' \in V \cup E$, we define

$$I_b(o, o') := \{i \in \{0, 1, 2, 3, 4\} \mid o < e_i < o'\}.$$

We say that o and o' are topologically connected with respect to a gw boson (denoted by $o \sim_w o'$) if and only if

$$gw \in \mathcal{C}(o) \cap \mathcal{C}(o') \cap [\mathcal{C}_{I_b(o, o')} \cup \mathcal{C}_{I_b(o', o)}],$$

meaning that an endpoint of a gw -string operator at o can be freely moved to o' .

For two objects $o_1, o_2 \in V \cup E$ and a Pauli label $w \in \{x, y, z\}$, we define a set $\mathcal{R}_w(o_1, o_2)$ to contain all gw -string operators terminating *only at* o_1 and o_2 . Here, we use the term “only at” to clarify that $\mathcal{R}_w(o_1, o_2)$ does not include “piecewise” string operators such as $R_{ac}R_{cb}$ for $R_{ac} \in \mathcal{R}_w(o_1, o_3)$ and $R_{cb} \in \mathcal{R}_w(o_3, o_2)$ whose endpoints on o_3 are not the same. Note that $\mathcal{R}_w(o_1, o_2)$ is nonempty if and only if $gw \in \mathcal{C}(o_1) \cap \mathcal{C}(o_2)$. We define

$$\Delta_w(o_1, o_2) := \begin{cases} \min_{R \in \mathcal{R}_w(o_1, o_2)} \text{wt}(R) & \text{if } \mathcal{R}_w(o_1, o_2) \neq \emptyset, \\ \infty & \text{otherwise,} \end{cases}$$

$$\Delta(o_1, o_2) := \min_{w \in \{x, y, z\}} \Delta_w(o_1, o_2),$$

where $\text{wt}(\cdot)$ denotes the weight of the given operator. Note that $\Delta_w(o_1, o_2)$ is either $\Delta(o_1, o_2)$ or ∞ for each w . We also define a “primed” distance as

$$\Delta'(o_1, o_2) := \min_{\substack{o'_1 \in \{o_1\} \cup \text{Ep}(o_1) \\ o'_2 \in \{o_2\} \cup \text{Ep}(o_2)}} \Delta(o'_1, o'_2), \quad (C2)$$

where $\text{Ep}(o)$ is the set of the endpoints of o , which is empty if o is a vertex and $\{v_{i,i\pm 1}\}$ if o is an edge e_i . We denote a chain sum of primed or unprimed distances simply as

$$\begin{aligned} \Delta^{(\prime)}(o_1, o_2, o_3, o_4, \dots, o_l) &:= \Delta^{(\prime)}(o_1, o_2) + \Delta^{(\prime)}(o_2, o_3) \\ &+ \Delta^{(\prime)}(o_3, o_4) + \dots \\ &+ \Delta^{(\prime)}(o_{l-1}, o_l). \end{aligned}$$

We recall that some of the notations defined above, such as $\mathcal{C}(\cdot)$, $\mathcal{R}_w(\cdot, \cdot)$, and $\Delta^{(\prime)}(\cdot, \cdot)$, depend on the pair of Pauli operators $(\bar{P}, \bar{Q})$ to measure. Therefore, when using this notation, we always suppose a specific distillation stage explicitly or implicitly.

2. Condition for a layout to be distance preserving

The following is a sufficient condition for the layout to be distance preserving:

Condition 2 ((Fault tolerance of the ancillary region)). For every distillation stage and each $w \in \{x, y, z\}$, if $gw \notin \mathcal{C}_i \cup \mathcal{C}_{i-2, i-1, i+1, i+2}$ (that is, $v_{i,i-1}$ and $v_{i,i+1}$ are not topologically connected with respect to gw), any operator equivalent to an operator in $\mathcal{R}_w(v_{i,i-1}, v_{i,i+1})$ should have a weight not less than d_i to prevent any possible logical errors on PAT_i .

To exhaustively address every operator equivalent to an operator in $\mathcal{R}_w(v_{i,i-1}, v_{i,i+1})$, we consider *cyclic piecewise green string (CPGS) operators*, which are trivial operators that can be written as products of nontrivial green string

operators with the same Pauli label. More formally, an L -segmented CPGS operator (for $L \geq 2$) has a form of

$$R_{\text{cyc}} = \prod_{j=0}^{L-1} R_w(o_j, \tilde{o}_j), \quad (\text{C3})$$

where $w \in \{x, y, z\}$, $o_j, \tilde{o}_j \in V \cup E$, $R_w(o_j, \tilde{o}_j) \in \mathcal{R}_w(o_j, \tilde{o}_j)$, and $o_j \not\sim_w \tilde{o}_j \sim_w o_{j+1}$ for each j (denoting $o_L := o_0$). Objects $o_1, \tilde{o}_1, \dots, o_{L-1}, \tilde{o}_{L-1}$ are called the endpoints of the CPGS operator. For example, when only e_1 among the five edges condenses

gz, $R_z(v_{4,0}, v_{0,1}) R_z(v_{1,2}, v_{2,3}) R_z(v_{2,3}, v_{4,0})$ is a three-segmented CPGS operator. It is worth noting that o_j and $\tilde{o}_j$ cannot be adjacent edges (such as e_0 and e_1) or an edge and one of its endpoints (such as e_0 and $v_{0,1}$). Without loss of generality, we can regard $[o_0, \tilde{o}_0, o_1, \tilde{o}_1, \dots, o_{L-1}, \tilde{o}_{L-1}, o_0]$ to be placed clockwise in order (namely, $o_0 < \tilde{o}_0 < o_1 < \dots < \tilde{o}_{L-1} < o_0$); see Appendix C4 below for its justification.

For R_{cyc} in Eq. (C3) to exist, we need $\mathcal{R}_w(o_j, \tilde{o}_j)$ to be nonempty and $o_j \not\sim_w \tilde{o}_j \sim_w o_{j+1}$ for each j . Hence, a necessary and sufficient condition for this is

$$\begin{aligned} g_w &\in \bigcap_{j=0}^{L-1} \left[\overbrace{\mathcal{C}(o_j) \cap \mathcal{C}(\tilde{o}_j)}^{(a)} \cap \overbrace{(\mathcal{C}_{I_b(\tilde{o}_j, o_{j+1})} \cup \mathcal{C}_{I_b(o_{j+1}, \tilde{o}_j)})}^{(b)} \setminus \overbrace{(\mathcal{C}_{I_b(o_j, \tilde{o}_j)} \cup \mathcal{C}_{I_b(\tilde{o}_j, o_j)})}^{(c)} \right] \\ &= \bigcap_{j=0}^{L-1} [\mathcal{C}(o_j) \cap \mathcal{C}(\tilde{o}_j) \cap \mathcal{C}_{I_b(\tilde{o}_j, o_{j+1})} \setminus \mathcal{C}_{I_b(o_j, \tilde{o}_j)}]. \end{aligned} \quad (\text{C4})$$

Here we require part (a) for $\mathcal{R}_w(o_j, \tilde{o}_j)$ to be nonempty, part (b) for $\tilde{o}_j \sim_w o_{j+1}$, and part (c) for $o_j \not\sim_w \tilde{o}_j$. The equality holds since $I_b(o_j, \tilde{o}_j) \subseteq I_b(o_{j+1}, \tilde{o}_j)$ thus $\mathcal{C}_{I_b(o_{j+1}, \tilde{o}_j)} \setminus \mathcal{C}_{I_b(o_j, \tilde{o}_j)} = \emptyset$, and also, $I_b(o_{j+1}, \tilde{o}_{j+1}) \subseteq I_b(\tilde{o}_j, o_j)$ thus $\mathcal{C}_{I_b(\tilde{o}_j, o_j)} \subseteq \mathcal{C}_{I_b(o_{j+1}, \tilde{o}_{j+1})}$.

For a given list of endpoints $[o_1, \tilde{o}_1, \dots, o_{L-1}, \tilde{o}_{L-1}]$, the corresponding CPGS operator exists for at least one Pauli label if and only if the right-hand side of Eq. (C4) is not empty, which is equivalent to

$$\bigcap_{j=0}^{L-1} [\mathcal{C}(o_j) \cap \mathcal{C}(\tilde{o}_j) \cap \mathcal{C}_{I_b(\tilde{o}_j, o_{j+1})}] \not\subseteq \bigcup_{j=0}^{L-1} \mathcal{C}_{I_b(o_j, \tilde{o}_j)}. \quad (\text{C5})$$

Provided that a CPGS operator R_{cyc} with $o_0 = v_{i-1,i}$ and $\tilde{o}_0 = v_{i,i+1}$ exists, $R_w(v_{i-1,i}, v_{i,i+1})$ is equivalent to $R_{\text{cyc}} R_w(v_{i-1,i}, v_{i,i+1}) = \prod_{j=1}^{L-1} R_w(o_j, \tilde{o}_j)$, thus we need to impose a requirement

$$\sum_{j=1}^{L-1} \Delta(o_j, \tilde{o}_j) \geq d_i. \quad (\text{C6})$$

In addition, we can equivalently use the primed distance defined in Eq. (C2) as

$$\sum_{j=1}^{L-1} \Delta'(o_j, \tilde{o}_j) \geq d_i, \quad (\text{C7})$$

since, if an edge e_j is an endpoint of R_{cyc} , then $e_j \sim_w v_{j,j \pm 1}$, thus Eq. (C6) also should hold if e_j is replaced with $v_{j,j \pm 1}$.

In summary, for given pairs of Pauli operators $\{(\overline{P}_{\text{LS}}^{(k)}, \overline{Q}_{\text{LS}}^{(k)})\}_{k=3}^8$ measured in the last six distillation stages, the following is a sufficient condition for the layout to be distance preserving.

Condition 3. For each distillation stage that measures $\overline{P}_{\text{LS}}^{(k)}$ and $\overline{Q}_{\text{LS}}^{(k)}$ (where $k \geq 3$), each $i \in \{0, 1, 2, 3, 4\}$, and each possible odd-length sequence of vertices and edges $[o_0, \tilde{o}_0, o_1, \tilde{o}_1, \dots, o_{L-1}, \tilde{o}_{L-1}, o_L]$ that are arranged clockwise in the pentagon and fulfills $o_0 = o_L = v_{i-1,i}$ and $\tilde{o}_0 = v_{i,i+1}$, if Eq. (C5) holds with respect to $(\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4)$ determined by $(\overline{P}_{\text{LS}}^{(k)}, \overline{Q}_{\text{LS}}^{(k)})$, then Eq. (C6) or equivalently Eq. (C7) should be satisfied.

In Fig. 16, we present three examples of potential CPGS operators for $i = 0$, which, respectively, give the following requirements:

- (a): $\mathcal{C}_{34} \subseteq \mathcal{C}_0 \cup \mathcal{C}_{12} \Rightarrow \Delta'(v_{01}, e_3) \geq d_0$,
- (b): $\mathcal{C}_{24} \subseteq \mathcal{C}_0 \cup \mathcal{C}_1 \cup \mathcal{C}_3 \Rightarrow \Delta'(v_{01}, e_2, e_4) \geq d_0$,
- (c): $\mathcal{C}_1 \subseteq \mathcal{C}_0 \cup \mathcal{C}_2 \cup \mathcal{C}_3 \cup \mathcal{C}_4 \Rightarrow \Delta'(e_1, v_{23}, v_{34}, v_{40}) \geq d_0$.

3. Verifying that the layouts in Fig. 6 are distance preserving

We now verify that the layout of Fig. 6(a) or Fig. 6(b) is distance preserving if Condition 1 is met. We set TRI_{out} , TRI_{α} , REC_{AB} , REC_{CD} , and TRI_{β} , respectively, as PAT_0 to PAT_4 . First thing to note is the primed distances between vertices and edges in the layout:

- (a) $\Delta'(o, o') = 0$ if o and o' are either neighboring edges or an edge and its endpoint (trivial case).
- (b) $\Delta'(o, o') = d_m + 1$ if $o \in \{v_{j-1,j}, e_{j-1}\}$ and $o' \in \{v_{j,j+1}, e_{j+1}\}$ for $j \in \{1, 4\}$.
- (c) $\Delta'(o, o') = d_Z$ if $o \in \{v_{j-1,j}, e_{j-1}\}$ and $o' \in \{v_{j,j+1}, e_{j+1}\}$ for $j \in \{2, 3\}$.

- (d) $\Delta'(v_{40}, e_2) = \Delta'(v_{40}, v_{23}) = \Delta'(e_0, v_{23}) = \max(d_m + 1, d_Z)$.
- (e) $\Delta'(o, o') \geq d_{\text{out}} + 1$ otherwise.

We suppose that the layout satisfies

$$d_m, d_Z < d_{\text{out}}, \quad (\text{C8a})$$

$$d_{\text{out}} - d_Z \leq d_m \leq 2d_Z, \quad (\text{C8b})$$

where the second one corresponds to item (i) of Condition 1, which will be justified later. In addition, we recall that

$$C_1 = \{\text{gz}\}, \quad C_4 = \{\text{gx}\}, \quad (\text{C9})$$

and, for each $k \geq 3$,

$$\overline{P}^{(k)}, \overline{Q}^{(k)} \in \mathcal{Z} := \{\overline{Z}_{\{\text{OAB}\}}, \overline{Z}_{\{\text{OAC}\}}, \overline{Z}_{\{\text{OAD}\}}, \overline{Z}_{\{\text{OBC}\}}, \overline{Z}_{\{\text{OBD}\}}, \overline{Z}_{\{\text{OCD}\}}, \overline{Z}_{\{\text{ABC}\}}, \overline{Z}_{\{\text{ABD}\}}, \overline{Z}_{\{\text{ACD}\}}, \overline{Z}_{\{\text{BCD}\}}, \overline{Z}_{\{\text{OABCD}\}}\}.$$

Although Condition 3 gives a vast amount of requirements, most of them are trivially true. We list the remaining (potentially) nontrivial requirements as follows:

$$i = 0 : C_{234} \not\subseteq C_0 \cup C_1 \Rightarrow \Delta'(v_{01}, e_2) = d_m + 1 \geq d_{\text{out}}, \quad (\text{C10a})$$

$$C_{134} \not\subseteq C_0 \cup C_2 \Rightarrow \Delta'(e_1, e_3) = d_Z \geq d_{\text{out}}, \quad (\text{C10b})$$

$$C_{12} \not\subseteq C_0 \cup C_{34} \Rightarrow \Delta'(e_2, v_{40}) = \max(d_m + 1, d_Z) \geq d_{\text{out}}, \quad (\text{C10c})$$

$$C_{123} \not\subseteq C_0 \cup C_4 \Rightarrow \Delta'(e_3, v_{40}) = d_m + 1 \geq d_{\text{out}}, \quad (\text{C10d})$$

$$C_{23} \not\subseteq C_0 \cup C_1 \cup C_4 \Rightarrow \Delta'(v_{01}, e_2) + \Delta'(e_3, v_{40}) = 2(d_m + 1) \geq d_{\text{out}}, \quad (\text{C10e})$$

$$C_{14} \not\subseteq C_0 \cup C_2 \cup C_3 \Rightarrow \Delta'(e_1, v_{23}, e_4) = 2d_Z \geq d_{\text{out}}, \quad (\text{C10f})$$

$$i = 1 : C_{340} \not\subseteq C_1 \cup C_2 \Rightarrow \Delta'(v_{12}, e_3) = d_Z \geq d_m, \quad (\text{C10g})$$

$$C_{240} \not\subseteq C_1 \cup C_3 \Rightarrow \Delta'(e_2, e_4) = d_Z \geq d_m, \quad (\text{C10h})$$

$$i = 2 : C_{301} \not\subseteq C_2 \cup C_4 \Rightarrow \Delta'(e_3, e_0) = d_m + 1 \geq d_Z, \quad (\text{C10i})$$

$$C_{30} \not\subseteq C_2 \cup C_4 \cup C_1 \Rightarrow \Delta'(e_3, e_0, v_{12}) = 2(d_m + 1) \geq d_Z, \quad (\text{C10j})$$

$$i = 3 : C_{012} \not\subseteq C_3 \cup C_4 \Rightarrow \Delta'(v_{34}, e_0) = d_m + 1 \geq d_Z, \quad (\text{C10k})$$

$$C_{402} \not\subseteq C_3 \cup C_1 \Rightarrow \Delta'(e_0, e_2) = d_m + 1 \geq d_Z, \quad (\text{C10l})$$

$$C_{02} \not\subseteq C_3 \cup C_4 \cup C_1 \Rightarrow \Delta'(v_{34}, e_0, e_2) = 2(d_m + 1) \geq d_Z, \quad (\text{C10m})$$

$$i = 4 : C_{013} \not\subseteq C_4 \cup C_2 \Rightarrow \Delta'(e_1, e_3) = d_Z \geq d_m, \quad (\text{C10n})$$

$$C_{012} \not\subseteq C_4 \cup C_3 \Rightarrow \Delta'(e_2, v_{34}) = d_Z \geq d_m. \quad (\text{C10o})$$

We can derive “(ii) $\overline{Z}_{\{\text{OCD}\}} \in \{\overline{P}^{(k)}\}_{k=1}^6$ ” in Condition 1 from Eq. (C10c) as follows: since its consequent is false, its presequent should be false for every distillation stage. From Eqs. (5), (C1), and (C9), we obtain

$$C_{12} \subseteq C_0 \cup C_{34} \equiv \text{gz} \notin C_2 \setminus C_0 \equiv \neg(\overline{Q}_{\text{out}}^{(k)} \neq \overline{Q}_{\text{AB}}^{(k)} = 1),$$

where “ $\neg$ ” is logical negation. The last proposition always holds except the case where $\overline{Q}^{(k)} = \overline{Z}_{\{\text{OCD}\}}$.

Likewise, we can derive “(iii) if $d_Z > 2d_m + 2$, $(\bar{Z}_{\{s\}})_{\pi/8}$ and $(\bar{Z}_{\{t\}})_{\pi/8}$ are not paired where (s, t) is any of (OAC, OBC), (OAD, OBD), (OAC, OAD), (OBC, OBD), (OCD, OABCD), (OAB, OABCD)” in Condition 1 from Eqs. (C10j) and (C10m): if $d_Z > 2d_m + 2$, the presequent of Eq. (C10j) should be false. Since

$$\mathcal{C}_{03} \subseteq \mathcal{C}_2 \cup \{gx, gz\} \equiv gy \notin \mathcal{C}_{03} \setminus \mathcal{C}_2 \equiv \neg \left(\bar{P}_{\text{out}}^{(k)} = \bar{Q}_{\text{out}}^{(k)} \wedge \bar{P}_{\text{CD}}^{(k)} = \bar{Q}_{\text{CD}}^{(k)} \wedge \bar{P}_{\text{AB}}^{(k)} \neq \bar{Q}_{\text{AB}}^{(k)} \right),$$

where “ $\wedge$ ” is logical AND, we need to avoid measuring $\bar{Z}_{\{s\}}$ and $\bar{Z}_{\{t\}}$ simultaneously for each $(s, t) \in \{(OAC, OBC), (OAD, OBD), (OCD, OABCD)\}$. Analogously, the negation of Eq. (C10m) implies

$$\neg \left(\bar{P}_{\text{out}}^{(k)} = \bar{Q}_{\text{out}}^{(k)} \wedge \bar{P}_{\text{AB}}^{(k)} = \bar{Q}_{\text{AB}}^{(k)} \wedge \bar{P}_{\text{CD}}^{(k)} \neq \bar{Q}_{\text{CD}}^{(k)} \right),$$

thus $(s, t) = (OAC, OAD), (OBC, OBD), (OAB, OABCD)$ are additionally prohibited.

All the other requirements in Eq. (C10) except Eqs. (C10c), (C10j), and (C10m) are always fulfilled regardless of the way to pair the $\pi/8$ -rotations. We prove that each of these requirements has a false presequent as follows, where “Preseq.” stands for the presequent:

- (a) Eq. (C10a): $\neg(\text{Preseq.}) \equiv gx \notin \mathcal{C}_{23} \setminus \mathcal{C}_0 \equiv \neg(\bar{P}_{\text{out}} \neq \bar{P}_{\text{AB}} = \bar{P}_{\text{CD}} = \mathbb{1})$, which holds for every $\bar{P} \in \mathcal{Z}$.
- (b) Eq. (C10b): $\mathcal{C}_{134} = \emptyset$.
- (c) Eq. (C10d): Counterpart of Eq. (C10a) where gx and $\bar{P}$ are, respectively, replaced with gz and $\bar{Q}$.
- (d) Eq. (C10e): $\neg(\text{Preseq.}) \equiv gy \notin \mathcal{C}_{23} \setminus \mathcal{C}_0 \equiv \neg(\bar{P}_{\text{ABCD}} = \bar{Q}_{\text{ABCD}} \wedge \bar{P}_{\text{out}} \neq \bar{Q}_{\text{out}})$, which holds for every pair $(\bar{P}, \bar{Q}) \in \mathcal{Z}^{\times 2}$ that satisfies $\bar{P} \neq \bar{Q}$.
- (e) Eq. (C10f): $\mathcal{C}_{14} = \emptyset$.
- (f) Eq. (C10g): $\neg(\text{Preseq.}) \equiv gx \notin \mathcal{C}_{03} \setminus \mathcal{C}_2 \equiv \neg(\bar{P}_{\text{AB}} \neq \bar{P}_{\text{out}} = \bar{P}_{\text{CD}} = \mathbb{1})$, which holds for every $\bar{P} \in \mathcal{Z}$.
- (g) Eq. (C10h): Counterpart of Eq. (C10g) where CD is replaced with AB.
- (h) Eq. (C10i): Counterpart of Eq. (C10g) where gx and $\bar{P}$ are, respectively, replaced with gz and $\bar{Q}$.
- (i) Eq. (C10k): Counterpart of Eq. (C10i) where CD is replaced with AB.
- (j) Eq. (C10l): Same as Eq. (C10h).
- (k) Eq. (C10n): Same as Eq. (C10i).
- (l) Eq. (C10o): Same as Eq. (C10k).

We lastly argue why the requirement “(i) $d_{\text{out}} - d_Z \leq d_m < 2d_Z$ ” is needed as follows: if $d_{\text{out}} - d_Z > d_m$, we additionally have a nontrivial requirement for $i = 0$:

$$\mathcal{C}_{24} \not\subseteq \mathcal{C}_0 \cup \mathcal{C}_1 \cup \mathcal{C}_3 \Rightarrow \Delta'(v_{01}, e_2, e_4) = d_m + d_Z + 1 \geq d_{\text{out}}.$$

The negation of its presequent implicates $\neg(\bar{P}_{\text{out}}^{(k)}, \bar{P}_{\text{CD}}^{(k)} \neq \mathbb{1} = \bar{P}_{\text{AB}}^{(k)})$, which prohibits the case of $\bar{P}^{(k)} = \bar{Z}_{\{\text{OCD}\}}$. This contradicts Condition 1. If $d_m > 2d_Z$, we have two additional nontrivial requirements (for $i = 1, 4$)

$$\mathcal{C}_{40} \not\subseteq \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3 \Rightarrow \Delta'(v_{01}, v_{23}, e_4) = 2d_Z \geq d_m,$$

$$\mathcal{C}_{01} \not\subseteq \mathcal{C}_4 \cup \mathcal{C}_2 \cup \mathcal{C}_3 \Rightarrow \Delta'(e_1, v_{23}, v_{34}) = 2d_Z \geq d_m.$$

However, these two contradict each other. They, respectively, give $\neg(\bar{P}_{\text{AB}}^{(k)}, \bar{P}_{\text{CD}}^{(k)} \neq \mathbb{1} = \bar{P}_{\text{out}}^{(k)})$ and $\neg(\bar{Q}_{\text{AB}}^{(k)}, \bar{Q}_{\text{CD}}^{(k)} \neq \mathbb{1} = \bar{Q}_{\text{out}}^{(k)})$, thus operators such as $\bar{Z}_{\{\text{ABC}\}}$ cannot be assigned to either $\bar{P}^{(k)}$ or $\bar{Q}^{(k)}$.

4. Justification of considering only clockwise CPGS operators

We now justify that, in the above discussion, it is sufficient to consider only CPGS operators with clockwise orders by proving the following claim:

Claim 2. For a given CPGS operator R_{cyc} in Eq. (C3) defined by $w \in \{x, y, z\}$ and $\{o_j, \tilde{o}_j\}_{j=0}^{L-1}$ where o_0 and $\tilde{o}_0$ are neighboring vertices, there exists a CPGS operator R'_{cyc} with endpoints $\{o'_j, \tilde{o}'_j\}_{j=0}^{L-1}$ that is arranged clockwise such that $\{o'_0, \tilde{o}'_0\} = \{o_0, \tilde{o}_0\}$ and $\text{wt}(R'_{\text{cyc}}) \leq \text{wt}(R_{\text{cyc}})$.

Proof. Let us consider a CPGS operator R_{cyc} in the claim. R_{cyc} can be represented by an undirected graph

$$G = (V \cup E, \{\{o_j, \tilde{o}_j\}\}_j),$$

referred to as the *schematic graph* of R_{cyc} . To avoid confusion, we refer to vertices and edges of G here as “nodes” and “links,” respectively, and “vertices” and “edges” only mean the elements of V and E , respectively. G can be embedded in the pentagon in a way that each node is placed on the corresponding vertex or the center of the corresponding edge. Each link of G indicates that its endpoints are connected by a gw-string operator in R_{cyc} . Note that G cannot have self-cycles (which correspond to trivial string operators) but may have parallel links.

Let us first assume that all the five edges do not condense gw. In this case, all the edges are isolated nodes in G and, since no two vertices are topologically connected with respect to gw, $\tilde{o}_j = o_{j+1}$ for each j . Thus, G is composed of a circuit on (at most five) vertices and some isolated nodes. The graph G embedded in the pentagon may have intersecting links, which correspond to intersecting gw-string operators. We can always redraw them not to intersect while keeping the supports of the string operators [7]. Note that the link in G between o_0 and $\tilde{o}_0$ (which are adjacent vertices) cannot intersect with another link, thus it still remains in the redrawn graph. If the redrawn graph has two or more cycles (i.e., simple circuits) including parallel links, we remove all the cycles except the one that contains a link between o_0 and $\tilde{o}_0$. This process leads to a new schematic graph G' and the corresponding CPGS operator R'_{cyc} that is not longer than R_{cyc} . Since R'_{cyc} does not have intersecting string operators, its endpoints can be relabeled to be arranged clockwise.

Let us now suppose that some edges condense gw. We define the *reduced* schematic graph $\tilde{G}$ where every set of nodes of $\tilde{G}$ that are topologically connected with respect to gw is merged into a single node. Thus, for each j , o_j , and $\tilde{o}_j$ correspond to different nodes in $\tilde{G}$, while $\tilde{o}_j$ and $\tilde{o}_{j+1}$ correspond to the same node. Therefore, $\tilde{G}$ contains a circuit on at most $[5 - (\text{number of edges condensing gw})]$ nodes and all the other nodes are isolated. By replacing intersecting links and leaving only one cycle from $\tilde{G}$ as done in the previous case where no edges condense gw, we can find a CPGS operator (that is not longer than R_{cyc}) that has a reduced schematic graph $\tilde{G}'$ with only

one cycle and no intersecting links. Note that $\tilde{G}'$ still contains a link between o_0 and $\tilde{o}_0$ because, if it intersects with another link (having an endpoint on the only edge between o_0 and $\tilde{o}_0$) in $\tilde{G}$, it means that o_0 and $\tilde{o}_0$ are topologically connected with respect to gw. By relabeling nodes appropriately (while keeping $\{o'_0, \tilde{o}'_0\} = \{o_0, \tilde{o}_0\}$), we can ensure $o'_j < \tilde{o}'_j < \tilde{o}'_{j+1}$ and $o'_j < o'_{j+1} < \tilde{o}'_{j+1}$, but the order between $\tilde{o}'_j$ and o'_{j+1} remains ambiguous unlike the previous case as they correspond to the same node in $\tilde{G}'$. If they have a wrong order ($o'_j < o'_{j+1} < \tilde{o}'_j < \tilde{o}'_{j+1}$), the links $(o'_j, \tilde{o}'_j)$ and $(o'_{j+1}, \tilde{o}'_{j+1})$ intersect, thus we can make them nonintersect by the same technique as above, which leads to the links $(o'_j, \tilde{o}'_{j+1})$ and $(\tilde{o}'_j, o'_{j+1})$. We then remove the link $(\tilde{o}'_j, o'_{j+1})$ that corresponds to a trivial string operator, which strictly shorten the CPGS operator. By repeating the above process until the schematic graph does not have intersecting links, we finally obtain the desired CPGS operator R'_{cyc} . ■

APPENDIX D: SIMULATIONS OF THE GROWING OPERATION

In this Appendix, we elaborate on the simulation method for the growing operation described in Sec. IV B, where errors are decoded using the concatenated MWPM decoder with postselection. We also present detailed numerical results that are not shown in the main text.

1. Methods

We simulate a circuit implementing the growing operation shown in Fig. 9(a), followed by d_m rounds of syndrome extraction. Each Bell state is prepared by initializing two qubits to $|+\rangle$ and $|0\rangle$, respectively, and applying a CNOT gate between them. Syndrome extraction is performed using the circuit in Fig. 20; see Appendix F for further details.

To calculate the $\bar{Z}$ ($\bar{X}$) failure rate, we consider preparing the $|\bar{0}\rangle$ ($|\bar{+}\rangle$) state in the initial patch with distance d_{cult} and measuring the final patch with distance d_m in the $\bar{Z}$ ($\bar{X}$) basis. These logical preparation and measurement are hypothetical components required only to define a logical observable of the circuit (deterministic when there are no errors) for using *Stim*. They are not presented in actual implementations, where the process is preceded by cultivation and followed by idling or lattice surgery. Since we aim to quantify the effect of errors during the growing operation in isolation from errors outside of it, the logical preparation and measurement are assumed to be perfect.

To calculate the logical gap using the concatenated MWPM decoder, we slightly modify the decoder implemented in Ref. [27] by treating the observable as a detector and running the decoder twice with different detector values. The logical gap is then determined from the difference

in log-likelihood weights between these two predictions. A decoding outcome is accepted only when the logical gap is greater than a predetermined threshold c_{gap} . This method is implemented in the latest version of the *color-code-stim* module [65].

We compute the logical error rate as $p_{\text{log}} = p_{\text{fail}}^Z + p_{\text{fail}}^X$, where p_{fail}^P is the obtained $\bar{P}$ failure rate. Note that this is a conservative estimation of the actual logical error rate, as $\bar{Y}$ errors are overcounted. The acceptance rate is computed as $p_{\text{acc}} = p_{\text{acc}}^Z p_{\text{acc}}^X$, where p_{acc}^Z and p_{acc}^X are, respectively, the acceptance rates of the two settings, assuming that decodings for Z and X failures are aborted independently.

2. Results

Now we present the results of our numerical simulations on the growing operation with varying p , d_{cult} , d_{m} , and c_{gap} . In Fig. 17, we plot p_{log} and p_{acc} against c_{gap} for $p \in \{5 \times 10^{-4}, 10^{-3}\}$, $d_{\text{cult}} \in \{3, 5\}$, and varying $d_{\text{m}} > d_{\text{cult}}$. The achievable minimum logical error rate is

$$\begin{cases} 1.0 \times 10^{-6} & \text{for } p = 5 \times 10^{-4}, d_{\text{cult}} = 3, \\ 6.9 \times 10^{-9} & \text{for } p = 5 \times 10^{-4}, d_{\text{cult}} = 5, \\ 5.0 \times 10^{-6} & \text{for } p = 10^{-3}, d_{\text{cult}} = 3, \\ 1.7 \times 10^{-7} & \text{for } p = 10^{-3}, d_{\text{cult}} = 5. \end{cases}$$

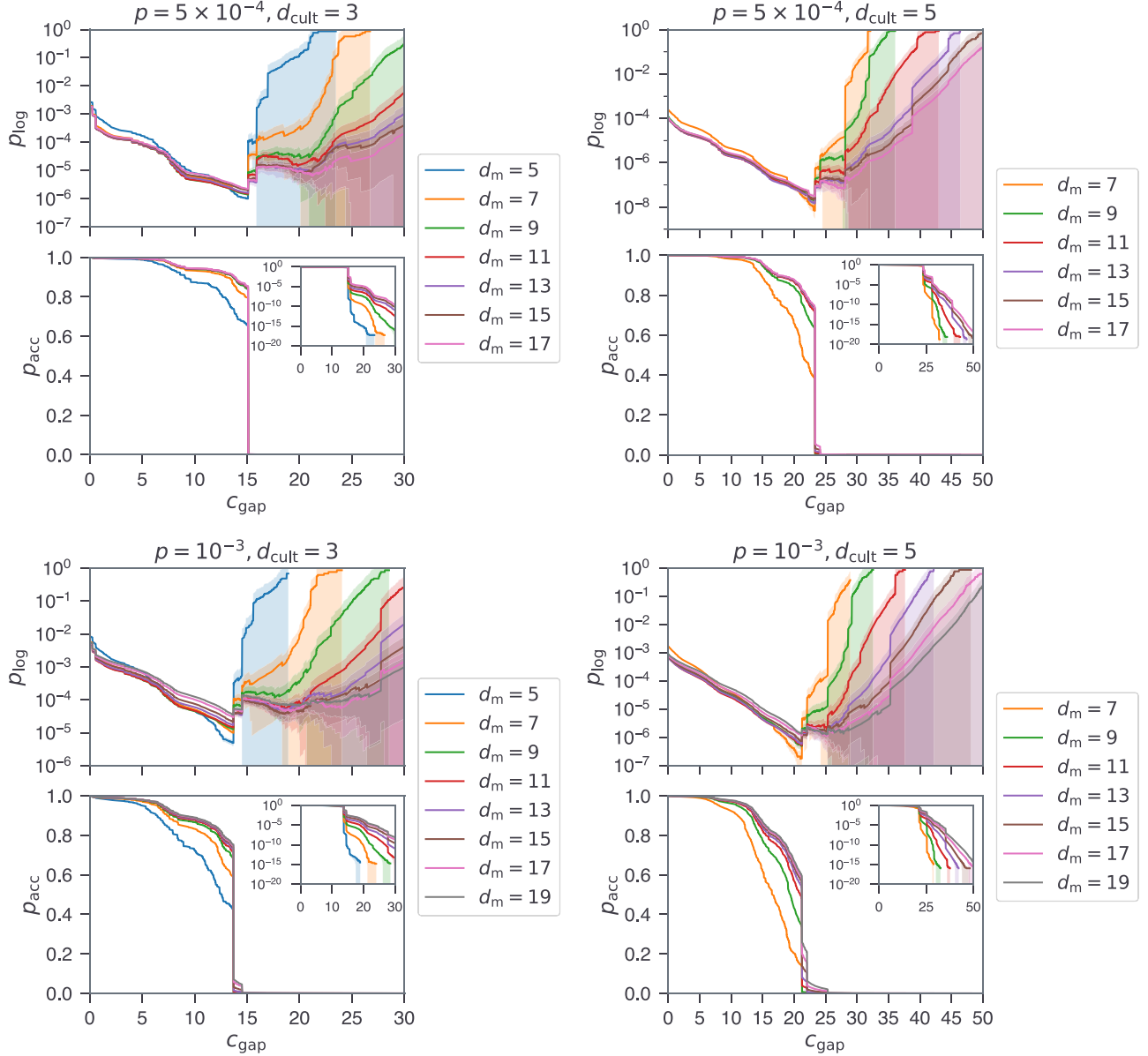


FIG. 17. Decoder simulation results for the growing operation with postselection. The logical error rate p_{log} and the acceptance rate p_{acc} are plotted against the logical gap threshold c_{gap} for physical noise strength $p \in \{5 \times 10^{-4}, 10^{-3}\}$ and pregrowing code distance $d_{\text{cult}} \in \{3, 5\}$, with varying postgrowing code distance $d_{\text{m}} > d_{\text{cult}}$. Insets in the acceptance rate plots display their logarithmic-scale versions. The shaded regions represents 99% confidence intervals.

Note that the logical error and acceptance rates show sudden jumps at

$$c_{\text{gap}} \approx \begin{cases} 15.0 & \text{for } p = 5 \times 10^{-4}, d_{\text{cult}} = 3, \\ 23.3 & \text{for } p = 5 \times 10^{-4}, d_{\text{cult}} = 5, \\ 13.7 & \text{for } p = 10^{-3}, d_{\text{cult}} = 3, \\ 21.0 & \text{for } p = 10^{-3}, d_{\text{cult}} = 5. \end{cases} \quad (\text{D1})$$

We conjecture that this phenomenon arises because the concatenated MWPM decoder may fail to find a minimum-weight solution. That is, the jump locations would correspond to the maximum possible logical gap values under the hypergraph MWPM decoder, which can always identify a minimum-weight solution. However, since the concatenated MWPM decoder may produce a higher-weight correction, the logical gap can occasionally exceed these maximum values with low probability. We leave the verification of this conjecture for future work.

Additionally, in Fig. 18, we illustrate the dependence of p_{log} on d_m with varying c_{gap} , which is not clearly visible in Fig. 17. Due to computational constraints, our simulations are limited to $d_m \leq 17$ when $p = 5 \times 10^{-4}$ and $d_m \leq 19$ when $p = 10^{-3}$. Beyond these limits, we extrapolate using

linear regression (on a logarithmic scale) based on the last three data points. Note that we do not extrapolate p_{acc} ; instead, we use its values at $d_m = 17$ or 19 for larger d_m 's. This approach does not overestimate performance, as p_{acc} increases with d_m , as shown in Fig. 17.

APPENDIX E: METHOD FOR NUMERICAL ANALYSIS

In this Appendix, we elaborate on the method for numerical analysis, which is outlined in Sec. V A

1. Logical error rates of patches

We assume that the logical error rate of multiple rounds of syndrome extraction is proportional to the number of rounds T . For a given noise strength p , the per-round $\bar{X}/\bar{Y}/\bar{Z}$ error rate of a triangular logical patch with distance d is denoted as $p_{\text{tri}}^{X/Y/Z}(p, d)$. For a rectangular logical patch with code distances d_X and d_Z in Fig. 2(d), its per-round $\bar{X}_1$ error rate is denoted as $p_{\text{rec}}^{X_1}(p, d_X, d_Z)$ and functions $p_{\text{rec}}^{Z_1}$, $p_{\text{rec}}^{X_2}$, and $p_{\text{rec}}^{Z_2}$ are defined analogously. These errors are caused by Pauli- X or Z error strings connecting opposite boundaries. Pauli- Y error strings connecting opposite

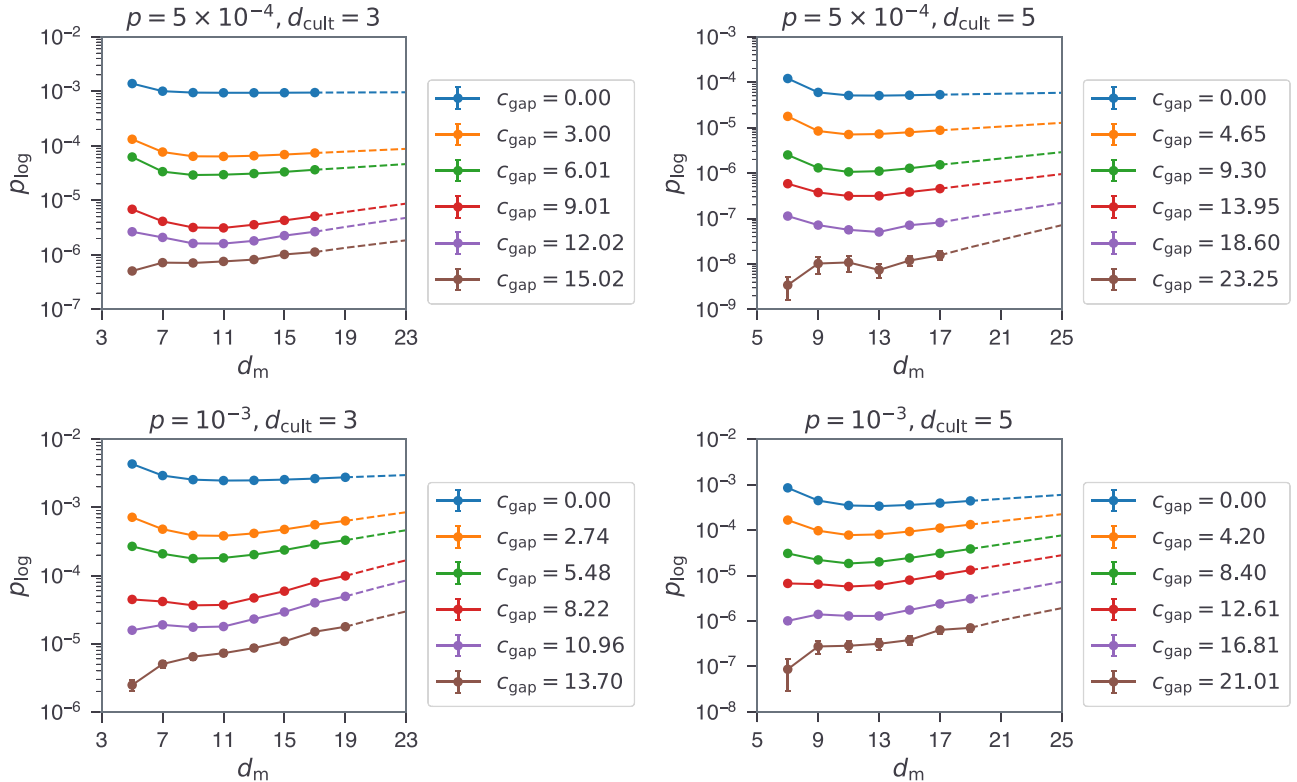


FIG. 18. Dependency of the logical error rate p_{log} on d_m for the growing operation with postselection, at $p \in \{5 \times 10^{-4}, 10^{-3}\}$ and $d_{\text{cult}} \in \{3, 5\}$, with varying c_{gap} . The values of c_{gap} are chosen evenly spaced between zero and the points of sudden jumps given in Eq. (D1). The error bars indicate 99% confidence intervals, while data points without error bars have confidence intervals smaller than the marker size. Extrapolations, shown as dashed lines, are obtained via linear fitting (on a logarithmic scale) using the last three data points.

boundaries incur correlations of $\bar{X}$ or $\bar{Z}$ errors between the two logical qubits, with error rates denoted as $p_{\text{rec}}^{X_1 X_2}$ and $p_{\text{rec}}^{Z_1 Z_2}$. In addition, “diagonal” blue string operators connecting the top left corner and the bottom right corner in Fig. 2(d) have weights of at least $\max(d_X, d_Z)$ and make correlated logical errors $\bar{X}_1 \bar{Z}_2$ (for bx), $\bar{Z}_1 \bar{X}_2$ (for bz), or $\bar{Y}_1 \bar{Y}_2$ (for by). We ignore such correlations in our error analysis, which can be justified because (i) diagonal error strings are significantly rarer than other error strings for sufficiently large code distances [66] and (ii) the resulting correlated logical errors are not specifically more detrimental than other logical errors (i.e., they are detectable by the final $\bar{X}$ measurements of the MSD circuit alike uncorrelated $\bar{Z}$ errors). In addition, we denote the logical error rates of the growing operation and the subsequent d_m rounds of syndrome extraction (which are jointly decoded with postselection) for the cultivation-MSD scheme as $p_{\text{grow}}^{X/Y/Z}(p, d_{\text{cult}}, d_m, c_{\text{gap}})$, where c_{gap} is the logical gap threshold for postselection (see Sec. IV B and Appendix D).

Through numerical simulations, we only can compute logical failure rates of observables, not individual logical error rates. (We say, e.g., $\bar{X}$ fails if a $\bar{Y}$ or $\bar{Z}$ error occurs.) To extract individual logical error rates from the logical failure rates of a triangular patch, we assume that $\bar{X}$ and $\bar{Z}$ errors occur with equal probability [27], while the probability of a $\bar{Y}$ error is r_y times this probability, where r_y is a given non-negative number. Namely, we assume

$$p_{\text{tri}}^X = p_{\text{tri}}^Z = \frac{p_{\text{tri}}^{\text{fail}}}{2(1 + r_y)}, \quad p_{\text{tri}}^Y = r_y p_{\text{tri}}^X,$$

where $p_{\text{tri}}^{\text{fail}}$ is the summation of the $\bar{X}$ and $\bar{Z}$ failure rates. Likewise, we suppose that

$$\begin{aligned} p_{\text{rec}}^{X_1} &= p_{\text{rec}}^{X_2} = \frac{p_{\text{rec}}^{Z, \text{fail}}}{2(1 + r_y)}, & p_{\text{rec}}^{X_1 X_2} &= r_y p_{\text{rec}}^{X_1}, \\ p_{\text{rec}}^{Z_1} &= p_{\text{rec}}^{Z_2} = \frac{p_{\text{rec}}^{X, \text{fail}}}{2(1 + r_y)}, & p_{\text{rec}}^{Z_1 Z_2} &= r_y p_{\text{rec}}^{Z_1}, \\ p_{\text{grow}}^X &= p_{\text{grow}}^Z = \frac{p_{\text{grow}}^{\text{fail}}}{2(1 + r_y)}, & p_{\text{grow}}^Y &= r_y p_{\text{grow}}^X, \end{aligned}$$

where $p_{\text{rec}}^{Z(X), \text{fail}}$ is the summation of the $\bar{Z}_1$ ($\bar{X}_1$) and $\bar{Z}_2$ ($\bar{X}_2$) failure rates of a rectangular patch, and $p_{\text{grow}}^{\text{fail}}$ is the summation of the $\bar{X}$ and $\bar{Z}$ failure rates of the growing operation.

We further assume that the output state of cultivation has probabilistic Pauli noise and define $p_{\text{cult}}^{X/Y/Z}(p, d_m)$ as its $\bar{X}/\bar{Y}/\bar{Z}$ error rate, satisfying

$$\begin{aligned} p_{\text{cult}}^X &= p_{\text{cult}}^Z = \frac{q_{\text{cult}}}{2 + r_y}, \\ p_{\text{cult}}^Y &= r_y p_{\text{cult}}^X, \end{aligned}$$

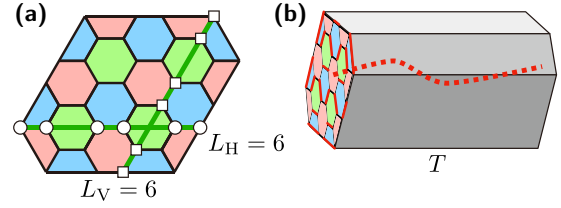


FIG. 19. (a) Patch for analyzing timelike error strings through a stability experiment, which encodes no logical qubits. The parameters L_H and L_V determine the size of the patch, and $L_H = L_V = 6$ in this example. (b) Spacetime picture of the stability experiment, which is composed of T rounds of syndrome extraction surrounded by two red temporal boundaries. The red dotted line represents an example of a timelike error string.

where q_{cult} is the output infidelity of cultivation.

In addition, we need to deal with timelike error strings that connect two red temporal boundaries of the ancillary region, such as T time consecutive Z -type check measurement errors on a single red face. For this, we consider a stability experiment [53] using a patch in the shape of Fig. 19(a), which is characterized by parameters L_H and L_V (i.e., the weights of the shortest green string operators connecting the pairs of opposite parallel boundaries horizontally and vertically). Note that the patch does not encode logical qubits as it only has green boundaries. In the experiment, the patch undergoes T rounds of syndrome extraction of the patch surrounded by two red temporal boundaries, as shown in Fig. 19(b). After decoding, we check whether a timelike error occurs from the product of the outcomes of the Z - or X -type checks on all the red and blue faces at any round, which should be $+1$ if there are no errors. We denote the timelike error rates as $p_{\text{TL}}^Z(p, L_H, L_V, T)$ and $p_{\text{TL}}^X(p, L_H, L_V, T)$, which, respectively, correspond to corrupted Z - and X -type checks. In our analysis, we only consider the cases of $L_H = L_V = T$ for simulations and assume that

$$p_{\text{TL}}^{Z/X}(p, L_H, L_V, T) = \frac{L_H L_V}{T^2} p_{\text{TL}}^{Z/X}(p, T, T, T).$$

2. Computation of the output infidelity and success probability of MSD

Each type of logical Pauli error that occurs during MSD can be mapped to a noise channel acted on the output and validation qubits between the last $\pi/8$ -rotation and the final $\bar{X}$ measurements of the validation qubits. The noise channel is of the form

$$\Lambda_{\bar{U}, p_{\text{err}}} : \bar{\rho} \mapsto (1 - p_{\text{err}}) \bar{\rho} + p_{\text{err}} \bar{U} \bar{\rho} \bar{U}^\dagger,$$

where $\bar{\rho}$ is the logical state of the output and validation qubits, p_{err} is the logical error rate, and $\bar{U}$ is a product of $\pi/2$ - or $(\pm\pi/4)$ -rotations in bases consisting

of $\bar{Z}$'s only. In Appendix H, we present rules to determine noise channels from various possible error sources in the logical patches and the ancillary region (including timelike errors), covering both the single-level and cultivation-MSD schemes. Following these rules, each error rate (p_{err}) of the noise channel is expressed in terms of the logical error rate functions (defined in Appendix E 1) and additional variables including T_{intv} and T_{idle} for the cultivation-MSD scheme.

Denoting the set of noise channels obtained by the rules as $\{\Lambda_{\bar{U}_i, p_i}\}_{i=1}^m$, the unnormalized output state after the final measurements is

$$\bar{\rho}_{\text{out}} = (\mathbb{1}_{\text{out}} \otimes \langle +++++ |_{\text{ABCD}}) \Lambda_{\text{noise}} (|\bar{\psi}_{\text{init}}\rangle \langle \bar{\psi}_{\text{init}}|),$$

where

$$\begin{aligned} \Lambda_{\text{noise}} &:= \Lambda_{\bar{U}_1, p_1} \circ \cdots \circ \Lambda_{\bar{U}_m, p_m}, \\ |\bar{\psi}_{\text{init}}\rangle &:= |\bar{A}_-\rangle_{\text{out}} \otimes |++++\rangle_{\text{ABCD}}, \\ |\bar{A}_-\rangle &:= \frac{1}{\sqrt{2}} (|\bar{0}\rangle + e^{-i\pi/4} |\bar{1}\rangle). \end{aligned}$$

The success probability q_{succ} of the scheme and the output infidelity q_{dist} are then, respectively, given as

$$q_{\text{succ}} = \text{Tr}(\bar{\rho}_{\text{out}}), \quad q_{\text{dist}} = 1 - \frac{1}{q_{\text{succ}}} \langle \bar{A}_- | \bar{\rho}_{\text{out}} | \bar{A}_- \rangle.$$

In Appendix I, we present analytic expressions of q_{succ} and q_{dist} as functions of the physical error rate p , the code distances, and the logical error rates of several patches.

To calculate q_{succ} and q_{dist} numerically, we need to explicitly specify the logical error rate functions, which vary depending on the decoder used to predict errors from syndrome outcomes. We employ the concatenated MWPM decoder, proposed in Ref. [27], which is suitable for our analysis since it is predicted to outperform other up-to-date matching-based decoders in terms of its logical failure rate under circuit-level noise, especially when p is sufficiently small ($\leq 10^{-3}$). Note that we use the same decoder

for postselection during the growing operation as well, thus it remains consistent. In Appendix G, we elaborate on our decoder simulations for memory and stability experiments, presenting the methods and the obtained results.

APPENDIX F: SELECTION OF THE ENTANGLING GATE SCHEDULE

One important factor to consider when running circuit-level error simulations is the scheduling of entangling gates in the syndrome-extraction circuit. Since the numerical results in the original concatenated MWPM decoder paper [27] are based on the schedule verified and optimized only for triangular patches, we need to adjust it to work well also with rectangular patches. More specifically, we identify 24 valid schedules with seven time steps that can properly extract check eigenvalues and, for each of them, simulate six rounds of syndrome extraction of a rectangular logical patch with $d_X = d_Z = 6$, obtaining the failure rates of four logical Pauli operators $\bar{Z}_1$, $\bar{Z}_2$, $\bar{X}_1$, and $\bar{X}_2$ by using the concatenated MWPM decoder. (Here, we say that a logical Pauli operator fails if a logical error anti-commuting with it occurs.) We then select the schedule that minimizes the worst-case failure rate (i.e., largest one among the four).

To elaborate on this process, it is as follows: checks on a hexagonal face are measured by using the circuits in Fig. 20(a), where each syndrome qubit for a Z -type (X -type) check is prepared to $|+\rangle$, undergo CZ (CNOT) gates with data qubits, and then are measured in the X basis. Six possible types of weight-4 faces that can be placed along boundaries are presented in Fig. 20(b) and the corresponding checks are measured in a similar way to be consistent. Note that the weight-4 face at the top left corner of Fig. 2(c) is regarded to have the same form as the second face (with b, c, d, e, h, i, j, k) of Fig. 20(b). The entangling gate schedule is specified by 12 positive integers $\mathcal{A} = [a, b, c, d, e, f; g, h, i, j, k, l]$, which contains all the integers from 1 to $\max \mathcal{A}$ (called the length of the schedule). Each integer represents the time step at which the corresponding entangling gate in Fig. 20(a) is applied.

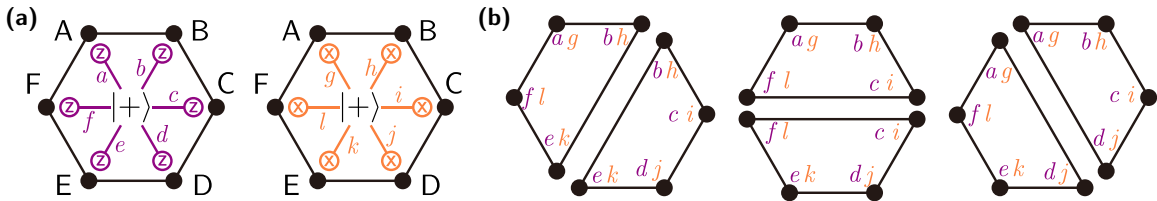


FIG. 20. (a) Syndrome extraction circuits for measuring Z -type (left) and X -type (right) checks on a hexagonal face. In each circuit, the syndrome qubit at the center of the face is prepared to $|+\rangle$, undergoes multiple controlled- Z (CZ) or controlled- X (CNOT) gates with data qubits, and is finally measured in the X basis. Twelve variables $a, b, \dots, l$ are positive integers specifying the time steps that the corresponding CNOT gates are applied. (b) Six types of weight-4 faces that can be placed along boundaries with the corresponding time-step variables of the CNOT gates, colored in purple (orange) for the measurements of Z -type (X -type) checks. The syndrome extraction circuits are in the same form as (a).

The first (last) half of the schedule is referred to as its Z-type (X -type) part.

The following conditions need to be satisfied for the schedule to be valid [22]:

- (1) Each qubit can be involved in at most one CNOT gate at each time step, meaning that (a, b, c, d, e, f) , (g, h, i, j, k, l) , (a, c, e, g, i, k) , and (b, d, f, h, j, l) are all tuples of distinct numbers.
- (2) X - and Z -type check measurements should not interfere with each other. For example, the stabilizer X of a Z -type syndrome qubit is propagated to Z operators on the surrounding data qubits. We should prevent a situation that an odd number among these operators are again propagated to an X -type syndrome qubit. Therefore, all the following 13 numbers should be positive:

$$\left\{ \begin{array}{l} (a-g)(b-h)(c-i)(d-j)(e-k)(f-l), \\ (a-g)(b-h)(f-l)(e-k), \\ (a-g)(b-h)(c-i)(f-l), \\ (a-g)(b-h)(c-i)(d-j), \\ (c-i)(d-j)(e-k)(f-l), \\ (a-g)(d-j)(e-k)(f-l), \\ (b-h)(c-i)(d-j)(e-k), \\ (a-k)(b-j), \quad (b-l)(c-k), \quad (c-g)(d-l), \\ (d-h)(e-g), \quad (e-i)(f-h), \quad (f-j)(a-i). \end{array} \right. \quad (\text{F1})$$

There are $4 \times 3 \times 2 = 24$ length-7 schedules satisfying the above conditions, which are

$$\begin{aligned} & [2, 1, 4, 5, 6, 3; 3, 2, 5, 6, 7, 4], \\ & [2, 3, 6, 5, 4, 1; 3, 4, 7, 6, 5, 2], \\ & [1, 2, 3, 6, 5, 4; 2, 3, 4, 7, 6, 5], \\ & [1, 4, 5, 6, 3, 2; 2, 5, 6, 7, 4, 3] \end{aligned}$$

and their variations considering 120° rotation (i.e., $\mathcal{A} \rightarrow [c, d, e, f, a, b; i, j, k, l, g, h]$) and the exchange of the X - and Z -type parts (i.e., $\mathcal{A} \rightarrow [g, h, i, j, k, l; a, b, c, d, e, f]$). To select one of these 24 schedules, we consider six rounds of syndrome extraction of a rectangular logical patch with distances $d_X = d_Z = 6$ and compute the failure rates of four logical Pauli operators $\bar{Z}_1, \bar{Z}_2, \bar{X}_1$, and $\bar{X}_2$ defined in Fig. 2(d) via Monte Carlo simulations. (Here, we say $\bar{Z}_1$ fails if an $\bar{X}_1$ or $\bar{Y}_1$ error occurs, and similarly for the other operators.) We then select the schedule that minimizes the largest one among these four failure rates. The computed logical failure rates are presented in Fig. 21 for

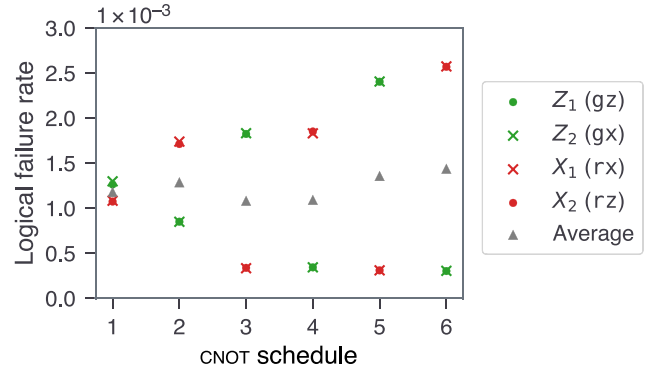


FIG. 21. Failure rates of the four logical Pauli operators $\bar{Z}_1, \bar{Z}_2, \bar{X}_1$, and $\bar{X}_2$ for six different entangling gate schedules in Eq. (F2), considering six rounds of syndrome extraction of a rectangular logical patch with distances $d_X = d_Z = 6$ under the circuit-level noise model of strength $p = 10^{-3}$, decoded via the concatenated MWPM decoder [27]. Note that $\bar{Z}_1, \bar{Z}_2, \bar{X}_1$, and $\bar{X}_2$ are, respectively, gz-, gx-, rx-, and rz-string operators. The averages of the four failure rates are also presented. We select schedule 1 that minimizes the largest logical failure rate among the four.

six schedules, which are, respectively,

$$\left\{ \begin{array}{l} 1. [3, 6, 5, 4, 1, 2; 4, 7, 6, 5, 2, 3], \\ 2. [4, 5, 6, 3, 2, 1; 5, 6, 7, 4, 3, 2], \\ 3. [5, 4, 1, 2, 3, 6; 6, 5, 2, 3, 4, 7], \\ 4. [2, 1, 4, 5, 6, 3; 3, 2, 5, 6, 7, 4], \\ 5. [1, 4, 5, 6, 3, 2; 2, 5, 6, 7, 4, 3], \\ 6. [1, 2, 3, 6, 5, 4; 2, 3, 4, 7, 6, 5]. \end{array} \right. \quad (\text{F2})$$

Here we only consider six out of the 24 schedules since the rectangular patch is symmetric under a 180° rotation (i.e., $\mathcal{A} \rightarrow [d, e, f, a, b, c; j, k, l, g, h, i]$), and exchanging the X - and Z -type parts simply swaps their corresponding failure rates ($\bar{X}_1 \leftrightarrow \bar{X}_2$ and $\bar{Z}_1 \leftrightarrow \bar{Z}_2$). Hence, we select the first one in Eq. (F2) as the default entangling gate schedule for our scheme, shown in Figs. 22(a) and 22(b).

By using one of the schedules in Eq. (F2), a single round of syndrome extraction can be done in eight time steps, where syndrome qubits for Z -type (X -type) checks are measured at time step 7 (8) and initialized at time step 8 (1). It is worth noting that two-body check measurements for lattice surgery in Figs. 7 and 8 also can be done during these eight time steps, as exemplified in Fig. 22(c). This is because, for each pair of neighboring qubits, one of them is involved in entangling gates during time steps 1 to 6, while the other is involved during time steps 2 to 7, implying that two extra entangling gates for a two-body check measurement can be executed at time steps 7 and 8. The additional syndrome qubit is initialized at time step 6 and measured at time step 1 of the next round.

In addition, we need to consider Y -type checks and mixed-Pauli checks as well, which may exist in domain

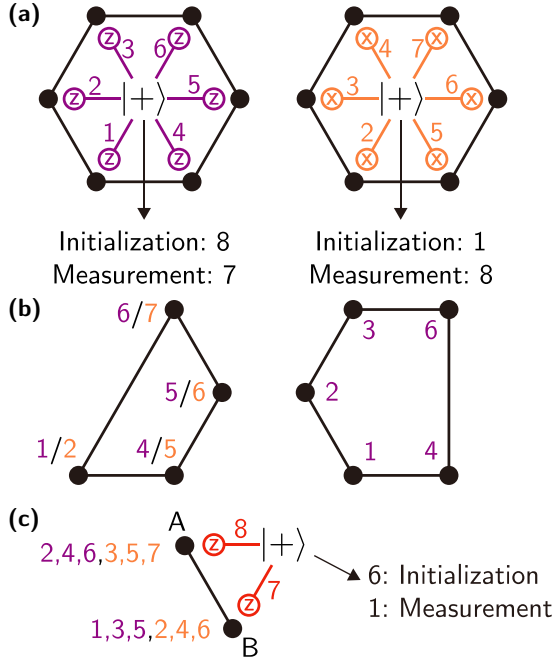


FIG. 22. (a) Selected entangling gate schedule for measuring a Z-type check (left) and an X-type check (right). Syndrome qubits for Z-type (X-type) checks are initialized at time step 8 (1) and measured at time step 7 (8). (b) Examples of schedules for weight-4 and 5 checks. Note that only the schedule for the Z-type check is presented on the weight-5 check since the face can support only one check. (c) Schedule for measuring a weight-2 check $Z_A \otimes Z_B$ in a domain wall. An additional syndrome qubit is prepared at time step 6, undergo CZ gates with qubits B and A at time steps 7 and 8, respectively, and is measured at time step 1.

walls. Such checks can be measured by replacing entangling gates with corresponding controlled-Pauli gates while keeping the schedule. For example, let us consider a Pauli-permuting domain wall ($x \leftrightarrow z$) crossing a face, with one side having qubits A, B, C, and F and the other side having qubits D and E in Fig. 20(a). In this case, we can replace the Z-type check S_f^Z with $S_1 := Z_A Z_B Z_C X_D X_E Z_F$ and apply CNOT gates instead of CZ gates on qubits D and E when measuring S_1 . The X-type check S_f^X is replaced with $S_2 := X_A X_B X_C Z_D Z_E X_F$ and measured analogously. Note that the schedule is still valid even if S_f^Z is replaced with S_2 and S_f^X is replaced with S_1 . We can verify that the second condition for the schedule to be valid (i.e., check measurements should not interfere each other) holds even if entangling gates are replaced like this. For example, in the above example, there is a risk that S_1 interferes with the Z-type check on the adjacent face containing qubits D and E. However, it does not happen since $(a - e)(b - d) > 0$ for all the 24 valid length-7 schedules, although it is not one of Eq. (F1). In general, six numbers $(a - e)(b - d)$, $(b - f)(c - e)$, $(c - a)(d - f)$, $(g - k)(h - j)$, $(h - l)(i - k)$, and $(i - g)(j - l)$ are all positive, thus we do not need to worry about this problem.

It is worth noting that the selected schedule is the same as the one used in Ref. [27] rotated by 60° , implying that the numerical results in Ref. [27] can be directly applied to inverted (base-up) triangular patches (including TRI_{out} and some auxiliary patches) in our MSD layouts. Our layouts also contain upright (base-down) triangular patches, but we assume for simplicity that their logical error rates are the same as those of inverted triangular patches having the same code distances. As numerical evidence for justifying this assumption, we simulate seven rounds of syndrome extraction of a distance-7 upright triangular patch at $p = 10^{-3}$ based on the selected schedule. The obtained $\bar{Z}$ and $\bar{X}$ failure rates are $(7.24 \pm 0.04) \times 10^{-4}$ and $(7.22 \pm 0.04) \times 10^{-4}$ (99% confidence intervals), respectively, which differ from the failure rates $(7.19 \pm 0.04) \times 10^{-4}$ obtained in Ref. [27] only by approximately 0.7%.

APPENDIX G: DECODER SIMULATIONS FOR MEMORY AND STABILITY EXPERIMENTS

In this Appendix, we provide details of our decoder simulations for memory and stability experiments using the concatenated MWPM decoder under circuit-level noise. We outline the simulation methods, present the corresponding results, and discuss our reasoning behind selecting the ansatz given in Eq. (11).

1. Methods for simulating memory and stability experiments

For a memory experiment with a triangular or rectangular patch, we consider $4d$ rounds of syndrome extraction (following the schedule in Fig. 22) for code distance d (by setting $d_X = d_Z = d$ for rectangular patches). This process is preceded and followed by Pauli-Z temporal boundaries; namely, all the physical qubits are initially prepared as $|0\rangle$ and finally measured in the Z basis. For a triangular patch, this means that the logical qubit encoded in the patch is initially prepared as $|\bar{0}\rangle$ and finally measured in the $\bar{Z}$ basis. For a rectangular patch, the two logical qubits are initially prepared as $|\bar{0}\rangle \otimes |\bar{+}\rangle$ and finally measured in the basis of $\bar{Z} \otimes \bar{X}$. Each round of syndrome extraction is conducted according to the schedule in Fig. 22(a). Note that this setting can only detect $\bar{Z}$ failures (for a triangular patch) or $\bar{Z}_1$ and $\bar{X}_2$ failures (for a rectangular patch). For complete analysis covering other types of failures, we consider the same setting again but with a schedule modified such that its X part and Z part are swapped.

For a stability experiment, we consider the patch in Fig. 19(a) with $L_H = L_V = T$, which encodes no logical qubits, undergoing T rounds of syndrome extraction with the same schedule. This process is preceded and followed by red temporal boundaries, namely, all the red edges are initially prepared as $|0\rangle |0\rangle + |1\rangle |1\rangle$ and measured in the Bell basis. If there are no errors, the product of red and blue checks with the same Pauli type should be always

+1 whenever they are measured. This fact can be used for determining whether a nontrivial timelike error string exists after decoding.

For simulating the above setting, we employ the *color-code-stim* module [27] with a slight modification to handle memory experiments with rectangular checks and stability experiments. The module automatically generates a noisy color-code circuit for given color-code lattice, entangling gate schedule, and bit-flip or circuit-level noise model, with detectors (i.e., products of measurement outcomes that are deterministic when there are no errors and can be used for decoding) annotated. It also provides features to simulate the circuit using the *Stim* library [67] and decode errors via the concatenated MWPM decoder [27], where each MWPM subroutine is implemented by the *PyMatching* library [68].

2. Results

For a triangular patch, we denote the per-round failure rate of the $\bar{X}$ ($\bar{Z}$) observable as $p_{\text{tri}}^{X(Z),\text{fail}} := p_{\text{tri}}^{Z(X)} + p_{\text{tri}}^Y$

and define $p_{\text{tri}}^{\text{fail}} := p_{\text{tri}}^{X,\text{fail}} + p_{\text{tri}}^{Z,\text{fail}}$. Similarly, for a rectangular patch, we denote the per-round logical failure rates as $p_{\text{rec}}^{Z_1(2),\text{fail}} := p_{\text{rec}}^{X_1(2)} + p_{\text{rec}}^{X_1X_2}$ and $p_{\text{rec}}^{X_1(2),\text{fail}} := p_{\text{rec}}^{Z_1(2)} + p_{\text{rec}}^{Z_1Z_2}$, and define $p_{\text{rec}}^{Z,\text{fail}} := p_{\text{rec}}^{Z_1,\text{fail}} + p_{\text{rec}}^{Z_2,\text{fail}}$ and $p_{\text{rec}}^{X,\text{fail}} := p_{\text{rec}}^{X_1,\text{fail}} + p_{\text{rec}}^{X_2,\text{fail}}$.

In Fig. 23, logical failure rates obtained from the memory and stability experiments are plotted for various values of p and d (or T for the stability experiments). Figures 23(a) and 23(b) are for triangular and rectangular patches, respectively, which show $p_{\text{tri}}^{\text{fail}}/T$ for triangular patches and $p_{\text{rec}}^{Z,\text{fail}}/T$ and $p_{\text{rec}}^{X,\text{fail}}/T$ for rectangular patches, where $T = 4d$. Figure 23(c) is for stability experiments, presenting the per-area failure rate $p_{\text{TL}}/T^2 = (p_{\text{TL}}^X + p_{\text{TL}}^Z)/T^2$, where $p_{\text{TL}}^{X/Z}$ is defined in Appendix E 1.

3. Ansatz selection

We now select the ansatz by which these per-round and per-area failure rates can be approximated, which is a function of p and d (denoting $d := T$ for stability experiments).

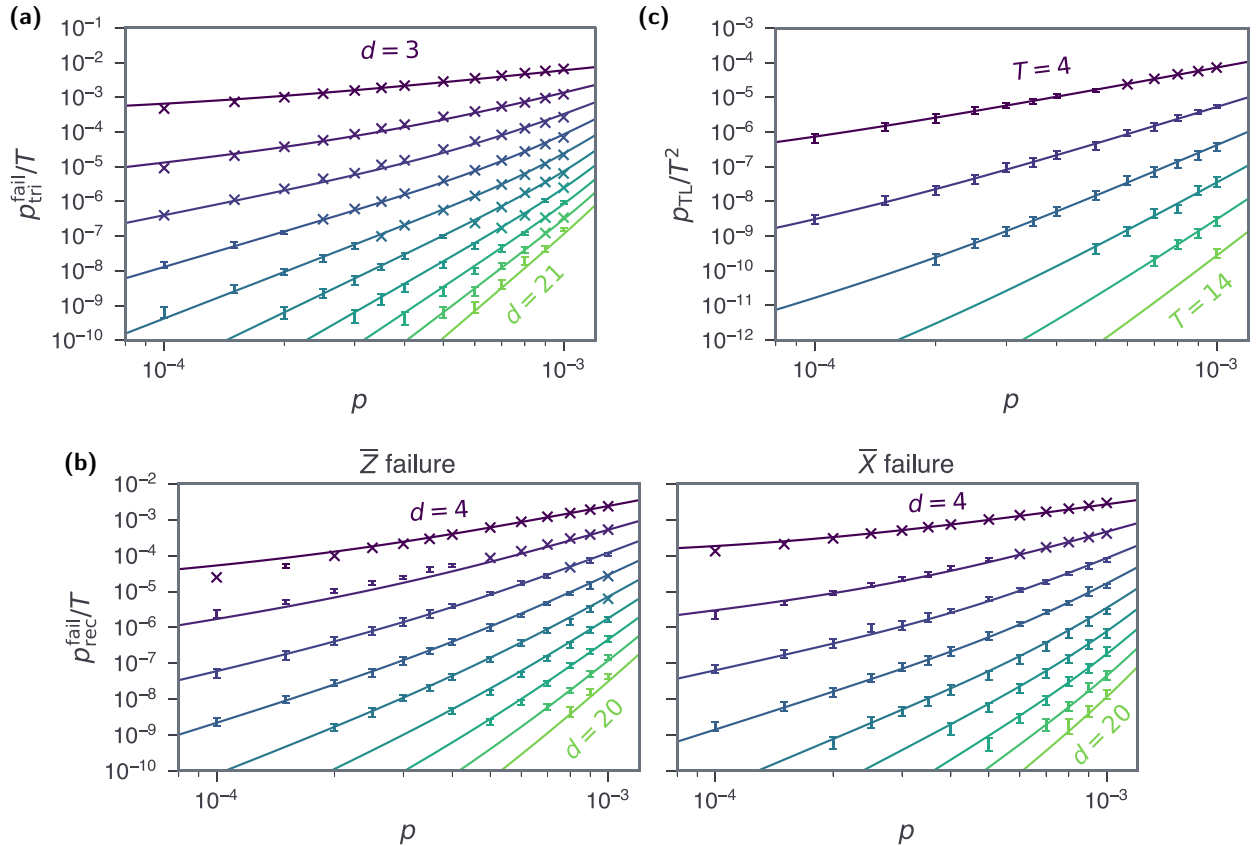


FIG. 23. Per-round or per-area logical failure rates plotted against the physical error rate p at varying d (code distance) or T (number of rounds), for (a) triangular patches, (b) rectangular patches, and (c) stability experiments. We consider $d \in \{3, 5, 7, \dots, 21\}$ for triangular patches, $d \in \{4, 6, 8, \dots, 20\}$ for rectangular patches, and $T \in \{4, 6, 8, \dots, 14\}$ for stability experiments. In (b), the per-round failure rates of $\bar{Z}$ and $\bar{X}$ (i.e., $p_{\text{rec}}^{Z,\text{fail}}/T$ and $p_{\text{rec}}^{X,\text{fail}}/T$) are separately presented. Each data point is marked as an error bar indicating its 99% confidence interval or as an “X” symbol if its relative margin of error (i.e., the ratio of half the width of its confidence interval to its center) is less than 10%. The solid lines are regression lines corresponding to individual d values, based on the ansatz in Eq. (11).

A representative ansatz commonly used in the literature [23,27,32,69] is

$$f(p, d) = \alpha \left(\frac{p}{p_{\text{th}}} \right)^{\beta d + \eta},$$

which depends on four parameters p_{th} , α , β , and η . Note that the per-round logical error rate of a surface code under circuit-level noise can be approximated by this ansatz with $(p_{\text{th}}, \alpha, \beta, \eta) \approx (0.01, 0.1, 0.5, 0.5)$ [69]. However, this ansatz is asymptotically valid for sufficiently small values of p , where higher-order terms of p are negligible. The ansatz cannot approximate the data in Fig. 23 sufficiently well, thus we add one higher-order term in the ansatz. The modified ansatz can be in the form of

$$f(p, d) = \alpha \left(\frac{p}{p_{\text{th}}} \right)^{\beta d + \eta} \left[1 + \epsilon \left(\frac{p}{p_{\text{th}}} \right)^{g(d)} \right], \quad (\text{G1})$$

where ϵ is an additional parameter and g is a function of d such that $g(d) > 0$ for every d . Since the form of the function g is unknown, we choose it among the seven candidates

$$\begin{aligned} g(d) &= \zeta, & g(d) &= \zeta d, & g(d) &= \zeta_0 + \zeta_1 d, \\ g(d) &= \zeta_0 + \zeta_1 d + \zeta_2 d^2, & g(d) &= d^\lambda, \\ g(d) &= \zeta d^\lambda, & g(d) &= \zeta_0 + \zeta_1 d^\lambda, \end{aligned}$$

which, respectively, have at most three additional parameters. Note that having more parameters in an ansatz does not always lead to better performance due to the risk of overfitting, which can reduce the model's ability to generalize to unseen data. To prevent overfitting and ensure generalizability, we use leave-one-out cross-validation (LOOCV) [70], a model validation technique where the model is trained on the dataset excluding a single data point and tested on that excluded data point,

iteratively for all data points. We then calculate the root-mean-square deviation (RMSD) of the tested data points as the LOOCV score, where a lower score indicates better performance. Here we note that the regressions are done through the least-squares method in the logarithmic scale with a transformed ansatz

$$\begin{aligned} \log f(p, d) &= \log \alpha + (\beta d + \eta)(\log p - \log p_{\text{th}}) \\ &\quad + \log [1 + \epsilon e^{(\log p - \log p_{\text{th}})g(d)}], \end{aligned}$$

using the function `curve_fit` in the Python library *SciPy* [71]. The LOOCV score is computed based on this logarithmic scale as well.

Table III presents the computed LOOCV scores for these seven candidates, all of which are significantly better than the ansatz without a higher-order term. The candidate $g(d) = \zeta d^\lambda$ shows the best overall performance, thus we select this as our ansatz. Note that $g(d) = d^\lambda$ performs better for the $\bar{Z}$ failure rate of rectangular patches, but we choose $g(d) = \zeta d^\lambda$ for consistency.

4. Ansatz parameter estimates

By fitting the data into the selected ansatz, we obtain the regression lines in Fig. 23. The estimated values of the ansatz parameters are as follows.

(a) Triangular patches [Fig. 23(a)]:

$$\begin{aligned} p_{\text{th}} &= 2.41 \times 10^{-3}, & \alpha &= 6.19 \times 10^{-4}, & \beta &= 0.537, \\ \eta &= -1.45, & \epsilon &= 27.2, & \zeta &= 0.404, & \lambda &= 0.933. \end{aligned}$$

(b) Rectangular patches, $\bar{Z}$ failure [Fig. 23(b), left]:

$$\begin{aligned} p_{\text{th}} &= 4.17 \times 10^{-3}, & \alpha &= 5.68 \times 10^{-4}, & \beta &= 0.439, \\ \eta &= -1.04, & \epsilon &= 88.1, & \zeta &= 0.927, & \lambda &= 0.332, \end{aligned}$$

TABLE III. Leave-one-out cross-validation (LOOCV) scores of several different ansatz candidates for logical failure rates, where g is the function in Eq. (G1). A lower LOOCV score indicates better generalizability. We select the ansatz with $g(d) = \zeta d^\lambda$, highlighted by (*).

Ansatz	LOOCV score			
	Triangular patches	Rectangular patches		Stability experiments
		$\bar{Z}$ failure	$\bar{X}$ failure	
No higher-order term ($\epsilon = 0$)	0.291	0.217	0.254	0.109
$g(d) = \zeta$	0.285	0.189	0.231	0.108
$g(d) = \zeta d$	0.129	0.195	0.138	0.0922
$g(d) = \zeta_0 + \zeta_1 d$	0.129	0.153	0.133	0.0905
$g(d) = \zeta_0 + \zeta_1 d + \zeta_2 d^2$	0.130	0.150	0.135	0.0905
$g(d) = d^\lambda$	0.162	0.125	0.159	0.0970
(*) $g(d) = \zeta d^\lambda$	0.129	0.145	0.134	0.0887
$g(d) = \zeta_0 + \zeta_1 d^\lambda$	0.129	0.164	0.140	0.0970

(c) Rectangular patches, $\bar{X}$ failure [Fig. 23(b), right]:

$$p_{\text{th}} = 3.07 \times 10^{-3}, \quad \alpha = 2.07 \times 10^{-4}, \quad \beta = 0.553, \\ \eta = -2.05, \quad \epsilon = 73.1, \quad \zeta = 0.515, \quad \lambda = 0.742.$$

(d) Stability experiments [Fig. 23(c)]:

$$p_{\text{th}} = 6.24 \times 10^{-3}, \quad \alpha = 6.91 \times 10^{-6}, \quad \beta = 0.601, \\ \eta = -1.61, \quad \epsilon = 543, \quad \zeta = 0.800, \quad \lambda = 0.389.$$

APPENDIX H: RULES TO DETERMINE NOISE CHANNELS FROM ERROR SOURCES DURING MSD

In this Appendix, we present the rules to determine noise channels from various possible error sources during MSD, as an extension of the discussion in Sec. V A. We here denote the logical-level support of a logical unitary operator $\bar{U}$ as $\text{supp}_{\text{L}} \bar{U}$.

1. Memory errors of logical patches

First, each logical patch suffers memory errors during or between lattice surgery. We include initialization and measurement errors of logical qubits (except for faulty T measurements) into this category for convenience. Importantly, during the merging operation of each stage for the single-level MSD scheme, we ignore $\bar{X}$ and $\bar{Y}$ errors of the qubits involving the stage. This is because new logical operators replacing such $\bar{X}$ and $\bar{Y}$ operators during the merging operation are represented by string-net operators having weights strictly larger than the code distances, as exemplified in Fig. 8(d). Likewise, in the cultivation-MSD scheme, $\bar{X}$ and $\bar{Z}$ errors can be ignored during merging operations.

For each stage k including the interval before the next merging operation (or including the final $\bar{X}$ measurements if $k = 8$), there exist noise channels $\Lambda_{\bar{U}, p_{\text{err}}}$ characterized as follows:

(1) $\bar{Z}$ error on each active qubit $\bar{q} \in \{\bar{q}_{\text{out}}, \bar{q}_A, \bar{q}_B, \bar{q}_C, \bar{q}_D\}$:

$$\bar{U} = \left(\bar{Z}_{\{\bar{q}\}} \right)_{\pi/2}, \\ p_{\text{err}} = \begin{cases} (d_m + 1) p^Z & \text{for the single-level scheme with any } k \\ & \text{and the cultivation-MSD scheme with } k = 8, \\ T_{\text{intv}} p^Z & \text{for the cultivation-MSD scheme with } k < 8, \end{cases}$$

where T_{intv} is the average number of rounds between successive stages and p^Z is the corresponding $\bar{Z}$ error rate $p_{\text{tri}}^Z(p, d_{\text{out}})$, $p_{\text{rec}}^{Z_1}(p, d_X, d_Z)$ (for qubits A and C), or $p_{\text{rec}}^{Z_2}(p, d_X, d_Z)$ (for qubits B and D).

(2) $\bar{X}$ error on each active qubit $\bar{q} \in \{\bar{q}_{\text{out}}, \bar{q}_A, \bar{q}_B, \bar{q}_C, \bar{q}_D\}$:

$$\bar{U} = \prod_{\substack{j \leq k \\ \bar{q} \in \text{supp}_{\text{L}} \bar{P}^{(j)}}} \bar{P}_{-\pi/4}^{(j)} \prod_{\substack{j \leq k \\ \bar{q} \in \text{supp}_{\text{L}} \bar{Q}^{(j)}}} \bar{Q}_{-\pi/4}^{(j)} \\ p_{\text{err}} = \begin{cases} \left(d_m + 1 - d_m \lambda_{\bar{q}}^{(k)} \right) p^X & \text{for the single-level scheme with any } k \\ & \text{and the cultivation-MSD scheme} \\ & \text{with } k = 8, \\ \left[T_{\text{intv}} - d_m \lambda_{\bar{q}}^{(k)} \right] p^X & \text{for the cultivation-MSD scheme} \\ & \text{with } k < 8, \end{cases}$$

where p^X is the corresponding $\bar{X}$ error rate $p_{\text{tri}}^X(p, d_{\text{out}})$, $p_{\text{rec}}^{X_1}(p, d_X, d_Z)$, or $p_{\text{rec}}^{X_2}(p, d_X, d_Z)$, and

$$\lambda_{\bar{q}}^{(k)} = \begin{cases} 1 & \text{if } \bar{q} \in \text{supp}_{\text{L}} \bar{P}^{(k)} \cup \text{supp}_{\text{L}} \bar{Q}^{(k)}, \\ 0 & \text{otherwise.} \end{cases}$$

(3) $\bar{Y}$ error on the output qubit $\bar{q}_{\text{out}}$ when it is active:

$$\bar{U} = (\bar{Z}_{\{0\}})_{\pi/2} \prod_{\substack{j \leq k \\ \bar{q}_{\text{out}} \in \text{supp}_L \bar{P}^{(j)}}} \bar{P}_{-\pi/4}^{(j)} \prod_{\substack{j \leq k \\ \bar{q}_{\text{out}} \in \text{supp}_L \bar{Q}^{(j)}}} \bar{Q}_{-\pi/4}^{(j)}$$

$$p_{\text{err}} = \begin{cases} (d_m + 1 - d_m \lambda_{\bar{q}_{\text{out}}}^{(k)}) p_{\text{tri}}^Y(p, d_{\text{out}}) & \text{for the single-level scheme with any } k \\ & \text{and the cultivation-MSD scheme with } k = 8, \\ [T_{\text{intv}} - d_m \lambda_{\bar{q}_{\text{out}}}^{(k)}] p_{\text{tri}}^Y(p, d_{\text{out}}) & \text{for the cultivation-MSD scheme with } k < 8. \end{cases}$$

(4) Correlated $\bar{Z}$ errors on each pair $(\bar{q}_1, \bar{q}_2) \in \{(\bar{q}_A, \bar{q}_B), (\bar{q}_C, \bar{q}_D)\}$, caused by Pauli- Y error strings:

$$\bar{U} = (\bar{Z}_{\{\bar{q}_1\}})_{\pi/2} (\bar{Z}_{\{\bar{q}_2\}})_{\pi/2},$$

$$p_{\text{err}} = \begin{cases} (d_m + 1) p_{\text{tri}}^{Z_1 Z_2}(p, d_X, d_Z) & \text{for the single-level scheme with any } k \\ & \text{and the cultivation-MSD scheme with } k = 8, \\ T_{\text{intv}} p_{\text{tri}}^{Z_1 Z_2}(p, d_X, d_Z) & \text{for the cultivation-MSD scheme with } k < 8. \end{cases}$$

(5) Correlated $\bar{X}$ errors on each pair $(\bar{q}_1, \bar{q}_2) \in \{(\bar{q}_A, \bar{q}_B), (\bar{q}_C, \bar{q}_D)\}$, caused by Pauli- Y error strings:

$$\bar{U} = \prod_{\substack{j \leq k \\ \bar{q}_1 \in \text{supp}_L \bar{P}^{(j)} \oplus \bar{q}_2 \in \text{supp}_L \bar{P}^{(j)}}} \bar{P}_{-\pi/4}^{(j)} \prod_{\substack{j \leq k \\ \bar{q}_1 \in \text{supp}_L \bar{Q}^{(j)} \oplus \bar{q}_2 \in \text{supp}_L \bar{Q}^{(j)}}} \bar{Q}_{-\pi/4}^{(j)}$$

$$p_{\text{err}} = \begin{cases} (d_m + 1 - d_m \lambda_{\bar{q}}^{(k)}) p_{\text{tri}}^{X_1 X_2}(p, d_X, d_Z) & \text{for the single-level scheme with any } k \\ & \text{and the cultivation-MSD scheme with } k = 8, \\ [T_{\text{intv}} - d_m \lambda_{\bar{q}}^{(k)}] p_{\text{tri}}^{X_1 X_2}(p, d_X, d_Z) & \text{for the cultivation-MSD scheme with } k < 8, \end{cases}$$

where “ $\oplus$ ” denotes the logical XOR operation.

(6) (Only for the single-level scheme) $\bar{W} \in \{\bar{X}, \bar{Y}, \bar{Z}\}$ error on each of qubits α and β :

$$\bar{U} = \begin{cases} \bar{P}_{\theta}^{(k)} & \text{for qubit } \alpha, \\ \bar{Q}_{\theta}^{(k)} & \text{for qubit } \beta, \end{cases} \quad p_{\text{err}} = \begin{cases} d_m p_{\text{tri}}^Z(p, d'_m) & \text{if } W = Z, \\ p_{\text{tri}}^W(p, d'_m) & \text{otherwise,} \end{cases}$$

where

$$\theta = \begin{cases} -\pi/4 & \text{if } W = X, \\ \pi/4 & \text{if } W = Y, \\ \pi/2 & \text{if } W = Z, \end{cases} \quad d'_m = \begin{cases} d_m & \text{if } k \geq 3, \\ d_Z - 1 & \text{otherwise.} \end{cases}$$

(7) (Only for the cultivation-MSD scheme) On each of qubits α and β , $\bar{Z}$, and $\bar{X}$ errors after the lattice surgery and $\bar{Y}$ errors after the magic state is prepared (including the growing operation if $d_m > d_{\text{cult}}$ and the idling operation):

$$\bar{U} = \begin{cases} \bar{P}_{\pi/2}^{(k)} & \text{for qubit } \alpha, \\ \bar{Q}_{\pi/2}^{(k)} & \text{for qubit } \beta, \end{cases}$$

$$p_{\text{err}} = p_{\text{tri}}^Z(p, d_m)/2 + p_{\text{tri}}^X(p, d_m)/2 + (d_m + 1) p_{\text{tri}}^Y(p, d_m) + (1 - \delta_{d_m, d_{\text{cult}}}) p_{\text{tri}}^Y(p, d_{\text{cult}}) + T_{\text{idle}} p_{\text{tri}}^Y(p, d_m),$$

where T_{idle} is the average number of rounds that the auxiliary patches idle before they are consumed.

- (8) (Only for the cultivation-MSD scheme) $\bar{X}$ and $\bar{Z}$ errors during the growing operation (if $d_m > d_{\text{cult}}$) and the idling operation on each of the two qubits α and β : two additional noise channels characterized by $(\bar{U}_+, p_{\text{err}})$ and $(\bar{U}_-, p_{\text{err}})$, where

$$\bar{U}_{\pm} = \begin{cases} \bar{P}_{\pm\pi/4}^{(k)} & \text{for qubit } \alpha, \\ \bar{Q}_{\pm\pi/4}^{(k)} & \text{for qubit } \beta, \end{cases}$$

$$p_{\text{err}} = \frac{1 - \delta_{d_m, d_{\text{cult}}}}{2} [p_{\text{tri}}^X(p, d_{\text{cult}}) + p_{\text{tri}}^Z(p, d_{\text{cult}})] + \frac{T_{\text{idle}}}{2} [p_{\text{tri}}^X(p, d_m) + p_{\text{tri}}^Z(p, d_m)].$$

A $\bar{X}$ or $\bar{Z}$ error flips the outcome of the lattice surgery, which results in a $\bar{P}_{\pi/4}^{(k)}$ or $\bar{P}_{-\pi/4}^{(k)}$ error depending on the measurement outcome.

2. Errors in the ancillary region

During the merging operation of stage k , the ancillary region may have spacelike and timelike error strings, which cause logical errors. For simpler calculations, we assume that irregular checks in domain walls have negligible effects on the fault tolerance of the color-code lattice.

We first consider spacelike error strings when $k \geq 3$. Certain Pauli- X (Z) error strings in the ancillary region may incur $\bar{Z}$ errors on the logical qubits involved in $\bar{P}_{\text{LS}}^{(k)}$ ($\bar{Q}_{\text{LS}}^{(k)}$). It is guaranteed that their weights are not less than the corresponding code distances, as shown in Appendix C. Hence, considering the case where the output qubit is involved in $\bar{P}_{\text{LS}}^{(k)}$ as an example, the output qubit has an additional $\bar{Z}$ error rate, which can be estimated from the $\bar{Z}_2$ error rate of an imaginary rectangular patch with distances $d_{X'} = d_V + 6$ and $d_{Z'} = d_{\text{out}} + 1$, where

$$d_H = \max(2d_Z, d_{\text{out}} + 1) + (N_m - N_{\text{m.side}})(d_m + 1),$$

$$d_V = \max[d_Z, N_{\text{m.side}}(d_m + 1)]$$

are the horizontal and vertical dimensions of the ancillary patch (defining $N_m = N_{\text{m.side}} = 1$ for the single-level MSD scheme). Here $d_{X'}$ has a constant term +6 for covering interface regions in addition to the ancillary patch. In the following, we list the estimated additional $\bar{Z}$ error rates of the logical qubits made by spacelike errors in the ancillary region during stage $k \geq 3$:

- (a) (Only for $k \geq 3$) Additional $\bar{Z}$ error rate on each qubit in $\text{supp}_L \bar{P}^{(k)} \subseteq \{\bar{q}_{\text{out}}, \bar{q}_A, \bar{q}_B, \bar{q}_C, \bar{q}_D\}$, which originates from horizontal Pauli- Z error strings in the ancillary region:

$$p_{\text{err.add}} = d_m p_{\text{rec}}^{Z_2}(p, d_V + 6, d_{Z'}),$$

where $d_{Z'} = d_{\text{out}} + 1$ for the output qubit and $d_Z = d_Z$ for the validation qubits.

- (b) (Only for $k \geq 3$) Additional $\bar{Z}$ error rate on each qubit in $\text{supp}_L \bar{Q}^{(k)} \subseteq \{\bar{q}_{\text{out}}, \bar{q}_A, \bar{q}_B, \bar{q}_C, \bar{q}_D\}$, which originates from horizontal Pauli- X error strings in the ancillary region:

$$p_{\text{err.add}} = d_m p_{\text{rec}}^{Z_1}(p, d_V + 6, d_{Z'}),$$

where $d_{Z'} = d_{\text{out}} + 1$ for the output qubit and $d_Z = d_Z$ for the validation qubits.

- (c) (Only for $k \geq 3$) Additional $\bar{Z}$ error rate on each qubit in $\text{supp}_L \bar{P}^{(k)} \Delta \text{supp}_L \bar{Q}^{(k)} \subseteq \{\bar{q}_{\text{out}}, \bar{q}_A, \bar{q}_B, \bar{q}_C, \bar{q}_D\}$, where $A \Delta B := A \cup B \setminus (A \cap B)$, which originates from horizontal Pauli- Y error strings in the ancillary region:

$$p_{\text{err.add}} = d_m p_{\text{rec}}^{Z_1 Z_2}(p, d_V + 6, d_{Z'}),$$

where $d_{Z'} = d_{\text{out}} + 1$ for the output qubit and $d_Z = d_Z$ for the validation qubits.

- (d) (Only for the single-level scheme with $k \geq 3$) Additional $\bar{Z}$ error rate on each of qubits α and β , which originates from vertical Pauli- Z and Y error strings (for qubit α) or vertical Pauli- X and Y error strings (for qubit β) in the ancillary region:

$$p_{\text{err.add}} = d_m p_{\text{rec}}^{X_i}(p, d_m + 1, d_H + 6) + d_m p_{\text{rec}}^{X_1 X_2}(p, d_m + 1, d_H + 6),$$

where $i = 1$ for qubit α and $i = 2$ for qubit β .

- (e) (Only for the cultivation-MSD scheme with $k \geq 3$) Additional $\bar{Y}$ error rate on each of qubits α and β (item 7 of Appendix H1), which originates from vertical Pauli- Z and Y error strings (for qubit α) or vertical Pauli- X and Y error strings (for qubit β) in the ancillary region:

$$p_{\text{err.add}} = \frac{N_{\text{m.side}}}{N_{\text{m}}} d_{\text{m}} [p_{\text{rec}}^{X_i}(p, d_{\text{m}} + 1, d_{\text{H}} + 6) + p_{\text{rec}}^{X_1 X_2}(p, d_{\text{m}} + 1, d_{\text{H}} + 6)] \\ + \frac{N_{\text{m}} - N_{\text{m.side}}}{N_{\text{m}}} d_{\text{m}} [p_{\text{rec}}^{Z_j}(p, d_{\text{V}} + 6, d_{\text{m}} + 1) + p_{\text{rec}}^{Z_1 Z_2}(p, d_{\text{V}} + 6, d_{\text{m}} + 1)],$$

where $(i, j) = (1, 2)$ for qubit α and $(i, j) = (2, 1)$ for qubit β . This is the average of error rates for the $2N_{\text{m}}$ auxiliary patches.

When $k < 3$, the single-level MSD scheme employs the simple layout as shown in Figs. 7(a) and 7(b). As in the case of $k \geq 3$, we consider an imaginary rectangular patch covering the thin ancillary region of the layout for estimating additional logical error rates from spacelike error strings. On the other hand, the cultivation-MSD scheme uses the full layout in Fig. 11 even for $k < 3$, and the ancillary region is not guaranteed to be distance-preserving unlike the cases of $k \geq 3$. For example, stage 1 involves the measurement of $\bar{P}_{\text{LS}}^{(1)} = \bar{Z}_{\text{A}} \otimes \bar{Y}_{\alpha}$, thus a $\bar{Z}$ error on qubit A is equivalent to a $\bar{Y}$ error on qubit α , meaning that there may exist gx-string operators with weight $\min(d_{\text{Z}}, d_{\text{m}} + 1)$ in the ancillary region that cause a $\bar{Z}$ error on qubit A. Hence, we instead consider an imaginary rectangular patch with distances $d_{\text{X}}' = d_{\text{V}}$ and $d_{\text{Z}}' = \min(d_{\text{Z}}, d_{\text{m}} + 1)$. We thus obtain the following additional error rates for $k < 3$:

- (a) (Only for the single-level scheme with $k < 3$) Additional $\bar{Z}$ error rate on each of qubits A, C (if $k = 1$) or qubits B, D (if $k = 2$):

$$p_{\text{err.add}} = d_{\text{m}} \max [p_{\text{rec}}^{Z_1}(p, 4, d_{\text{Z}}), p_{\text{rec}}^{Z_2}(p, 4, d_{\text{Z}})] \\ + d_{\text{m}} p_{\text{rec}}^{Z_1 Z_2}(p, 4, d_{\text{Z}}).$$

Note that we take the maximum between $p_{\text{rec}}^{Z_1}$ and $p_{\text{rec}}^{Z_2}$ due to ambiguity from a Pauli-permuting domain wall.

- (b) (Only for the cultivation-MSD scheme with $k < 3$) Additional $\bar{Z}$ error rate on each of qubits A, C (if $k = 1$) or qubits B, D (if $k = 2$):

$$p_{\text{err.add}} = d_{\text{m}} p_{\text{rec}}^{Z_i}(p, d_{\text{V}} + 6, \min(d_{\text{Z}}, d_{\text{m}} + 1)) \\ + d_{\text{m}} p_{\text{rec}}^{Z_1 Z_2}(p, d_{\text{V}} + 6, \min(d_{\text{Z}}, d_{\text{m}} + 1))$$

where $i = 2$ for qubits A and B and $i = 1$ for qubits C and D. Note that $p^{(k)}$ involves qubits A or B and $q^{(k)}$ involves qubits C or D.

Next, the ancillary region may also have timelike error strings. A timelike error string that corrupts red X -type checks (e.g., d_{m} consecutive measurement errors of a red X -type check) flips the measurement outcome of $\bar{P}^{(k)}$, as its value is determined from the product of X -type check outcomes in the ancillary region; see Fig. 8(a). In the single-level scheme, this is equivalent to a $\bar{X}$ error on qubit α after the lattice surgery. In the cultivation-MSD scheme, this makes a $\bar{P}_{\pi/4}^{(k)}$ or $\bar{P}_{-\pi/4}^{(k)}$ error on the output and validation qubit, depending on the measurement outcome of $\bar{P}^{(k)} \otimes \bar{Y}_{\alpha}$, which is the same effect as a $\bar{X}$ error during the growing operation (item 8 in Appendix H 1). Considering timelike error strings corrupting Z -type checks as well, we obtain the following noise channels for each stage k :

- (a) (Only for the single-level scheme) Additional $\bar{X}$ error rate on each of qubits α and β :

$$p_{\text{err.add}} = \begin{cases} p_{\text{TL}}^X(p, d_{\text{H}} + 6, d_{\text{V}} + 6, d_{\text{m}}) & \text{for qubit } \alpha \text{ when } k > 3, \\ p_{\text{TL}}^Z(p, d_{\text{H}} + 6, d_{\text{V}} + 6, d_{\text{m}}) & \text{for qubit } \beta \text{ when } k > 3, \\ \max [p_{\text{TL}}^X(p, d_{\text{Z}}, 4, d_{\text{m}}), p_{\text{TL}}^Z(p, d_{\text{Z}}, 4, d_{\text{m}})] & \text{when } k < 3. \end{cases}$$

- (b) (Only for the cultivation-MSD scheme) Additional error rate on p_{err} of item 8 in Appendix H 1:

$$p_{\text{err.add}} = \begin{cases} p_{\text{TL}}^X(p, d_{\text{H}} + 6, d_{\text{V}} + 6, d_{\text{m}})/2 & \text{for qubit } \alpha, \\ p_{\text{TL}}^Z(p, d_{\text{H}} + 6, d_{\text{V}} + 6, d_{\text{m}})/2 & \text{for qubit } \beta. \end{cases}$$

3. Errors from non-Clifford components

The non-Clifford components of our MSD schemes comprise the faulty T measurement for the single-level scheme and cultivation for the cultivation-MSD scheme. We now explore how errors in these components affect the output logical states.

For the single-level scheme, based on the argument in Sec. III B, we handle errors caused by faulty T measurements as follows:

- (a) (Only for the single-level scheme) For each stage k , each of qubits α and β additionally has a $\bar{X}$ error rate of $2p/3$, a $\bar{Y}$ error rate of $2p/3$, and a $\bar{Z}$ error rate of $5p/3$.

For the cultivation-MSD scheme, a $\bar{X}$ or $\bar{Z}$ error on a magic state has the same effect as the memory error during the growing operation, which is covered in item 8 of Appendix H 1. A $\bar{Y}$ error can be handled just as an additional $\bar{Y}$ error. Hence, the effects of errors from cultivation are as follows for each stage k :

- (a) (Only for the cultivation-MSD scheme) Additional error rate on p_{err} of item 8 in Appendix H 1:

$$p_{\text{err.add}} = \frac{1}{2} [p_{\text{cult}}^X + p_{\text{cult}}^Z].$$

- (b) (Only for the cultivation-MSD scheme) Additional $\bar{Y}$ error rate on each of qubits α and β :

$$p_{\text{err.add}} = p_{\text{cult}}^Y.$$

APPENDIX I: ANALYTIC EXPRESSIONS OF THE SUCCESS PROBABILITY AND OUTPUT INFIDELITY OF MSD

In this Appendix, we present the analytic expressions of the success probability q_{succ} and the output infidelity q_{dist} of the single-level and cultivation-MSD schemes, as functions of code distances ($d_{\text{out}}, d_X, d_Z, d_m$), physical noise strength p , and the logical error rates of several patches. The expressions are obtained via the method in Sec. V A and Appendix H. For simplicity, we assume that, for a parameter $r_y \geq 0$,

$$\begin{aligned} p_{\text{tri}}(p, d) &:= p_{\text{tri}}^X(p, d) = p_{\text{tri}}^Z(p, d) = p_{\text{tri}}(p, d)/r_y, \\ p_{\text{rec}}^X(p, d_X, d_Z) &:= p_{\text{rec}}^{X_1}(p, d_X, d_Z) = p_{\text{rec}}^{X_2}(p, d_X, d_Z) = p_{\text{rec}}^{X_1 X_2}(p, d_X, d_Z)/r_y, \\ p_{\text{rec}}^Z(p, d_X, d_Z) &:= p_{\text{rec}}^{Z_1}(p, d_X, d_Z) = p_{\text{rec}}^{Z_2}(p, d_X, d_Z) = p_{\text{rec}}^{Z_1 Z_2}(p, d_X, d_Z)/r_y, \\ p_{\text{cult}} &:= p_{\text{cult}}^X = p_{\text{cult}}^Z = p_{\text{cult}}^Y/r_y, \\ p_{\text{grow}}(p, d_{\text{cult}}, d_m, c_{\text{gap}}) &:= p_{\text{grow}}^X(p, d_{\text{cult}}, d_m, c_{\text{gap}}) = p_{\text{grow}}^Z(p, d_{\text{cult}}, d_m, c_{\text{gap}}) = p_{\text{grow}}^Y(p, d_{\text{cult}}, d_m, c_{\text{gap}})/r_y, \\ L_H L_V p_{\text{TL}}(p, T) &:= p_{\text{TL}}^X(p, L_H, L_V, T) = p_{\text{TL}}^Z(p, L_H, L_V, T), \end{aligned}$$

where the last line implies that $p_{\text{TL}}^X(p, L_H, L_V, T) = p_{\text{TL}}^Z(p, L_H, L_V, T) \propto L_H L_V \propto (\text{Area of the patch})$.

1. Single-level MSD scheme

We define

$$\begin{aligned} p_{\text{out}} &:= p_{\text{tri}}(p, d_{\text{out}}), & p_{\text{anc.out}} &:= p_{\text{rec}}^Z(p, d_V + 6, d_{\text{out}} + 1), \\ p_{\text{rec.X}} &:= p_{\text{rec}}^X(p, d_X, d_Z), & p_{\text{anc.rec}} &:= d_m(1 + r_y)p_{\text{rec}}^Z(p, d_V + 6, d_Z), \\ p_{\text{rec.Z}} &:= r_y(d_m + 1)p_{\text{rec}}^Z(p, d_X, d_Z), & p_{\text{anc.rec.pre}} &:= d_m(1 + r_y)p_{\text{rec}}^Z(p, 4, d_Z), \\ p_m &:= (2d_m + 1 + r_y)p_{\text{tri}}(p, d_m), & p_{\text{anc.m}} &:= d_m(1 + r_y)p_{\text{rec}}^X(p, d_m + 1, d_H + 6), \\ p_{\text{timelike}} &:= (d_H + 6)(d_V + 6)p_{\text{TL}}(p, d_m), \end{aligned}$$

where $d_H := \max(d_{\text{out}} + 1, 2d_Z)$ and $d_V := \max(d_m + 1, d_Z)$. We present terms of q_{succ} and q_{dist} up to $O(p^3)$, $O(p^2\bar{p})$, and $O(\bar{p}\bar{p}')$, where $\bar{p}$ and $\bar{p}'$ are any of the symbols defined above. In addition, we show only the lowest-order term among nonconstant terms having comparable orders (e.g., if the expression has a term involving $pp_{\text{rec.X}}$, terms involving $p^2p_{\text{rec.X}}$ are omitted). The obtained expressions of q_{dist} and q_{succ} are as follows:

$$\begin{aligned}
q_{\text{dist}} &\approx 35 \left(\frac{7}{3}p \right)^3 + \frac{16d_m + 19 + r_y}{4} p_{\text{out}} + d_m (7 + r_y) p_{\text{anc.out}} \\
&\quad + \frac{7}{2} \left(\frac{7}{3}p \right)^2 \left(15p_m + 22p_{\text{anc.m}} + 8p_{\text{anc.rec.pre}} + \frac{11(d_H + 6)(d_V + 6) + 16d_Z}{(d_H + 6)(d_V + 6)} p_{\text{timelike}} \right) \\
&\quad + \left(8p_{\text{rec.Z}} + 4p_{\text{anc.rec}} + \frac{27 + 19r_y}{8} p_{\text{rec.X}} \right) \left(\frac{14}{3}p + p_m + 2p_{\text{anc.m}} + p_{\text{timelike}} \right) \\
&\quad + 4p_{\text{rec.X}} p_{\text{rec.Z}} + 2p_{\text{rec.X}} p_{\text{anc.rec}} + \frac{31 + 68r_y}{8} p_{\text{rec.X}}^2, \\
q_{\text{succ}} &\approx 1 - 35p - \frac{15}{2}p_m - 11p_{\text{anc.m}} - 4p_{\text{anc.rec.pre}} - \frac{11(d_H + 6)(d_V + 6) + 16d_Z}{2(d_H + 6)(d_V + 6)} p_{\text{timelike}} - \frac{16(2 + r_y)}{r_y} p_{\text{rec.Z}} \\
&\quad - \frac{4(5 + 3r_y)}{1 + r_y} p_{\text{anc.rec}} - \frac{4d_m + 71 + 35r_y}{4} p_{\text{rec.X}} - 2(1 + r_y) p_{\text{out}}.
\end{aligned}$$

It is worth noting that q_{dist} does not contain a term of $O(p_{\text{rec.X}})$, which is consistent with the argument in Sec. III A showing that the MSD circuit is tolerant to a single-location $\bar{X}$ error on one of the validation qubits.

2. Cultivation-MSD scheme

We define

$$\begin{aligned}
p_{\text{out}} &:= p_{\text{tri}}(p, d_{\text{out}}), \\
p_{\text{rec.X}} &:= (T_{\text{intv}} - d_m) p_{\text{rec}}^X(p, d_X, d_Z), \\
p_{\text{rec.Z}} &:= r_y T_{\text{intv}} p_{\text{rec}}^Z(p, d_X, d_Z), \\
p_m &:= [(1 + r_y)(T_{\text{idle}} + d_m) + 1] p_{\text{tri}}(p, d_m), \\
p_{\text{timelike}} &:= (d_H + 6)(d_V + 6) p_{\text{TL}}(p, d_m), \\
p_{\text{magic}} &:= (1 + r_y) \left[p_{\text{cult}} + (1 - \delta_{d_m, d_{\text{cult}}}) p_{\text{grow}}(p, d_{\text{cult}}, d_m, c_{\text{gap}}) \right], \\
p_{\text{anc.out}} &:= p_{\text{rec}}^Z(p, d_V + 6, d_{\text{out}} + 1), \\
p_{\text{anc.rec}} &:= d_m(1 + r_y) p_{\text{rec}}^Z(p, d_V + 6, d_Z), \\
p_{\text{anc.rec.pre}} &:= d_m(1 + r_y) p_{\text{rec}}^Z(p, d_m + 1, d_V + 6), \\
p_{\text{anc.m}} &:= d_m(1 + r_y) p_{\text{rec}}^X(p, d_m + 1, d_H + 6),
\end{aligned}$$

where d_H and d_V are defined in Eq. (8), T_{intv} is the average number of rounds between successive stages, T_{idle} is the average number of rounds that auxiliary patches idle before they are consumed. Note that T_{intv} and T_{idle} can be estimated by simulating the procedure of the cultivation-MSD scheme, as described in Sec. IV C and implemented Ref. [42]. We compute terms of q_{succ} and q_{dist} up to $O(\bar{p}\bar{p}'\bar{p}'')$, where $\bar{p}$, $\bar{p}'$, and $\bar{p}''$ are any of the symbols defined above and show only the lowest-order term (except a constant) among terms having comparable orders. The obtained expressions of q_{dist} and q_{succ} are then as follows:

$$\begin{aligned}
q_{\text{dist}} &\approx 35 \left(p_{\text{magic}} + p_m + p_{\text{anc.m}} + \frac{1}{2} p_{\text{timelike}} \right)^3 + 28p_{\text{anc.rec.pre}} \left(p_{\text{anc.m}} + p_{\text{magic}} + p_m + \frac{1}{2} p_{\text{timelike}} \right)^2 \\
&\quad + \left(\frac{27 + 19r_y}{4} p_{\text{rec.X}} + 16p_{\text{rec.Z}} + 8p_{\text{anc.rec}} + 6p_{\text{anc.rec.pre}}^2 \right) \left(p_{\text{magic}} + p_m + p_{\text{anc.m}} + \frac{1}{2} p_{\text{timelike}} \right) \\
&\quad + \frac{19T_{\text{intv}} - 3d_m + (T_{\text{intv}} - d_m)r_y}{4} p_{\text{out}} + d_m (7 + r_y) p_{\text{anc.out}} \\
&\quad + \frac{31 + 68r_y}{8} p_{\text{rec.X}}^2 + \left[\left(\frac{5}{2} + r_y \right) p_{\text{anc.rec.pre}}^2 + 2p_{\text{anc.rec}} + 4p_{\text{rec.Z}} \right] p_{\text{rec.X}},
\end{aligned}$$

$$q_{\text{succ}} \approx 1 - 15 \left(p_{\text{magic}} + p_{\text{m}} + p_{\text{anc.m}} + \frac{1}{2} p_{\text{timelike}} \right) - \left(\frac{71 + 35r_y}{4} + \frac{d_{\text{m}}}{T_{\text{intv}} - d_{\text{m}}} \right) p_{\text{rec.X}} - \frac{16(2 + r_y)}{r_y} p_{\text{rec.Z}} - \frac{20 + 12r_y}{1 + r_y} p_{\text{anc.rec}} - 4p_{\text{anc.rec.pre}}.$$

APPENDIX J: DEPENDENCY ANALYSIS OF MSD PERFORMANCE ON r_y

In Fig. 24, we plot the output infidelities q_{dist} of our single-level and cultivation-MSD schemes against r_y for several combinations of parameters at $p \in \{5 \times 10^{-4}, 10^{-3}\}$, showing that q_{dist} does not significantly depend on r_y . Even if we compare the two extreme cases of $r_y = 0$ and $r_y = 1$, the difference in infidelity is at most only a factor of approximately 1.5.

APPENDIX K: INTEGRATED CULTIVATION + GROWING SIMULATIONS

When analyzing the cultivation-MSD scheme in Sec. V, we handled the cultivation and growing operations as two separate modules. In other words, we assumed that the cultivation process outputs a magic state with a given infidelity and success probability, which is inputted to the subsequent growing operation. We simulated each operation independently and combined the results. This was for keeping the modularity of our scheme by treating the cultivation part as a “black box.” However, in an actual process,

the “interface” between these two modules may have non-negligible effects on the overall performance of the scheme, thus simulating the two modules independently may give inaccurate results.

In this Appendix, we analyze such effects in the case of using the scheme in Ref. [26] for cultivation. We combine the cultivation and growing circuits into a single circuit and simulate it jointly. Note that, as done in Ref. [26], we replace T with S in the cultivation circuit for its efficient simulation using *Stim* [67] and track the value of $\bar{Y}$. Consequently, the cultivation logical error rate is estimated by doubling the obtained logical error rate of the modified circuit, following the same assumption in Ref. [26].

The main difference from our original analysis arises from detectors connecting the cultivation and growing parts. Originally, the last layer of detectors in the cultivation circuit for its simulation connects the final logical check measurements and the following virtual perfect syndrome measurements after the cultivation. Similarly, the first layer of detectors (involving the initial patch) in the growing circuit connects the first syndrome measurements and the virtual perfect logical preparation layer before the growth. By combining the two circuits, these two

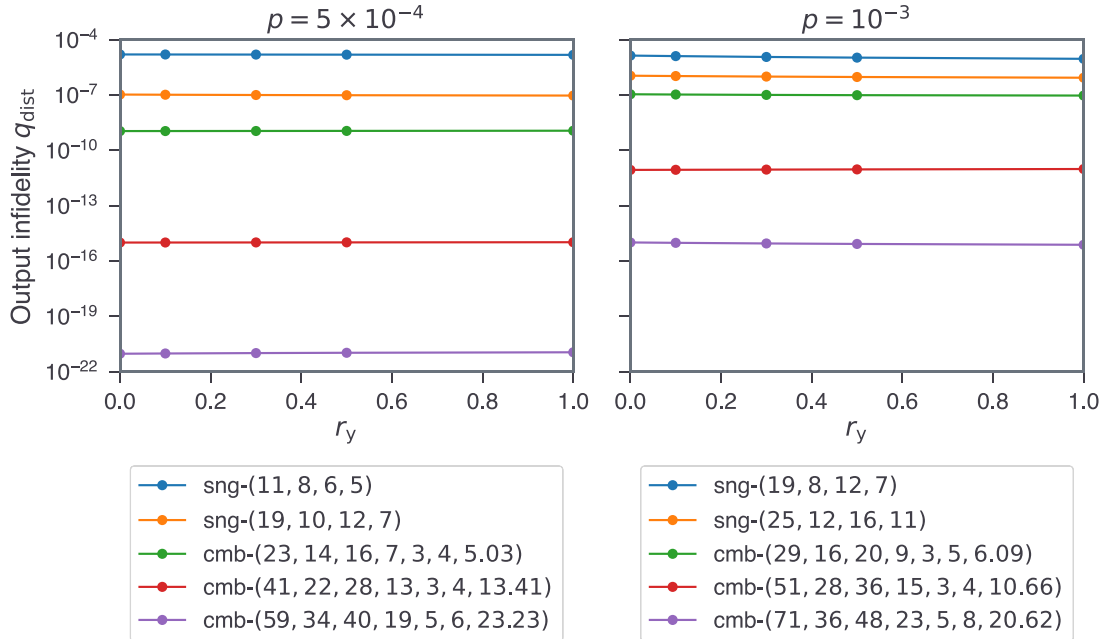


FIG. 24. Dependency of the output infidelity on r_y at $p \in \{5 \times 10^{-4}, 10^{-3}\}$ for several variants of the single-level and cultivation-MSD schemes, denoted as “sng- $(d_{\text{out}}, d_X, d_Z, d_{\text{m}})$ ” and “cmb- $(d_{\text{out}}, d_X, d_Z, d_{\text{m}}, d_{\text{cult}}, N_{\text{m}}, c_{\text{gap}})$ ”, respectively.

layers are merged naturally into a single layer of detectors. Importantly, the values of these detectors are now used for postselection of cultivation, not for decoding the growing operation. Namely, we abort cultivation if any of these detectors gives -1 , which is more probable than in the original scenario (assuming perfect syndrome measurements after the cultivation), implying that the success probability of cultivation may be reduced by this modification.

We now present the results obtained from our simulations. First, Table IV shows the success rates of the cultivation module within the integrated cultivation + growing circuit, alongside those from the original circuit for comparison. This aligns with our expectation that integration would decrease the success rate, although the observed differences are not significant ($<4\%$ difference) except in the case of $(p, d_{\text{cult}}) = (10^{-3}, 5)$.

An end-to-end resource cost analysis is presented in Figs. 25 and 26, showing the logical error rate as a function of the effective time cost and the spacetime cost, respectively, for $p \in \{5 \times 10^{-4}, 10^{-3}\}$.

TABLE IV. Success rates $q_{\text{cult}}^{\text{succ}}$ of cultivation for the original circuit [26] and the integrated cultivation + growing circuit.

p	d_{cult}	$q_{\text{cult}}^{\text{succ}}$ (original)	$q_{\text{cult}}^{\text{succ}}$ (integrated)
5×10^{-4}	3	0.83	0.80
5×10^{-4}	5	0.35	0.35
10^{-3}	3	0.65	0.64
10^{-3}	5	0.15	0.12

For comparison, these figures also contain the curves (dashed lines) obtained by combining results from our original simulations.

The time cost analyzed in Fig. 25 is the one that directly affects the cost of our cultivation-MSD scheme, as it affects T_m (average number of rounds between successive stages). The discrepancy in the time costs between the integrated and separate scenarios is particularly notable for $(p, d_{\text{cult}}) = (10^{-3}, 5)$, where they differ by at most about 1.7 times, implying that the spacetime cost of the cultivation-MSD scheme can be increased by a

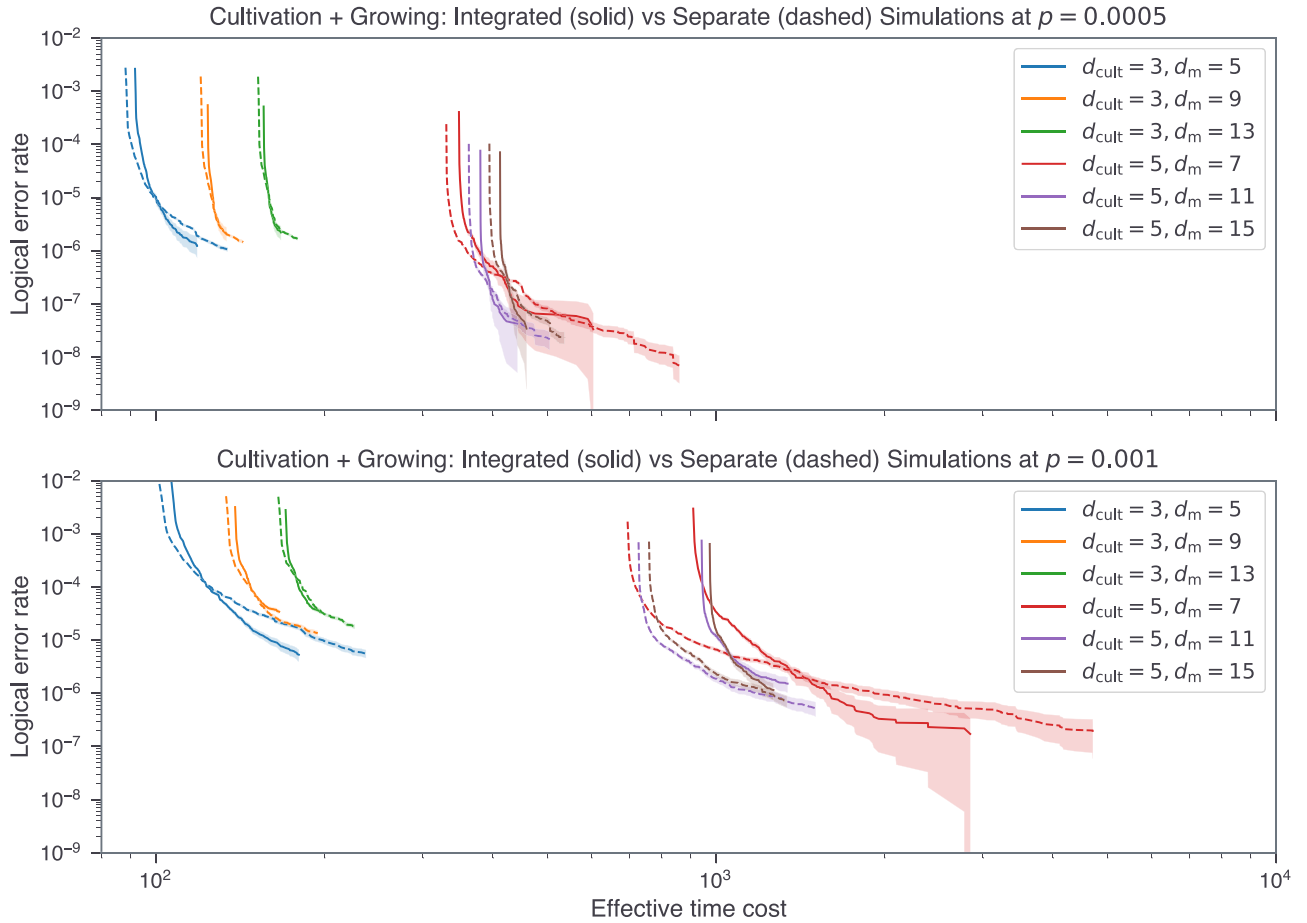


FIG. 25. Logical error rate of the cultivation + growing circuit, plotted against the effective time cost for $p \in \{5 \times 10^{-4}, 10^{-3}\}$ and various combinations of d_{cult} and d_m . For comparison, integrated scenarios (where the cultivation and growing modules are simulated jointly) and separate scenarios (where the two modules are simulated separately and the results are combined straightforwardly) are represented by solid and dashed lines.

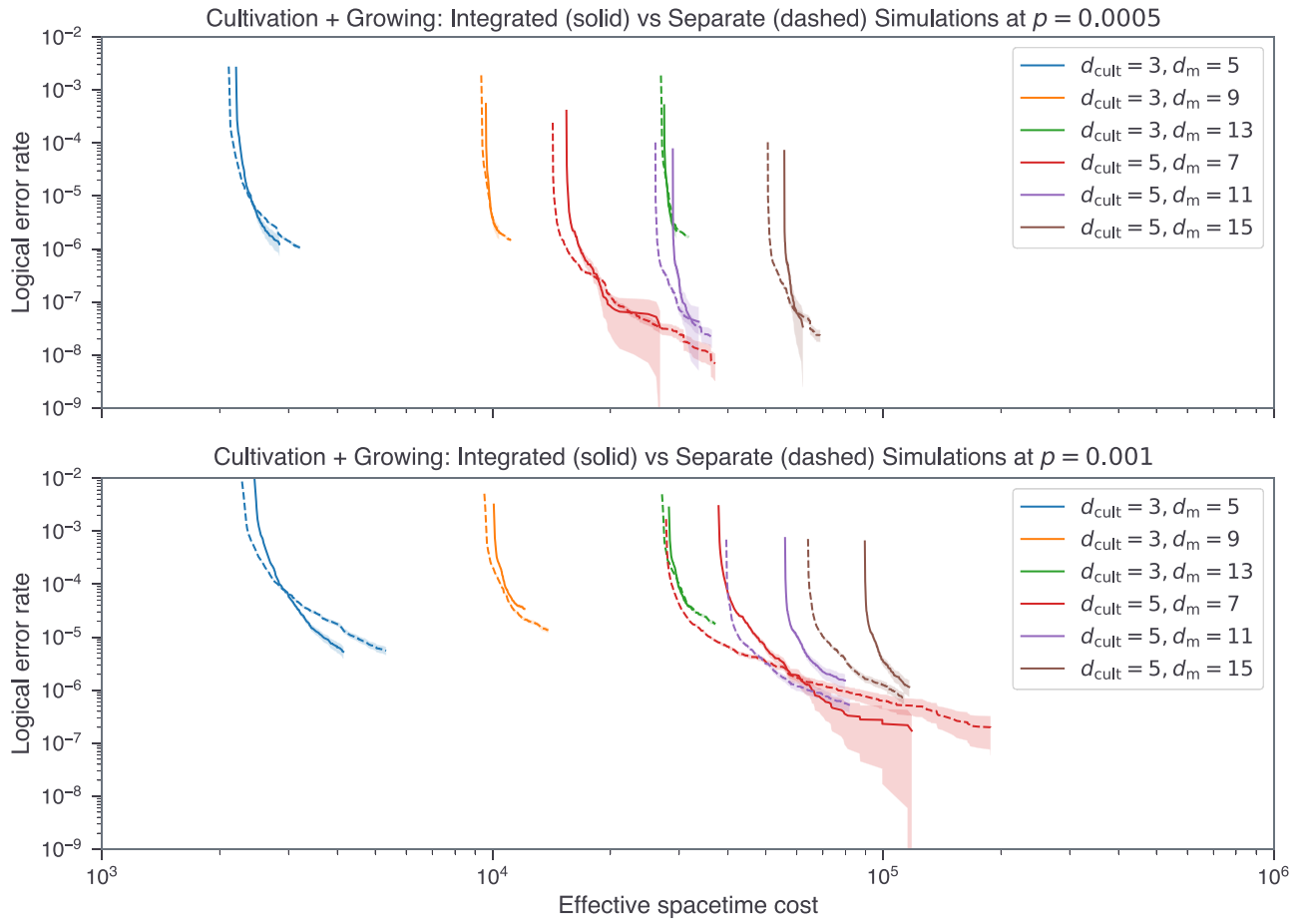


FIG. 26. Logical error rate of the cultivation + growing circuit, plotted against the effective spacetime cost for $p \in \{5 \times 10^{-4}, 10^{-3}\}$ and various combinations of d_{cult} and d_m . For comparison, integrated and separate scenarios are represented by solid and dashed lines.

similar factor. Additionally, the minimum achievable logical infidelity of the cultivation-MSD scheme also can be affected. Although the large confidence intervals for the logical error rates in Fig. 25 make precise estimation difficult, we conjecture that the minimum infidelity could increase by approximately 3–30 times for both $p = 5 \times 10^{-4}$ and 10^{-3} . This conjecture is based on the observation that the minimum logical error rates for the integrated scenario are roughly 1.5–3 times higher than those for the separated scenario when $d_{\text{cult}} = 5$ and $d_m \in \{11, 15\}$.

The spacetime cost analyzed in Fig. 26 is not directly related to our resource analysis for MSD, but we place the figure here for readers who are interested in a “color-code-only” cultivation scheme that does not involve conversion to a surface code unlike the scheme in Ref. [26].

The codes we have used for the above simulations are available in the *color-code-stim* package [65].

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, New York, 2010), 10th ed.

[2] H. Bombin and M. A. Martin-Delgado, Topological quantum distillation, *Phys. Rev. Lett.* **97**, 180501 (2006).
 [3] H. Bombin, in *Quantum Error Correction*, edited by D. A. Lidar and T. A. Brun (Cambridge University Press, Cambridge, 2013), Chap. 19, p. 455.
 [4] S. B. Bravyi and A. Y. Kitaev, Quantum codes on a lattice with boundary, *ArXiv:quant-ph/9811052*.
 [5] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Topological quantum memory, *J. Math. Phys.* **43**, 4452 (2002).
 [6] A. J. Landahl, J. T. Anderson, and P. R. Rice, Fault-tolerant quantum computing with color codes, *ArXiv:1108.5738*.
 [7] M. S. Kesselring, J. C. Magdalena de la Fuente, F. Thomsen, J. Eisert, S. D. Bartlett, and B. J. Brown, Anyon condensation and the color code, *PRX Quantum* **5**, 010342 (2024).
 [8] F. Thomsen, M. S. Kesselring, S. D. Bartlett, and B. J. Brown, Low-overhead quantum computing with the color code, *Phys. Rev. Res.* **6**, 043125 (2024).
 [9] L. Postler, S. Heußen, I. Pogorelov, M. Rispler, T. Feldker, M. Meth, C. D. Marciniak, R. Stricker, M. Ringbauer, R. Blatt, P. Schindler, M. Müller, and T. Monz, Demonstration of fault-tolerant universal quantum gate operations, *Nature* **605**, 675 (2022).
 [10] C. Ryan-Anderson, *et al.*, Implementing fault-tolerant entangling gates on the five-qubit code and the color code, *ArXiv:2208.01863*.

- [11] D. Bluvstein, *et al.*, Logical quantum processor based on reconfigurable atom arrays, *Nature* **626**, 58 (2024).
- [12] L. Postler, F. Butt, I. Pogorelov, C. D. Marciniak, S. Heußen, R. Blatt, P. Schindler, M. Risppler, M. Müller, and T. Monz, Demonstration of fault-tolerant Steane quantum error correction, *PRX Quantum* **5**, 030326 (2024).
- [13] N. Lacroix, *et al.*, Scaling and logic in the color code on a superconducting quantum processor, *ArXiv:2412.14256*.
- [14] A. Kubica and M. E. Beverland, Universal transversal gates with color codes: A simplified approach, *Phys. Rev. A* **91**, 032330 (2015).
- [15] A. J. Landahl and C. Ryan-Anderson, Quantum computing by color-code lattice surgery, *ArXiv:1407.5103*.
- [16] S. Bravyi and A. Kitaev, Universal quantum computation with ideal Clifford gates and noisy ancillas, *Phys. Rev. A* **71**, 022316 (2005).
- [17] B. W. Reichardt, Quantum universality from magic states distillation applied to CSS codes, *Quantum Inf. Process.* **4**, 251 (2005).
- [18] A. M. Meier, B. Eastin, and E. Knill, Magic-state distillation with the four-qubit code, *ArXiv:1204.4221*.
- [19] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Surface codes: Towards practical large-scale quantum computation, *Phys. Rev. A* **86**, 032324 (2012).
- [20] S. Bravyi and J. Haah, Magic-state distillation with low overhead, *Phys. Rev. A* **86**, 052329 (2012).
- [21] J. O’Gorman and E. T. Campbell, Quantum computation with realistic magic-state factories, *Phys. Rev. A* **95**, 032338 (2017).
- [22] M. E. Beverland, A. Kubica, and K. M. Svore, Cost of universality: A comparative study of the overhead of state distillation and code switching with color codes, *PRX Quantum* **2**, 020341 (2021).
- [23] D. Litinski, Magic state distillation: Not as costly as you think, *Quantum* **3**, 205 (2019).
- [24] C. Chamberland and K. Noh, Very low overhead fault-tolerant magic state preparation using redundant ancilla encoding and flag qubits, *npj Quantum Inf.* **6**, 91 (2020).
- [25] T. Itogawa, Y. Takada, Y. Hirano, and K. Fujii, Efficient magic state distillation by zero-level distillation, *PRX Quantum* **6**, 020356 (2025).
- [26] C. Gidney, N. Shutty, and C. Jones, Magic state cultivation: Growing T states as cheap as CNOT gates, *ArXiv:2409.17595*.
- [27] S.-H. Lee, A. Li, and S. D. Bartlett, Color code decoder with improved scaling for correcting circuit-level noise, *Quantum* **9**, 1609 (2025).
- [28] A. G. Fowler, Two-dimensional color-code quantum computation, *Phys. Rev. A* **83**, 042310 (2011).
- [29] B. J. Brown, K. Laubscher, M. S. Kesselring, and J. R. Wootton, Poking holes and cutting corners to achieve Clifford gates with the surface code, *Phys. Rev. X* **7**, 021029 (2017).
- [30] M. S. Kesselring, F. Pastawski, J. Eisert, and B. J. Brown, The boundaries and twist defects of the color code and their applications to topological quantum computation, *Quantum* **2**, 101 (2018).
- [31] N. Fazio, R. Harper, and S. Bartlett, Logical noise bias in magic state injection, *ArXiv:2401.10982*.
- [32] D. Litinski, A game of surface codes: Large-scale quantum computing with lattice surgery, *Quantum* **3**, 128 (2019).
- [33] E. Knill, Quantum computing with realistically noisy devices, *Nature* **434**, 39 (2005).
- [34] H. Goto, Minimizing resource overheads for fault-tolerant preparation of encoded states of the Steane code, *Sci. Rep.* **6**, 19578 (2016).
- [35] C. Chamberland and A. W. Cross, Fault-tolerant magic state preparation with flag qubits, *Quantum* **3**, 143 (2019).
- [36] A. Kitaev and L. Kong, Models for gapped boundaries and domain walls, *Commun. Math. Phys.* **313**, 351 (2012).
- [37] C. Gidney and C. Jones, New circuits and an open source decoder for the color code, *ArXiv:2312.08813*.
- [38] C. Gidney, M. Newman, P. Brooks, and C. Jones, Yoked surface codes, *Nat. Commun.* **16**, 4498 (2025).
- [39] H. Bombín, M. Pant, S. Roberts, and K. I. Seetharam, Fault-tolerant postselection for low-overhead magic state preparation, *PRX Quantum* **5**, 010302 (2024).
- [40] S. C. Smith, B. J. Brown, and S. D. Bartlett, Mitigating errors in logical qubits, *Commun. Phys.* **7**, 386 (2024).
- [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] S.-H. Lee, msd-magic-state-prep-cycle-simulation, 2025, <https://github.com/seokhyung-lee/msd-magic-state-prep-cycle-simulation>.
- [43] P. Sarvepalli and R. Raussendorf, Efficient decoding of topological color codes, *Phys. Rev. A* **85**, 022317 (2012).
- [44] N. Delfosse, Decoding color codes by projection onto surface codes, *Phys. Rev. A* **89**, 012317 (2014).
- [45] N. Maskara, A. Kubica, and T. Jochym-O’Connor, Advantages of versatile neural-network decoding for topological codes, *Phys. Rev. A* **99**, 052351 (2019).
- [46] 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).
- [47] N. Delfosse and N. H. Nickerson, Almost-linear time decoding algorithm for topological codes, *Quantum* **5**, 595 (2021).
- [48] E. Sabo, A. B. Alosious, and K. R. Brown, Trellis decoding for qudit stabilizer codes and its application to qubit topological codes, *ArXiv:2106.08251*.
- [49] A. Kubica and N. Delfosse, Efficient color code decoders in $d \geq 2$ dimensions from toric code decoders, *Quantum* **7**, 929 (2023).
- [50] J. Zhang, Y.-C. Wu, and G.-P. Guo, Facilitating practical fault-tolerant quantum computing based on color codes, *Phys. Rev. Res.* **6**, 033086 (2024).
- [51] Y. Takada, Y. Takeuchi, and K. Fujii, Ising model formulation for highly accurate topological color codes decoding, *Phys. Rev. Res.* **6**, 013092 (2024).
- [52] L. Berent, L. Burgholzer, P.-J. H. S. Derks, J. Eisert, and R. Wille, Decoding quantum color codes with MaxSAT, *Quantum* **8**, 1506 (2024).
- [53] C. Gidney, Stability experiments: The overlooked dual of memory experiments, *Quantum* **6**, 786 (2022).
- [54] J. Ramette, J. Sinclair, N. P. Breuckmann, and V. Vuletić, Fault-tolerant connection of error-corrected qubits with noisy links, *npj Quantum Inf.* **10**, 58 (2024).

- [55] D. Litinski and F. v. Oppen, Lattice surgery with a twist: Simplifying Clifford gates of surface codes, *Quantum* **2**, 62 (2018).
- [56] A. G. Fowler, A. M. Stephens, and P. Groszkowski, High-threshold universal quantum computation on the surface code, *Phys. Rev. A* **80**, 052312 (2009).
- [57] A. M. Stephens, Fault-tolerant thresholds for quantum error correction with the surface code, *Phys. Rev. A* **89**, 022321 (2014).
- [58] B. Heim, K. M. Svore, and M. B. Hastings, Optimal circuit-level decoding for surface codes, *ArXiv:1609.06373*.
- [59] O. Higgott, T. C. Bohdanowicz, A. Kubica, S. T. Flammia, and E. T. Campbell, Improved decoding of circuit noise and fragile boundaries of tailored surface codes, *Phys. Rev. X* **13**, 031007 (2023).
- [60] Note that these thresholds are *scaling* thresholds, not *crossing* thresholds. That is, they correspond to the parameter p_{th} in the subthreshold ansatz given by Eq. (11), rather than the intersection points of curves for different code distances. These two thresholds generally differ, as the ansatz breaks down near threshold. Although the crossing threshold is commonly used as a standard definition in the literature, the scaling threshold is more relevant in the low-error regime as we consider in our analysis.
- [61] J. Zhang, Y.-C. Wu, and G.-P. Guo, Facilitating practical fault-tolerant quantum computing based on color codes, *Phys. Rev. Res.* **6**, 033086 (2024).
- [62] H. Zhou, C. Zhao, M. Cain, D. Bluvstein, C. Duckering, H.-Y. Hu, S.-T. Wang, A. Kubica, and M. D. Lukin, Algorithmic fault tolerance for fast quantum computing, *ArXiv:2406.17653*.
- [63] N. Fazio, M. Webster, and Z. Cai, Low-overhead magic state circuits with transversal CNOTs, *ArXiv:2501.10291*.
- [64] S.-H. Lee, Color-code-msd-data, 2025, <https://github.com/seokhyung-lee/color-code-msd-data>.
- [65] S.-H. Lee, Color-code-stim, 2024, <https://github.com/seokhyung-lee/color-code-stim>.
- [66] Assuming $d_X \geq d_Z$, there exist $N_{\text{diag}} := \binom{l}{r}$ shortest paths for diagonal error strings, where $l := d_X/2 - 1$ and $r := (d_X - d_Z)/2$. For vertical error strings (causing $\bar{X}_1$ or $\bar{X}_2$ errors), there are approximately $N_v := d_Z \cdot 2^{d_X/2-1}$ shortest paths. Comparing these two values, $N_{\text{diag}}/N_v < 0.1$ for $d_X \geq 8$ or $d_Z \geq 6$, and $N_{\text{diag}}/N_v < 0.05$ for $d_X \geq 14$ or $d_Z \geq 8$. In general, we obtain $N_{\text{diag}}/N_v < 2/[d_Z\sqrt{\pi(d_X-2)}]$ from the inequality $\binom{l}{r} \leq \binom{l}{l/2} < \sqrt{2/(\pi l)}2^l$. Hence, although diagonal and vertical error strings have the same minimum weight, they differ significantly in their frequency of occurrence, assuming that circuit-level noise acts at similar levels in both directions.
- [67] C. Gidney, Stim: A fast stabilizer circuit simulator, *Quantum* **5**, 497 (2021).
- [68] O. Higgott, PyMatching: A python package for decoding quantum codes with minimum-weight perfect matching, *ACM Trans. Quantum Comput.* **3**, 16 (2022).
- [69] A. G. Fowler and C. Gidney, Low overhead quantum computation using lattice surgery, *ArXiv:1808.06709*.
- [70] B. Efron and G. Gong, A leisurely look at the bootstrap, the jackknife, and cross-validation, *Am. Stat.* **37**, 36 (1983).
- [71] P. Virtanen, *et al.*, SciPy 1.0 Contributors, SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python, *Nat. Methods* **17**, 261 (2020).