

Composite-dimensional topological codes with boundaries and defects

Mohamad Mousa ^{*}

Department of Physics, Purdue University, West Lafayette, Indiana 47907, USA

Amit Jamadagni  and Eugene Dumitrescu [†]

Computational Sciences and Engineering Division, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831, USA



(Received 8 October 2025; revised 8 April 2026; accepted 22 June 2026; published 16 July 2026)

We introduce new algorithms and provide example constructions of stabilizer models for the gapped boundaries, domain walls, and 0D defects of Abelian composite-dimensional twisted quantum doubles. Using the physically intuitive concept of condensation, our algorithm explicitly describes how to construct the boundary and domain-wall stabilizers starting from the bulk model. This extends the utility of Pauli stabilizer models in describing nontranslationally invariant topological orders with gapped boundaries. To highlight this utility, we provide a series of examples, including a new family of quantum error-correcting codes where the double of $\mathbb{Z}_4$ is coupled to instances of the double semion (DS) phase. We discuss the codes' utility in the burgeoning area of quantum error correction with an emphasis on the interplay between deconfined anyons, logical operators, error rates, and decoding. We also augment our construction, built using algorithmic tools to describe the properties of explicit stabilizer layouts at the microscopic lattice level, with dimensional counting arguments and macroscopic-level constructions building on pants decompositions. The latter outlines how such codes' representation and design can be automated. Our results are validated by a series of error-correcting threshold calculations comparing our codes' performance with that of standard surface codes. To do so, we introduce a composite-dimensional belief-propagation decoder with ordered statistics that utilizes combination sweeps. Going beyond our worked-out examples, we expect our explicit step-by-step algorithms to pave the path for higher-dimensional codes to be discovered and implemented in near-future architectures that take advantage of various hardware platforms.

DOI: [10.1103/qr5v-1rg6](https://doi.org/10.1103/qr5v-1rg6)

I. INTRODUCTION

Quantum error correction distills many noisy physical quantum degrees of freedom into a smaller number of logical quantum degrees of freedom with exponentially suppressed error rates [1–3]. Further, topological fault-tolerant quantum computation utilizes topological order to encode and, with low error rates, manipulate logical degrees of freedom encoded in a nonlocal logical subspace [4,5]. Recently, a number of qubit experiments have demonstrated how topological codes' logical error rates are suppressed with increasing system size. Recent examples include 31 superconducting qubits [6], 49 qubits in a neutral-atom array [7], 81 neutral-atom platform [8], as well as many others [9–16].

As the number of qubits increases, it is of growing importance to determine which quantum error-correcting codes are best suited to maximize performance on different quantum

hardware architectures. In this work, we provide a direction to help make this determination. We work within the context of the family of topological codes based on Abelian twisted quantum doubles [4,17], which encompasses generalizations to the well-studied surface code. Here, we explore the interplay of three generalizations to the surface code: local Hilbert space dimension, boundaries and defects, and spatial anisotropy to provide new routes towards error correction.

One natural generalization of the qubit-based surface code is to leverage higher-dimensional $d > 2$ local Hilbert spaces as a resource (qudits) and use higher-dimensional codes. This route is promising because qudits have shown promise in accelerating the simulation of high-dimensional quantum systems [18,19], compiling algorithms [20–22], magic-state distillation [23,24], and quantum control [25,26]. As a result, experiments have been developed to realize qudit systems using architectures based on donor spins in silicon [27], ultracold atoms and molecules [28,29], optical photons [30,31], superconducting circuits [32–35], trapped ions [36,37], and vacancy centers [38,39].

As we explore in this work, higher-dimensional composite local Hilbert spaces provide a route to simultaneously utilize multiple topological phases in a single quantum error-correcting code. This direction is less explored, since translationally invariant systems are simpler to study [40]. Translationally invariant (generalized) Pauli stabilizer codes

^{*}Contact author: mmousa@purdue.edu

[†]Contact author: dumitrescu@ornl.gov

in prime dimensions were shown to be equivalent to copies of higher-dimensional toric codes $D(\mathbb{Z}_d)$ [41]. Additionally, for composite-dimension qudits, translationally invariant Pauli stabilizer models can realize the bulk of Abelian twisted quantum doubles [42]. These works of Refs. [41–44], while covering all qudit dimensions, did not address the case of translationally noninvariant models. Here, we explore spatially anisotropic codes with an additional emphasis on the richness that composite-dimension qudits offer. This spatial anisotropy is known to enable new properties like universality with non-Abelian islands [45].

However, constructing spatially anisotropic codes requires a careful treatment of the boundaries that separate distinct topological phases [46–55]. This was absent from previous works that considered Pauli stabilizer models of Abelian twisted quantum doubles, for example [42]. In particular, the existing abstract constructions [47,50] of these boundaries and defects may not be experimentally efficient in certain cases. For example, while the bulk of each phase separately has a representation in terms of familiar Pauli stabilizer models, it is unclear if the gapped boundaries and defects can also have a Pauli stabilizer description. Answering this question has practical consequences for implementation as well as for assessing the computing power of these codes. A simple systematic method to construct these boundaries and 0D defects within the Pauli stabilizer framework is still lacking.

In this work, we show that boundaries, domain walls, and point-like defects of Abelian twisted quantum doubles can be written in terms of generalized Pauli stabilizers, thus generalizing the results of Ref. [42]. By providing an algorithm to construct them starting from the bulk theory, we prove this *constructively*. Our construction relies on local condensation, so it does not introduce new degrees of freedom [54,55], non-Pauli stabilizers [47,49,50], nor rely on lattice defects [56]. It also stays within the framework of topological phases, without reverting to gauging symmetries of their underlying symmetry-protected phases [54]. It also automatically produces topologically complete models, meaning that all logical operators are macroscopic [53,57]. Even when using other choices for the initial quantum double stabilizers, such as spatially disconnected stabilizers [53], our construction automatically produces models with topological order. As such, the resulting Hamiltonian for the boundary will also be described by local stabilizers. This construction offers both a simple, intuitive understanding of the boundaries and an efficient method that can automate boundary construction for various phases and lattices. Importantly, by relying on anyon condensation as opposed to consistency relations [47,50,58,59], it extends the systematic construction of Pauli stabilizers to also include the 0D defects [60] of Abelian twisted quantum doubles, which is, to our knowledge, unique to this work.

Our construction further facilitates the exploration of a new family of codes that comprises generalized holes, boundaries, and defects of higher-dimensional codes. For simplicity, we exemplify this new family of codes by studying the smallest composite number $d = p \times q$ where $p = q = 2$. This could be realized by four-dimensional qudits or, alternatively, in the near term, by grouping qubits in pairs. We find new qualitative properties for higher-dimensional codes, such as tunable error

biases, which can be exploited to simplify noise models [61]. Their syndrome extraction process can also exhibit advantages compared to the qubit case.

II. SUMMARY OF MAIN RESULTS

This section summarizes the main results and outlines our work. Readers primarily interested in explicit code constructions, decoding algorithms, and physical examples may focus on the background formalism in Sec. III and the new results in Sec. IV. Readers seeking a general and systematic framework for constructing gapped boundaries, domain walls, and defects within a Pauli stabilizer formalism may consult Sec. V. The global properties of the resulting codes, including ground-state degeneracy and logical operator structure, are analyzed in Sec. VI using both microscopic and macroscopic approaches.

Section III describes the background and quantum double and twisted quantum double formalisms used throughout. Specifically, we work with the double of $\mathbb{Z}_4$, $D(\mathbb{Z}_4)$, as a base model. We discuss periodic and open boundary conditions. In addition to the smooth and rough boundaries, we introduce an even boundary that is distinct and leads to different types of planar codes. Further, we use our boundary construction algorithm to derive a set of domain walls that generalize Bombin’s [56] $\mathbb{Z}_2$ twist. Section III B then describes the bulk and boundaries of the double-semion phase model, which may be obtained by boson condensing the parent $\mathbb{Z}_4$ model. Throughout, we provide examples of how different combinations of boundaries and domain walls (codimension-1 defects) realize logical codes. For example, codes of logical dimension $D = 4^M$ may be constructed via planar or spherical ($M = 0$), open boundary condition planar codes ($M = 1$), surface or toric ($M = 2$), or higher $2M$ -boundary surface or M -torus or other combinations involving twists ($M > 2$). Interestingly, even boundaries lead to qubit (rather than logical $d = 4$ -qudits) codes. Taken together, we have codes with logical dimensions $D = 4^M 2^{M'}$.

Building on these constructions and intuition, in Sec. IV we go beyond codes with a single bulk phase. The main tool for doing so, generalizing the notion of punctures, is the condensation of *local* DS patches (codimension-0 defects as opposed to global condensation [42]), from the parent $\mathbb{Z}_4$ phase. In this composite and nontranslationally invariant code, we find that $N + 1$ DS patches realize an N -logical qubit code. Combining this result with deformations of the parent $\mathbb{Z}_4$ model, we get a total composite logical dimension of $D = 4^M \times 2^{N+M'}$. We provide the logical Pauli operators that act on the patch codes’ qubit topological codespace.

After a brief review of stabilizer codes and the noise models for qudit-based codes, we then detail how such codes can be decoded. This involves a variety of modified minimum-weight-perfect-matching decoder (MWPM), an integer linear programming decoder (ILP), and a belief propagation with ordered statistics and combination sweep decoder (BP-OSD-CS). We then use these decoders to find code capacity thresholds for the five codes considered.

After describing our new codes and their decoding strategies, Sec. V presents rigorous constructions and the main systematic methodology that led to our codes’ discovery.

Specifically, we provide Algorithm 2, which concretely illustrates how to gauge from a parent code—meaning to insert boundaries, domain walls, and 0D defects into Abelian twisted quantum doubles—to the new composite codes. We emphasize the concept of anyon condensation, which is used to conceptually understand high-level code properties, such as their interfaces. In addition, condensation results in certain code-gauging strategies. These tools can be used to alter codes from the bottom up (i.e., microscopic $\rightarrow$ macroscopic) within a generalized Pauli stabilizer formalism. Importantly, we carefully examine how condensation depends on both pre-existing bulk and boundary degrees of freedom.

To provide intuition and to see the algorithm in action, a series of well-known codes and their boundaries are rederived within this formalism in Sec. VC. To rigorously describe our DS patch codes, we then systematically constructed boundaries and domain walls in Secs. VD and VE respectively. As an additional illustrative example, highlighting further possible modifications, the generalization from the $\mathbb{Z}_2$ twist to $\mathbb{Z}_4$ is presented in Example V.4. An important example of the domain wall between $\mathbb{Z}_4$ and DS is treated heuristically in Example V.5 and, using the full machinery of our construction, in Appendix A. Further examples, such as the construction of 0D defects, are detailed in Appendix B.

To insert these topological components into different topological codes under a unified framework, Sec. VI computes the ground-state degeneracies in the presence of *multiple* boundaries, domain walls, and defects. Based on microscopic commuting projector stabilizer models, these calculations rely on counting arguments that take into account the order of old and new constraints and the changing dependencies on one another. They are central to deriving the logical dimension of our family of new quantum-error-correcting codes. Ultimately, the properties of the topological codes do not depend on the exact layout, which enables us to then analyze the codes at scale.

In Sec. VIB, a generic code description that is agnostic to concrete stabilizer models is used to compute the codespace dimension and provide intuition about the logical operators of different codes. This is carried out using the pants decomposition of any genus- g 2D orientable manifold with the topological components of the codes. This confirms the results of Sec. VI and illustrates how interested readers may further modify or generalize our codes as desired. Finally, Sec. VII concludes by discussing this work's implications, open questions, and future research directions.

III. BACKGROUND

Kitaev's quantum double models [4,62] are both physical realizations of the mathematical quantum-double construction by Drinfeld [63] as well as concrete lattice instances of Dijkgraaf-Witten topological quantum field theories (TQFTs) [64]. In addition, they underpin a family of quantum-error-correcting codes (QECCs) [4].

The bulk of a TQFT A is described by a Hamiltonian, which we denote as H_A . Generically, the TQFT is spatially supported on a surface $\mathcal{M}_A$, which may have a boundary $\partial\mathcal{M}_A$. In addition to the bulk interactions, the boundary is described by $H_{\partial A}$ living on $\partial\mathcal{M}_A$. Further, two distinct TQFTs A, B supported on surfaces $\partial\mathcal{M}_A$ and $\partial\mathcal{M}_B$ must have a do-

main wall between them if $\partial\mathcal{M}_A \cap \partial\mathcal{M}_B \neq \emptyset$. This domain wall is described by a Hamiltonian $H_{A \cap B}$ which lives on the common boundary of the two TQFTs $\partial\mathcal{M}_A \cap \partial\mathcal{M}_B$. Altogether, the composite system is described by a Hamiltonian,

$$H_{\text{tot}} = \sum_{\mathcal{M}_A} H_A + \sum_{\mathcal{M}_B} H_B + \sum_{\partial\mathcal{M}_A} H_{\partial A} + \sum_{\partial\mathcal{M}_B} H_{\partial B} + \sum_{\partial\mathcal{M}_A \cap \partial\mathcal{M}_B} H_{A \cap B}. \quad (1)$$

It is evident that the boundaries of a TQFT are trivial specific examples of the domain walls between this TQFT and the vacuum TQFT. Conversely, gapped domain walls between two topologically nontrivial phases are closely related to their boundaries with the vacuum. In fact, as is discussed in Sec. VE, a gapped domain wall between two topological phases A and B is precisely the gapped boundary of the topological order $A \otimes B$ [47,65].

In the lattice picture, the models are defined on a 2D orientable surface $\mathcal{M}$, possibly containing a spatial boundary $\partial\mathcal{M}$. The surface is then triangulated into a lattice Γ consisting of vertices V , edges E , and plaquettes P (faces). Each spatial boundary $\partial\mathcal{M}$ has to be associated with a consistent boundary theory that is compatible with the bulk theory [4,48–51].

In (twisted) quantum double models [4,17], a qudit, taking values in a finite group G , is associated with each edge of the lattice $e \in E$ [66]. These qudits live in a Hilbert space $\mathcal{H}_e$ spanned by the orthonormal basis generated by the group elements $\{|g\rangle_e : g \in G\}$. The total Hilbert space is then the tensor product of all the edge Hilbert spaces $\mathcal{H}_{\text{tot}} = \bigotimes_{e \in E} \mathcal{H}_e$. The case where the group is $\mathbb{Z}_2$ corresponds to the toric code, where a qubit is associated with each edge [4]. Since the basic results in this paper can also be shown in Abelian groups, we focus on the case of the cyclic group $\mathbb{Z}_4$, which has a simple implementation using two qubits or $d = 4$ qudits (hereafter referred to as qudits). In addition, the Hamiltonian terms are built from the generalized Pauli stabilizers [67], defined below.

Remark III.1. With appropriate definitions, all physical properties of the model do not depend on the microscopic details of the lattice. Any triangulation of the surface $\mathcal{M}$ gives rise to the same topological phase [4,17]. For simplicity, we follow the most prominent conventions, as they appear in the literature, and work with a square lattice.

In Sec. III A, we review the $\mathbb{Z}_4$ surface code, which is the generalization of the surface code for the case of qudits [4]. In addition, by deriving the gapped boundaries and domain walls, we use $\mathbb{Z}_4$ as a tool to review defects and edges. Afterward, in Sec. III B, we review a double-semion code construction, which is a twisted quantum-double version of the $\mathbb{Z}_2$ toric code [17], and is also closely related to the $\mathbb{Z}_4$ code [42].

A. The $D(\mathbb{Z}_4)$ quantum double

In higher-dimensional Hilbert spaces, the stabilizers of codes are built out of the generalized Pauli X and Z operators [67], which are also called the shift and clock operators. In the rest of the paper, we simply refer to them as Pauli operators.

For the case of $D(\mathbb{Z}_N)$, they are defined as

$$X = \sum_{0 \leq j \leq N-1} |j+1\rangle\langle j|, \quad Z = \sum_{0 \leq j \leq N-1} e^{2\pi i j/N} |j\rangle\langle j|. \quad (2)$$

For $N = 4$, the explicit matrix representations at a given site are

$$X = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -i \end{pmatrix}. \quad (3)$$

They obey the following relations:

$$X^4 = Z^4 = 1, \quad ZX = iXZ, \quad ZX^\dagger = -iX^\dagger Z. \quad (4)$$

Using these operators, we define for every vertex v and plaquette p the star operators $A(v)$ and the plaquette operators $B(p)$. We also introduce shorthand pictorial symbols for later use.

1. $D(\mathbb{Z}_4)$ bulk

The bulk Hamiltonian of $\mathbb{Z}_4$ is then

$$H_{\mathbb{Z}_4} = - \sum_{v \in V} (A(v) + \text{H.c.}) - \sum_{p \in F} (B(p) + \text{H.c.}), \quad (5)$$

where H.c. denotes the Hermitian conjugate, and the sum is over vertices and plaquettes belonging to the bulk region,

$$A(v) = \begin{array}{c} X \\ | \\ X^\dagger \end{array} = \blacklozenge, \quad B(p) = \begin{array}{c} Z^\dagger \\ \square \\ Z \end{array} = \blacksquare. \quad (6)$$

Since all the terms of the Hamiltonian commute, they define a stabilizer code generated by all the star and plaquette operators,

$$\mathcal{S}_{\mathbb{Z}_4} = \langle A, B \rangle. \quad (7)$$

Excitations of the bulk. There are 16 local excitations (including the vacuum) generated by the different fusions of the pure electric charges $\{1, e, e^2, e^3\}$ and the pure magnetic fluxes $\{1, m, m^2, m^3\}$ satisfying $e^4 = m^4 = 1$. These are gapped excitations from the Hamiltonian perspective, as they cost finite energy to create. Alternatively, in the stabilizer picture, they are either errors or logical operators, depending on whether they violate stabilizers or not, respectively. Lastly, from the TQFT perspective, they can be thought of as localized particles. In particular, they are called anyons as they can have exotic statistics which are neither fermionic nor bosonic [68]. We switch between all three pictures depending on the context. Here, we treat them as anyons for the moment, and denote the anyons of the quantum double $D(\mathbb{Z}_4)$ by $\mathcal{A}(\mathbb{Z}_4)$.

When two anyons come close together, they can be regarded as one emergent anyon; this process is called fusion [69]. In Abelian theories, fusion outcomes are deterministic, which means that the fusion of any two anyons results in a single new anyon [4, 70, 71]. Explicitly, we have

$$e^p m^q \otimes e^r m^s = e^{p+r} m^{q+s} \quad p, q, r, s \in \{0, 1, 2, 3\}. \quad (8)$$

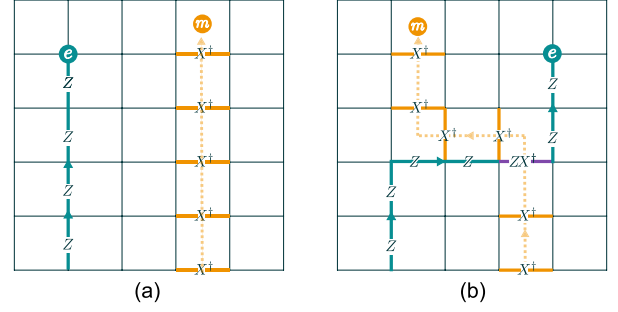


FIG. 1. (a) Strings for creating e and m anyons in $\mathbb{Z}_4$. (b) Braiding and e and an m anyons, the two strings overlap at the purple edge.

Equivalently, we can also write this using the fusion symbols,

$$a \otimes b = \sum_m N_{ab}^m m \quad a, b, m \in \mathcal{A}(\mathbb{Z}_4), \quad (9)$$

where the fusion symbol $N_{ab}^m = \delta_{a+b, m}$. The fusion of anyons in $\mathbb{Z}_4$ thus forms a group $\mathcal{F} = \mathbb{Z}_4 \times \mathbb{Z}_4$.

In addition to fusion rules, anyons carry an intrinsic topological spin given by the diagonal $\mathcal{T}$ matrix,

$$\mathcal{T}(e^p m^q) = i^{pq}, \quad (10)$$

where we used the shorthand $\mathcal{T}(a) = \mathcal{T}_{aa}$ since it is a diagonal matrix. We can then see that all the pure electric charges, $\{1, e, e^2, e^3\}$, the pure magnetic fluxes, $\{1, m, m^2, m^3\}$, and the dyon $e^2 m^2$ are bosons.

As is common in topological phases, bosonic excitations (anyons with topological spin = 1) can still braid nontrivially with other excitations. The braiding of two anyons a and b around each other produces a $U(1)$ phase denoted here by $B(a, b)$. In Abelian theories, this phase can be directly computed from the $\mathcal{T}$ matrix as [4]

$$B(a, b) = \frac{\mathcal{T}(a \otimes b)}{\mathcal{T}(a) \mathcal{T}(b)}. \quad (11)$$

In Abelian theories, braiding is thus completely determined by the topological spin of the generators e and m , which is i in this case. Applying Eqs. (9)–(11), we have

$$B(e^p m^q, e^r m^s) = i^{ps+qr}. \quad (12)$$

The braiding statistics can also be read from the microscopic string operators that create and transport the anyons, as shown in Fig. 1. Note that strings with the opposite direction have the inverse operators. This means that if the m string goes down or left, it acts with X , and if the e string goes left or down, it acts with $Z^\dagger$. This follows from the implicit orientation of the lattice fixed through the definition of the Hamiltonian terms (5). All other anyonic braiding statistics can be deduced using these two generators.

The braiding statistics of anyons play a crucial role in defining the $\mathbb{Z}_4$ models as surface, toric, or other related quantum-error-correcting codes. Closed Wilson loops of anyons commute with all the Hamiltonian terms and thus define or implement logical operators acting within the codespace [72]. The caveat is that if two loops of anyons braid nontrivially, then their actions are not independent. In

addition, loops that are deformed into one another, via the action of the terms in the Hamiltonian, are also equivalent.

2. $D(\mathbb{Z}_4)$ boundaries

The original model for fault-tolerant quantum computing by anyons was defined on a torus [4], which is a closed manifold, so there was no need to discuss what occurs at the boundaries. Introducing smooth and rough boundaries, a seminal follow-on work [73] defined $D(\mathbb{Z}_2)$ surface (planar) codes, which have become ubiquitous QECCs.

In this section, we focus on the generalization of the $\mathbb{Z}_2$ case to the gapped boundaries of TQFTs described by (twisted) quantum doubles based on the $D(\mathbb{Z}_4)$. Note that our notion of boundaries, thus far, corresponds to a boundary between a nontrivial TQFT and the topologically trivial vacuum, where no anyons exist. This means that at the bulk-to-boundary map, a bulk deconfined anyon (an anyon that does not need extra energy to propagate) may only condense (be identified with the vacuum) or become confined depending on the details of the boundary. The confined anyons are still gapped, and they live on the $(1+1)$ D boundary. In fact, the $(1+1)$ D gapped boundaries of topological phases are completely determined by the anyons they condense [65,74]. These anyons for a certain boundary form a subgroup of the fusion group, which is called a Lagrangian subgroup. This subgroup $\mathcal{L}$ satisfies three properties [71,74,75],

- (1) All anyons $a \in \mathcal{L}$ are bosons, $\mathcal{T}(a) = 1$.
- (2) All anyons $a, b \in \mathcal{L}$ braid trivially with each other $B(a, b) = 1$.
- (3) For any anyon $c \notin \mathcal{L}$, there exists an anyon $a' \in \mathcal{L}$ such that $B(a', c) \neq 1$.

Intuitively, the Lagrangian subgroup describes a group of anyons that can be consistently identified with the vacuum. In addition, identifying this group with the vacuum will confine all other anyons outside the subgroup [76]. From the spin and braiding statistics of the $\mathbb{Z}_4$ model, we can find three boundaries,

$$\begin{aligned} \mathcal{L}_{\text{smooth}} &= \{1, m, m^2, m^3\}, & \mathcal{L}_{\text{rough}} &= \{1, e, e^2, e^3\}, \\ \mathcal{L}_{\text{even}} &= \{1, e^2, m^2, e^2 m^2\}. \end{aligned} \quad (13)$$

The first two boundaries generalize the smooth and rough boundaries of the $\mathbb{Z}_2$ surface code to the $\mathbb{Z}_4$ case. The last boundary is a new boundary that condenses all anyons with even exponents.

Suppose the model lives on a square surface $\mathcal{M}$ with a boundary, $\partial\mathcal{M}$ Fig. 2. Then the Hamiltonian in Eq. (5) has to be changed to include boundary terms in order to realize these three boundaries,

$$\begin{aligned} H_{\mathbb{Z}_4, \text{Smooth}} &= H_{\mathbb{Z}_4} - \sum_{v \in \partial\mathcal{M}} (A(v)) + \text{H.c.}, \\ H_{\mathbb{Z}_4, \text{rough}} &= H_{\mathbb{Z}_4} - \sum_{p \in \partial\mathcal{M}} (B(p)) + \text{H.c.}, \\ H_{\mathbb{Z}_4, \text{Even}} &= H_{\mathbb{Z}_4} - \sum_{p, e \in \partial\mathcal{M}} (B^2(p) + X^2(e)) + \text{H.c.} \end{aligned} \quad (14)$$

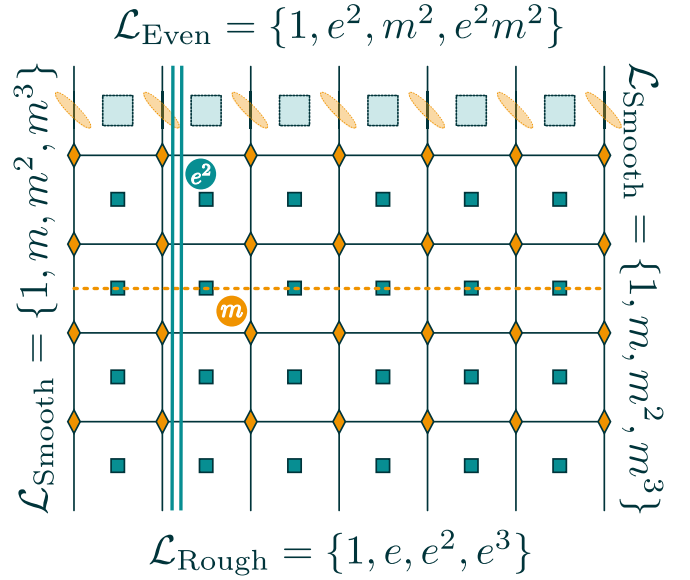


FIG. 2. A logical-qubit $\mathbb{Z}_4$ surface code composed of three different boundaries. The logical $\tilde{Z}$ operator is a vertical string operator, transporting an e^2 particle between the rough and even boundaries. The logical $\tilde{X}$ operator is a horizontal string operator transporting an m particle between the two smooth boundaries.

Explicitly, the three boundaries can be constructed using the stabilizers in

$$\begin{aligned} A'(v) &= \begin{array}{c} X \\ | \\ X^\dagger \end{array} = \text{diamond} \quad , \quad \begin{array}{c} X^\dagger \\ | \\ X \end{array} = \text{diamond} \\ B'(p) &= \begin{array}{c} z^\dagger \\ \text{U-shape} \\ z \end{array} = \text{square with } z^\dagger \quad , \quad \begin{array}{c} z^\dagger \\ \text{U-shape} \\ z \end{array} = \text{square with } z^\dagger \\ B'^2(p) &= \begin{array}{c} z^2 \\ \text{U-shape} \\ z^2 \end{array} = \text{square with } z^2 \quad , \quad \begin{array}{c} z^2 \\ \text{U-shape} \\ z^2 \end{array} = \text{square with } z^2 \\ X'^2(e) &= \begin{array}{c} | \\ X^2 \end{array} = \text{orange oval} \end{aligned} \quad (15)$$

We see that the boundary terms are derived from the bulk terms with possibly a higher power or restricted support. We formalize this notion in Sec. V, where a general algorithm is given for constructing all boundaries of Abelian twisted quantum doubles in terms of Pauli stabilizers. A $\mathbb{Z}_4$ surface code is then defined by a choice of these boundaries. We provide one example here, while other examples can be treated analogously using the methods described in Sec. VI.

Example III.1 ($\mathbb{Z}_4$ on a Surface). Consider a $\mathbb{Z}_4$ code on a square surface with two opposing (left/right) smooth boundaries as well as an even (top) and a rough (bottom) boundary, as shown in Fig. 2. We use the usual notation, where a solid line represents an e particle (pure electric charge) living on the

direct lattice, while dashed lines represent an m particle (pure magnetic flux) living on the dual lattice.

The ground state is twofold degenerate. We discuss methods for computing the ground-state degeneracy (GSD) for different surfaces and boundaries in Sec. VI. The logical operators can be chosen to be the vertical worldline of a e^2 that condenses on the even and the rough boundaries, together with the horizontal m -worldline string that condenses on the smooth boundaries. From the braiding equation, Eq. (12), we can see that $B(e^2, m) = -1$. From the fusion equation Eq. (9), we have the relation that $(e^2)^2 = 1$. The square of the *entire* m string condenses on the top even boundary, and it is therefore identified with the identity. Since the two logical operators anticommute and square to identity, they generate the single-qubit Pauli group, and we define the e^2 and m worldline strings as the $\tilde{Z}$ and $\tilde{X}$ logical operators, respectively. Note that, had the top boundary been rough, rather than even, the code would have defined a logical $d = 4$ qudit.

3. $D(\mathbb{Z}_4)$ domain walls and defects

In addition to boundaries with the vacuum, topological phases can have boundaries with other compatible phases or, in particular, with the same phase. Those boundaries are called domain walls to distinguish them from boundaries with the vacuum. They are also called defects, as they usually correspond to defects in the lattice [56]. We reserve the term defect for the 0D ends of a finite domain wall [60,77,78]. Here, we focus on invertible domain walls within $D(\mathbb{Z}_4)$. An invertible domain wall is a wall that can be followed by another wall, producing the transparent (trivial) domain wall that acts as the identity [69]. In $D(\mathbb{Z}_4)$, we have four invertible walls. Labeled by their action on the generators e and m , they are

$$\begin{aligned} W^1 &: (e, m) \mapsto (e, m), \\ W^{e \leftrightarrow e^{-1}} &: (e, m) \mapsto (e^{-1}, m^{-1}), \\ W^{e \leftrightarrow m} &: (e, m) \mapsto (m, e), \\ W^{e \leftrightarrow m^{-1}} &: (e, m) \mapsto (m^{-1}, e^{-1}). \end{aligned} \quad (16)$$

The duality domain wall $W^{e \leftrightarrow m}$ is also known in the literature as a twist [56]. The inverse domain wall $W^{e \leftrightarrow e^{-1}}$ was trivial in the $D(\mathbb{Z}_2)$ case. Domain walls can be brought together to form new domain walls. The new product is called stable if no local interaction near the domain wall can change it [79]. This is denoted by $W^i \otimes W^j = W^k$. In the $\mathbb{Z}_4$ case, the identity wall (transparent) naturally satisfies $W^1 \otimes W^i = W^i \otimes W^1 = W^i$. All walls square to the transparent wall, $W^i \otimes W^i = W^1$. Further, $W^{e \leftrightarrow e^{-1}} \otimes W^{e \leftrightarrow m} = W^{e \leftrightarrow m^{-1}}$, and $W^{e \leftrightarrow m} \otimes W^{e \leftrightarrow m^{-1}} = W^{e \leftrightarrow e^{-1}}$. It is then clear that the domain walls of $\mathbb{Z}_4$ form a

$\mathbb{Z}_2 \times \mathbb{Z}_2$ Abelian group under fusion. The Pauli stabilizers of these domain walls are given in Eqs. (17) and (18),

$$\begin{aligned} A^{\text{op}}(v) &= \begin{array}{c} X \\ X^\dagger \end{array} \begin{array}{c} X^\dagger \\ X \end{array} = \text{red diamond} \\ B^{\text{op}}(p) &= \begin{array}{c} Z^\dagger \\ Z \end{array} \begin{array}{c} Z \\ Z^\dagger \end{array} = \text{red diamond} \\ X^2 X^2(e) &= \frac{X^2}{X^2} = \text{blue diamond} \\ D^{(1)}(p) &= \begin{array}{c} Z^\dagger \\ Z \end{array} \begin{array}{c} Z^\dagger \\ Z \end{array} = \text{blue diamond} \\ D^{(2)}(p) &= \begin{array}{c} Z^\dagger \\ Z \end{array} \begin{array}{c} Z^\dagger \\ Z \end{array} = \text{blue diamond} \\ E^{(1)}(p) &= \begin{array}{c} Z^\dagger \\ Z \end{array} \begin{array}{c} Z^\dagger \\ Z \end{array} = \text{blue diamond} \\ E^{(2)}(p) &= \begin{array}{c} Z^\dagger \\ Z \end{array} \begin{array}{c} Z^\dagger \\ Z \end{array} = \text{blue diamond} \end{aligned} \quad (17)$$

$$\begin{aligned} T(v, p) &= \begin{array}{c} X^\dagger \\ Z^\dagger \end{array} \begin{array}{c} X^\dagger \\ Z \end{array} = \text{purple diamond} \\ XZ(e) &= \begin{array}{c} X^\dagger \\ X \end{array} \begin{array}{c} Z^\dagger \\ Z \end{array} = \text{purple diamond} \\ U(v) &= \begin{array}{c} X^\dagger \\ X^\dagger \end{array} \begin{array}{c} X^\dagger \\ X^\dagger \end{array} = \text{purple diamond} \\ Q(p) &= \begin{array}{c} Z^\dagger \\ XZ \end{array} \begin{array}{c} Z^\dagger \\ XZ \end{array} = \text{purple diamond} \\ T^{\text{op}}(v, p) &= \begin{array}{c} X^\dagger \\ Z^\dagger \end{array} \begin{array}{c} X^\dagger \\ Z \end{array} = \text{purple diamond} \\ XZ^\dagger(e) &= \begin{array}{c} X^\dagger \\ X^\dagger \end{array} \begin{array}{c} Z^\dagger \\ Z \end{array} = \text{purple diamond} \\ U^{\text{op}}(v) &= \begin{array}{c} X^\dagger \\ X^\dagger \end{array} \begin{array}{c} X^\dagger \\ X^\dagger \end{array} = \text{purple diamond} \\ Q^{\text{op}}(p) &= \begin{array}{c} Z^\dagger \\ X^\dagger Z \end{array} \begin{array}{c} Z^\dagger \\ X^\dagger Z \end{array} = \text{purple diamond} \end{aligned} \quad (18)$$

The exact layout of the stabilizers implementing the four domain walls is shown in Fig. 3. At the end of a finite domain wall, a 0D defect lives. These defects can be used for quantum computing in their own right [49,50,56,80–86]. Section V discusses how domain walls, as well as 0D defects, can be constructed systematically from bulk stabilizers. Note that the domain walls in Fig. 3 realize their mappings in any $\mathbb{Z}_n$ for appropriate generalized Pauli matrices, Eq. (3) (but they do not necessarily exhaust all the invertible domain walls for $\mathbb{Z}_n$ with $n > 4$).

B. Double semion twisted quantum double

We are interested in the error-correcting properties of the $\mathbb{Z}_4$ phase and its closely related phases. In this section, we review the doubled semion (DS) phase, which is closely related to $\mathbb{Z}_4$ as we discuss later in Sec. V. The doubled semion (DS) phase is an Abelian topological-order first realized as a string-net model [87]. It is a twisted $\mathbb{Z}_2$ gauge theory. It is also the simplest twisted quantum double, which is a generalization of the Kitaev quantum double [17]. All twisted quantum doubles with Abelian orders can be described in terms of Pauli stabilizers [42].

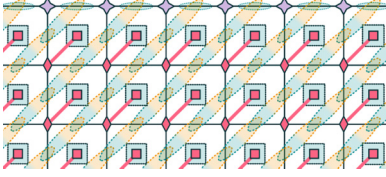


FIG. 5. Stabilizers for the unique boundary of the DS phase. At this boundary, only the boson b condenses.

From the DS's anyonic braiding statistics, Eq. (24), we see that condensing the b boson confines all other nontrivial anyons at the boundary. From the correspondence of Lagrangian subgroups and boundaries, DS has only one boundary.

$$H_{\text{DS, Boundary}} = H_{\text{DS}} - \sum_{v, e \in \partial \mathcal{M}} (A^2(v) + Z^2(e)). \quad (27)$$

As before, the boundary terms are inherited from the bulk terms. The stabilizers of the boundaries are

$$A^2(v) = \begin{array}{c} \text{---} \\ | \\ \text{---} \\ | \\ \text{---} \end{array} = \text{---} \diamond \text{---}, \quad Z^2(e) = \text{---} \text{---} = \text{---} \text{---} \quad (28)$$

Figure 5 shows the exact layout of the stabilizers at this unique boundary of the DS phase.

This means that, in the absence of any punctures, any DS surface code has $\text{GSD} = 1$. On a torus, however, the model will have $\text{GSD} = 4$ just like the toric code. In addition, the only invertible domain wall between two DS phases is the transparent one.

IV. COMPOSITE TOPOLOGICAL CODES

In this section, we present an investigation of the utility of using composite-dimensional qudit codes exemplified by the $D(\mathbb{Z}_4)$ surface code, the related even code introduced in Sec. III A, and the DS code introduced in Ref. [42] and reviewed in Sec. III B. In addition, we also discuss spatially anisotropic topological codes (nontranslationally invariant codes). These are exemplified by a hybrid DS- $\mathbb{Z}_4$ patch code introduced below in Sec. IV D 4. We discuss the construction of the codes from Pauli stabilizers, their boundary conditions, and their logical operators. Finally, we present numerical simulations to determine their thresholds and logical error rates subject to different noise models.

The $\mathbb{Z}_2$ surface code had only two types of boundaries: smooth and rough [73] and only one type of nontrivial invertible domain wall, namely the one implementing the $e \leftrightarrow m$ duality. With these topological properties, many constructions that are variants of the toric and surface codes have been proposed. These constructions include, but are not limited to, adding punctures to increase the number of qubits in the $\mathbb{Z}_2$ surface code [3,90,91], and adding twists that affect the GSD [56,88,92]. Although these constructions had diverse properties, they were limited by the possibilities afforded by the dimensionality and group-theoretic properties of $\mathbb{Z}_2$. A similar structure is also common for the slightly higher-dimensional qutrits [21,93–95] and, in fact, all prime-dimensional qudits.

As reviewed in Secs. III A and III B, two new properties are unlocked for the $\mathbb{Z}_4$ surface code: the existence of a new boundary (the even boundary), and the existence of child phases accessible through condensation (the DS phase). This is in addition to the new domain walls introduced in Eq. (16). The details of the methodological construction of these topological components are deferred to Sec. V D. In this section, we focus on four main codes: (i) the $\mathbb{Z}_4$ planar code, (ii) the $\mathbb{Z}_4$ even code, (iii) the DS code, and (iv) the DS- $\mathbb{Z}_4$ hybrid patch code. We begin with an introduction to stabilizer-code metrics.

A. Stabilizer codes

Assume we have a certain Hilbert space $\mathcal{H}$ with an associated group of linear operators $\mathcal{G}$. A stabilizer code $\mathcal{S}$ is any Abelian subgroup of operators $\mathcal{S} \subset \mathcal{G}$. For example, $\mathcal{S} = \langle A_v, B_p \rangle$ for the toric code, Eq. (6) [2]. The code subspace $\mathcal{C}$ is a subset of vectors (codewords) in the Hilbert space that are fixed (stabilized) by the elements of $\mathcal{S}$,

$$\mathcal{C} = \{ |c\rangle \in \mathcal{H} \mid s |c\rangle = |c\rangle \forall s \in \mathcal{S} \}.$$

Usually, the stabilizer code is implemented using commuting Hamiltonian terms (with local terms in the case of topological codes). Such a Hamiltonian has the code subspace as elements of the ground subspace. The dimension of the code space $\dim(\mathcal{C})$ is the logical subspace used for quantum computing. Assuming the code is constructed from logical qubits (which is the case for qubit- and dim-4 qudits), the number $k \in \mathbb{N}$ is defined as

$$k = \log_2(\dim(\mathcal{C})).$$

The number of physical qubits used to implement the system is usually denoted by n . For commensurability, we continue to count qudit-based codes in terms of qubits, including necessary ancilla qubits. This offers a conservative estimate of the new codes, assuming that only native qubits are available. We quickly note that dim-4 qudits are actually easier to manufacture from qubits than dim-3 qutrits since they do not need an extra ancilla [95]. However, the total overhead requires more careful treatment.

Logical operators are a subset of operators $L \in \mathcal{G}$ that commute with the stabilizers $[L, s] = 0, \forall L \in \mathcal{L}, s \in \mathcal{S}$. Their action on the codespace is quotiented by the stabilizers, since the stabilizers' action is trivial on any codeword. The set of distinct logical operators is then $\tilde{\mathcal{L}} = \mathcal{L}/\mathcal{S}$. For any operator, its weight is the cardinality of its support set. This is simply the number of qudits on which it acts. Since in topological codes any operator can be multiplied by stabilizers that act trivially, the weight usually refers to the minimum weight within a coset. For a stabilizer code, the code distance is the minimum weight of all nontrivial logical operators and is denoted by [2],

$$d = \min_{l \in \tilde{\mathcal{L}}, l \neq S} (||l||).$$

The three numbers n, k , and d are important metrics for any code; the code is usually denoted by the triplet $[[n, k, d]]$ [2].

Many bounds exist for quantum-error-correcting codes; most of them are generalizations of known classical bounds

[96,97]. Importantly, the following bound applies for topological stabilizer codes on a Euclidean surface [98]:

$$kd^2 \leq cn,$$

for some constant c that depends on the weight of the stabilizers and the dimension of qudits constituting the system. This bound shows that the family of topological codes cannot simultaneously satisfy the criteria for a good quantum code [99],

$$k/n \geq c_1 \quad \text{and} \quad d/n \geq c_2, \quad (29)$$

for constants c_1 and c_2 . Different lattice geometries, such as hyperbolic surface codes or more general qLDPC codes, have been shown to evade this bound [100–102]. Even though Abelian topological codes were shown not to satisfy a good encoding rate, and cannot implement transversal universal quantum computing without either magic-state distillation or non-Abelian ingredients [45,103–110], their relative accessibility both theoretically and experimentally continues to drive efforts to implement quantum computing and quantum error-correction [9,11–14,16].

B. Noise model

Modeling noise is an important procedure for better designing quantum-error-correction codes with realistic noise resilience. It further affects the decoding strategies and the thresholds obtained from simulations. For the case of qubits, two standard error models are widely used. First, the pure bit-flip (or phase flip) Pauli error channel. For a given physical error rate p , this channel applies an X operator with probability p independently on each qubit. The pure X channel is often used as a zeroth-order test for codes obtaining the code capacity threshold. Second, the depolarizing noise model applies either X , Y , or Z with probability $p/3$ each [111–113].

For the case of qudits, we consider widely used generalizations to these two channels [114–116]. For simplicity, we express these channels in terms of $d = 4$ qudits. The generalization for a general dimension is straightforward. For a given physical error rate p , we have a pure X noise channel that acts as the identity with a probability of $(1 - p)$ and acts with either X , X^2 , or X^3 with probability $p/3$ each. A pure Z -noise channel can be defined in a similar manner. For the depolarizing noise, the channel again acts as the identity with probability $(1 - p)$, and acts as $X^a Z^b$ for $\{a, b\} \subset \{0, 1, 2, 3\}$ and $(a, b) \neq (0, 0)$ with probability $p/15$ each [114–116].

We note that this is a conservative noise model that should work well if the qudits are actually made out of qubits, since it treats all the powers of the generalized Pauli operators equally. If the experimental system has native qudits, one can expect that errors are further suppressed by their powers. For example, an excitation to the second excited state has a quadratically suppressed probability.

C. Decoding

There is another way of thinking about codes, which is more amenable to classical error correction. In a Pauli stabilizer code, the check matrix H contains the code's stabilizers in its row space. The symplectic form of H separates Z and X

terms as: $H = [H_z | H_x]$ and thus has a size $||S|| \times 2n$, where $||S||$ is the cardinality of the generators of the stabilizer group. A valid logical ground state g is a $2n$ -dimensional vector satisfying $H \cdot g = 0$.

Generically, the system is initialized in one of the ground states (or a superposition of them), and the check matrix is then repeatedly measured to detect errors. If a detectable error occurs, the erroneous state x does not satisfy some stabilizers $H \cdot x = s$. Here, s is called the syndrome. The job of a decoder for a stabilizer code is essentially to estimate x from s . This is done by finding a correction c such that $H \cdot (x + c) = 0$. The weight of an error is defined as the number of qudits it nontrivially acts on. For example, $e = [1 \ 0 \ 2 \ 0 \ \dots \ 0]^T$. This means there is a Z error on qudit 1 and a Z^2 error on qudit 3. Such an error has a weight equal to 2: $wt(e) = 2$. This is in line with the definition for qubits [2,117]. Ideally, all errors with a Hamming weight less than $t = \lfloor \frac{d-1}{2} \rfloor$ are correctable [96]. However, in practice, this may not hold for approximate decoders.

There are two major kinds of decoders. First, quantum maximum likelihood (QML) decoders estimate the most likely individual error x . This problem was shown to be NP-complete [118]. Their most famous algorithms include minimum-weight perfect matching (MWPM) [3,119–121], union-find [122], renormalization group [116,123,124], belief propagation [125,126], neural networks, and reinforcement learning [127–132]. These decoders work to estimate the single error that occurred, usually by assuming that the lower-weight errors are more probable. These are highly parallelizable algorithms that are standard for simulations and hardware implementations [130,133–137].

Second, degenerate quantum maximum likelihood (DQML) decoders attempt to solve the more useful question: what was the logical ground state before the error? That is, instead of estimating only the most likely error x , they estimate the most likely logical equivalence class of errors. This problem is harder than finding the most likely error, and it is #P-complete [138]. One prominent algorithm for these decoders is the BSV decoder, which is based on tensor networks [139–141].

There exist efficient packages for sampling qubit- and qudit-based circuits [142,143]. In addition, many implementations for different types of decoders are available [144–150]. However, all of these decoders either assume qubits, prime-dimensional qudits (or treat composite dimensions as Galois fields), or CSS codes. The case of dimension-4 qudits and non-CSS codes is less developed. In the following, we report three prominent types of decoders that we have developed.

1. MWPM-4 decoder

The most straightforward way to decode $\mathbb{Z}_4$ is to adapt MWPM [3,119–121] to qudits by splitting the problem into even and odd errors. The intuition is simply that even errors can be paired together $2 + 2 \equiv 0 \pmod{4}$ while odd errors need to be added similarly $1 + 3 \equiv 0 \pmod{4}$. We call this the MWPM-4 decoder. We illustrate with the case of the $\mathbb{Z}_4$ surface code with pure X noise as an example. Even syndromes arise from the X^2 errors, while odd syndromes arise from the X and X^3 errors. One can then split the syndrome

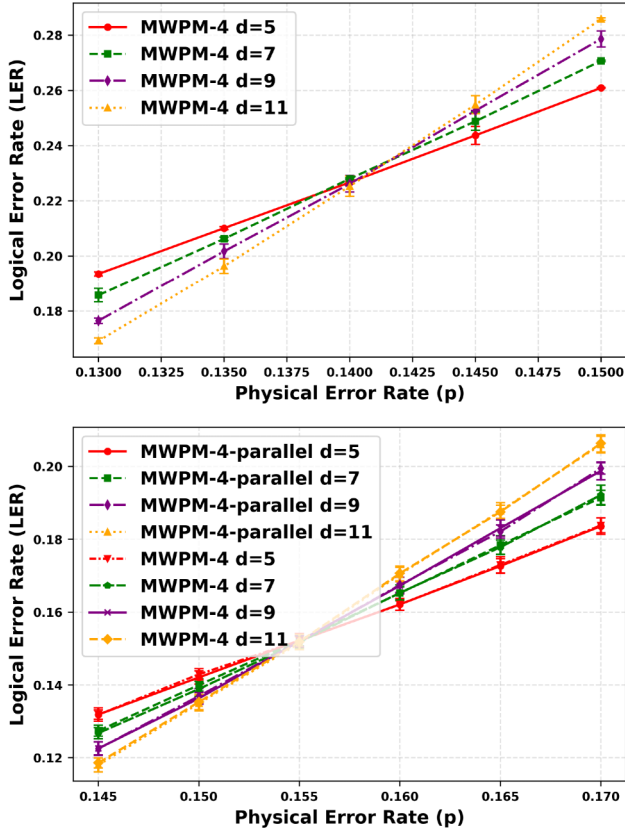


FIG. 6. (Top) Code capacity threshold of the $\mathbb{Z}_4$ surface code, Eq. (6) subject to pure X noise using the MWPM-4 decoder. The threshold is around 14%, which is comparable to the 15.25% of Ref. [114]. (Bottom) Code capacity threshold of the even code of Sec. IV D 2 subject to the same noise. The threshold is around 15.5%, which is higher than the surface code. The sequential and parallel forms of the decoder behave very similarly.

vector s into the odd part: $s_{\text{odd}} \equiv s \pmod{2}$, and the even part: $s_{\text{even}} \equiv \frac{s(s-1)(3-s)}{2} \pmod{2}$. We can then decode the odd syndromes together. Note that after the modulus operation, the decoder only sees a binary string of 1s and 0s, so it tries to match two X errors together [$1 + 1 \not\equiv 0 \pmod{4}$], for example. After decoding the odd part, there are even residuals. These even residuals, along with originally even syndromes, can be decoded together in one last step. Of course, this lost information degrades the decoder's performance and does not fully exploit the Hilbert space of the qudits.

Evidently, using this naive decoder, one finds a code capacity threshold of 14% for the $\mathbb{Z}_4$ surface code, as shown in Fig. 6, which is close to the RG decoders' result of around 15.25% in Ref. [114]. Both decoders are well below the hashing bound of 18.9%, which is expected to be saturated from statistical mechanical arguments [114, 115]. Even though they are not optimal, this choice of decoder is incredibly fast, as it only scales by a constant factor over standard MWPM decoders. To tackle more codes, such as the even code introduced in Sec. IV D 2, care must be taken when extracting syndromes and splitting them into their parity parts. As some stabilizers have even orders, they give even syndromes if the error is odd. This can be easily tackled by accounting

for the parity of the order of the stabilizers. Furthermore, instead of following the sequential steps described above, one can parallelize the procedure by decoding the s_{odd} and s_{even} parts simultaneously. The even part is guaranteed to succeed, while the odd part may leave residuals that require another round of decoding. The number of rounds is again the same; however, the second even round is much less dense than in the sequential version.

We found that the two methods give nearly identical results near the threshold for the even code. In the surface code, two odd checks can trigger an even syndrome if they touch the same odd check, preventing the parallel decoder from reaching the threshold. This problem does not affect the even decoder, as the logical operators square to identity. While MWPM-4 is already fast, we note that this caveat may be useful for certain hardware implementations. We finally note that the same scheme can readily work for any dimension that is a power of two (MWPM- $2^{\log_2(d)}$). As for the depolarizing noise channel discussed in Sec. IV B, the decoder struggles with it, giving thresholds of 17.25% for the $\mathbb{Z}_4$ surface code and 17.75% for the even code. While this decoder can handle CSS codes, it requires significant adaptations to handle non-CSS codes. Given its relatively suboptimal performance, we chose to pursue other decoding strategies.

2. ILP decoder

Certainly, MWPM-4 is nowhere near optimal despite its speed. To establish accurate code thresholds, we opted for integer linear programming (ILP) decoders. Evidently, the decoding can be cast as a direct formulation of the most-likely-error problem [151]. Given a check matrix H and a syndrome s , the problem is to minimize the Hamming weight of the estimated error x subject to $H \cdot x = 0$. These decoders are widely used for benchmarking other decoders [152–155]. These decoders are usually very slow as they find the optimal solution. They rely on software packages such as CPLEX or Gurobi that solve the problem with exact integers [156, 157]. The main advantage of these decoders, aside from their accuracy, is that they can handle any type of code or noise. They also handle the $\mathbb{Z}_4$ logic easily by using auxiliary variables to handle the modulus relations. These decoders are not to be confused with linear-programming (LP) decoders [158–162], which involve a relaxation step to continuous variables followed by an ordered statistics decoding (OSD) phase.

Using this decoder, we can find thresholds for all the codes discussed in this work. We give representative results here while summarizing all results in Sec. IV D. Using these more accurate decoders, we find a code-capacity threshold of 18% for the $\mathbb{Z}_4$ surface code and 17.5% for the even code, as shown in Fig. 7. The higher thresholds of the $\mathbb{Z}_4$ surface code compared to that of the $\mathbb{Z}_2$ surface code can be understood from the fact that there are different error channels, enabling the decoder to use different syndromes to better predict the correction. Alternatively, if we compare a qubit-based code with a qudit-based code such that each four-dimensional qudit is composed of two qubits, then for the same physical error rate, one qubit constituting the qudit code experiences almost half the physical error that an individual qubit experiences inside the qubit code. While we expect both codes to have

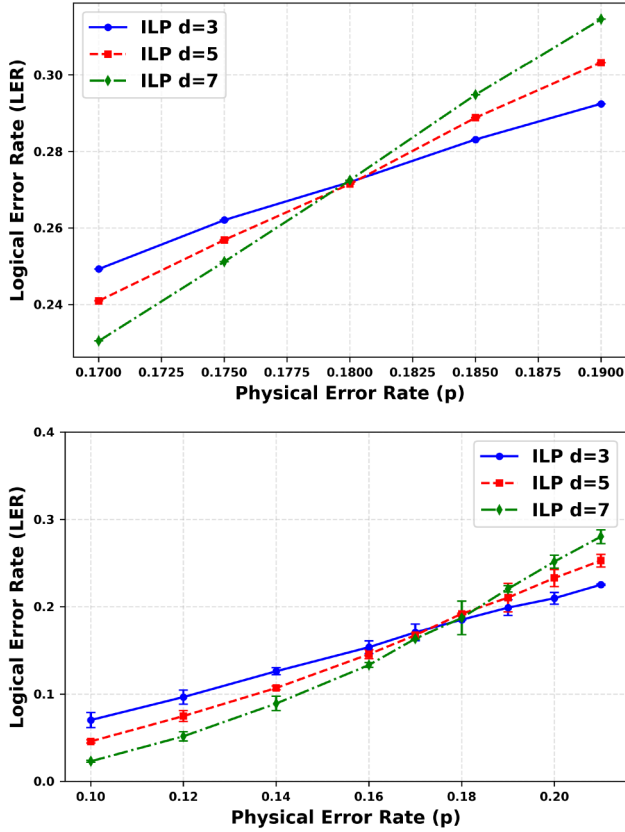


FIG. 7. (Top) Code capacity threshold of the $\mathbb{Z}_4$ surface code, Eq. (6), subject to pure X noise using the ILP decoder. The threshold is around 18%, which is comparable to the theoretical maximum 18.9% of Refs. [114,115]. (Bottom) Code capacity threshold of the even code of Sec. IV D 2 subject to the same noise. The threshold is around 17.5%.

thresholds close to 18.9%, which is the hashing bound [115], two reasons might contribute to this suboptimality. First, this is a QML decoder and not a DQML decoder. Second, we have set time limits for its runs due to computational time constraints. Evidently, the logical error rate is lower for the even code as the possible logical error paths are fewer.

3. BP-OSD-4 decoder

The two extremes of a very fast but less optimal decoder (MWPM-4) and the slow but accurate decoder (ILP) call for a middle ground where relatively fast decoding gives adequate results. To achieve this, we turned to belief-propagation decoding, which is a widely used classical decoder [163]. This message-passing algorithm again tries to find the minimum-weight error x such that $H \cdot x = s$. It computes marginal probabilities for individual qubits based on the information from their adjacent parity and check operators. Importantly, when the Tanner graph [164] of the check matrix is a tree, it exploits this structure to perform fewer summations and is guaranteed to converge [165,166]. It was also shown to do well if the graph is sufficiently tree-like [167–169].

The problem with loops is that they lead to split beliefs: in the quantum case, there are multiple possible solutions because of quantum degeneracy [126,170]. Two solutions,

$H \cdot x_1 = s$ and $H \cdot x_2 = s$, can each receive a fractional estimate. This returns the solution $x = x_1 + x_2$; however, this only guarantees $H \cdot x \equiv s \pmod{2}$, which is not equal to s in general (for the qubit case, $H \cdot x = 0$, which fails for any nonzero syndrome). This problem is addressed by the ordered-statistics decoder (OSD), which is designed to handle situations where the BP algorithm does not converge. Importantly, the OSD step uses the fractional results of the first BP step as soft-decision inputs. This method was used in classical decoding to lower error floors, and has been used relatively recently in a quantum setting [125,126,171]. We defer the specific implementation of the message-passing stage to Appendix C, while describing the OSD stage here.

We assume that we are dealing with pure X noise for simplicity, as described in Sec. IV B. When the belief propagation of Appendix C fails, it returns fractional values for qudits instead of integer values. This fractional soft information is represented by a $n \times 4$ matrix q , where the entries of row q_i represent the probabilities of the i th qudit having experienced an error of $(1, X, X^2, X^3)$. Intuitively, the qudits that are physically further from the error receive high-confidence estimates. Assume qudit i is such a qudit far from the error, the estimated error vector generically looks like $q_i = (0.90, 0.05, 0, 0.05)$, which means we are quite confident this qudit did not experience an error. Another example of reliable qudits may be a pair of qudits that experienced an error while the surrounding qudits did not. In this case, the BP stage provides a confident estimate of the errors on the two isolated qudits.

The first step in the OSD stage is then to order the qudits in the check matrix by the BP algorithm's confidence in their values, from least reliable to most reliable. The OSD stage is based on the observation that while the check matrix is not full-rank and cannot be used to directly find the error by inversion ($x = H^{-1} \cdot s$), we can still find a full-rank submatrix H_A inside the ordered check matrix $\tilde{H} = [H_A | H_B]$, where inversion is possible (i.e., such that $H_A^{-1} \cdot s = x_A$). In the following, we refer to the two sets of qudits as $\{A\}$ (less reliable and with independent columns) and $\{B\}$ (more reliable). The full estimated error is then $x = [x_A | x_B]$, where x_B is formed from the hard decision of the decoder BP stage.

The second step is to actually invert the submatrix H_A to find the zeroth error estimate $x_A^{[0]} = H^{-1} \cdot s_A$. In binary codes, the underlying Hilbert space is just the field $\mathbb{F}_2$ where standard Gaussian elimination can be used to invert any square matrix with nonzero determinant. This is the widely used implementation of this step in qubit-based codes [126,148,150]. The important difference is that the four-dimensional qudits are a ring and not a field, as evidenced by the noninvertible element $2 \times 2 = 0 \equiv \pmod{4}$. The straightforward solution is to use Smith's normal form (SNF) [172,173], which can handle rings to solve the equation $H_A \cdot x = s$ by decomposing $H_A = UDV$ where U and V are invertible, and D is diagonal. The equation then becomes $U \cdot D \cdot V \cdot x = s$, which can be rearranged to $D \cdot y = t$, with $y := V \cdot x$ and $t := U^{-1} \cdot s$.

While the SNF method is mathematically clear, the diagonalization step is performed only once for the code-check matrix. In each shot, the two matrix multiplications to revert from the auxiliary variables y and s dominate the computation. A faster method is to perform Gaussian elimination in two

steps. We reduce rows with unit pivots $\{1, 3\}$. Since these units are invertible, we can also normalize so that the pivots are all 1. Then we reduce over the even pivots without normalizing. These are only row operations, so they are cheaper than the full SNF; however, again, this is done once per code. The bottleneck is the per-shot computation, as we discuss below. After these two reducing steps, the check matrix becomes

$$\overline{H}_A = \begin{pmatrix} I_{r_1} & P_1 & P_f \\ 0 & 2I_{r_2} & 2P'_1 \\ 0 & 0 & 0 \end{pmatrix}. \quad (30)$$

Here, I_{r_1} and I_{r_2} are identity matrices corresponding to the unit and even pivots r_1 and r_2 , respectively. P_1 and P'_1 only have $\{0, 1\}$ entries, and P_f has free qudits n_f with no restricted entries. This is Howell's form of the matrix [174,175]. The crucial point is that, for the even pivots, there is ambiguity in both the SNF and Howell's form. Since an equation like $2x = 0$ has two solutions ($x \in \{0, 2\}$) over $\mathbb{Z}_4$, we have to do a combination sweep to determine which choice is better. Since the number of possibilities grows as $2^{|r_2|}$ where $|r_2|$ is the number of even pivots, we cannot explore all the possibilities. This situation is similar to the combination sweep of Ref. [125].

We devised a simple way to tackle this. First, we note that the BP soft information might have been confident that certain qudits should take one of the two choices. We then assign all the qudits of the even pivots to their preferred BP choices. This is the baseline greedy solution. We then try flipping single qudits from the least reliable γ qudits among the even pivots. This step can be denoted by r_2 -CS-1. Higher combination sweeps (r_2 -CS- ω) can also be considered. We empirically found that truncating at $\omega = 2$ is satisfactory. It gives total trials $1 + \gamma + \binom{\gamma}{2}$. For codes of sizes $d \in \{3, 5, 7, 9\}$, a value of $\gamma = 16$ was used. After these trials, we select the trial with the lowest Hamming weight.

This is a zeroth-order solution since it only solved for the least-reliable qudits $\{A\}$ while fixing the most reliable qudits $\{B\}$ to their BP beliefs. Higher-order OSD involves flipping some of the more reliable qudits $\{B\}$ and determining whether this reduces the Hamming weight of the estimated error. This is called a combination sweep (CS) [126,150,176]. Since the combinations very quickly become intractable, algorithms usually focus on the lowest-weight combinations, such as flipping a single or two qubits at a time [126,150,176]. In our qudit case, each qudit must be randomly flipped to one of the three values $\{1, 2, 3\}$, which adds a constant factor of 3 for the single-qudit sweeps and a factor of 3^2 for the pair sweeps. However, some qudits are constrained by even pivots, and thus some of the flipping choices are not solvable. This is checked for every trial, and the trial is skipped if it is unsolvable.

The combination sweep step for the reliable qudits is implemented in multiple ways. For example, one can set a number of free qudits λ and try all single and pair combinations in this set [126]. This involves $3 \times \lambda + 9 \times \binom{\lambda}{2}$ trials. Alternatively, we can set a number Ω and try all combinations of Hamming weight Ω over the entire free-qudit space n_f [125]. For $\Omega = 2$, this involves $3 \times n_f + 9 \times \binom{n_f}{2}$. In practice, we found it useful to set an upper bound on the number of trial qudits controlled by the parameter Γ . We opted for

ALGORITHM 1. OSD-CS for $\mathbb{Z}_4$.

```

1: procedure OSDH,  $s, q, \omega = 2, \Omega = 2, \gamma, \Gamma$ 
2:   Order  $H$ 's columns by  $q$  confidence to form  $\tilde{H} = [H_A|H_B]$ .
3:   Reduce  $\tilde{H}$  to Howell's form.
4:   Assign hard beliefs for even pivot qudits  $r_2$  and free qudits  $n_f$ .
5:   Initialize:  $e_{\text{best}} \leftarrow \emptyset, w_{\text{min}} \leftarrow \infty, q_{\text{best}} \leftarrow \infty$ .
6:   for even trial sweep  $q_\sigma \in \text{even pivots } r_2$  ( $\sigma \leq \gamma$ ) do
7:     for free trial sweep  $q_\tau \in \text{free qudits } n_f$  ( $\tau \leq \Gamma$ ) do
8:       if any even pivot has an odd rhs then
9:         continue  $\triangleright$  The trial is unsolvable
10:      end if
11:      Compute estimated error  $e$ .
12:      if  $\text{wt}(e) < w_{\text{min}}$  or ( $\text{wt}(e) = w_{\text{min}}$  and  $q < q_{\text{best}}$ ) then
13:         $e_{\text{best}} \leftarrow e, w_{\text{min}} \leftarrow \text{wt}(e)$ 
14:      end if
15:    end for
16:  end for
17:  return  $e_{\text{best}}$ 
18: end procedure

```

using single combinations up to $3 \times \Gamma$ and pairs up to Γ . This brings the total trials to $9 \times [\Gamma + \binom{\Gamma}{2}]$. We used n_f -CS-2 with Γ up to 19. The algorithm is summarized in Algorithm 1.

An example of a threshold calculation is shown in Fig. 8. Here, the DS code is simulated against pure X noise, achieving a threshold of 21%. This is a 10% improvement from the same calculation with machine-learning tools using non-Pauli codes, which gave 9.5% [177].

We finally note that the combination sweep for the free qudits n_f or the even pivots can be done in parallel, avoiding explicitly running the double for loops of Algorithm 1.

D. Numerical simulations of qudit codes

In this section, we present the numerical results of decoding new composite topological codes using the noise model of Sec. IV B and the decoders of Sec. IV C. We begin by providing the explicit stabilizer layouts for these codes. These

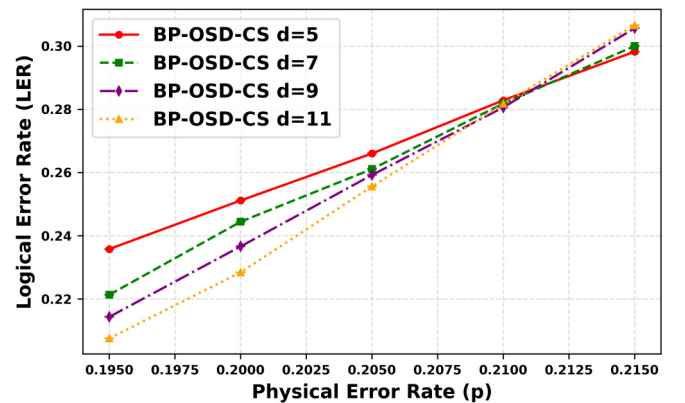


FIG. 8. Threshold calculation of the DS code on a torus of Sec. IV D 3 subject to pure X noise. This was done with 10^6 shots with belief-propagation with OSD and CS as illustrated in Algorithm 1 and in Appendix C.

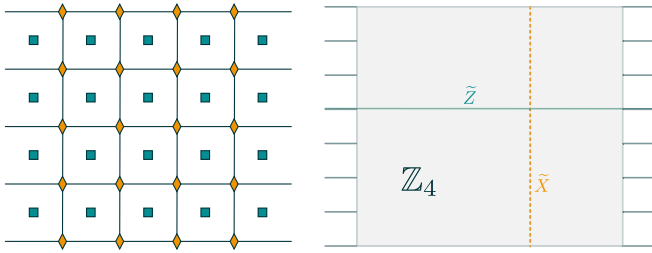


FIG. 9. $D(\mathbb{Z}_4)$ surface code with opposite smooth and rough boundary conditions, and which has $\text{GSD} = 4$. The logical operators $\tilde{Z}$ and $\tilde{X}$ are a string of Pauli Z s and a string of Pauli X s, respectively.

include the even code introduced in Fig. 2, the DS code explained in Sec. IV D 3, and a new hybrid DS- $\mathbb{Z}_4$ code.

1. Planar $\mathbb{Z}_4$ code benchmarking

We begin by discussing the $\mathbb{Z}_4$ surface code. This is the simplest topological code using qudits, and there is extensive literature on its thresholds. The code layout is identical to that of the well-known surface code but with qudits instead, as shown in Fig. 9. From statistical mechanical arguments, a threshold of 18.9% is expected for the pure X noise [115] while renormalization group (RG) decoders have achieved a threshold of around 15.4%. Our simplistic MWPM-4 decoder, described in Sec. IV C 1 achieves a threshold of 14%, which is comparable to the RG decoders. However, both the integer linear programming (ILP) solver and the belief-propagation decoder (BP) achieve a threshold of around 18%, which is much closer to the theoretical maximum. For the depolarizing noise, we find thresholds of 27% and 26.5% for ILP and BP, respectively.

2. Even code results

The even code introduced in Fig. 10 is a simple variant of the $\mathbb{Z}_4$ surface code. It effectively restricts the code to its even sectors, thereby realizing a logical qubit within the logical qudit of the $\mathbb{Z}_4$ surface code. This is achieved by changing one of the rough boundaries of the surface code to a new boundary we call the even boundary; see Sec. III A 2. This changes the logical $\tilde{Z}$ operator to a string composed of Z^2 Pauli operators as shown in Fig. 10. This lowers the logical

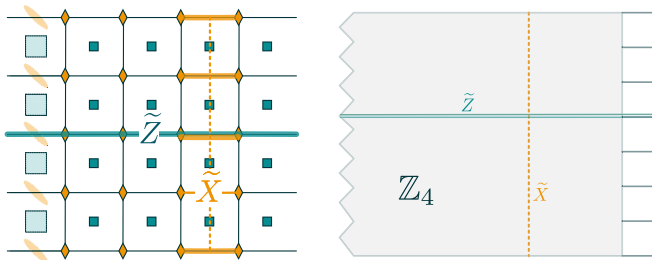


FIG. 10. $D(\mathbb{Z}_4)$ surface code with two opposite smooth boundaries, a rough, and an even boundary (here denoted by a zigzag boundary). It has a $\text{GSD} = 2$. The logical operators $\tilde{Z}$ and $\tilde{X}$ are a string of Pauli Z^2 s (denoted with double lines) and a string of Pauli X s, respectively.

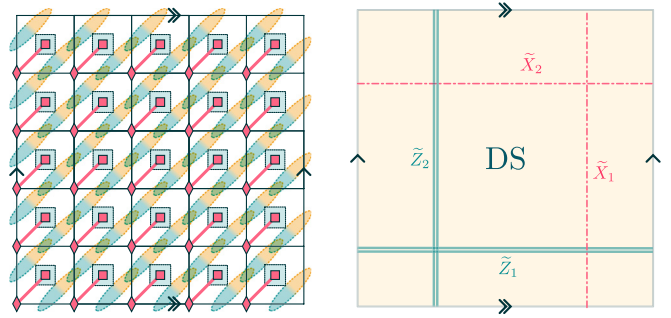


FIG. 11. Double Semion code on a torus which has $\text{GSD} = 4$. The logical operators $\tilde{Z}$ are strings of Pauli Z^2 creating the boson b . While the logical $\tilde{X}$ are strings of Pauli X and Z s creating the semion s ; see Fig. 4.

error rate by at least half, as the odd Z operators cannot induce a logical error. We could not find any results for this code in the literature, although we expect its thresholds to be similar to those of the $\mathbb{Z}_4$ surface code, as they share the same bulk. Indeed, we found for the pure X noise a code capacity threshold of 17.5% and 18% for the ILP and BP decoders, and only 15.5% for the MWPM-4 decoder. For the depolarizing noise, ILP and BP yielded 27% and 26%, respectively. The MWPM-4 decoder cannot handle the complex depolarizing noise over $\mathbb{Z}_4$. Importantly, there is a biased suppression of pure Z relative to pure X noise due to the asymmetric nature of this code. While they have the same thresholds, the logical error rate for the Z noise was found to be significantly lower (by almost a factor of 2 at $d = 5$ and $p = 10\%$).

3. DS code results

A DS code can be defined on a torus, just like the toric code [4,17]. The logical operators of the DS code on a torus are shown in Fig. 11. The code can be realized using qubits and non-Pauli stabilizers, as it is a twisted quantum double. It was simulated using machine learning decoders under pure X noise, achieving a threshold of 9.5% and a threshold of 10.5% for depolarizing noise [177]. The same reference reports thresholds of 7.6% (pure X) and 7.4% (depolarizing) with the MWPM decoder, and 9.4% (pure X) and 10.5% (depolarizing) with multilayer-perceptron decoding.

Equivalently, the code has a description in terms of generalized Pauli stabilizers for four-dimensional qudits [42]. Using this latter description, we were able to simulate the code using an ILP decoder, obtaining thresholds of 17.5% and 20.5% for pure X and depolarizing noise, respectively. Moreover, the BP decoder achieves thresholds of 20.5% and 22%, respectively. These numbers are almost double the thresholds for the qubit case due to the fact that half of the anyons of the $\mathbb{Z}_4$ are confined in the DS phase; see Fig. 15 below. These confined anyons translate to error strings that the decoder can see and reliably correct since it sees the whole string and not only the two ends.

These simulations used up to 5×10^6 shots and a capped ILP time of 12 s per shot. This time limit explains the less-expected result of the ILP solver achieving a lower threshold than BP. We also note that BP with OSD can match the ILP accuracy with a very exhaustive OSD stage as discussed in

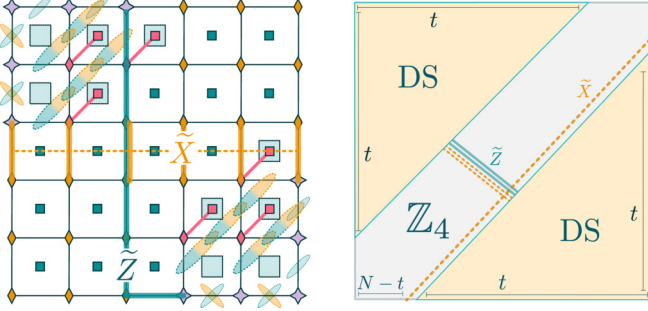


FIG. 12. Layout for a hybrid DS- $\mathbb{Z}_4$ code on a plane which has $\text{GSD} = 2$. The logical operators $\tilde{Z}$ are strings of Pauli Z^2 creating the boson b . While the logical $\tilde{X}$ are strings of Pauli X . Alternatively, a string of simultaneous X^2 and Z^2 can also create a logical $\tilde{X}$ between the two DS patches.

Sec. IV C 3. Finally, similar to the qubit case, the twisted nature of the code results in the depolarizing noise threshold being close to the pure noise. This can be easily seen in the Pauli-stabilizer case simulated here, as condensing the anyon $e^2 m^2$ induces an equivalence between X and Z errors.

4. Hybrid DS- $\mathbb{Z}_4$ code results

Finally, the most interesting case is when we spatially combine DS and $\mathbb{Z}_4$ phases to form a spatially anisotropic composite-dimensional code. One simple layout for this code is shown in Fig. 12. This code combines two phases with two thresholds. As we found in Secs. IV D 1 and IV D 3, the DS code has a higher pure X -noise threshold, while the planar code has a higher depolarizing-noise threshold. It is hard to determine a priori the new thresholds for the new code, given the BP algorithm as a decoder.

We still expect that the logical error rate will be lower than that of the planar code, as the average weight of the logical $\tilde{X}$ operators is now higher. The domain wall between the DS and $\mathbb{Z}_4$ phases condenses $\{1, e^2 m^2\}$ of the $\mathbb{Z}_4$ anyons, while all anyons of DS can propagate into the $\mathbb{Z}_4$. In contrast to our prior examples, involving a single TQFT, the $\mathcal{L}_{\mathbb{Z}_4\text{-DS}}$ domain wall is noninvertible. This implies that particles do not tunnel across the interface symmetrically. For example, the e anyon cannot tunnel into the DS phase as it is confined. This is natural since the two phases have a different number of anyons. We defer the details of how this domain wall is constructed, and its relation to anyon condensation, to Example V.5. Since the wall is noninvertible, its action cannot be read from the generators $\{e, m\}$ alone. A more complete description of domain walls and their relationships to degeneracy is discussed in Sec. VI B. For illustration, we now give the DW's action on each anyon of $\mathbb{Z}_4$,

$$W^{\mathbb{Z}_4\text{-DS}} : \begin{cases} 1, e^2 m^2 & \mapsto 1, \\ em, e^3 m^3 & \mapsto s, \\ em^3, e^3 m & \mapsto \bar{s}, \\ e^2, m^2 & \mapsto b, \\ \text{otherwise} & \mapsto \text{confined}. \end{cases}$$

One can naturally explore other layouts for these codes. For example, we can push the patches to the bulk of the $\mathbb{Z}_4$ as in Fig. 13, also see Example VI.7.

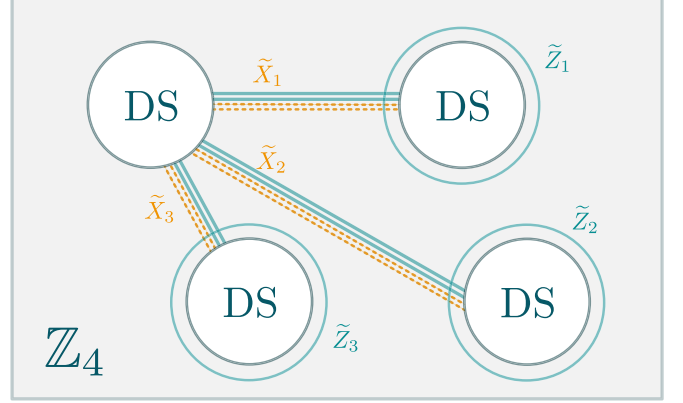


FIG. 13. Hybrid three-logical-qubit $\mathbb{Z}_4$ -DS patch code composing of four DS patches inside a $\mathbb{Z}_4$ bulk. The logical $\tilde{Z}$ operators consist of e loops (solid lines) around three patches. The logical $\tilde{X}$ operators are an $e^2 m^2$ (two solid lines and two dashed lines) string between three patches and one patch.

The patches can exchange $e^2 m^2$ anyons between them. These strings commute and square to one $e^4 m^4 = 1$. We therefore identify them with the Pauli $\tilde{X}_i$ operators on each of the three logical qubits. In addition to patches exchanging $e^2 m^2$ s, we can also form noncontractible loops, of e or m world lines, around each patch, which are confined in the DS phase, and therefore cannot be contracted to trivial loops inside the patch. These loops are depicted in Fig. 13. Interestingly, the loops again square to unity, this time because the e^2 anyons are *deconfined* inside DS and, e.g., the e^2 loop can be contracted. Further, the e or m world-line loops anticommute with the Pauli $\tilde{X}_i$ operators as $e^2 m^2$ and e anticommute, see Eq. (12). They can then be used to define the Pauli $\tilde{Z}_i$ logical operators acting on each new logical qubit.

To summarize, this section presents the thresholds of different codes subject to different noise models and using ILP, BP-OSD-CS, and MWPM-4 decoders (Table I).

The computing resources used to produce the results in Table I consisted of a single node with 64 cores (multicore processing on a single node), with a maximum job run time of 24 hours for the ILP decoder. However, for the latest belief propagation-based decoder, the run times are generally

TABLE I. Code capacity thresholds across different noise models and decoders.

No.	Codes	Noise	ILP	BP-CS	MWPM-4
1	Planar $\mathbb{Z}_2$	Pure X	11%	10.75%	10.3%
		Depol.	18%	16%	15.25%
2	Planar $\mathbb{Z}_4$	Pure X	18%	18%	14%
		Depol.	27%	26.5%	17.25%
3	Even	Pure X	17.5%	18%	15.5%
		Depol.	27%	26%	17.75%
4	DS	Pure X	17.5%	21%	N/A
		Depol.	20.5%	22%	N/A
5	DS- $\mathbb{Z}_4$	Pure X	18.5%	19%	N/A
		Depol.	26.5%	26.5%	N/A

TABLE II. Code parameters n , k , and d for each evaluated code family.

No.	Codes	n	k	d
1	Planar $\mathbb{Z}_2$	$2d^2$	1	d
2	Planar $\mathbb{Z}_4$	$4d^2$	2	d
3	Even	$4d^2$	2	d
4	DS	$4d^2$	4	d
5	DS- $\mathbb{Z}_4$	$4d^2$	2	d

less than 10 hours. We used between 5×10^5 and 10^8 shots. These results show that using the same number of qudits, one can still achieve different thresholds or logical error rates by virtue of their connectivity. Indeed, as shown in Table II, the encoding rates and number of qubits (assuming each qudit is made from two qubits) are comparable for these codes, while they still exhibit different behaviors as evidenced in Table I. As shown in Fig. 14, the logical error rates are successively suppressed as we change the codes. For instance, both the even and the hybrid code use the same asymptotic number of qudits, however, the hybrid code has a lower logical error rate. This shows that spatial structure, composite-dimensionality, and stabilizer connectivity are three valuable resources for topological codes.

V. MICROSCOPIC BOUNDARY FORMULATION

To make the results and codes above rigorous, this section derives the lattice models presented and provides an algorithm for doing so. Our algorithm takes as input a set of stabilizers and returns a new set of stabilizers describing the new Abelian twisted quantum double, which may contain boundaries and domain walls in addition to bulk terms. We present examples highlighting its use in constructing DS patches embedded in a $\mathbb{Z}_4$ parent phase at both a concrete and a conceptual level.

Conceptually, we show how an understanding of the classification of boundaries can aid their construction from the bulk Hamiltonian terms. This is manifested in a concrete lattice-level algorithm. The concrete construction of the new codes in the previous section is a by-product of this algorithm. Its utility extends well beyond that. The intuition and perspective presented here can be especially useful for the recently proposed schemes that use multiple phases to perform universal quantum computing [178,179].

This section is organized into five subsections. Section V A lays out the classification of the boundaries and discusses the differences from other works; Sec. V B explains at a conceptual level why boundaries can be constructed from bulk

TABLE III. Rosetta stone with cocycle ($K \subseteq G, \omega$) and Lagrangian subgroup $\mathcal{L}$ descriptions of $\mathbb{Z}_4$ boundaries.

Boundary	K	$[\omega]$	$\mathcal{L}$
Rough	$\{0\}$	trivial	$\{1, e, e^2, e^3\}$
Smooth	$\{0, 1, 2, 3\}$	trivial	$\{1, m, m^2, m^3\}$
Even	$\{0, 2\}$	trivial	$\{1, e^2, m^2, e^2 m^2\}$

TABLE IV. Translation between subgroup and cocycle ($K \subseteq G, \omega$) and Lagrangian subgroup $\mathcal{L}$ descriptions of the boundaries of the DS phase.

Boundary	K	$[\omega]$	$\mathcal{L}$
DS-Vacuum	$\{0\}$	trivial	$\{1, b\}$

terms; Sec. V C gives the concrete lattice-level algorithm for their constructions; and Secs. V D and V E give examples of boundary and domain wall construction. The construction of 0D defects is deferred to Appendix B.

A. Boundaries classification

The classification of the boundaries and domain walls of a twisted quantum double based on the (not necessarily Abelian) group G is determined by the Lagrangian algebras of the topological order [51,65]. As seen in Sec. III A 2, the set of mutually compatible condensing anyons completely specifies a boundary. In the case of untwisted quantum doubles, these Lagrangian subgroups are in one-to-one correspondence with a subgroup $K \subseteq G$ and a function $\omega : K \times K \rightarrow U(1)$ that associates a $U(1)$ phase to each ordered pair of elements in the subgroup K . This function is a 2-cocycle since it obeys

$$\delta\omega(g_1, g_2, g_3) = \frac{\omega(g_2, g_3)\omega(g_1, g_2g_3)}{\omega(g_1g_2, g_3)\omega(g_1, g_2)} = 1, \quad \forall g_1, g_2, g_3 \in G, \quad (31)$$

$$\omega(e, g) = \omega(g, e) = 1, \quad \forall g \in G. \quad (32)$$

The first line is the trivial coboundary condition, while the second line is a gauge choice that can always be made [47]. Further, the function ω is defined only up to an equivalence class $[\omega] \in H^2(K, U(1))$.

As an example, we provide a translation between the two descriptions for the case of $\mathbb{Z}_4$ in Table III.

In the case of twisted quantum doubles, extra constraints apply to the boundary [52]. For example, the DS phase is based on the group $\mathbb{Z}_2$, which has two subgroups, namely the trivial subgroup and the whole group. However, since it is a twisted quantum double with a nontrivial 3-cocycle, it cannot have a boundary with a nontrivial subgroup [52] which is presented in Table IV.

In what follows, we do not deal with the classification in terms of cocycles directly. Instead, we only use the physically intuitive Lagrangian subgroup of anyons explicitly, while the cocycle picture remains implicit. This is contrary to Refs. [47,49,50,52] that abstractly formulated how to construct microscopic lattice models for a (twisted) quantum double's gapped boundary or domain walls using cocycles. In the case of quantum doubles, the models are formed by defining subgroup variants of the star and plaquette operators

TABLE V. Summary of changes from toric code to toric code with a noncontractible twist.

Order	Operators				Δn_m	Δl_m
$m = 2$	+3 XZ	+3 F	-3 A	-3 B	0	-1

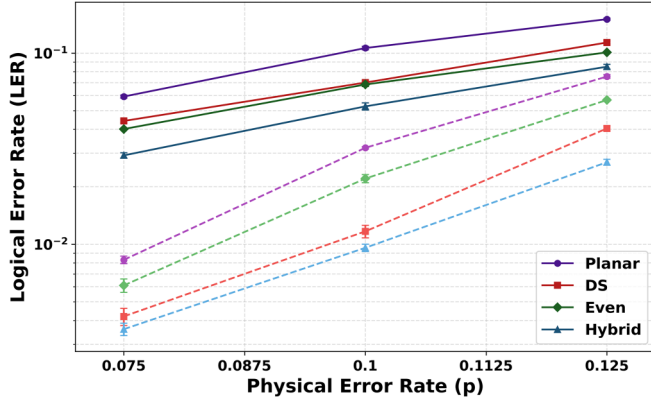


FIG. 14. Comparison of logical error rates among different codes subject to pure X noise using the belief-propagation algorithm discussed in Sec. IV C. Solid lines represent $d = 3$ while dashed lines represent $d = 7$. The plot shows monotonic error suppression between the different codes.

(6). The modified star operators have a modified action that associates a $U(1)$ phase depending on the qudits' values. Consequently, they are not readily implementable experimentally. This issue also arises for 1D domain walls and 0D defects that require an extra $U(1)$ phase (those with a nontrivial 2-cocycle) [47,49,50,52]. From another perspective, these abstract terms, while correct, are not derived from the bulk terms.

Other constructions were proposed recently to obtain Pauli stabilizers for the boundaries of Abelian (twisted) quantum doubles [53]. One important difference is that our algorithm also produces the 0D defects. In addition, it produces topologically complete models, meaning that the only local operators that commute with the stabilizers are products of other stabilizers [53,57]. Other choices for the initial (twisted) quantum double stabilizers are still valid for our algorithm. For example, spatially disconnected stabilizers introduced in Ref. [53], do not require extra steps to ensure topological completion in our construction.

B. Boundaries by local condensation

The operators presented in the previous works [47,49,50,52], while sufficient to mathematically study the properties of each boundary, do not provide a simple prescription for experimental implementation. To this point, for Abelian twisted quantum doubles, a description of the bulk in terms of generalized Pauli stabilizers was recently provided [42]. However, a simple construction of Pauli stabilizers for the boundaries and domain walls has still been lacking. Condensation of anyons is a suitable tool for studying boundaries [65,180,181]. Inspired by the correspondence between the classification of boundaries and Lagrangian subgroups of the bulk, we present an algorithm to construct different boundaries, domain walls, and 0D defects of Abelian topological orders described by (twisted) quantum doubles using local condensation.

Our construction of the boundaries is similar to the procedure for condensing the bulk of the untwisted quantum double to produce twisted quantum doubles [42], but with two modifications. First, we condense a Lagrangian group instead of

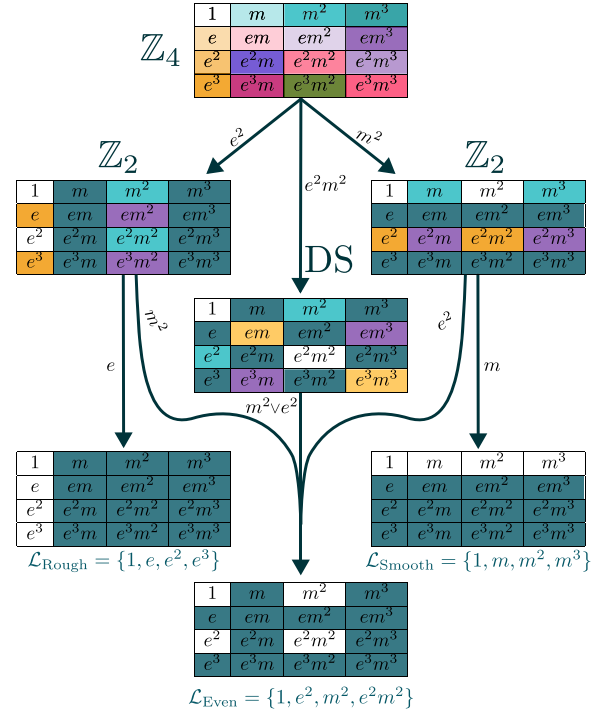


FIG. 15. Different condensation paths starting from a $D(\mathbb{Z}_4)$ bulk. The top three paths $\{e^2, m^2, e^2 m^2\}$ produce $\mathbb{Z}_2$ toric codes with different domain walls with the $\mathbb{Z}_4$ and DS phase, respectively. Further condensing $\{m^2, e^2, e^2 \sim m^2\}$ produces the even boundary of the $\mathbb{Z}_4$. Anyon labels with the same color are identified, while those with a dark background are confined. Seen differently, the final paths to $\mathbb{Z}_4$ boundaries are also different boundaries for the $\mathbb{Z}_2$ and DS embedded in the $\mathbb{Z}_4$.

a partial condensation that leaves nontrivial topological order afterward. Second, we only condense a region in space that leaves the boundary terms we are interested in.

Partial condensation. In $\mathbb{Z}_p$, where p is a prime, only rough and smooth boundaries exist. Thus, all punctures in $\mathbb{Z}_p$ are either rough or smooth or a mixture of the two. Anyon condensation is much richer in composite-dimensional qudits $\mathbb{Z}_{pq}$. This can be illustrated with the example of $\mathbb{Z}_4$ ($p = q = 2$), as summarized in Fig. 15. Starting with the bulk theory ($\mathbb{Z}_4$), one can condense any set of bosons that mutually braid trivially, relaxing the last requirement of Sec. III A 2 (that anyons outside the condensed subgroup braid nontrivially with an anyon in the group). For example, condensing only e^2 in $\mathbb{Z}_4$ is allowed and results in the $\mathbb{Z}_2$ toric code [182], which can be thought of as a new kind of puncture inside the $\mathbb{Z}_4$. This condensation is partial in the algebraic sense since, after condensation, nontrivial topological order remains. For example, condensing e^2 leaves the anyon classes $\{[1], [e], [m^2], [em^2]\}$. These are the remaining classes because condensing e^2 identifies $1 \sim e^2$, $e \sim e^3$, $m^2 \sim e^2 m^2$, and $em^2 \sim e^3 m^2$. The rest of the anyons (e.g., m) are now confined, as they braid nontrivially with one of the condensed anyons (e.g., e^2). Confined anyons require energy to propagate through the phase, and they are no longer elementary excitations. This process is shown in Fig. 15, where confined anyons have a darker background and anyons of the same color belong to the same class. Thus, condensing e^2 yields

the $\mathbb{Z}_2$ toric code, where the spins and braiding relations are inherited from the $\mathbb{Z}_4$ operator algebra. In the same way, as discussed in Sec. III B, condensing $e^2 m^2$ gives the DS phase. In general, all twisted quantum doubles with Abelian orders can be constructed similarly by condensing certain quantum doubles [42].

Maximal condensation. On the other hand, condensing a Lagrangian subgroup (Sec. III A 2) leaves only the trivial topological order. One such Lagrangian subgroup is $\mathcal{L}_{\text{rough}} = \{1, e, e^2, e^3\}$. This subgroup is generated by condensing either e or e^3 . If we condense e after condensing e^2 , we end up with the trivial topological order where all deconfined anyons are identified with identity. This is shown at the bottom left of Fig. 15.

Local condensation. Condensing a set of anyons globally, whether a Lagrangian group or one of its subgroups, switches the topological phase into another one. However, if the condensation is carried out locally, we end up with two phases with a domain wall at their interface. If one of the phases is the vacuum, then the interface is a boundary (maximal condensation); otherwise, it is a domain wall between two phases (partial condensation). In the $\mathbb{Z}_4$ example, condensing $\mathcal{L}_{\text{rough}} = \{1, e, e^2, e^3\}$ over a region R of space results in vacuum in that region. Additionally, the boundary ∂R corresponds to the rough boundary of $\mathbb{Z}_4$ Eq. (13). Similarly, condensing e^2 over a region R results in $\mathbb{Z}_2$ over that region. The boundary ∂R is a domain wall between $\mathbb{Z}_4$ and $\mathbb{Z}_2$. Condensing m^2 over R would have resulted in the $\mathbb{Z}_2$ phase over that region. However, it corresponds to a different domain wall between $\mathbb{Z}_4$ and $\mathbb{Z}_2$. The correspondence between anyon condensation and boundaries ensures that all domain walls can be constructed this way.

The takeaway from the $\mathbb{Z}_4$ example is that condensing anyons over a region R should be enough to construct all possible boundaries at ∂R . Since any domain wall between two phases is a boundary of a related phase, we focus on boundaries without loss of generality. We consider concrete examples of domain walls in Sec. V E and finite domain walls ending with 0D defects in Appendix B.

C. Algorithm to construct boundary stabilizers

We now describe the algorithm for constructing the boundary stabilizers. Suppose we have a (twisted) quantum double, which has the bulk Hamiltonian,

$$H = - \sum_v A(v) - \sum_p B(p) - \sum_e C(e) + \text{H.c.}, \quad (33)$$

where the $C(e)$ are edge terms required for the case of twisted quantum doubles [e.g., Eq. (20)]. Here, the summation is over all vertices V , edges E , and plaquettes P as the model is assumed to live on a closed surface $\mathcal{M}$. As before, the star and plaquette operators are the standard tensor products of generalized Pauli operators; see Eq. (5). The terms of this Hamiltonian generate a stabilizer group, which we denote by $\mathcal{S}_H$,

$$\mathcal{S}_H = \langle A, B, C \rangle. \quad (34)$$

We want to modify both the Hamiltonian and the stabilizer group to have a spatial boundary ∂R bounding a region $R \subset$

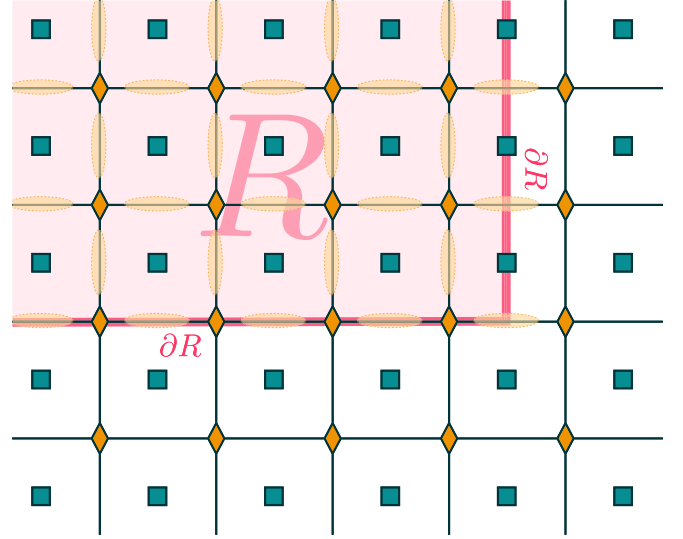


FIG. 16. After the first two steps of the algorithm, the edge operators X (yellow ellipses) are measured inside the selected region R and also on the boundary ∂R .

$\mathcal{M}$ of the lattice. In addition, as boundaries are in one-to-one correspondence with Lagrangian subgroups (the anyons they condense), we have to choose the kind of boundary corresponding to a Lagrangian subgroup $\mathcal{L}$. We do this in six steps:

First, we choose a 2D region $R \subset \mathcal{M}$ in the lattice with a 1D boundary ∂R and choose a Lagrangian subgroup $\mathcal{L}$ for the boundary. The type of the boundary is independent of its microscopic spatial geometry (one does not need to fix a specific shape of the boundary beforehand). The boundary can live on the direct or dual lattice, or a mixture of them. One choice for the boundary is shown by the pink line in Fig. 16.

Second, we add all ribbon operators $F^a(t)$ that create the condensed anyons in $\mathcal{L}$ inside R or on ∂R to the Hamiltonian. In the case of dyons, for example, condensing $em \in \mathcal{L}$, the shortest ribbon acts on two edges. We call these the new stabilizer terms $\mathcal{S}_{\mathcal{L}}$,

$$\mathcal{S}_{\mathcal{L}} = \{F^a(t) \mid a \in \mathcal{L}\}. \quad (35)$$

Third, we remove the old stabilizers that do not commute with the new ones. The physical interpretation is that the removed operators correspond to the smallest contractible loops of anyons that become confined and cannot tunnel through the boundary to the vacuum. The set of now confined stabilizers is

$$\mathcal{S}_C = \{h \in \mathcal{S}_H \mid \exists l \in \mathcal{S}_{\mathcal{L}}, [h, l] \neq 0\}. \quad (36)$$

Fourth, we construct the new stabilizer group $\mathcal{S}_{H'}$,

$$\mathcal{S}_{H'} = \mathcal{S}_{\mathcal{L}} \cup (\mathcal{S}_H \setminus \mathcal{S}_C). \quad (37)$$

The generators of this stabilizer group are denoted by $J(v, p)$ and they fall into one of three categories: (a) old stabilizer group $\mathcal{S}_H$ generators $A(v), B(p), C(e)$ that continue to commute with the new ones in $\mathcal{S}_{\mathcal{L}}$, (b) the new stabilizers of the condensed anyons $\mathcal{S}_{\mathcal{L}}$, and (c) local products of removed old stabilizer terms $J(v, p) = \prod_{\text{sc}} A(v)B(p)C(e)$ that commute with the new terms $\mathcal{S}_{\mathcal{L}}$. The terms in (c) are promoted from being generated in the old code to being generators in the new

ALGORITHM 2. Boundary stabilizers for H .

```

1: procedure BDRY ( $H, R, \mathcal{L}_R$ )
Require: A region  $R$  with boundary  $\partial R$  and a Lagrangian
subgroup  $\mathcal{L}_R$ .
2:   Measure the stabilizers  $\mathcal{S}_\mathcal{L}$  that create anyons in  $\mathcal{L}_R$ 
inside  $R$  or on  $\partial R$ .
3:   Remove stabilizers  $\mathcal{S}_C \in \mathcal{S}_H$  that do not commute with  $\mathcal{S}_\mathcal{L}$ .
4:   Construct  $\mathcal{S}_{H'} = \mathcal{S}_\mathcal{L} \cup (\mathcal{S}_H \setminus \mathcal{S}_C)$ .
5:   while condensed qudits exist do
6:     Erase trivialized qudits.
7:     Restrict boundary stabilizers to their nontrivialized support.
8:   end while
9:   return  $H_{\partial R}$ 
10: end procedure

```

code. For example, it may be the case that $A(v_1)$ and $B(p_1)$ do not commute with the new stabilizer group $\mathcal{S}_\mathcal{L}$ but that their generated product $J(v_1, p_1) = A(v_1)B(p_1)$ does. In this case, we add $J(v_1, p_1)$ as a generator of the new stabilizer group $\mathcal{S}_{H'}$.

Fifth, we erase any qudits that are trivially constrained. By a trivially constrained qudit, we mean that there are no degrees of freedom left for the qudit. In other words, the qudit is completely disentangled from the rest of the system and can be safely discarded without affecting the properties of the underlying topological code. These include the qudits inside R or on the boundary ∂R as they are acted on by the stabilizers in $\mathcal{S}_\mathcal{L}$.

Sixth, restrict the stabilizers at the boundary to the nontrivial qudits only. If, after this step, new qudits are condensed, then repeat step five until no more qudits are condensed [183]. Keeping them results in a boundary that strictly follows the boundary ∂R . We choose to erase all trivialized qudits for clarity. The new Hamiltonian is the summation of the generators of $\mathcal{S}_{H'}$,

$$H' = - \sum_{v \in \text{Bulk}} A(v) - \sum_{p \in \text{Bulk}} B(p) - \sum_{e \in \text{Bulk}} C(e) - \sum_{(v,p) \in \partial R} J(v,p), \quad (38)$$

where A , B , and C are the bulk stabilizers that do not overlap with ∂R . $J(v, p)$ represents a generic boundary term that can depend on a vertex and an adjacent plaquette at the same time. The algorithm is summarized in Algorithm 2.

D. Examples of boundary construction

To illustrate this algorithm, we consider, in order of increasing complexity, the explicit construction of a few example boundaries. First up is the well-known smooth boundary of the surface code. The star and plaquette operators are given by Eq. (6):

$$A(v) = \begin{array}{c} X \\ \hline X^\dagger \end{array} = \diamond, \quad B(p) = \begin{array}{c} Z^\dagger \\ \square \\ Z \end{array} = \blacksquare \quad (39)$$

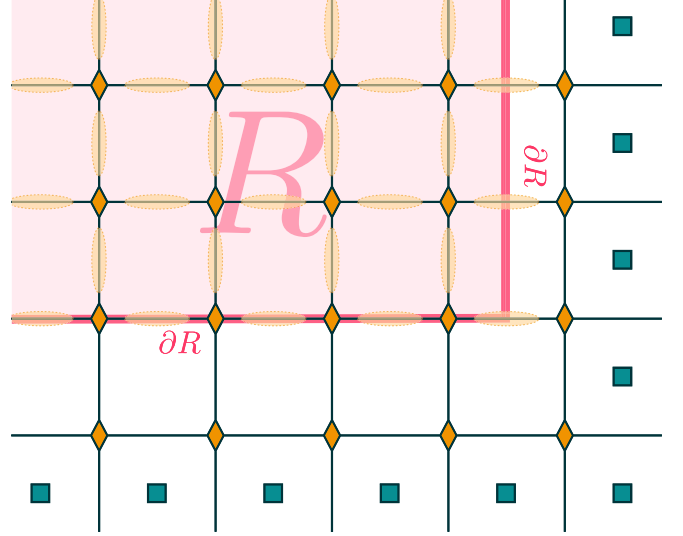


FIG. 17. After the fourth step of the algorithm, the stabilizers that do not commute with the new X operators are removed. These include any plaquette that shares any edge with an X operator.

Example V.1 (Smooth boundary for the $\mathbb{Z}_4$ surface code).

First, we choose a connected region R . As illustrated in Fig. 16, R is bounded by ∂R , denoted by the pink path. We wish for this smooth boundary to condense m particles, so the Lagrangian subgroup is $\mathcal{L} = \{1, m, m^2, m^3\}$. Second, to condense the Lagrangian subgroup anyons, we measure the shortest ribbon operator that creates the generator (X) at each qubit inside R and at the pink path ∂R . Figure 16 shows the resulting configuration after the first two steps. Third, we need to remove the old $B(p)$ stabilizers that do not commute with the measured X operators. This includes all the plaquettes inside R as well as those that share any edge with ∂R . The star operators commute, and they are therefore not removed. Fourth, since the only products of the old operators that commute with the new stabilizers are the star operators, there is nothing to do in this step. Importantly, those star operators located at the boundary ∂R will still commute with the new Hamiltonian, so they remain. After the first four steps, we reach the configuration shown in Fig. 17.

In the fifth step, for clarity, we remove the edges that have condensed to vacuum as they are decoupled from the rest of the system. Any edge measured with the short string local- X operator condenses into a trivial vacuum. After condensing these edges, we arrive at Fig. 18(a). The star operators on the boundary ∂R now have a support of one edge. This is written in a consistent notation in Fig. 19.

This is a perfectly valid boundary for the surface code. The seemingly rough boundary is actually a smooth boundary because it condenses m anyons. Had we chosen the pink path to lie in the dual lattice only, we would not have needed to deal with this, as exemplified by the other vertical pink line. Removing the edges that are now touched by only one X operator gives us the (geometrically) smooth boundary [73] illustrated in Fig. 19(b). Sixth, we restrict the new stabilizers to their support on the nontrivial qudits. Some of the vertices now have a star that is supported on only three edges.

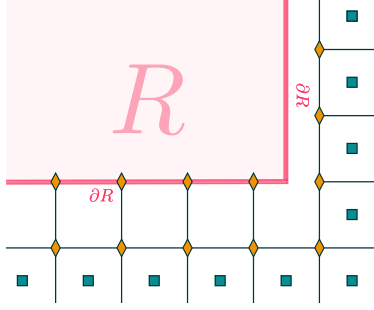


FIG. 18. After removing the qudits measured by the X operator, star operators have one edge support at the boundary.

In Example V.1, the fourth step was trivial because of the simplicity of the surface code. In the next example, we treat the slightly more complicated case of a new boundary in $\mathbb{Z}_4$.

Example V.2 (Even boundary for the $\mathbb{Z}_4$ surface code).

First, let us construct the even boundary of the $\mathbb{Z}_4$ surface code (13) on the spatial boundary ∂R located at the pink path in Fig. 20(a), taking the region R extending to infinity as before. Second, we measure the ribbons creating the anyons of the Lagrangian subgroup $\mathcal{L}_{\text{even}} = \{1, e^2, m^2, e^2 m^2\}$. In this case, the condensation algebra is generated by the X^2 (yellow tilted ellipses) and Z^2 (blue tilted ellipses) operators. The configuration after the first two steps is shown in Fig. 20(a). Third, we remove any star or plaquette stabilizers that share any edge with the stabilizers Z^2 or X^2 , respectively. After the third step, we reach the configuration in Fig. 20(b). Fourth, the products of the old stabilizers that we promote to generators are $(A(v))^2$ (purple four-sided star) and $(B(p))^2$ (big dotted square):

$$A^2(v) = \begin{array}{c} X^2 \\ | \\ X^2 \\ | \\ X^2 \end{array} = \text{purple star}, \quad B^2(p) = \begin{array}{c} Z^2 \\ \square \\ Z^2 \end{array} = \text{big dotted square} \quad (40)$$

These stabilizers have to be added to the Hamiltonian, as shown in Fig. 21.

Fifth, we now remove the trivialized qudits. Since a pair of degree two constraints is simultaneously applied to a four-dimensional degree of freedom, all edges in the bulk of the

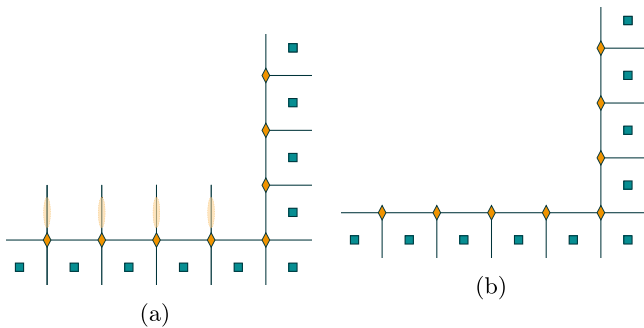
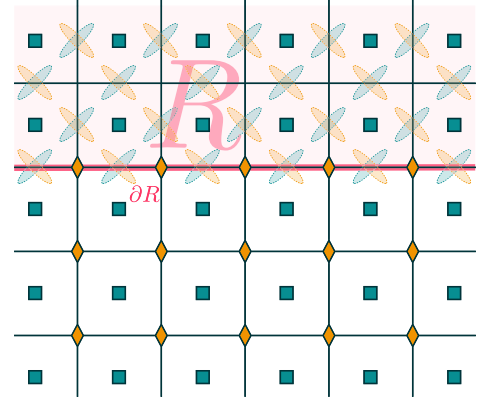
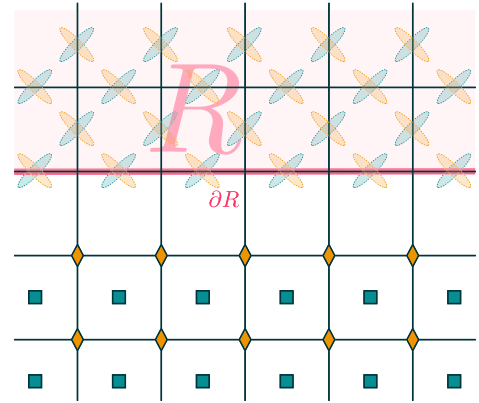


FIG. 19. (a) Smooth boundary for the $\mathbb{Z}_4$ surface code that strictly follows spatial boundary ∂R . (b) After the last step of the algorithm, we recover the familiar smooth boundary for $\mathbb{Z}_4$.



(a)



(b)

FIG. 20. The first three steps of the algorithm to construct the even boundary of $\mathbb{Z}_4$. (a) Region R along with its boundary ∂R . After the second step of the algorithm, the X^2 and Z^2 operators are measured inside R and on ∂R . These operators condense the anyons in the even boundary subgroup $\mathcal{L}_{\text{even}}$. (b) In the third step, the stabilizers that do not commute with the measured operators are removed. These are any plaquette or star operators that share an edge with a measured X^2 or Z^2 .

region R , or on the pink boundary ∂R , condense to the vacuum. In general, the qudits inside R or on ∂R always condense. We reach Fig. 22(a).

Observe that the support of the new star $A^2(v)$ operators on the boundary ∂R is reduced to a single edge. Additionally, the new plaquette operators $B^2(p)$ now have support on three edges. After restricting the stabilizers on the boundary to their nontrivialized supports, no more qudits are condensed, and we are done. We arrive at Fig. 22(b), which describes the even boundary that condenses the $\{1, e^2, m^2, e^2 m^2\}$ anyons. This gives the boundary in Eq. (14),

$$H_{\mathbb{Z}_4, \text{even}} = H_{\mathbb{Z}_4} - \sum_{p, e \in \partial \mathcal{M}} (B^2(p) + X^2(e)) + \text{H.c.} \quad (41)$$

In these examples, we only cared to check the commutation of operators that are not Hermitian. However, the shift and clock operators in any dimension are unitary, and the following

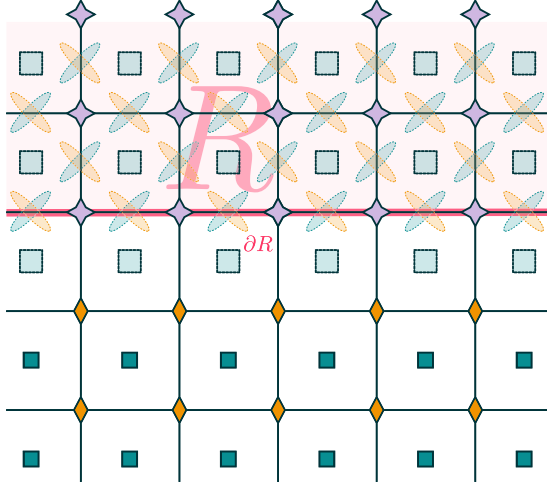


FIG. 21. In the fourth step, the products of removed stabilizers that commute with the measured operators are added. These are the square of plaquette $B^2(p)$ denoted by big dotted light blue squares, and the square of stars $A^2(v)$ denoted by four-sided mauve stars.

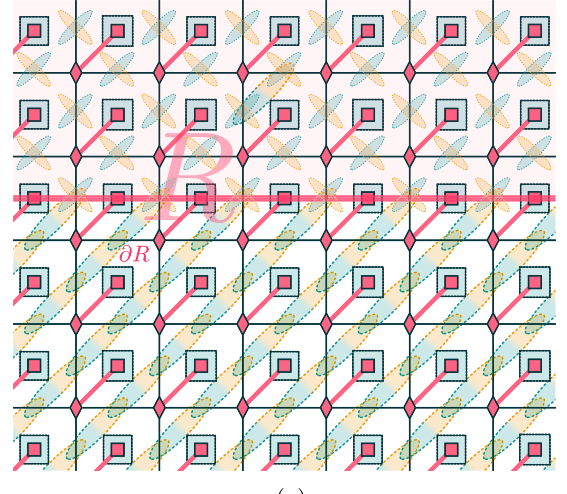
remark shows that one needs only to consider the operator or its Hermitian conjugate.

Remark V.1. Assume we have two commuting unitary operators $[A, B] = 0$, $AA^\dagger = A^\dagger A = \mathbb{I}$ and $BB^\dagger = B^\dagger B = \mathbb{I}$. Then it is simple to show that $[A, B^\dagger] = [A^\dagger, B] = 0$.

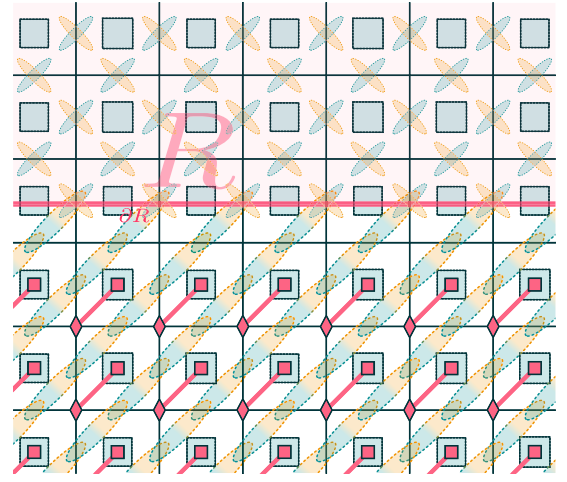
Finally, to construct different adjacent boundaries, one just applies the algorithm sequentially. For example, after obtaining the even boundary for the $\mathbb{Z}_4$ surface code on one spatial boundary, one can proceed to construct the smooth boundary on any other spatial boundary.

Example V.3 (Boundary for the Doubled Semion (DS) phase). The DS phase (20) has only one boundary with vacuum, where the boson b condenses.

First, let us choose a spatial boundary ∂R located at the pink path in Fig. 23(a), taking the region R extending to infinity. Second, we measure the ribbons creating the anyons of the Lagrangian subgroup $\mathcal{L}_{DS} = \{1, b\}$. In this case, the condensation subgroup is generated by the X^2 (yellow tilted

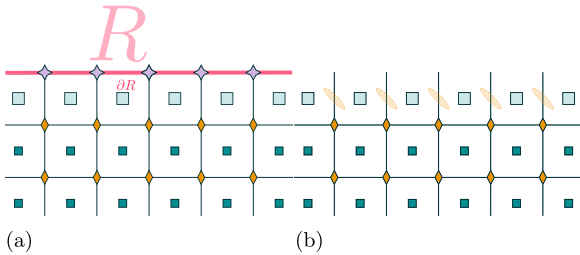


(a)



(b)

FIG. 23. The first three steps of the algorithm to construct the unique boundary of DS. (a) Region R along with its boundary ∂R . After the second step of the algorithm, the X^2 and Z^2 operators are measured inside R and on ∂R . We removed the $X^2 Z^2$ for visual clarity (they are also now dependent on the edge stabilizers X^2 and Z^2). (b) In the third step, the stabilizers that do not commute with the measured operators are removed. These are the F operators (20) that share an edge with a measured X^2 or Z^2 .



(a)

(b)

FIG. 22. Steps five and six in the construction of the even boundary for the $\mathbb{Z}_4$ surface code. (a) In the fifth step, we erase trivialized qudits. These are all qudits inside R or on ∂R . (b) In the sixth step, boundary stabilizers are restricted to their nontrivialized support. $B^2(p)$ act on three edges, while $A^2(v)$ became an X^2 edge operator.

ellipses) or Z^2 (blue tilted ellipses) operators:

$$X^2(e) = \frac{X^2}{\text{yellow ellipse}}, \quad Z^2(e) = \frac{Z^2}{\text{blue tilted ellipse}} \quad (42)$$

The configuration after the first two steps is shown in Fig. 23(a). Third, we remove any F stabilizers that share any edge with the stabilizers Z^2 or X^2 . The X^2 and plaquettes $B^2(p)$ continue to commute with the new operators. In fact, after adding the edge operators X^2 and Z^2 , we can remove the old $X^2 Z^2$ or $B^2(p)$ stabilizers in the bulk of R . After the third step, we reach the configuration in Fig. 23(b). Fourth, the

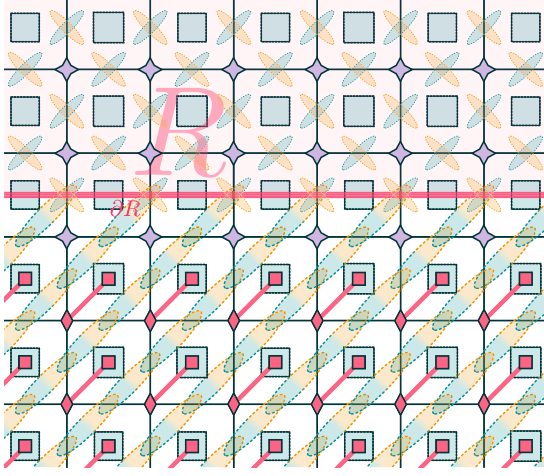


FIG. 24. In the fourth step, the products of removed stabilizers that commute with the measured operators are added. These are the square of star operators $A^2(v)$ denoted by four-sided mauve stars.

products of the old stabilizers that we promote to generators are $A^2(v) = F^2(v, p)B^2(p)$ (purple four-sided star):

$$F^2(v, p)B^2(p) = \begin{array}{c} \text{Diagram of } F^2(v, p) \end{array} \times \begin{array}{c} \text{Diagram of } B^2(p) \end{array} = A^2(v) = \begin{array}{c} \text{Diagram of } A^2(v) \end{array} = \text{Purple four-sided star} \quad (43)$$

These stabilizers have to be added to the Hamiltonian, as shown in Fig. 24.

Fifth, we now remove the trivialized qudits. All edges in the bulk of the region R , or on the pink boundary ∂R , condense to the vacuum. We reach Fig. 25(a).

Observe that the support of the X^2Z^2 operators on the boundary ∂R is reduced to a single edge. Additionally, the new star operators $A^2(v)$ now have support on three edges.

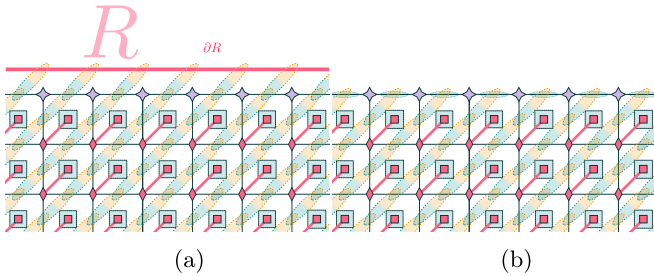


FIG. 25. Steps five and six in the construction of the unique boundary for the DS stabilizer code. (a) In the fifth step, we erase trivialized qudits. These will be all qudits inside R or on ∂R . (b) In the sixth step, boundary stabilizers are restricted to their nontrivialized support. $A^2(v)$ acts on three edges, while the two-edge operator $X^2Z^2(e)$ became a Z^2 single-edge operator.

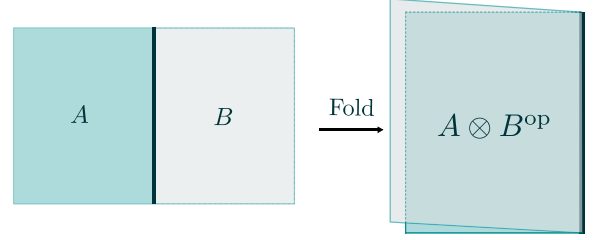


FIG. 26. The folding trick shows how the domain wall between phases A and B is the same as a boundary of the phase $A \otimes B^{\text{op}}$. B^{op} is the opposite phase of B , meaning that it has undergone reflection in the direction perpendicular to the wall.

After restricting the stabilizers on the boundary to their non-trivialized supports, no more qudits are condensed, and we are done. We arrive at Fig. 25(b), which describes the unique DS boundary that condenses the $\{1, b\}$ anyons. This gives the boundary Hamiltonian

$$H_{\text{DS-bdry}} = H_{\text{DS}} - \sum_{v, e \in \partial \mathcal{M}} (A^2(v) + X^2(e)) + \text{H.c.} \quad (44)$$

E. Examples of domain walls construction

The same algorithm to construct boundaries can be used to construct domain walls through the folding trick [47]. In this section, we provide representative examples of this procedure. One advantage of constructing domain walls via stabilizer gauging is that it avoids defining star operators that explicitly depend on the 2-cocycle of the boundary. Since the second cohomology group of Abelian cyclic groups is trivial, $H^2(\mathbb{Z}_m, U(1)) = 0$, we have not encountered this situation in the prior examples. Naturally, a nontrivial 2-cocycle appears when considering the more generic defects (domain walls) within and between topological orders. In particular, $H^2(\mathbb{Z}_n \times \mathbb{Z}_n, U(1)) = \mathbb{Z}_n$. If we denote the group action additively, then we have the following 2-cocycles: $\omega_s((a_1, a_2), (b_1, b_2)) = e^{\frac{2\pi i s}{n}(a_1 b_2 - a_2 b_1)}$ for $s \in \{0, 1, \dots, n-1\}$.

The domain walls of a quantum double based on the group G are the boundaries of the quantum double of the group $G \times G$. This is known as the folding trick [47] and is shown schematically in Fig. 26.

Thus, the algorithm works for the domain walls as well. We just need to account for folding. This is done by first placing the two topological orders on top of each other with the top layer mirrored (i.e., with a reflection around the domain wall). We treat the double layer as a single layer, realizing the quantum double $D(G \times G)$. If it is a quantum double, it has the Hamiltonian

$$H = - \sum_v (A_1(v) + A_2(v)) - \sum_p (B_1(p) + B_2(p)) + \text{H.c.} \quad (45)$$

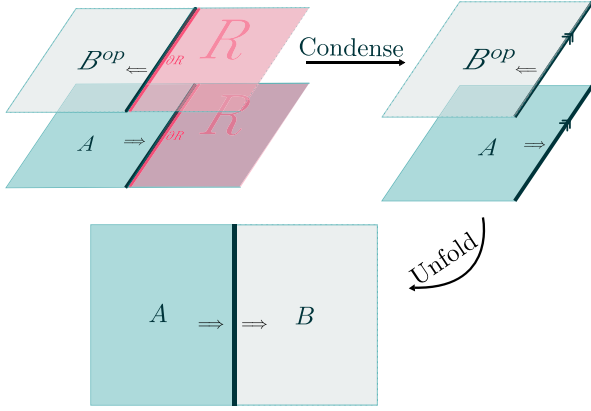


FIG. 27. Constructing a domain wall using condensation. Before running the algorithm, we fold the top layer (B) into its mirrored version (B^{op}). An arrow is drawn to show how the orientation flips from B^{op} to B when folding (unfolding). After running the algorithm, we unfold the top layer to have a domain wall between the two phases.

where the subscript denotes the layer (the top layer is given index 2). Concretely, we have the following stabilizers:

$$A_1(v) = \begin{array}{c} X_1 \\ \hline X_1^\dagger \\ \hline X_1^\dagger \\ \hline X_1 \end{array} = \diamond_1, \quad B_1(p) = \begin{array}{c} z_1^\dagger \\ \hline \square \\ \hline z_1 \end{array} z_1 = \blacksquare_1$$

$$A_2(v) = \begin{array}{c} X_2 \\ \hline X_2^\dagger \\ \hline X_2^\dagger \\ \hline X_2 \end{array} = \diamond_2, \quad B_2(p) = \begin{array}{c} z_2^\dagger \\ \hline \square \\ \hline z_2 \end{array} z_2^\dagger = \blacksquare_2$$

(46)

Note how the stabilizers for the top layer are the mirrored versions of the first-layer stabilizers. We proceed as before, choosing a region R with a boundary ∂R and following Algorithm 2 with the appropriate Lagrangian subgroup. Finally, to obtain the boundary as a defect between the two adjacent topological orders, we unfold the two layers. This procedure is depicted schematically in Fig. 27.

As an example, let us construct the $e \leftrightarrow m$ domain wall of the $\mathbb{Z}_4$ surface code. Note that by changing the definition of the generalized Pauli operators to the correct dimension, this procedure naturally generalizes to any $\mathbb{Z}_n$.

Example V.4 (Domain Wall ($e \leftrightarrow m$) for $\mathbb{Z}_4$ (or $\mathbb{Z}_n$)). Before starting the algorithm, we place the two copies of the surface code on top of each other. Denote the anyons of the bottom layer by $\{1, e_1, m_1, \dots\}$ and for the top layer by $\{1, e_2, m_2, \dots\}$. The top layer is a mirrored version of the bottom layer. This is to ensure we obtain theories with the same orientation after unfolding, as shown in Fig. 27.

We then begin the algorithm. First, we choose a region R extending to infinity. The boundary ∂R lives on the direct lattice in layer 1 but on a dual line in layer 2. This boundary was chosen simply for convenience because of the nature of the condensed anyons, but any other choice works; see Fig. 28. Second, we add the shortest ribbon operators that create the anyons of the Lagrangian subgroup generated by $\mathcal{L}_{e \leftrightarrow m} =$

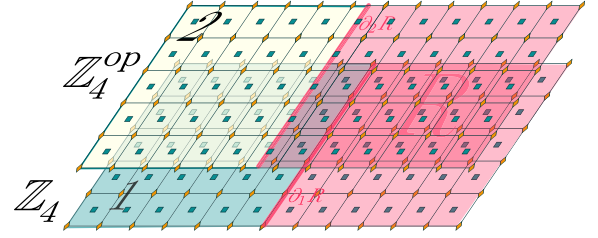


FIG. 28. Region of condensation R for the domain wall $W^{e \leftrightarrow m}$ of the $\mathbb{Z}_4$ surface code. The stabilizers of the top layer are reflected around the domain wall as defined in Eq. (45). The boundary ∂R lives on the direct lattice in layer 1 and on the dual lattice in layer 2.

$\text{Span}(\{e_1 m_2^{-1}, e_2 m_1^{-1}\})$. These operators are generated by

$$e_1 m_2^{-1}(\Rightarrow) = \begin{array}{c} X_2 \\ \hline z_1 \end{array} = \begin{array}{c} \text{orange ribbon} \\ \hline \text{blue ribbon} \end{array}, \quad e_1 m_2^{-1}(\Uparrow) = z_1 \begin{array}{c} X_2 \\ \hline \end{array} = \begin{array}{c} \text{blue ribbon} \\ \hline \text{orange ribbon} \end{array}$$

$$m_1 e_2^{-1}(\Rightarrow) = \begin{array}{c} z_2 \\ \hline X_1 \end{array} = \begin{array}{c} \text{orange ribbon} \\ \hline \text{blue ribbon} \end{array}, \quad m_1 e_2^{-1}(\Uparrow) = X_1^\dagger \begin{array}{c} z_2 \\ \hline \end{array} = \begin{array}{c} \text{blue ribbon} \\ \hline \text{orange ribbon} \end{array}$$

(47)

Here, the arrows indicate the direction of propagation of the anyons (the direction of the ribbons creating them).

Note that we require at least a ribbon with two edges to realize the dyons. And we had to take into account the mirrored nature of layer 2. For example, the conventions for right and left are reversed in layer 2 compared to layer 1. It can be easily checked that they all commute with each other. The configuration after the first two steps is shown in Fig. 29.

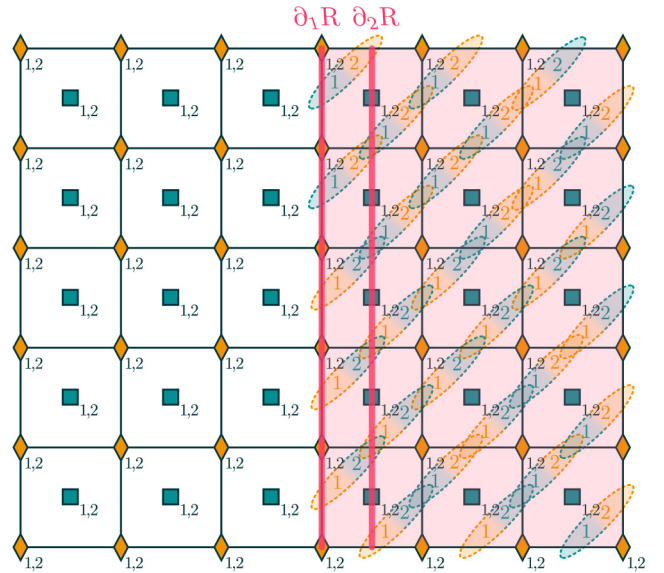


FIG. 29. Top view of the two layers after the first two steps of the algorithm. The short ribbon operators are added to the region of condensation R . Note how the boundary $\partial R = \partial_1 R \cup \partial_2 R$ has a component on the direct lattice on layer 1 and on the dual lattice on layer 2 denoted by the lines $\partial_1 R$ and $\partial_2 R$ respectively. The stars and plaquettes here are the summation of the operators of the two layers, Eq. (45). Only a subset of the measured ribbon operators is shown as they overlap.

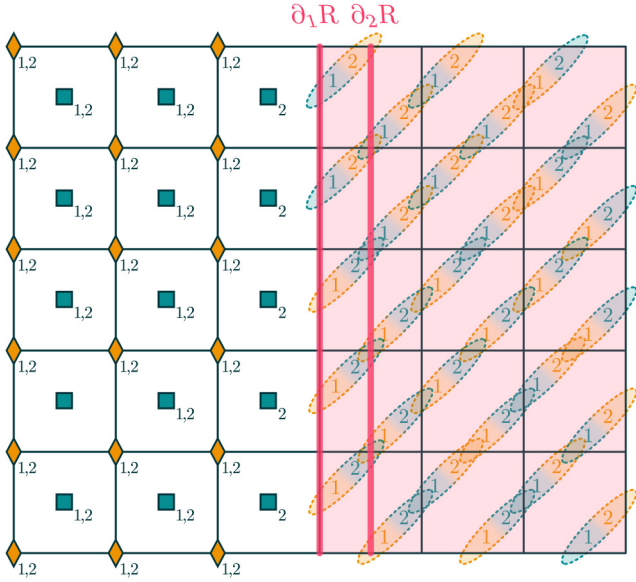


FIG. 30. After the third step, the stabilizers that do not commute with the ribbon operators are removed. Note how the plaquettes adjacent to the boundary $\partial_1 R$ now live in the second layer only.

In the third step, we remove the old stabilizers that do not commute with the new ones. These will include all the old star and plaquette stabilizers of both layers in the bulk of the region R . In addition, we remove the star operators of both layers on the boundary $\partial_1 R$. We remove the plaquettes $B_1(p)$ of the second layer adjacent to the boundary $\partial_1 R$. The configuration after the third step is shown in Fig. 30.

In the fourth step, we form new stabilizers that are products of the old ones and commute with the new stabilizers. These are given by products of the removed stars with their adjacent removed plaquettes,

$$A_1^\dagger(v)B_2(p) = \begin{array}{c} \begin{array}{|c|} \hline z_2^\dagger \\ \hline \end{array} \\ \begin{array}{|c|} \hline z_2 x_1^\dagger \\ \hline \end{array} \\ \begin{array}{|c|} \hline x_1 \\ \hline \end{array} \end{array} = \begin{array}{c} \begin{array}{|c|} \hline 2 \\ \hline \end{array} \\ \begin{array}{|c|} \hline 1 \\ \hline \end{array} \end{array}, \quad (48)$$

$$B_1^\dagger(p)A_2(v) = \begin{array}{c} \begin{array}{|c|} \hline z_1 x_2 \\ \hline \end{array} \\ \begin{array}{|c|} \hline z_1^\dagger \\ \hline \end{array} \end{array} = \begin{array}{c} \begin{array}{|c|} \hline 1 \\ \hline \end{array} \\ \begin{array}{|c|} \hline 2 \\ \hline \end{array} \end{array}$$

After adding the new stabilizers to the model, we reach the configuration shown in Fig. 31. In the fifth step, we erase trivialized qudits in Fig. 32. These are all the qudits in the bulk of R or on its two-component boundary $\partial_1 R$ and $\partial_2 R$. Note that the edges of the second layer that lie above $\partial_1 R$ are not condensed. After condensing qudits, we also removed all stabilizers that only act on trivialized qudits for clarity; see Fig. 33.

The sixth step is to restrict the stabilizers at the boundary to their nontrivialized support. This results in the following

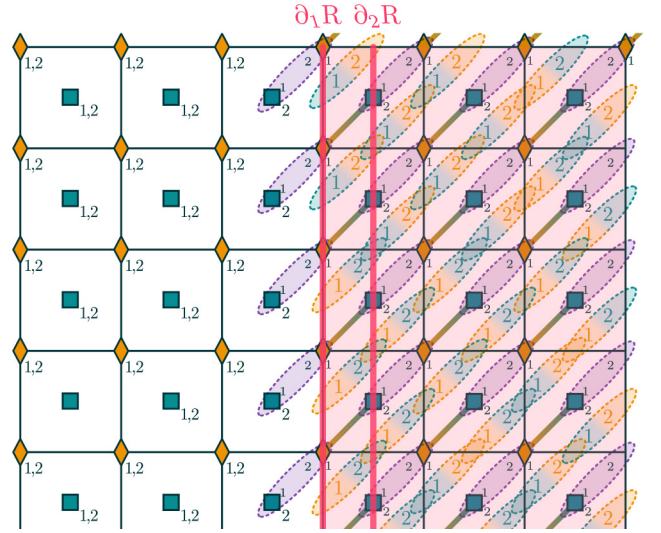


FIG. 31. After the fourth step, products of old stabilizers that commute with the ribbons are added, Eq. (48).

stabilizers:

$$XZ(e) = \begin{array}{c} \begin{array}{|c|} \hline x_1 \\ \hline \end{array} \end{array} \xrightarrow{\text{Unfold}} \begin{array}{c} \begin{array}{|c|} \hline x \\ \hline \end{array} \end{array} = \begin{array}{c} \begin{array}{|c|} \hline 1 \\ \hline \end{array} \end{array}$$

$$T(v,p) = \begin{array}{c} \begin{array}{|c|} \hline z_1 x_2 \\ \hline \end{array} \\ \begin{array}{|c|} \hline z_1^\dagger \\ \hline \end{array} \end{array} \xrightarrow{\text{Unfold}} \begin{array}{c} \begin{array}{|c|} \hline z \\ \hline \end{array} \end{array} = \begin{array}{c} \begin{array}{|c|} \hline 1 \\ \hline \end{array} \end{array} \quad (49)$$

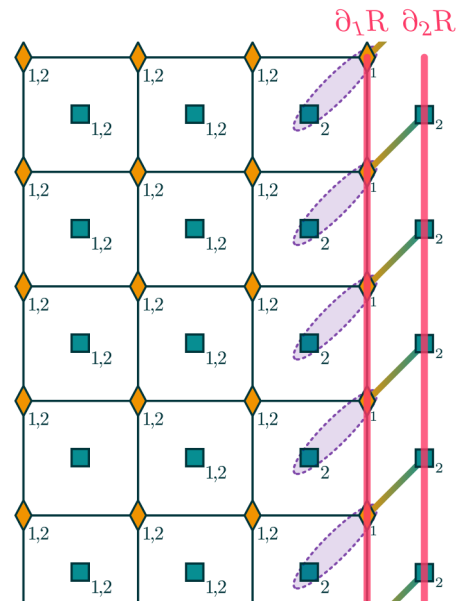


FIG. 32. After the fifth step, the qudits in R or on the boundary $\partial R = \partial_1 R \cup \partial_2 R$ are erased. In particular, on $\partial_1 R$ only one qudit lives now.

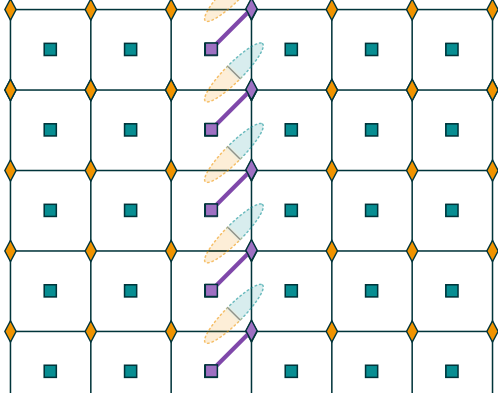


FIG. 33. In the last step of the algorithm, we restrict the stabilizers to their nontrivialized support. After unfolding, we have the twist operators $T(v, p)$ (purple) that live on a vertex and an adjacent plaquette, and also the $XZ(e)$ (ellipse) operators connecting two edges.

Finally, we unfold layer 2 to be put on the right. This results in the stabilizer of the bulk of the two layers being the same again, redoing the horizontal reflection. At each edge, now only one qudit lives, so we can drop the layers' index unambiguously. Importantly, right at the $\partial_1 R$ boundary, only the qudits in the second layer survive. We reach the $W^{e \leftrightarrow m}$ domain wall for $\mathbb{Z}_4$. This is also the domain wall for this duality for any $\mathbb{Z}_n$ by changing the definitions of the generalized Pauli matrices accordingly. In particular, for $\mathbb{Z}_2$ we interpret the X and Z as the ordinary two-dimensional Pauli matrices, which are Hermitian.

It is clear that one can also construct finite domain walls by only condensing a smaller region than considered in Fig. 27. In particular, we will be interested in constructing certain domain walls between a parent bulk theory [e.g., $D(\mathbb{Z}_4)$] and a finite patch within it that is obtained by condensation (e.g., DS). In this case, one can directly apply the condensation without having to use the folded picture. We use this to illustrate one way the domain wall between $D(\mathbb{Z}_4)$ and DS can be constructed.

Example V.5 (Domain wall between $D(\mathbb{Z}_4)$ and DS). Since DS is obtained from $D(\mathbb{Z}_4)$ by condensation, one can condense anyons creating DS in only a finite patch of space. These are the $e^2 m^2$ anyons of $\mathbb{Z}_4$. First, we choose a region R with boundary ∂R . We want to construct the domain wall that doesn't condense any of the DS phase anyons. Second, we add the short ribbon operators $X^2 Z^2$ that create the $e^2 m^2$ anyons inside R or on ∂R . After the first two steps, we arrive at Fig. 34.

Third, we remove any $A(v)$ or $B(p)$ stabilizers that do not commute with the measured stabilizers $X^2 Z^2$. After the third step, we reach the configuration shown in Fig. 35. Fourth, the products of the old stabilizers that we promote to generators are $F(v, p) = A(v)B(p)$ (a connected pink star and a plaquette) and also the $B^2(p)$ operators (big dashed square). These stabilizers have to be added to the Hamiltonian as shown in Fig. 36.

Fifth, we now remove the trivialized qudits. In this case, no edges are condensed. Subsequently, the sixth step is also

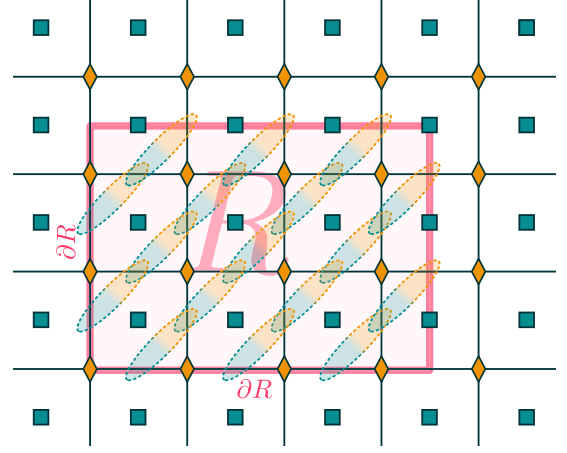


FIG. 34. After the two steps of the algorithm, the edge operators $X^2 Z^2$ creating $e^2 m^2$ are measured in the region R and also on the boundary ∂R .

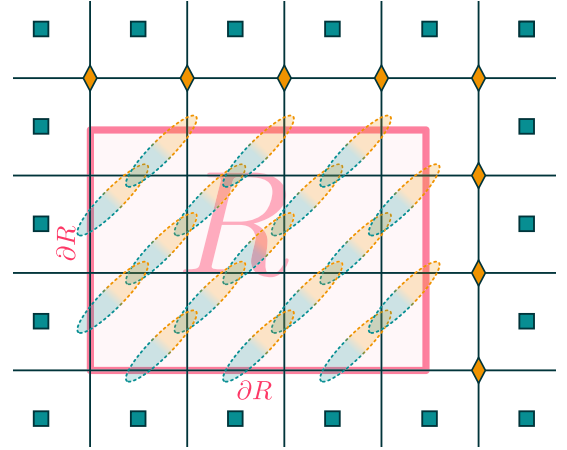


FIG. 35. After the third step, old stabilizers that do not commute with the measured ribbons are removed. These are the star and plaquette operators of $D(\mathbb{Z}_4)$ that share at least one noncommuting edge with $X^2 Z^2$.

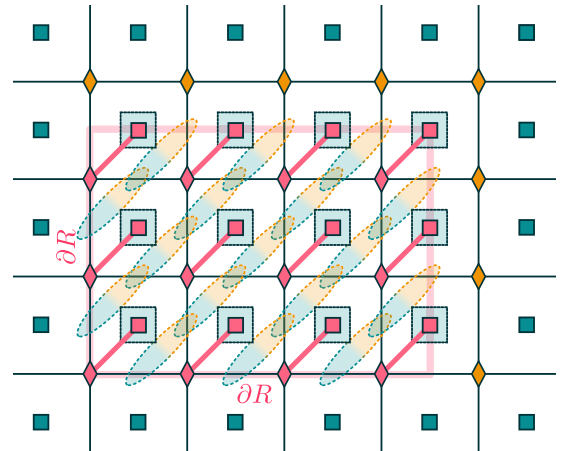


FIG. 36. In the fourth step, the products of removed stabilizers that commute with the measured operators are added. These are the $F(v, p)$ (a connected pink star and a plaquette) and the $B^2(p)$ operators (big dashed square).

trivial. This example shows that in a certain situation, one does not need to do the folding. In Appendix A, we corroborate this by deriving the domain walls using the folding method.

VI. CALCULATING LOGICAL GSD

To confirm that our Abelian (twisted) quantum doubles are quantum error-correcting codes, we follow Kitaev's original logic [4] and rigorously compute the ground-state degeneracy. This topological degeneracy corresponds to the dimension of the logical code space that is acted upon by logical operators as previously described in Sec. IV.

The ground-state projector has, by definition, the ground states in its image and all excited states in its kernel. Thus, the most straightforward method to compute the ground-state degeneracy is to explicitly construct the ground-state projector and take its trace [184]. However, because of the tensor product structure, this is not scalable in practice.

In this section, we detail two complementary methods for efficiently calculating, modifying, and reasoning about the GSD for composite topological codes. The first method utilizes the microscopic stabilizer description, where both local (e.g., stabilizer order) and global (e.g., boundary conditions) information is used to calculate the GSD. This is detailed below in Sec. VI. Alternatively, one can use a coarse-grained macroscopic description to compute the GSD using knowledge of logical ribbon operators and pants decomposition as detailed in Sec. VIB, which follows Ref. [79].

A. GSD using microscopic stabilizers

In this subsection, we assume that we are given a concrete stabilizer layout for a system, and then compute its GSD. Begin with an unconstrained Hilbert space of dimension D . In the simple and most common case, all qudits have the same dimension d and thus, $D = d^{\#E}$ where $\#E$ is the number of edges in the system, each supporting a single qudit [66]. We then consider the n mutually commuting stabilizer group generators $\{O_i\}$, each with order m_i (i.e., $O_i^{m_i} = 1$). Note that, in general, for higher-dimensional qudits ($d > 2$), these generators do not have to be Hermitian.

Next, one must determine and eliminate the l lingering dependencies between the presumed generators. That is, remove l of the presumed generators, which are actually generated from products of the others and are not truly independent. In the examples we consider below, the dependencies are organized with respect to generators of a given order. Since each *independent* generator O_i simultaneously constrains the ground-state manifold, by a factor $1/m_i$, the ground-state degeneracy is

$$\text{GSD} = \frac{D}{\prod_{|m|=2,3,\dots,m_{\max}} m^{n_m - l_m}}. \quad (50)$$

Here, n_m and l_m are the numbers of generators and dependencies that have order m , respectively.

By definition, the GSD of topological theories does not depend on the system size nor its microscopic details, e.g., the particular triangulation of the surface. As we see in the examples below, we can therefore use any minimal example

to calculate the GSD and then extend the result for a larger or more complicated, but topologically equivalent, system.

Updating GSD. If the code is regauged to add boundaries or domain walls, as described in Algorithm 2, we may sequentially update the GSD by only accounting for the changes. We summarize the results of applying Algorithm 2 as they relate to constraint dimensions and dependencies. This summary culminates in equations for computing the updated GSD' as a function of the old one. Suppose we know the GSD for a certain configuration realizing stabilizer group $\mathcal{S}_H$ with generators $s_H = \{O_i\}$ with $O_i^{m_i} = 1$ and dependencies l_m . We can apply the algorithm to obtain a different configuration with Hilbert space dimension D' , stabilizer group $\mathcal{S}_{H'}$, generators $s_{H'} = \{O'_j\}$, with $O'_j{}^{m'_j} = 1$ and dependencies l'_m . After the algorithm is complete, a set of new generators $O'_j \in s_{H'}$ are added to the new Hamiltonian H' ,

$$s_{\text{new}} = \{O'_j \in s_{H'} \setminus s_H\}. \quad (51)$$

In addition, a set of generators $O_i \in s_H$ are removed from the old Hamiltonian H ,

$$s_{\text{rem.}} = \{O_i \in s_H \setminus s_{H'}\}. \quad (52)$$

Finally, the erasure of qubits may result in a change in the dimension of the Hilbert space. These three components (two sets of generators plus the changes in the total dimension of the Hilbert space) are all the data needed from the algorithm to relate the new GSD' to the old one.

The introduction of new, or removal of old, degrees of freedom is accounted for by the ratio of total Hilbert space dimensions $r_D = D'/D$.

Order by order, the total change in the constraints is computed by taking the difference of the number of order m constraints $\Delta n_m = n'_m - n_m$ and by also tracking the change in dependencies $\Delta l_m = l'_m - l_m$.

After this bookkeeping, we can write the new degeneracy in terms of the old one as

$$\text{GSD}' = \text{GSD} \frac{r_D}{\prod_{|m|=2,3,\dots,m_{\max}} m^{\Delta n_m - \Delta l_m}}. \quad (53)$$

1. Examples using minimal stabilizer models

To get a feeling for Eq. (53), let us start with Example VI.1 where we modify the surface code in Fig. 37 with $\text{GSD} = 2$ to make a cylinder with two rough boundaries.

Example VI.1 ($\mathbb{Z}_2$ on a cylinder with two rough boundaries). Starting from the surface code, suppose we want to build the configuration with $\mathbb{Z}_2$ on a cylinder with two rough boundaries as shown in Fig. 38.

We track the changes from the layout in Fig. 37. The edges went from 18 to 15 and thus,

$$r_D = 2^{18-15} = 2^{-3}.$$

There are only order-2 operators. The number of plaquettes did not change while the number of stars decreased by 2, so

$$\Delta n_2 = -2.$$

Finally, while the dependencies were $l_2 = 0$ in the surface code, here the product of all plaquettes gives us the identity

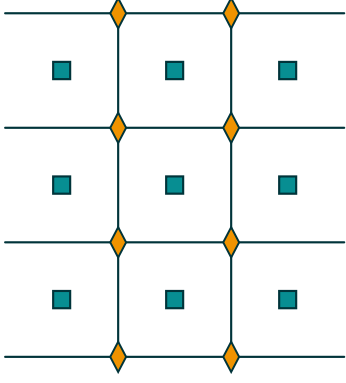


FIG. 37. $D(\mathbb{Z}_2)$ surface code with opposite smooth and rough boundary conditions, and which has $\text{GSD} = 2$.

$\prod_p B(p) = \mathbb{I}$, so we have $l'_2 = 1$. This gives

$$\Delta l_2 = 1.$$

We then compute the GSD from the direct application of Eq. (53) as

$$\text{GSD}' = \frac{r_D}{2^{\Delta n_2 - \Delta l_2}} = \frac{2^{-3}}{2^{-2-1}} \text{GSD} = 2. \quad (54)$$

Example VI.2 (Toric code through folding a cylinder). A minimal example for the toric code is depicted in Fig. 39. The toric code can be made starting with a cylinder with nine faces, 18 edges, and eight vertices, as in Fig. 39. Turning it into a torus equates to enforcing periodic boundary conditions on the remaining two boundaries. This process involves removing three edges, removing one star operator as it becomes dependent on the others, and removing three plaquettes. We then have

$$\begin{aligned} \text{GSD}_{\text{Toric}} &= \frac{2^{\Delta(\#E)}}{2^{\Delta(\#A_v)} 2^{\Delta(\#B_p)}} \times \text{GSD}_{\text{Cylinder}} \\ &= \frac{2^{-3}}{2^{-1} 2^{-3}} \times 2 = 4 \end{aligned} \quad (55)$$

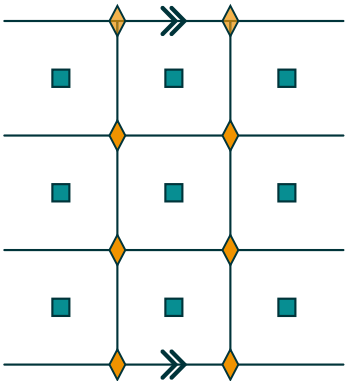


FIG. 38. Minimal stabilizer model for the $\mathbb{Z}_2$ on a cylinder with two rough boundaries. The double arrows indicate periodic boundary conditions. Consequently, the semi-transparent diamonds should not be counted.

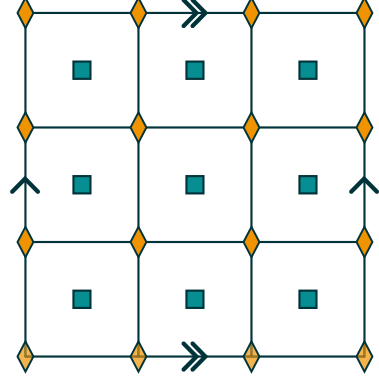


FIG. 39. Minimal stabilizer model for toric code.

This method of constructing a minimal model to count the degeneracy becomes especially useful for more complicated configurations. All one needs to do is make sure that no product of the operators gives the identity. This is the same condition as requiring that all the operators are independent. An interesting domain wall, which was discussed is the $e \leftrightarrow m$ of the toric code in Sec. V 4. We can compute the GSD for this model using this simple model construction.

Example VI.3 ($\mathbb{Z}_2$ on a Torus with a contractible $e \leftrightarrow m$ twist). A minimal example for the toric code with a contractible twist is depicted in Fig. 40. We perform Algorithm 2 from the toric code and carefully count the changes. We remove two A stabilizers and two B stabilizers and add one T , U , and Q . They are all of order $m = 2$. In total, we have $\Delta n_2 = 3 - 4 = -1$. Now, let us examine how the dependencies are modified. Since the product of the remaining A operators, and similarly that of all B operators, no longer constitutes the global identity, the two old dependencies are removed. However, note that the product of all A s, B s, Q s, U s, and T s does constitute the identity, so there is one dependency of order 2. Altogether, we have $\Delta l_2 = 1 - 2 = -1$ and since no qubits were added or removed $r_D = 1$. The exponent on m in Eq. (53) then becomes $\Delta n_m - \Delta l_m = 0$ such that

$$\text{GSD}_{1\text{-Twist}} = \frac{\text{GSD}_{\text{TC}}}{2^{-1+1}} = 4. \quad (56)$$

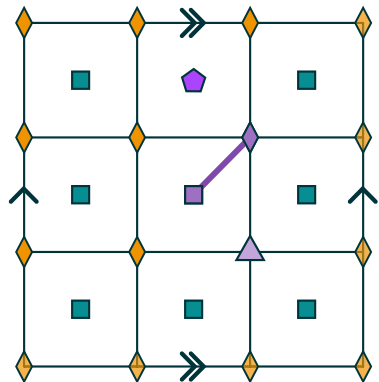


FIG. 40. Toric code with a single finite twist.

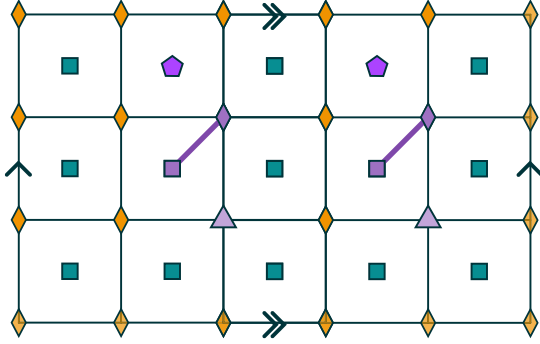


FIG. 41. Stabilizers for the Toric code with two twists.

This is correct because the codespace is the same as before, although a global phase factor -1 may now be applied due to the presence of the lone twist [56].

What about an additional twist? We can again use Eq. (53) to obtain a formula for N twists as we now show.

Example VI.4 ($\mathbb{Z}_2$ on a torus with N twists). Take the configuration with one twist, described by Eq. (56), as the initial configuration with fourfold GSD. Adding another twist (not connected to the first one) does not change the dimension or number of degrees of freedom ($r_D = 1$). We will again add one T , one U , and one Q operator; see Eq. (49), remove two star operators, and remove two plaquette operators as shown in Fig. 41.

Now, let us count the dependencies. Since we already had one twist, which combined the two dependencies of the toric code into a single dependency, the total number of dependencies does not change this time, $\Delta l_2 = 0$. The new degeneracy is

$$\text{GSD}_{2\text{-Twists}} = \text{Eq. } 56 \times 1 \times 2 \times 1 = 8. \quad (57)$$

We can recursively apply the same calculation when adding more twists, such that for N twists

$$\text{GSD}_{N\text{-Twists}} = \text{GSD}_{\text{Toric } (N-1)\text{-Twists}} \times 2 = 2^{N+1}. \quad (58)$$

This result is interpreted as the logical Hilbert space of N twists, coming with $N - 1$ qubits appended to the original toric code's fourfold degeneracy.

Example VI.5 ($\mathbb{Z}_2$ on a Torus with a non-contractible twist). Let us restart with a clean version of the 18-qubit, 9-plaquette, and 9-star toric code. The new configuration with the noncontractible twist is shown in Fig. 42.

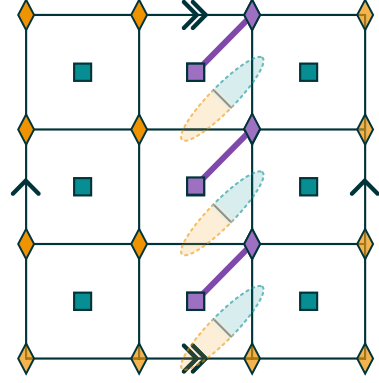
We summarize the changes from the toric code using Table V.

We then immediately read off the degeneracy as

$$\text{GSD}_{\text{non-con. Twist}} = \frac{1}{2^{0-(-1)}} \times \text{GSD}_{\text{Toric Code}} = 2 \quad (59)$$

This example follows the result of Ref. [185] where it was shown that adding a noncontractible twist gauges the toric code's $\tilde{X}_1 \tilde{Z}_2$ degree of freedom, hence adding one *logical* constraint resulting in a *single* logical qubit [185].

Remark VI.1. In all the examples discussed so far, changing from $\mathbb{Z}_2$ to $\mathbb{Z}_m$ we just change the order of the operators from 2 to m in Eq. (53).

FIG. 42. Stabilizer model for the $\mathbb{Z}_2$ toric code with one noncontractible twist (an $e \leftrightarrow m$ domain wall along one handle of the torus).

Having developed Algorithm 2 to gauge stabilizer codes, and Eq. (53) to count and update degeneracies, we are now in a position to prove the results presented in Sec. IV D 4, which concern twisted quantum doubles coexisting. Since the twisted quantum doubles can be formed by condensation from quantum doubles [42], the degeneracy of such systems can be proved with minimal illustrative models. Continuing with the arguments above, we begin by considering a $D(\mathbb{Z}_4)$ toric code with $\text{GSD} = 4^2 = 16$. We now proceed to condense finite spatial regions and determine the GSDs.

Example VI.6 ($\mathbb{Z}_4$ with a contractible DS patch). Start from the $D(\mathbb{Z}_4)$ toric code and condense a single region that contains the DS phase. Any model with a single contractible (meaning it does not wrap around either handle of the torus) patch has the same degeneracy as this construction. Before condensing the DS patch, we had $\prod_v A_v = 1$ and $\prod_p B_p = 1$, which meant that one of the plaquettes and one of the star operators is dependent for a total of two order four dependencies ($l_4 = 2$) and zero order two dependencies ($l_2 = 0$) such that $l = l_2 + l_4 = 2$. To condense a DS patch, we need to add two $X^2 Z^2$ ribbons, as in Fig. 43.

Afterward, we remove three star operators and three plaquette operators that do not commute with the $X^2 Z^2$ ribbons. We then add products of the removed operators that commute

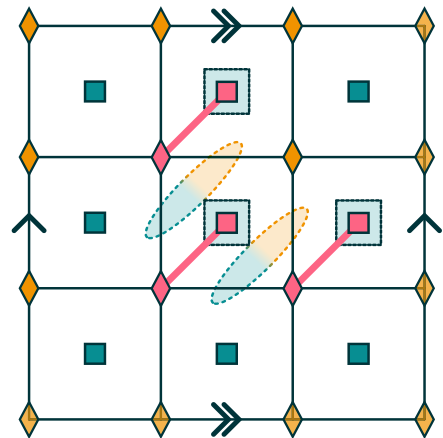
FIG. 43. Finite contractible DS patch embedded in a $D(\mathbb{Z}_4)$ toric code.

TABLE VI. Changes from a $\mathbb{Z}_4$ code to a $\mathbb{Z}_4$ code with a contractible DS patch.

Order	Operators			Δn_m	Δl_m
$m = 2$	$+2 X^2 Z^2$	$+3 B^2$		+5	+1
$m = 4$	$+3 F$	$-3 A$	$-3 B$	-3	-1

with the new stabilizers. These are the $F(v, p)$ and $B^2(p)$ operators; see Eq. (19). We add three F operators on the removed vertices and three $B^2(p)$ operators on the removed plaquettes.

Note that we have two new constraint relations again. First, we have $\prod_{v,p} A(v)B(p)F(v, p) = 1$ because this is the same as multiplying all star and plaquette operators before adding the DS patch. Second, we have $(\prod_{p \in DS} B^2(p))(\prod_{p \notin DS} B(p))^2 = 1$. Here, the product is over the new plaquettes, which live inside or adjacent to the right or top of the DS patch, with the square of the rest of the old plaquettes that were not removed. Thus, $l' = 2$ and $r_l = 2/2 = 1$. The crucial observation is that the first constraint removes an operator with order four, for example, one $F(v, p)$, while the second constraint means that we are double-counting an operator of order 2, taken here as $B^2(p)$ [186]

The counting of stabilizers is then summarized in Table VI. We then immediately read off the degeneracy as

$$\text{GSD}_{\mathbb{Z}_4-1\text{-DS}} = \frac{1}{2^{5-1}4^{-3-(-1)}} \times \text{GSD}_{\mathbb{Z}_4} = 16. \quad (60)$$

Intuitively, the new contractible patch does not obstruct any of the old loops (logical operators) and does not add any linearly independent ground states either.

We can add more DS patches, and the situation becomes similar to the case of $\mathbb{Z}_2$ with many twists.

Example VI.7 ($\mathbb{Z}_4$ on a Torus with NDS patches). Starting with example VI.6 with one DS patch, we can add one more patch as shown in Fig. 44.

In the old configuration we had the relations

$$\prod_{v,p} A(v)B(p)F(v, p) = 1, \\ \left(\prod_{p \in DS} B^2(p) \right) \left(\prod_{p \notin DS} B(p) \right)^2 = 1.$$

As before, in the second constraint, the product is over the new plaquettes which live inside or to the right or top of the

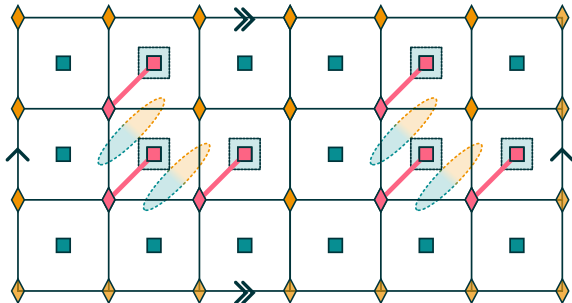


FIG. 44. 2 contractible DS patches.

TABLE VII. Table summarizing changes from $\mathbb{Z}_4$ code to $\mathbb{Z}_4$ and DS half and half contractible code.

Order	Operators			Δn_m	Δl_m
$m = 2$	$+2 X^2 Z^2$	$+3 B^2$		+5	0
$m = 4$	$+3 F$	$-3 A$	$-3 B$	-3	0

DS patch and the square of the rest of the old plaquettes that were not removed. This means $l_2 = 1$ and $l_4 = 1$. To form the second patch, we first remove three star operators $A(v)$ and three plaquette operators $B(p)$. We proceed by adding two $X^2 Z^2$ operators, three new $F(v, p)$ operators, and three new plaquette operators $B^2(p)$. The constraints remain the same. Consequently, $l'_2 = 1$, $l'_4 = 1$ and $\Delta l_2 = \Delta l_4 = 0$. The counting of the stabilizers and dependencies is then summarized in Table VII.

The ground-state degeneracy for one contractible patch of DS is then

$$\text{GSD}_{\mathbb{Z}_4-2\text{-DS}} = \frac{1}{2^{5-0}4^{-3-(0)}} \times \text{GSD}_{\mathbb{Z}_4-1\text{-DS}} = 32. \quad (61)$$

Adding more patches does not change the constraints, as we saw. For N patches, we then have the following degeneracy, which is valid for $N \geq 1$:

$$\text{GSD}_{\mathbb{Z}_4-N\text{-DS}} = 4^{\frac{N+3}{2}}. \quad (62)$$

Example VI.8 (Torus with half $\mathbb{Z}_4$ and half DS phase). Let us start with a $\mathbb{Z}_4$ on a torus. We want to calculate the degeneracy of a phase where half of the torus is a $\mathbb{Z}_4$ QD, and the other half is in the doubled semion (DS) phase as shown in Fig. 45. Since here the DS lives on a cylinder, the boundary conditions are different from just a DS patch.

In this case, we also need to specify the order of the operators since they are different which is shown in Table VIII.

We then immediately read off the degeneracy as

$$\text{GSD}_{\mathbb{Z}_4\text{-DS}} = \frac{1}{2^{12}4^{-6-(-1)}} \times \text{GSD}_{\mathbb{Z}_4} = 8. \quad (63)$$

This can be understood intuitively since the full $\mathbb{Z}_4$ loops can no longer traverse the torus as some of them get identified or confined when they pass through the DS phase. Another way to calculate the GSD will make this clearer in Sec. VI B. We saw in Example VI.6 that when the DS phase occupied a contractible region, the degeneracy was the same as without the DS phase. The stabilizer counting using minimal models is then a powerful tool for exploring modifications to topological orders in a sequential way. Paired with Algorithm 2, one can imagine using a computer program to automate the construction of topological quantum error-correction codes.

TABLE VIII. Summary of changes from $\mathbb{Z}_4$ toric code to $\mathbb{Z}_4$ sharing the torus with DS phase.

Order	Operators			Δn_m	Δl_m
$m = 2$	$+6 X^2 Z^2$	$+6 B^2$		+12	1
$m = 4$	$+6 F$	$-6 A$	$-6 B$	-6	-1

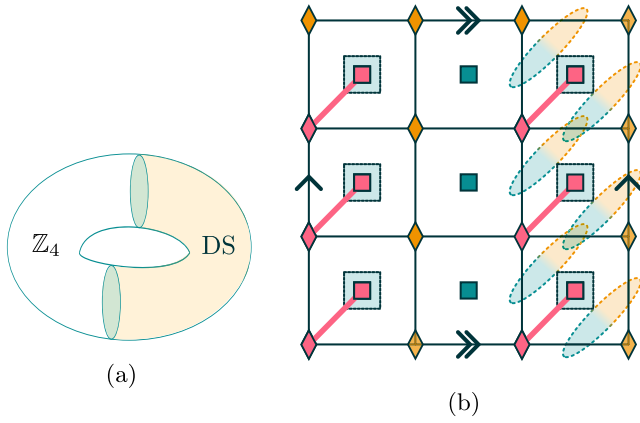


FIG. 45. $\mathbb{Z}_4$ toric code with half the torus as DS phase. (a) Sketch for the configuration with $\mathbb{Z}_4$ toric code with the DS phase occupying half the torus. (b) Minimal lattice model for the configuration on the right. Compare with Example VI.11, which uses a different method.

B. GSD using macroscopic pants decomposition

Dimensional counting is useful for deriving the logical properties, such as ground-state degeneracy, from microscopic configurations, as we saw in the last subsection. This subsection offers a complementary, top-down, rather than bottom-up, intuitive methodology, involving only high-level macroscopic information, for calculating the GSD by counting the number of noncontractible logical loops of anyons in our model. This is based on Ref. [79]. We use these methods to explore and validate results for the introduced composite codes.

Example VI.9 ($D(\mathbb{Z}_2)$ on a Cylinder with two smooth boundaries). Let us take the cylinder with smooth boundary conditions, as shown in Fig. 46.

While there are two noncontractible loops of anyons around the circumference of the cylinder, only the Z loop is nontrivial. Further, an X string between the two smooth boundaries of the cylinder is noncontractible and nontrivial. The Z loop and the X string anticommute, and we have $\text{GSD} = 2$. Evidently, all we needed was the information about the topological surface the theory lives on, along with the types of boundaries it has. The specific microscopic details were not used.

The heuristic method of counting loops can be readily generalized to any orientable surface with or without boundaries. The exact stabilizer formulation is not needed to compute the

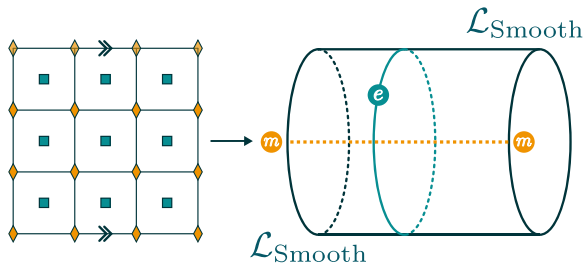


FIG. 46. (Left) Microscopic stabilizer model for the $\mathbb{Z}_2$ code on a cylinder with two smooth boundaries. (Right) The macroscopic topological data needed to compute the GSD of the model.

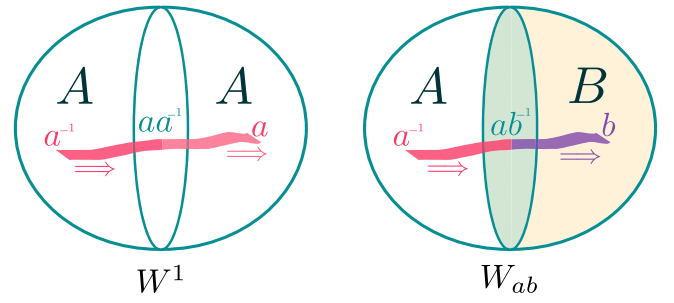


FIG. 47. (Left) The trivial domain wall denoted by W^1 transports a to a . Alternatively, this transparent domain wall condenses aa^{-1} . (Right) A domain wall W can transport anyon a to b , thus condensing ab^{-1} . The W_{ab} matrix entry is the dimension of the fusion space of the anyon ab^{-1} on the domain wall W on a sphere. Intuitively, it measures in how many independent ways the anyon a in phase A tunnels into an anyon b in phase B .

ground-state degeneracy once the topological data are fixed. The subtlety is that one should avoid overcounting or counting dependent loops. This issue can be tackled for a broad class of topological codes, as we now discuss.

In topology, it is important to systematically classify how arbitrary surfaces are constructed in terms of simpler primitive structures. For example, how standard spheres and holes can be glued to form any connected and simply-connected two-complex is a well-studied problem [187]. In this spirit, this section uses the results of Ref. [79] to validate the results in Sec. IV D 4.

The construction relies on two facts. First, as shown in Fig. 48 below, any orientable 2D manifold can be decomposed

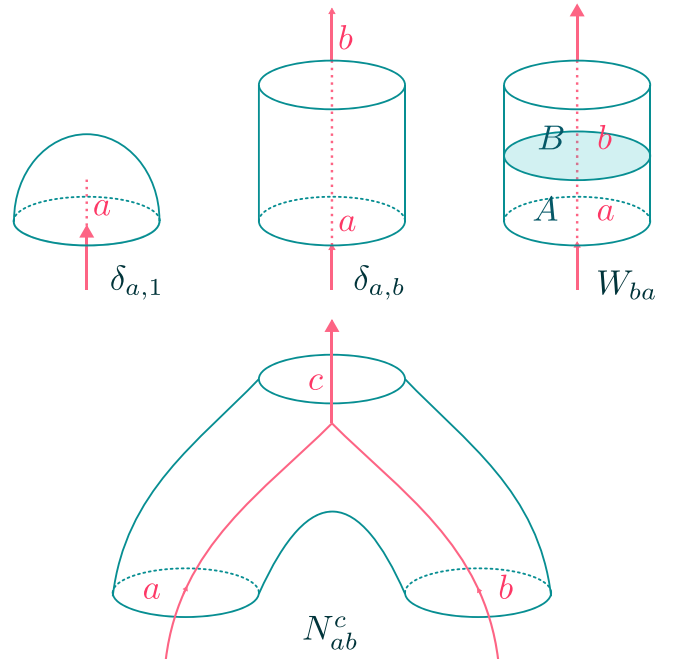


FIG. 48. Any orientable 2D manifold can be decomposed into caps, cylinders, and pants. Each component is associated with a tensor with indices in the anyons of the bulk theory.

into caps, cylinders, and pants [79,188]. Second, there is a correspondence between domain walls (boundaries) and a tunneling or transfer matrix that describes the domain wall's action on anyons [79].

To define the tunneling matrix W [79], consider the case where two phases A and B share a domain wall W between them as shown in Fig. 47. Note that we use W for the domain wall and its associated matrix. We define W_{ab} to be the dimension of the fusion space ($\mathcal{V}$) of this configuration on a sphere (S^2): $W_{ab}^1 := \dim[\mathcal{V}(S^2, b, W, a^{-1})] \in \mathbb{N}$ [79]. This is also the number of independent tunneling channels that tunnel the anyon a in A to the anyon b in B . Since gapped domain walls respect the braiding statistics of the two phases, we have the following relations [79]:

$$WS^B = S^A W, \quad WT^B = T^A W. \quad (64)$$

Here, S^A matrix encodes braiding statistics of anyons in phase A . The T^A matrix is a diagonal matrix that encodes their topological twist or spin. We will also only consider stable domain walls whose GSD cannot change due to local perturbations. This gives us the constraint [79]

$$W_{ia} W_{jb} \leq \sum_{kc} (N^D)_{ij}^k W_{kc} (N^A)_{ab}^c, \quad (65)$$

where the $(N^A)_{ab}^c \in \mathbb{N}$ is the dimension of the fusion channels of anyons a and b into c . Using this construction, and given a certain spatial topology, any surface can be decomposed into caps, cylinders, and W matrices corresponding to different domain walls between phases. This is shown in Fig. 48. It actually suffices to consider caps, domain walls, and pants since a cylinder can be formed with one pants and capping one of its boundaries. The two types of cylinders are included for convenience. The GSD can be computed by enumerating how many nontrivial independent anyonic loops exist. This process corresponds to concatenating the topological components of each configuration, known as the pants' decomposition.

Systematically, the method can be described as follows. First, we begin by taking as input a 2D surface $\mathcal{M}$ that can have multiple spatial boundaries $\partial\mathcal{M}_i$. The surface may also be divided into different topological phases with appropriate domain walls W^j between them. Boundaries are described as domain walls with the vacuum W^i for uniformity. Second, we decompose the 2D surface using a pants decomposition into the 2D components described in Fig. 48. This isolates different bulk phases that have domain walls between them into different elementary 2D components. Multiple pants decompositions exist for the same configuration, and it is not crucial to use the minimal one. The different decompositions are related by a set of topological moves that naturally do not change the topology of the surface nor affect the GSD calculated here [79,188]. We also do not discuss how this decomposition is formally carried out, since in all the examples, finding a decomposition is relatively easy. Third, we attach to every 2D component its appropriate tensor from Fig. 48. The tensors take values in the anyon labels of the bulk (boundary) of their respective topological phases. In this step, we also need to choose labels for the indices. Two indices that are connected should have the same label. In the end, no free indices should remain. Fourth, the GSD is the scalar resulting

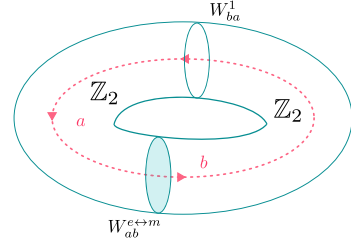


FIG. 49. Pants decomposition for the $D(\mathbb{Z}_2)$ twisted toric code, i.e., containing one noncontractible twist. It consists of two $D(\mathbb{Z}_2)$ cylinders glued together on one side by the transparent domain wall and by the $W^{e \leftrightarrow m}$ on the other side.

from contracting the indices with the same label. This process is better illustrated with examples.

1. Examples using pants decomposition

Example VI.10 ($\mathbb{Z}_2$ on a Torus with a non-contractible twist II). As a first example, we recalculate the case of the toric code with one noncontractible twist, Fig. 49.

The twist is simply the domain wall $e \leftrightarrow m$. The matrix of this domain wall is given by

$$W^{e \leftrightarrow m} = \begin{matrix} & \begin{matrix} 1 & e & m & em \end{matrix} \\ \begin{matrix} 1 \\ e \\ m \\ em \end{matrix} & \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{matrix}. \quad (66)$$

We show the order of the basis for clarity. We can decompose the torus in this case into two cylinders. They intersect at two domain walls, one of which is the trivial domain wall $W_{ab}^1 = \delta_{a,b}$ and the other one is $W^{e \leftrightarrow m}$. We then have

$$\text{GSD} = \sum_{a,b} W_{a,b}^{e \leftrightarrow m} W_{ab}^1 = \sum_{a,b} W_{ab}^{e \leftrightarrow m} \delta_{a,b} = \text{Tr}(W^{e \leftrightarrow m}) = 2, \quad (67)$$

agreeing with Ref. [185] and our result using stabilizers model in Example VI.5.

Example VI.11 ($\mathbb{Z}_4$ with a non-contractible DS patch). Another example is the case of $\mathbb{Z}_4$ and a noncontractible DS patch on a torus. We have a 4×16 domain wall W because DS has four anyons $\{1, s, \bar{s}, b\}$ while $\mathbb{Z}_4$ has 16 anyons labeled by $e^i m^j$ where $0 \leq i, j \leq 3$. The condensation procedure used to obtain DS from $\mathbb{Z}_4$ informs us that the $e^2 m^2$ anyons in $\mathbb{Z}_4$ are identified with the vacuum in DS , while the semion s condenses with $\{em, e^3 m^3\}$ etc. The condensation and tunneling are exactly the same for DS anyons, as each one is its own antiparticle. We then have the W matrix,

$$W^\dagger = \begin{matrix} & \begin{matrix} 1 & em & e^2 & m^2 & e^2 m^2 & em^3 & e^3 m & e^3 m^3 \end{matrix} \\ \begin{matrix} 1 \\ s \\ \bar{s} \\ b \end{matrix} & \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \end{matrix}. \quad (68)$$

We only show the nonzero columns of the matrix. As shown in Fig. 50, the torus can be decomposed into a $D(\mathbb{Z}_4)$ and a DS cylinder, which meet at two copies of the same domain

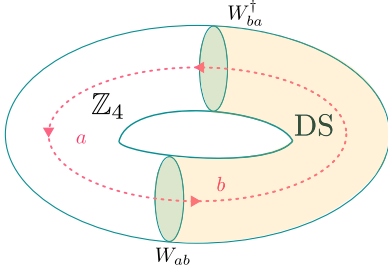


FIG. 50. Pants decomposition of the torus with half $\mathbb{Z}_4$ and half DS phases. It consists of two cylinders glued at both ends with the unique domain wall W between the two phases.

wall. The GSD is then

$$\text{GSD} = \sum_{a,b} W_{ab} W_{ba}^\dagger = \text{Tr}(W W^\dagger) = 8, \quad (69)$$

which confirms the results of Example VI.8.

One can also treat the example with a contractible DS patch. While this was a relatively complicated example using stabilizer constraints, it has an easier solution using pants decomposition.

Example VI.12 ($D(\mathbb{Z}_4)$ with one contractible DS patch). For a one-patch code, as shown in Fig. 51, we decompose the torus into a $\mathbb{Z}_4$ cylinder, $\mathbb{Z}_4$ pants, and a DS cap.

All the domain walls between two $\mathbb{Z}_4$ phases are trivial and denoted here by W^1 . The domain wall from $\mathbb{Z}_4$ to DS is denoted by W , which is given in Example VI.11. Finally, we attach to the pants the $D(\mathbb{Z}_4)$ fusion coefficients $N_{i,j}^k = \delta_{i+j,k}$ and to the cap the delta operator. The GSD is then

$$\begin{aligned} \text{GSD} &= \sum_{a,j,i,u,k} W_{aj}^1 N_{i,j}^k W_{ui}^\dagger \delta_{u,1} W_{ka}^1 = \sum_{a,i} N_{i,a}^a W_{i,1}^\dagger \\ &= \sum_a N_{1,a}^a W_{1,1}^\dagger = \sum_a = 16, \end{aligned} \quad (70)$$

where in the second step we used $N_{i,a}^a = \delta_{i,1}$. In the last step, we simply sum over the anyon types of $D(\mathbb{Z}_4)$. This agrees with the results of Example VI.6.

Example VI.13 ($D(\mathbb{Z}_4)$ on a torus with N DS patches). Let us start with the case of two patches. We use the decomposition shown in Fig. 52.

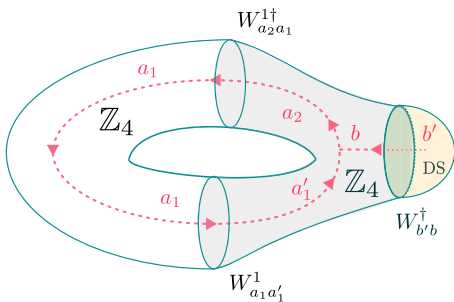


FIG. 51. Pants decomposition of the $\mathbb{Z}_4$ toric code with one contractible DS patch. Because of the trivial wall W^1 , we actually have $a_1 = a'_1$ and $a_2 = a'_1$.

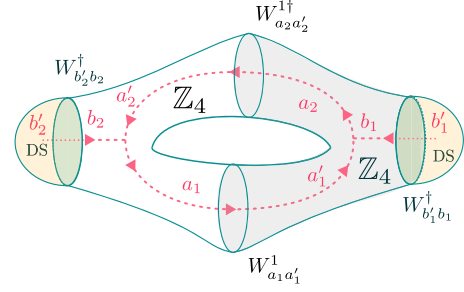


FIG. 52. Pants decomposition of the $\mathbb{Z}_4$ toric code with two contractible DS patches. Because of the trivial domain walls, we have $a_i = a'_i$.

We then compute the GSD of the two patches as

$$\begin{aligned} \text{GSD} &= \sum_{a,b} W_{a_1 a'_1}^1 N_{a_1, b_1}^{a_2} W_{b_1, b'_1}^\dagger \delta_{b'_1, 1} W_{a_2, a'_2}^\dagger N_{a'_2, b_2}^{a_1} W_{b'_2, b_2}^\dagger \delta_{b'_2, 1} \\ &= \sum_{a,b} W_{b_1, 1}^\dagger N_{a_1, b_1}^{a_2} W_{b_2, 1}^\dagger N_{a_2, b_2}^{a_1} \\ &= \sum_{a,b} W_{b_1, 1}^\dagger \delta_{a_1 + b_1, a_2} W_{b_2, 1}^\dagger \delta_{b_2, -b_1} \\ &= \sum_{a_1, b_1} W_{b_1, 1}^\dagger W_{-b_1, 1}^\dagger = 16 \times 2. \end{aligned} \quad (71)$$

In the second line, we simplified using the deltas from the transparent domain walls and from the caps. Since the theory is Abelian, in the third line we replaced $N_{a,b}^c$ with $\delta_{a+b,c}$. In the last line, we observe that a_1 has 16 choices corresponding to the anyons of $\mathbb{Z}_4$. While b_1 have only two choices corresponding to $\{1, e^2 m^2\}$ because of the domain wall W .

The pattern continues when adding new patches. If we add N patches, then the ground-state degeneracy can be computed as follows (for $N \geq 1$):

$$\begin{aligned} \text{GSD} &= \sum_{a,b} W_{b_1, 1}^\dagger N_{a_1, b_1}^{a_2} W_{b_2, 1}^\dagger N_{a_2, b_2}^{a_3} \\ &\quad \dots W_{b_{N-1}, 1}^\dagger N_{a_{N-1}, b_{N-1}}^{a_N} W_{b_N, 1}^\dagger N_{a_N, b_N}^{a_1} \\ &= \sum_{a,b} W_{b_1, 1}^\dagger \delta_{a_2, a_1 + b_1} W_{b_2, 1}^\dagger \delta_{a_3, a_2 + b_2} \\ &\quad \dots W_{b_{N-1}, 1}^\dagger \delta_{a_N, a_{N-1} + b_{N-1}} W_{b_N, 1}^\dagger \delta_{b_N, a_1 - a_N} \\ &= 16 \times 2^{N-1}. \end{aligned} \quad (72)$$

We note that $W_{x,1}^\dagger$ is only nonzero when $x = 1$ or $x = e^2 m^2$ as can be seen from Eq. (68). We then have only two choices for the indices b_i , and there are $N - 1$ of them (since b_N is dependent). This gives 2^{N-1} choices. After picking the free $N - 1$ b 's, making any arbitrary choice for a_1 from the 16 anyons of $\mathbb{Z}_4$ determines the rest of a 's as well as b_N . This gives $16 \times 2^{N-1}$ choices. This agrees with Example VI.7.

Cases with open boundaries can also be treated similarly since we can cap them with the vacuum. This is equivalent to the fact that boundaries of a phase are domain walls between the phase and the vacuum. However, this restricts us in the case in which the boundary is of one type. Equivalently, it is not clear how to include the case in which domain walls intersect.

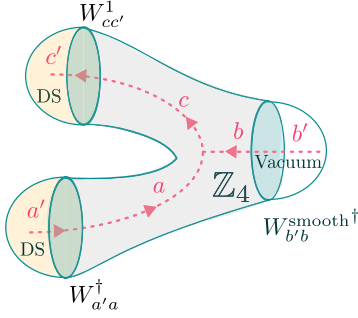


FIG. 53. Pants decomposition of the $\mathbb{Z}_4$ toric code with two contractible DS patches and a smooth boundary.

Example VI.14 (DS Patches in a $\mathbb{Z}_4$ bulk with smooth boundaries). The Pants decomposition can also be used with a single open boundary. In this case, one needs to add a cap representing the vacuum phase. We consider the configuration of two DS patches inside a $D(\mathbb{Z}_4)$, which itself has a smooth boundary. This has the same degeneracy as the configuration shown in Fig. 13 with two patches rather than four. The pants decomposition is shown in Fig. 53.

One can imagine stretching (topologically deforming) the vacuum cap until it becomes the outer boundary:

$$\begin{aligned}
 \text{GSD} &= \sum_{a,b,c,a',b',c'} \delta_{a',1} W_{a'a}^\dagger N_{a,b}^c W_{b'b}^{\text{smooth}\dagger} \delta_{b',1} W_{c'c} \delta_{c',1} \\
 &= \sum_{a,b,c} W_{1a}^\dagger \delta_{b,c-a} W_{1b}^{\text{smooth}\dagger} W_{c1} \\
 &= \sum_{a,b,c} W_{1a}^\dagger \delta_{b,1} \delta_{c-a,b} W_{1b}^{\text{smooth}\dagger} W_{c1} \\
 &= \sum_a W_{1a}^\dagger W_{-a1} = 2.
 \end{aligned} \tag{73}$$

In the second line, we enforced the delta functions. In the third line, since $a, c \in \{1, e^2 m^2\}$, the only pure flux b that can appear as $(a - c)$ is the vacuum state, 1. Finally, summing over the two choices of a gives us $\text{GSD} = 2$. The case with N patches can be treated similarly to Example VI.13

VII. CONCLUSIONS

In this work, we have defined a new family of Abelian quantum error-correcting codes. In addition to conventional boundaries of quantum doubles [47,73], our codes contain finite-sized condensates of a twisted quantum double of the parent quantum double. We have explicitly illustrated this using the group $G = \mathbb{Z}_4$ and generalized the DS global condensation [42] to a spatially local condensation. We derived the logical operators and showed that, in addition to whatever background logical operators of the parent $D(\mathbb{Z}_4)$, $N - 1$ logical qubits are added via N patches. This result explicitly shows how topological qubits of distinct dimensions (two and four) can coexist in a single code. We have described why, from the perspective of simplifying and improving error correction in the near term, this approach is desirable.

To prove our results, we derived the technical machinery along two complementary directions. First, we provided a

concrete algorithm to gauge one stabilizer code into another. This was based on the concept of anyon condensation and, in an experimental setting, can be interpreted as measuring a particular set of stabilizers. Building on this algorithm, we derived the ground-state degeneracies by modding out the constraints from the total Hilbert-space dimension. In addition to this microscopic lattice perspective, we showed how the number of logical operators, as loops of ribbon operators, can be counted and constructed in terms of topological primitives such as cylinders, pants, and caps.

Our work opens the path for multiple research directions. Our first motivation is that these codes should be realized in experiments. In this light, it will be interesting to compare the error rates and explore the tradeoffs exhibited between $\mathbb{Z}_2$ codes, DS codes, and composite $\mathbb{Z}_4$ – DS codes. In addition to being experimentally explored, these questions can be explored by QECC simulations that more holistically include the realistic costs of real-time decoding and the burden this places on code performance in experiments.

Lastly, our work opens the path for more theoretical analysis of codes and a grand classification of quantum topologies. For example, our results here have, for the sake of brevity and to provide concrete examples, been limited to low-dimensional Abelian cyclic groups. In addition to exploring non-Abelian groups, one may also perform a similar analysis with quantum doubles generated by higher-dimensional cyclic groups. In such cases, there is room for more types of anyon condensation (both in variety and in the number of times this procedure can be performed). It would be interesting to see how such concepts can be judiciously applied to lower the experimental resource burden in QECC and also to engineer more exotic quasiparticle types for computation.

Note added. Recently, we became aware of Ref. [189], which analyzes phase transitions between DS and $\mathbb{Z}_4$ phases. While complementary, our work focuses on the quantum-error-correcting codes derived from the judicious use of these phases. It would be interesting to investigate the intersection of these two concepts in future studies.

ACKNOWLEDGMENTS

We thank Shawn Cui, Birgit Kaufmann, Ralph Kaufmann, Phil Lotshaw, Paul Kairys, Sohan Ghosh, and Tyler Ellison for useful discussions and comments. We thank Sabre Kais for access to the Bell computing cluster at Purdue University. M.M., A.G., and E.D. are supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Research Program, Early Career Award under Contract No. ERKJ420. This research was performed during M.M.'s participation in the Graduate Research at ORNL (GRO) Educational Program. This research used resources of the Compute and Data Environment for Science (CADES) at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. ORNL is managed by UT-Battelle, LLC, under Contract No. DE-AC05-00OR22725 for the U.S. Department of Energy.

This manuscript has been authored by UT-Battelle, LLC, under Contract No. DE-AC0500OR22725 with the U.S. Department of Energy. The United States Government retains

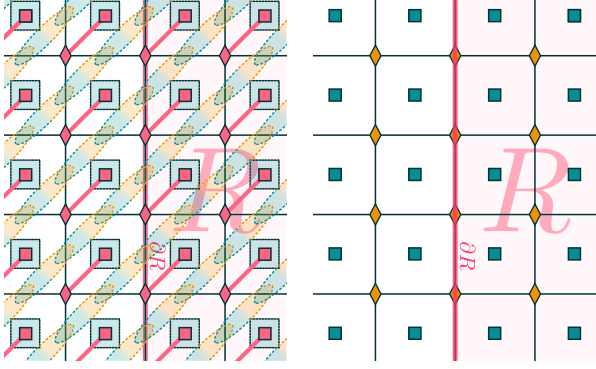


FIG. 54. (Left) Region of condensation R with its boundary ∂R in layer 1 (the bottom unmirrored DS layer). (Right) Region of condensation R with its boundary ∂R in layer 2 (the top mirrored $\mathbb{Z}_4$ layer).

and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for the United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan.

DATA AVAILABILITY

There are no publicly available research data or software supporting this manuscript. Requests for further information or data should be sent to the authors.

APPENDIX A: CONSTRUCTION OF DOMAIN WALL BETWEEN DS- $D(\mathbb{Z}_4)$

In this appendix, we derive the domain wall between the DS phase and the $\mathbb{Z}_4$ phase using Algorithm 2. This proves the more heuristic method given in Example V.5. Before starting the algorithm, we place the $D(\mathbb{Z}_4)$ on top while keeping the DS phase at the bottom. Denote the anyons of the bottom layer

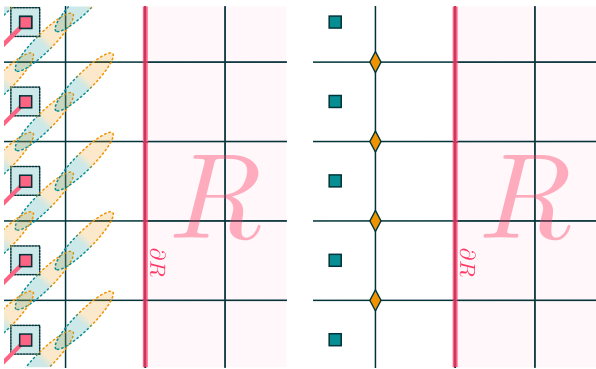


FIG. 55. After the third step, the stabilizers that do not commute with the measured ribbon operators are removed. These include all stabilizers of both layers in R . In layer 1 (left), $B_2(p)$ and X^2Z^2 touching the boundary ∂R were removed. In layer 2 (right), $B_2(p)$ and $A_2(v)$ touching ∂R were removed.

by $\{1, s_1, \bar{s}_1, b_1\}$ and those of the top layer by $\{1, e_2, m_2, \dots\}$. The top layer is a mirrored version of the bottom layer, as shown in Fig. 27. The Hamiltonian is a mirrored version of the $\mathbb{Z}_4$ Hamiltonian Eq. (46). Here, the mirrored nature of the top layer is fully exploited. Importantly, the anyon e_2m_2 with twist $\mathcal{T}(e_2m_2) = i$ in the unfolded picture is mapped to the anyon $e_2m_2^{-1}$ with $\mathcal{T}^{\text{op}}(e_2m_2) = i$ in the folded version because the handedness changes with mirror reflection.

We then begin the algorithm. First, we choose a region R extending to infinity. The boundary ∂R lives on the direct lattice in both layers, see Fig. 54. Second, we add the shortest ribbon operators creating anyons of the Lagrangian subgroup generated by $\mathcal{L}_{\mathbb{Z}_4-DS} = \{s_1e_2m_2^{-1}, \bar{s}_1e_2m_2, b_1e_2^2, b_1m_2^2, 1_1e^2m_2, \dots\}$. We show a subset of the generators for illustration,

$$\begin{aligned} s_1e_2^{-1}m_2(\Rightarrow) &= \left[\begin{array}{c} X_1X_2 \\ Z_1Z_2 \end{array} \right], & s_1e_2^{-1}m_2(\Uparrow) &= \left[\begin{array}{c} X_1^\dagger X_2 \\ Z_1Z_2^\dagger \end{array} \right] \\ \bar{s}_1e_2^{-1}m_2^{-1}(\Rightarrow) &= \left[\begin{array}{c} X_1^\dagger X_2^\dagger \\ Z_1Z_2 \end{array} \right], & \bar{s}_1e_2^{-1}m_2^{-1}(\Uparrow) &= \left[\begin{array}{c} X_1X_2^\dagger \\ Z_1Z_2^\dagger \end{array} \right] \\ b_1e_2^2(\Rightarrow) &= \left[\begin{array}{c} X_1^2X_2^2 \\ Z_1^2Z_2^2 \end{array} \right], & b_1m_2^2(\Uparrow) &= \left[\begin{array}{c} Z_1^2Z_2^2 \\ X_1^2X_2^2 \end{array} \right] \end{aligned} \quad (\text{A1})$$

Here, the arrows indicate the direction of propagation of the anyons (i.e., the direction of the ribbons creating them).

After the second step, the ribbons in Eq. (A1) are added to R and along ∂R . In the third step, we remove the old stabilizers that do not commute with the new ones. These include all the stabilizers of both layers in the bulk of the region R . In addition, we remove the star operators $A_2(v)$ of layer 2 on the boundary ∂_1R . We remove the plaquettes $B_2(p)$ of the second layer adjacent to the boundary ∂_1R . We remove the X^2Z^2 operators of the DS phase touching ∂R . The configuration after the third step is shown in Fig. 55.

In the fourth step, we form new stabilizers that are products of the old ones and commute with the new stabilizers. These are given by products of the removed stars with their adjacent removed plaquettes, along with the F operators of layer 1. In addition, we include the product of the plaquette squared of the two layers,

$$\begin{aligned} F_1(v,p)A_2(v)B_2(p) &= \frac{X_1Z_1^\dagger X_2Z_2}{X_1^\dagger X_2} \left[\begin{array}{c} Z_1^\dagger Z_2^\dagger \\ X_1X_2 \\ X_1^\dagger X_2^\dagger \\ X_1Z_1Z_2 \end{array} \right] Z_1Z_2^\dagger \xrightarrow{\text{Restrict}} \left[\begin{array}{c} X_1^\dagger X_2 \\ Z_1^\dagger Z_2^\dagger \end{array} \right] \\ F_1(v,p)B_2(p) &= \frac{X_1Z_1^\dagger Z_2}{X_1^\dagger X_1} \left[\begin{array}{c} Z_1^\dagger Z_2^\dagger \\ X_1Z_1Z_2 \\ X_1^\dagger X_2^\dagger \end{array} \right] Z_1Z_2^\dagger \xrightarrow{\text{Restrict}} \left[\begin{array}{c} X_1Z_1^\dagger Z_2 \\ X_1^\dagger X_2^\dagger \end{array} \right] \\ B_1^2(p)B_2^2(p) &= Z_1^2Z_2^2 \left[\begin{array}{c} Z_1^\dagger Z_2^\dagger \\ Z_1^\dagger Z_2^\dagger \\ Z_1^\dagger Z_2^\dagger \\ Z_1^\dagger Z_2^\dagger \end{array} \right] Z_1^2Z_2^2 \xrightarrow{\text{Restrict}} \left[\begin{array}{c} Z_1^2Z_2^2 \\ Z_1^2Z_2^2 \\ Z_1^2Z_2^2 \\ Z_1^2Z_2^2 \end{array} \right] \end{aligned} \quad (\text{A2})$$

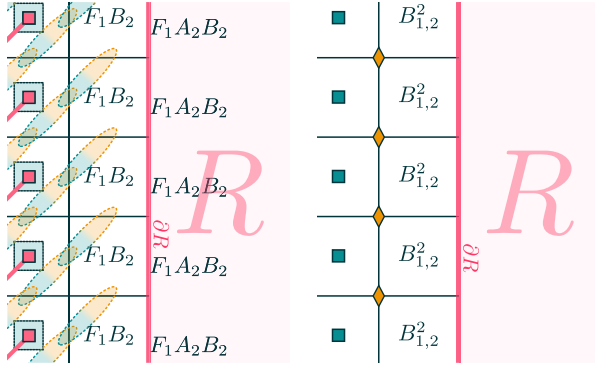


FIG. 56. After the fifth step, the products of old stabilizers that commute with the ribbons are added Eq. (A2). We only show here the stabilizers near the boundary. In addition, qudits in R or on ∂R were erased.

We denote the placement of the stabilizer relative to the boundary. After adding the new stabilizers to the model, we reach the configuration in Fig. 31.

In the fifth step, we erase trivialized qudits. These are all the qudits in the bulk of R or on its boundary ∂R . After condensing qudits, we also remove all stabilizers that only act on trivialized qudits for clarity, Fig. 56.

The sixth step is to restrict the stabilizers at the boundary to their nontrivialized support. This results in the stabilizers shown in Eq. (A2). Finally, we unfold layer 2 and place it on the right. This restores the stabilizer of the bulk of layer 2 ($\mathbb{Z}_4$) to those of the same again as the unmirrored $D(\mathbb{Z}_4)$ by redoing the horizontal reflection. At each edge, now only one qudit remains, so we can omit the layers' indices unambiguously. This results in the following stabilizers:

$$\begin{aligned}
 F_1(v, p)A_2(v)B_2(p) &= \frac{X_1^\dagger X_2}{X_1^\dagger X_1} \xrightarrow{\text{Unfold}} \frac{X^\dagger}{X} \\
 F_1(v, p)B_2(p) &= \frac{X_1^\dagger Z_1^\dagger Z_2^\dagger}{X_1^\dagger X_1^\dagger} \xrightarrow{\text{Unfold}} \frac{XZ^\dagger}{X^\dagger} \\
 B_1^2(p)B_2^2(p) &= \frac{Z_1^2 Z_2^2}{Z_1^2 Z_2^2} \xrightarrow{\text{Unfold}} \frac{Z^2}{Z^2}
 \end{aligned} \tag{A3}$$

Note how the vertical edges corresponding to ∂R are now condensed. These are perfectly valid stabilizers for the domain wall between the DS and $D(\mathbb{Z}_4)$; however, it is desirable to have lower-weight stabilizers. This is done by taking the first stabilizer $X_1^\dagger X_2 = 1$ at the Hilbert space level. This maps the two qudits into one qudit. A mapping of two-qudit operators to this new degree of freedom stitches the lattice back, giving

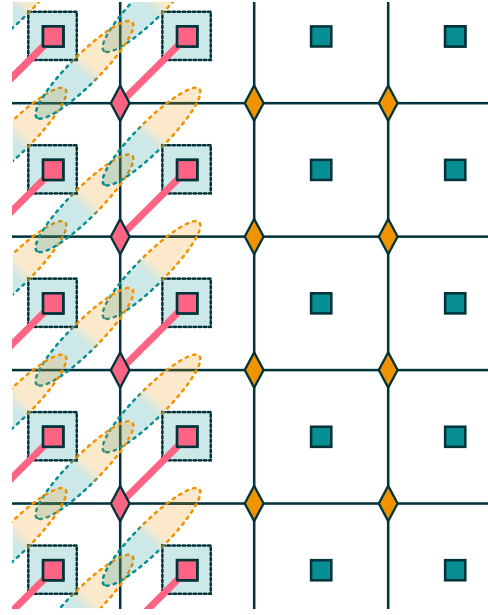


FIG. 57. After stitching the stabilizers at the boundary together, we retrieve the domain wall between the DS and $D(\mathbb{Z}_4)$ phases.

the familiar stabilizers

$$\begin{aligned}
 F_1(v, p)B_2(p) &= \frac{XZ^\dagger}{X^\dagger} \xrightarrow{\text{Stitch}} \frac{XZ^\dagger}{X^\dagger} = \text{red square} \\
 B_1^2(p)B_2^2(p) &= \frac{Z^2}{Z^2} \xrightarrow{\text{Stitch}} \frac{Z^2}{Z^2} = \text{blue square}
 \end{aligned} \tag{A4}$$

After stitching the lattice back, we find the domain wall between the DS and $\mathbb{Z}_4$ phases shown in Fig. 57. This proves the results using the more heuristic method used in Example V.5 with direct condensation.

APPENDIX B: EXAMPLE OF A 0D DEFECT CONSTRUCTION

In this appendix, we use Algorithm 2 along with the folding method to construct the inverse domain wall of the $D(\mathbb{Z}_4)$ code $[(e, m) \rightarrow (e^{-1}, m^{-1})]$. More importantly, this exemplifies how finite domain walls ending with 0D defects can be constructed systematically.

Before beginning the algorithm, we place two copies of the $D(\mathbb{Z}_4)$ and denote the bottom layer with index 1 and the top layer with index 2. First, we select three regions. For regions R_1 and R_3 , we construct the trivial domain wall, while for region R_2 we construct the inverse domain wall. The regions are shown in Fig. 58.

Second, we measure the ribbon operators corresponding to the Lagrangian subgroup of the respective region and domain wall. In regions R_1 and R_3 , we measure the ribbons of the Lagrangian subgroup that create the trivial domain wall $\mathcal{L}_{\text{trivial}} = \{1, e_1 e_2^{-1}, m_1 m_2^{-1}\}$ while for region R_2 we measure

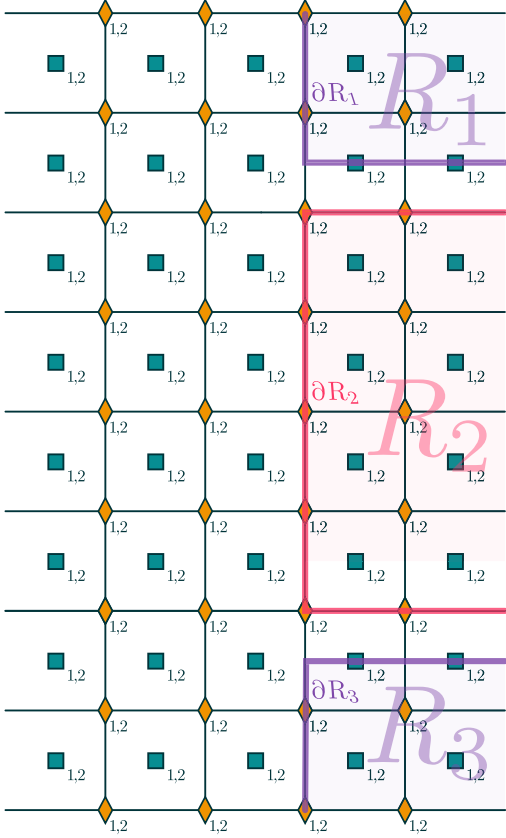


FIG. 58. Regions used to create the finite $(e, m) \rightarrow (e^{-1}, m^{-1})$ defect of $D(\mathbb{Z}_4)$. The three boundaries do not overlap and completely cover the left half-plane of the doubled layer. Stabilizers have two subscripts corresponding to both layers.

the Lagrangian subgroup that creates the inverse domain wall. The ribbons are all one-edge operators (B1):

$$\begin{aligned}
 e_1 e_2 (\Rightarrow) &= \frac{Z_1 Z_2^\dagger}{}, \quad m_1 m_2 (\Uparrow) = \frac{X_1^\dagger X_2^\dagger}{}, \\
 e_1 e_2 (\Uparrow) &= \left| \frac{Z_1 Z_2}{}, \quad m_1 m_2 (\Rightarrow) = \left| \frac{X_1 X_2^\dagger}{}, \right. \\
 e_1 e_2^{-1} (\Rightarrow) &= \frac{Z_1 Z_2}{}, \quad m_1 m_2^{-1} (\Uparrow) = \frac{X_1^\dagger X_2^\dagger}{}, \\
 e_1 e_2^{-1} (\Uparrow) &= \left| \frac{Z_1 Z_2}{}, \quad m_1 m_2^{-1} (\Rightarrow) = \left| \frac{X_1 X_2^\dagger}{}.
 \end{aligned} \tag{B1}$$

Third, we remove the old stabilizers that do not commute with the measured operators. These include all the stabilizers in the bulk of the three regions. In addition, we remove the star operators on the boundaries ∂R_1 , ∂R_2 , and ∂R_3 , along with the plaquettes adjacent to the three boundaries. After this step, we reach the configuration shown in Fig. 59.

In the fourth step, we add products of removed stabilizers that commute with the measured operators. These consist

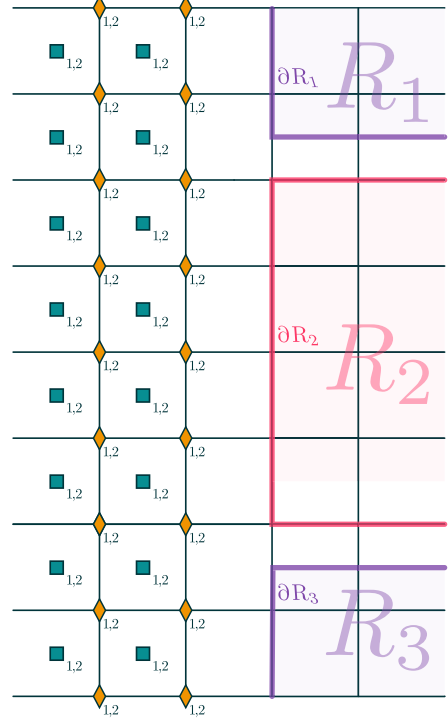


FIG. 59. After the third step, the stabilizers that did not commute with measured operators are removed. These are all stabilizers from both layers that share at least one edge with the bulks or the boundaries of the three regions.

of products of star operators of both layers $A_1(v)A_2(v)$ or $A_1(v)A_2^\dagger(v)$ for the trivial and the inverse domain wall, respectively. Similarly, products of plaquettes of each layer $B_1(p)B_2(p)$ or $B_1(p)B_2^\dagger(p)$ are added. Note that the operators of the second layer are mirrored since it is folded, as shown below:

$$\begin{aligned}
 A_1(v)A_2(v) &= \frac{X_1^\dagger X_2}{X_1^\dagger X_2^\dagger} \xrightarrow{\text{Restrict}} \frac{X_1^\dagger X_2}{}, \\
 B_1(p)B_2(p) &= \frac{Z_1^\dagger Z_2^\dagger}{Z_1 Z_2} \xrightarrow{\text{Restrict}} \frac{Z_1^\dagger Z_2^\dagger}{Z_1 Z_2}, \\
 A_1(v)A_2^\dagger(v) &= \frac{X_1^\dagger X_2^\dagger}{X_1 X_2} \xrightarrow{\text{Restrict}} \frac{X_1^\dagger X_2^\dagger}{}, \\
 B_1(p)B_2^\dagger(p) &= \frac{Z_1^\dagger Z_2}{Z_1 Z_2^\dagger} \xrightarrow{\text{Restrict}} \frac{Z_1^\dagger Z_2}{Z_1 Z_2^\dagger}.
 \end{aligned} \tag{B2}$$

These operators have to be added in place of the old, removed stabilizers everywhere. However, they are most important near the boundary, as they create the domain wall. Thus, we do not need to figure out those terms except at the boundaries. Importantly, right at the OD defect, the stars must satisfy both boundaries. This is the first time that the order of the group matters. So far, everything was applicable to all $D(\mathbb{Z}_n)$. In

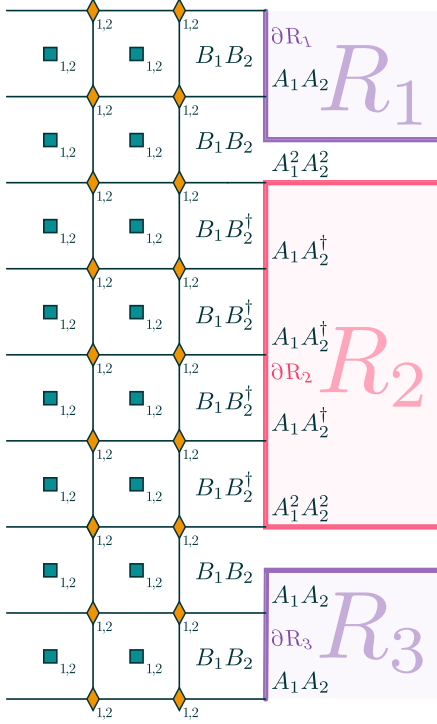


FIG. 60. The configuration after the fourth and fifth step. In the fourth step, products of old stabilizers that commute with the new measured operators are added. We only added them near the boundary, as the rest of them will be condensed. After the fifth step, trivialized qutrits are erased.

$D(\mathbb{Z}_4)$, the square of stars $A_1^2(v)A_2^2(v)$ commutes with both boundaries. Fifth, we erase the trivialized qutrits in the bulk of the three regions along with their boundaries. In Fig. 60, the new plaquette and star operators are shown at the three boundaries, while the trivialized qutrits are erased.

Sixth, we restrict the operators to their nontrivialized support. This results in the operators in the last column of Eq. (B2). Since there are no more qutrits to condense, the construction is complete. We can unfold the two layers, dropping the layer indices. The operators at the boundary transform to

$$\begin{aligned}
 A_1(v)A_2(v) &= \frac{X_1^\dagger X_2}{\text{Unfold}} \rightarrow \frac{X^\dagger}{X} \\
 B_1(p)B_2(p) &= \frac{Z_1^\dagger Z_2}{Z_1 Z_2} \xrightarrow{\text{Unfold}} \frac{Z^\dagger}{Z} \\
 A_1(v)A_2^\dagger(v) &= \frac{X_1^\dagger X_2^\dagger}{\text{Unfold}} \rightarrow \frac{X^\dagger}{X^\dagger} \\
 B_1(p)B_2^\dagger(p) &= \frac{Z_1^\dagger Z_2^\dagger}{Z_1 Z_2^\dagger} \xrightarrow{\text{Unfold}} \frac{Z^\dagger}{Z}
 \end{aligned} \tag{B3}$$

This is a valid finite-domain wall that maps anyons to corresponding anyons with a 0D defect at each end. We see that the qutrits on the vertical line have been condensed. If it is desirable to have lower-weight stabilizers, we can stitch the two sides together. This is formally done by imposing the $XX^\dagger$ or XX two-edge operators at the Hilbert-space level. This stitching maps the two qutrits into one. The action of any operator that acted on the original qutrits can now be mapped onto this new degree of freedom. For the stabilizers right at the boundary, we have

$$\begin{aligned}
 B_1(p)B_2(p) &= \frac{Z^\dagger}{Z} \xrightarrow{\text{Stitch}} \frac{Z^\dagger}{Z} \\
 &= B(p) = \blacksquare \\
 B_1(p)B_2^\dagger(p) &= \frac{Z^\dagger}{Z} \xrightarrow{\text{Stitch}} \frac{Z^\dagger}{Z} \\
 &= B^{\text{op}}(p) = \blacklozenge
 \end{aligned} \tag{B4}$$

The star operators that share an edge with the trivial wall remains unchanged, while those sharing an edge with the inverse domain become in new star operators $A^{\text{op}}(v)$. Right at the 0D defect, we have the order 2 operator X^2X^2 , which cannot stitch the two sides back. We are therefore left with four weight-five operators denoted by $D^{(1)}$, $D^{(2)}$, $E^{(1)}$, and $E^{(2)}$:

$$\begin{aligned}
 A^{\text{op}}(v) &= \frac{X^\dagger}{X^\dagger} = \blacklozenge & D^{(1)}(p) &= \frac{Z^\dagger}{Z} = \blacktriangleleft \\
 B^{\text{op}}(p) &= \frac{Z^\dagger}{Z} = \blacklozenge & D^{(2)}(p) &= \frac{Z^\dagger}{Z} = \blacktriangleright \\
 X^2X^2(e) &= \frac{X^2}{X^2} = \blacklozenge & E^{(1)}(p) &= \frac{Z^\dagger}{Z} = \blacktriangleleft \\
 & & E^{(2)}(p) &= \frac{Z^\dagger}{Z} = \blacktriangleright
 \end{aligned} \tag{B5}$$

Heuristically, to construct the inverse wall, we choose a vertical line in the direct lattice l and an adjacent vertical line in the dual lattice $\tilde{l}$. For the plaquettes intersecting $\tilde{l}$, we take the adjoint of their Pauli operators acting on edges intersecting l . For the star operators intersecting l , we apply the adjoint of their Pauli operators acting on edges intersecting $\tilde{l}$. With these new operators, we reach a stabilizer code for the defect that still requires weight-five operators at the two ends. The new stabilizer placement configuration is shown in Fig. 61 after the unfolding and stitching.

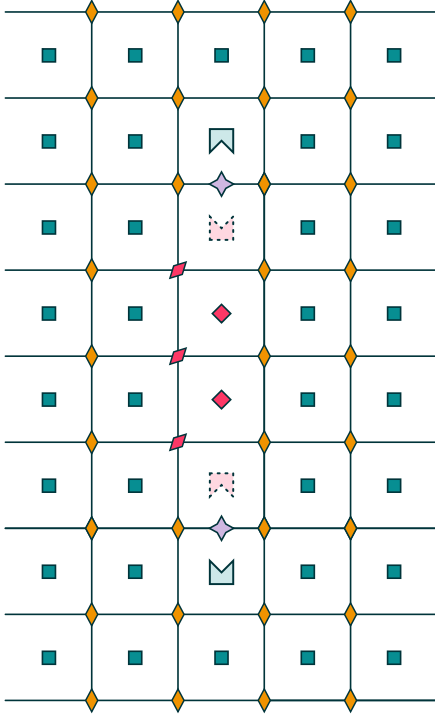


FIG. 61. The final stabilizers for the defect after unfolding and stitching. Two weight-five operators now live at each end of the finite domain wall. The stabilizers are defined in Eq. (B5).

APPENDIX C: BELIEF PROPAGATION

We describe here the belief-propagation implementation in more detail. It follows the case for qubits closely [125,126] generalized to the four-dimensional qudits with extra modifications due to the fact that $\mathbb{Z}_4$ is not a field. We operate in the usual setting in which the problem is stated as a Tanner graph [164] where qudit and check nodes are denoted by the variables v and u , respectively. The check node u_i is connected to the data node v_j with an edge if $H_{ij} \neq 0$. Furthermore, generalizing the qubit case, the edges are now weighted by H_{ij} .

Given as input a syndrome vector s with the same length as the number of code checks, the goal is to estimate the error vector x , which has the same length as the number of qudit nodes. For each check node u_i that has $V(u_i)$ neighboring data qudits, it naturally satisfies the equation $\sum_{j=1}^{V(u_i)} H_{ij} x_{v_j} = s_{u_i}$. Here, the goal is to find an estimated error x that satisfies all the constraints. This is the usual $H \cdot x = s$, but recast in terms of the graph variables. The check-to-data messages used a kind of an XOR operation in the qubit case (that can be captured with a simple product operation since it is defined over $\mathbb{Z}_2$) [125]. This naturally generalizes to a convolution for the $\mathbb{Z}_4$ case. The convolution of two vectors f and g given the syndrome s is defined as

$$(f \oplus g)(s) = \sum_{t \in \mathbb{Z}_4} f(t) \cdot g(s - t \bmod 4). \quad (\text{C1})$$

Here, we use functional notation for the entries of the vectors. For example, for a vector g , $g(0)$ is its first element. It is also useful to define iterated convolution (usually over the data

qudit neighbors of a check node $[V(u_i)]$,

$$\bigotimes_{k \neq j} \phi_k = \phi_1 \oplus \cdots \oplus \phi_{j-1} \oplus \phi_{j+1} \oplus \cdots \oplus \phi_{V(u_i)}.$$

This naively scales as $O(4^{V(u_i)})$. However, it can be computed using the forward-backward check-node update in $O(16 \times V(u_i))$ or the fast Fourier transform $O(4 \log(4) \times V(u_i))$ [190–192]. We describe the convolution using the simpler naive approach, while numerically, we have used the forward–backward check-node update.

We now begin the algorithm. First, we take as an input the probability of physical errors p , assuming for simplicity that they are uniform and over a pure Pauli channel (e.g., pure X noise). The messages are four-dimensional vectors carrying the information for each power of X . The probability of failure for any physical noise channel is then $P(v) = (1 - p, \frac{p}{3}, \frac{p}{3}, \frac{p}{3})$, assuming the noise model described in Sec. IV B. The initial messages from the data qudits to the check nodes are then $m_{v \rightarrow u}(a) = P(v, a)$, and those from the check nodes to data qudits is just $m_{u \rightarrow v}(a) = \frac{1}{4}$ since we do not yet have any preference regarding the estimated error.

In the next iteration, the check node u_i projects the messages it has into contributions to the syndrome it sees from its neighbor data qudit v_i . This is captured in successive steps. We first calculate the probability that the qudit contributes y value to the syndrome: $\phi_j(a) = \sum_{\substack{y \in \mathbb{Z}_4 \\ H_{ij} \cdot y \equiv a \pmod{4}}} m_{v_j \rightarrow u_i}(y)$. The check nodes send their estimated errors to their neighboring data qudits by performing a convolution over all the

ALGORITHM 3. Qudit belief propagation decoder.

```

1: procedure QUDIT-BPH,  $s, p, \alpha, T_{\max}$ 
2:   Initialize priors  $P(v, x) \leftarrow (1 - p, p/3, p/3, p/3)$  for all
   qudits  $v$ .
3:   Set initial messages:  $m_{v \rightarrow u}(a) \leftarrow P(v, a)$  and  $m_{u \rightarrow v}(a) \leftarrow 1/4$ .
4:   for  $t = 1$  to  $T_{\max}$  do
5:     Check-to-Data: Project incoming messages:
6:      $\phi_k(y) = \sum_{H_{ik}z \equiv y \pmod{4}} m_{v_k \rightarrow u_i}(z)$ .
7:     Convolve to find check messages:
8:      $m_{u_i \rightarrow v_j}(a) \propto (\bigotimes_{k \neq j} \phi_k)(s_{u_i} - H_{ij}a \pmod{4})$ .
9:     Normalize  $m_{u_i \rightarrow v_j}(a)$ .
10:    Data-to-Check: Compute qudit belief:
11:     $q_{v_j}(x) \propto P(v_j, x) \prod_{u_c} m_{u_c \rightarrow v_j}(x)$ .
12:    Divide out check message:
13:     $m_{v_j \rightarrow u_i}^{\text{new}}(x) \leftarrow \frac{q_{v_j}(x)}{m_{u_i \rightarrow v_j}(x)}$ .
14:    Update with damping:
15:     $m_{v_j \rightarrow u_i}(x) \leftarrow \alpha \cdot m_{v_j \rightarrow u_i}^{\text{new}}(x) + (1 - \alpha) \cdot m_{v_j \rightarrow u_i}^{\text{old}}(x)$ .
16:    Hard Decision: Estimate error:
17:     $\hat{x}_{v_j} \leftarrow \arg\max_x q_{v_j}(x)$ .
18:    if  $H\hat{x} \equiv s \pmod{4}$  then
19:      return  $\hat{x}$   $\triangleright$  BP converged
20:    end if
21:  end for
22:  return Soft decisions  $q$   $\triangleright$  Pass to OSD-0
23: end procedure

```

neighbors except the receiver,

$$m_{u_i \rightarrow v_j}(a) \propto \left(\bigotimes_{\substack{k=1 \\ k \neq j}}^{V(u_i)} \phi_k \right) (s_{u_i} - H_{ij} \cdot a \bmod 4). \quad (\text{C2})$$

The message is normalized such that $\sum_{b \in \mathbb{Z}_4} m_{u_i \rightarrow v_j}(b) = 1$. After the data node receives its messages, it updates its belief according to

$$q_{v_j}(a) \propto P(v_j, a) \prod_{u_i \in U(v_j)} m_{u_i \rightarrow v_j}(a). \quad (\text{C3})$$

Here, $U(v_j)$ denotes the check nodes neighboring the data node v_j . After that, the data node sends the ratio of its cumulative belief to the received belief from each check node,

$$m_{v_j \rightarrow u_i}(a) = \frac{q_{v_j}(a)}{m_{u_i \rightarrow v_j}(a)}. \quad (\text{C4})$$

Optionally, we apply a damping scheme to improve numerical stability. We have found that this parameter is not particularly crucial near the thresholds. However, a nonzero value close to

1 was empirically found to help convergence,

$$m_{v_j \rightarrow u_i}^{(t)}(a) = \alpha \cdot m_{v_j \rightarrow u_i}^{\text{new}}(a) + (1 - \alpha) \cdot m_{v_j \rightarrow u_i}^{(t-1)}(a). \quad (\text{C5})$$

Finally, we use the qudit's belief to compute a hard decision estimate of the error: $\hat{x}_{v_j} = \text{argmax}(q_{v_j})$. This new vector takes the index with the highest probability for all data qudits. Finally, we check convergence $H \cdot \hat{x} = s$ every few iterations to reduce computational cost. If the estimated error satisfies the syndrome, the algorithm halts; otherwise, it continues with another iteration. When the algorithm fails to converge after reaching a preset maximum number of iterations T_{max} , it passes the soft decisions vectors q to the OSD step described in Sec. IV C 3. The algorithm is summarized in Algorithm 3.

We note that for the depolarizing channel, one has vectors of length 16 instead of four. We could also use log probabilities to avoid underflow. However, near the threshold, this is not an issue, and using raw probabilities is slightly faster. We also note that, recently, a paper appeared that shows promise in modifying the Tanner graph in order to avoid the OSD step altogether [193]. It would be interesting to see whether an adaptation of this method to qudit codes could provide further improvements.

-
- [1] P. W. Shor, Scheme for reducing decoherence in quantum computer memory, *Phys. Rev. A* **52**, R2493 (1995).
 - [2] D. Gottesman, Stabilizer codes and quantum error correction, Ph.D. thesis, California Institute of Technology, Pasadena, CA (1997).
 - [3] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Topological quantum memory, *J. Math. Phys.* **43**, 4452 (2002).
 - [4] A. Kitaev, Fault-tolerant quantum computation by anyons, *Ann. Phys.* **303**, 2 (2003).
 - [5] C. Nayak, S. H. Simon, A. Stern, M. Freedman, and S. Das Sarma, Non-Abelian anyons and topological quantum computation, *Rev. Mod. Phys.* **80**, 1083 (2008).
 - [6] K. J. Satzinger, Y.-J. Liu, A. Smith, C. Knapp, M. Newman, C. Jones, Z. Chen, C. Quintana, X. Mi, A. Dunsworth, C. Gidney, *et al.*, Realizing topologically ordered states on a quantum processor, *Science* **374**, 1237 (2021).
 - [7] D. Bluvstein, S. J. Evered, A. A. Geim, S. H. Li, H. Zhou, T. Manovitz, S. Ebadi, M. Cain, M. Kalinowski, D. Hangleiter, *et al.*, Logical quantum processor based on reconfigurable atom arrays, *Nature (London)* **626**, 58 (2024).
 - [8] P. Sales Rodriguez, J. M. Robinson, P. N. Jepsen, Z. He, C. Duckering, C. Zhao, K.-H. Wu, J. Campo, K. Bagnall, M. Kwon, *et al.*, Experimental demonstration of logical magic state distillation, *Nature (London)* **645**, 620 (2025).
 - [9] A. Paetznick, M. P. da Silva, C. Ryan-Anderson, J. M. Bello-Rivas, J. P. Campora, A. Chernoguzov, J. M. Dreiling, C. Foltz, F. Frachon, J. P. Gaebler, *et al.*, Demonstration of logical qubits and repeated error correction with better-than-physical error rates, [arXiv:2404.02280](https://arxiv.org/abs/2404.02280).
 - [10] R. S. Gupta, N. Sundaresan, T. Alexander, C. J. Wood, S. T. Merkel, M. B. Healy, M. Hillenbrand, T. Jochym-O'Connor, J. R. Wootton, T. J. Yoder, *et al.*, Encoding a magic state with beyond break-even fidelity, *Nature (London)* **625**, 259 (2024).
 - [11] R. Acharya, I. Aleiner, R. Allen, T. I. Andersen, M. Ansmann, F. Arute, K. Arya, A. Asfaw, J. Atalaya, R. Babbush, *et al.*, Suppressing quantum errors by scaling a surface code logical qubit, *Nature (London)* **614**, 676 (2023).
 - [12] D. Bluvstein, H. Levine, G. Semeghini, T. T. Wang, S. Ebadi, M. Kalinowski, A. Keesling, N. Maskara, H. Pichler, M. Greiner, V. Vuletić, and M. D. Lukin, A quantum processor based on coherent transport of entangled atom arrays, *Nature (London)* **604**, 451 (2022).
 - [13] V. V. Sivak, A. Eickbusch, B. Royer, S. Singh, I. Tsioutsios, S. Ganjam, A. Miano, B. L. Brock, A. Z. Ding, L. Frunzio, *et al.*, Real-time quantum error correction beyond break-even, *Nature (London)* **616**, 50 (2023).
 - [14] S. Krinner, N. Lacroix, A. Remm, A. Di Paolo, E. Genois, C. Leroux, C. Hellings, S. Lazar, F. Swiadek, J. Herrmann, *et al.*, Realizing repeated quantum error correction in a distance-three surface code, *Nature (London)* **605**, 669 (2022).
 - [15] C. Ryan-Anderson, J. G. Bohnet, K. Lee, D. Gresh, A. Hankin, J. P. Gaebler, D. Francois, A. Chernoguzov, D. Lucchetti, N. C. Brown, T. M. Gatterman, S. K. Halit, K. Gilmore, J. A. Gerber, B. Neyenhuis, D. Hayes, and R. P. Stutz, Realization of real-time fault-tolerant quantum error correction, *Phys. Rev. X* **11**, 041058 (2021).
 - [16] R. Acharya, D. A. Abanin, L. Aghababaie-Beni, I. Aleiner, T. I. Andersen, M. Ansmann, F. Arute, K. Arya, A. Asfaw, N. Astrakhantsev, *et al.*, Quantum error correction below the surface code threshold, *Nature (London)* **638**, 920 (2025).
 - [17] Y. Hu, Y. Wan, and Y.-S. Wu, Twisted quantum double model of topological phases in two dimensions, *Phys. Rev. B* **87**, 125114 (2013).
 - [18] M. Meth, J. Zhang, J. F. Haase, C. Edmunds, L. Postler, A. J. Jena, A. Steiner, L. Dellantonio, R. Blatt, P. Zoller, *et al.*,

- Simulating two-dimensional lattice gauge theories on a qudit quantum computer, *Nat. Phys.* **21**, 570 (2025).
- [19] N. P. D. Sawaya, T. Menke, T. Ha Kyaw, S. Johri, A. Aspuru-Guzik, and G. G. Guerreschi, Resource-efficient digital quantum simulation of d -level systems for photonic, vibrational, and spin- s Hamiltonians, *npj Quantum Inf.* **6**, 49 (2020).
- [20] A. Bocharov, M. Roetteler, and K. M. Svore, Factoring with qutrits: Shor's algorithm on ternary and metaplectic quantum architectures, *Phys. Rev. A* **96**, 012306 (2017).
- [21] P. Gokhale, Y. Ding, T. Tomesh, M. Suchara, M. Martonosi, and F. T. Chong, Asymptotic improvements to quantum circuits via qutrits, in *Proceedings of the 46th International Symposium on Computer Architecture (ISCA)* (ACM Press, New York, NY, 2019), pp. 554–566.
- [22] J. Chu, X. He, Y. Zhou, *et al.*, Scalable algorithm simplification using quantum and logic, *Nat. Phys.* **19**, 126 (2023).
- [23] E. T. Campbell, H. Anwar, and D. E. Browne, Magic-state distillation in all prime dimensions using quantum Reed–Muller codes, *Phys. Rev. X* **2**, 041021 (2012).
- [24] E. T. Campbell, Enhanced fault-tolerant quantum computing in d -level systems, *Phys. Rev. Lett.* **113**, 230501 (2014).
- [25] J. Lindon, A. Tashchilina, L. W. Cooke, and L. J. LeBlanc, Complete unitary qutrit control in ultracold atoms, *Phys. Rev. Appl.* **19**, 034089 (2023).
- [26] E. O. Kiktenko, A. S. Nikolaeva, P. Xu, G. V. Shlyapnikov, and A. K. Fedorov, Scalable quantum computing with qudits on a graph, *Phys. Rev. A* **101**, 022304 (2020).
- [27] I. Fernández De Fuentes, T. Botzem, M. A. I. Johnson, *et al.*, Navigating the 16-dimensional Hilbert space of a high-spin donor qudit with electric and magnetic fields, *Nat. Commun.* **15**, 1380 (2024).
- [28] N. B. Vilas, P. Robichaud, C. Hallas, G. K. Li, L. Anderegg, and J. M. Doyle, An optical tweezer array of ultracold polyatomic molecules, *Nature (London)* **628**, 282 (2024).
- [29] S. Chaudhury, S. T. Merkel, A. Silberfarb, I. H. Deutsch, and P. S. Jessen, Quantum control of the hyperfine spin of a Cs atom ensemble, *Phys. Rev. Lett.* **99**, 163002 (2007).
- [30] M. Kues, C. Reimer, P. Roztock, L. R. Cortés, S. Sciara, B. Wetz, Y. Zhang, A. Cino, S.-T. Chu, B. E. Little, *et al.*, On-chip generation of high-dimensional entangled quantum states and their coherent control, *Nature (London)* **546**, 622 (2017).
- [31] Y. Chi, J. Huang, Z. Zhang, J. Mao, Z. Zhou, X. Chen, C. Zhai, J. Bao, T. Dai, H. Yuan, *et al.*, A programmable qudit-based quantum processor, *Nat. Commun.* **13**, 1166 (2022).
- [32] B. L. Brock, S. Singh, A. Eickbusch, V. V. Sivak, A. Z. Ding, L. Frunzio, S. Girvin, and M. Devoret, Quantum error correction of qudits beyond break-even, *Nature (London)* **641**, 612 (2025).
- [33] L. B. Nguyen, Y. Chi, Z. Zhang, Y. Sun, X. Chen, Z. Zhai, B. Tang, and Y. Yang, Empowering a qudit-based quantum processor by traversing the dual bosonic ladder, *Nat. Commun.* **15**, 7117 (2024).
- [34] S. Roy, X. Wu, H. J. Kim, J. Lee, Y. Zhao, A. Senanian, C. S. Wang, O. C. Wetherbee, L. Zhang, B. Cole, C. P. Larson, E. Yelton, K. Arora, P. L. McMahon, *et al.*, Synthetic high angular momentum spin dynamics in a microwave oscillator, *Phys. Rev. X* **15**, 021009 (2025).
- [35] Z. Wang, R. W. Parker, E. Champion, and M. S. Blok, High- e_j/e_c transmon qudits with up to 12 levels, *Phys. Rev. Appl.* **23**, 034046 (2025).
- [36] F. M. Leupold, P. Jurcevic, C. Hempel, G. A. Kazakov, M. Giustina, R. Blatt, and C. F. Roos, Sustained state-independent quantum contextual correlations from a single ion, *Phys. Rev. Lett.* **120**, 180401 (2018).
- [37] M. Ringbauer, J. Hapla, C. T. Schmiegelow, K. Kieling, R. Blatt, and C. F. Roos, A universal qudit quantum processor with trapped ions, *Nat. Phys.* **18**, 1053 (2022).
- [38] C. Adambukulam, B. C. Johnson, A. Morello, and A. Laucht, Hyperfine spectroscopy and fast, all-optical arbitrary state initialization and readout of a single, ten-level ^{73}Ge vacancy nuclear spin qudit in diamond, *Phys. Rev. Lett.* **132**, 060603 (2024).
- [39] V. A. Soltamov, P. V. Klimov, N. A. Zakharenko, E. B. Monakhov, and P. G. Baranov, Excitation and coherent control of spin qudit modes in silicon carbide at room temperature, *Nat. Commun.* **10**, 1678 (2019).
- [40] J. Haah, Commuting Pauli Hamiltonians as maps between free modules, *Commun. Math. Phys.* **324**, 351 (2013).
- [41] J. Haah, Classification of translation invariant topological Pauli stabilizer codes for prime dimensional qudits on two-dimensional lattices, *J. Math. Phys.* **62**, 012201 (2021).
- [42] T. D. Ellison, Y.-A. Chen, A. Dua, W. Shirley, N. Tantivasadakarn, and D. J. Williamson, Pauli stabilizer models of twisted quantum doubles, *PRX Quantum* **3**, 010353 (2022).
- [43] H. Bombín, G. Duclos-Cianci, and D. Poulin, Universal topological phase of two-dimensional stabilizer codes, *New J. Phys.* **14**, 073048 (2012).
- [44] H. Bombín, Structure of 2D topological stabilizer codes, *Commun. Math. Phys.* **327**, 387 (2014).
- [45] K. Laubscher, D. Loss, and J. R. Wootton, Universal quantum computation in the surface code using non-Abelian islands, *Phys. Rev. A* **100**, 012338 (2019).
- [46] H. Bombin and M. A. Martin-Delgado, Family of non-Abelian Kitaev models on a lattice: Topological condensation and confinement, *Phys. Rev. B* **78**, 115421 (2008).
- [47] S. Beigi, P. W. Shor, and D. Whalen, The quantum double model with boundary: Condensations and symmetries, *Commun. Math. Phys.* **306**, 663 (2011).
- [48] A. Kitaev and L. Kong, Models for gapped boundaries and domain walls, *Commun. Math. Phys.* **313**, 351 (2012).
- [49] I. Cong, M. Cheng, and Z. Wang, Topological quantum computation with gapped boundaries, *arXiv:1609.02037*.
- [50] I. Cong, M. Cheng, and Z. Wang, Hamiltonian and algebraic theories of gapped boundaries in topological phases of matter, *Commun. Math. Phys.* **355**, 645 (2017).
- [51] L. Kong, X.-G. Wen, and H. Zheng, Boundary-bulk relation in topological orders, *Nucl. Phys. B* **922**, 62 (2017).
- [52] A. Bullivant, Y. Hu, and Y. Wan, Twisted quantum double model of topological order with boundaries, *Phys. Rev. B* **96**, 165138 (2017).
- [53] Z. Liang, B. Yang, J. T. Iosue, and Y.-A. Chen, Operator algebra and algorithmic construction of boundaries and defects in $(2+1)\text{d}$ topological Pauli stabilizer codes, *arXiv:2410.11942*.
- [54] Y. Li, Z. Song, A. Kubica, and I. H. Kim, Domain walls from SPT-sewing, *arXiv:2411.11967*.
- [55] M. Li, X. Yang, and X.-Y. Dong, Gapped boundaries of Kitaev's quantum double models: A lattice realization of anyon

- condensation from Lagrangian algebras, *Phys. Rev. B* **113**, 035150 (2026).
- [56] H. Bombin, Topological order with a twist: Ising anyons from an Abelian model, *Phys. Rev. Lett.* **105**, 030403 (2010).
- [57] S. Bravyi, M. B. Hastings, and S. Michalakis, Topological quantum order: Stability under local perturbations, *J. Math. Phys.* **51**, 093512 (2010).
- [58] Y. Hu, Y. Wan, and Y.-S. Wu, Boundary Hamiltonian theory for gapped topological orders, *Chin. Phys. Lett.* **34**, 077103 (2017).
- [59] Y. Hu, Z.-X. Luo, R. Pankovich, Y. Wan, and Y.-S. Wu, Boundary Hamiltonian theory for gapped topological phases on an open surface, *J. High Energy Phys.* **01** (2018) 134.
- [60] M. Barkeshli, C.-M. Jian, and X.-L. Qi, Theory of defects in Abelian topological states, *Phys. Rev. B* **88**, 235103 (2013).
- [61] D. K. Tuckett, A. S. Darmawan, C. T. Chubb, S. Bravyi, S. D. Bartlett, and S. T. Flammia, Tailoring surface codes for highly biased noise, *Phys. Rev. X* **9**, 041031 (2019).
- [62] M. de Wild Propitius, Topological interactions in broken gauge theories, Ph.D. thesis, University of Amsterdam, Amsterdam (1995).
- [63] V. G. Drinfel'd, Quantum groups, *J. Sov. Math.* **41**, 898 (1988).
- [64] R. Dijkgraaf and E. Witten, Topological gauge theories and group cohomology, *Commun. Math. Phys.* **129**, 393 (1990).
- [65] L. Kong, Anyon condensation and tensor categories, *Nucl. Phys. B* **886**, 436 (2014).
- [66] The terms edge and qudit are essentially synonymous in this work since each edge always supports a qudit.
- [67] V. Gheorghiu, Standard form of qudit stabilizer groups, *Phys. Lett. A* **378**, 505 (2014).
- [68] F. Wilczek, Quantum mechanics of fractional-spin particles, *Phys. Rev. Lett.* **49**, 957 (1982).
- [69] L. Kong and Z.-H. Zhang, An invitation to topological orders and category theory, *arXiv:2205.05565*.
- [70] M. Gould, Quantum double finite group algebras and their representations, *Bull. Aust. Math. Soc.* **48**, 275 (1993).
- [71] L. Wang and Z. Wang, In and around Abelian anyon models, *J. Phys. A: Math. Theor.* **53**, 505203 (2020).
- [72] H. Bombin, An introduction to topological quantum codes, *arXiv:1311.0277*.
- [73] S. B. Bravyi and A. Y. Kitaev, Quantum codes on a lattice with boundary, *arXiv:quant-ph/9811052*.
- [74] A. Kapustin and N. Saulina, Topological boundary conditions in Abelian Chern–Simons theory, *Nucl. Phys. B* **845**, 393 (2011).
- [75] P. Etingof, D. Nikshych, V. Ostrik, and W. E. Meir, Fusion categories and homotopy theory, *arXiv:0909.3140*.
- [76] M. Levin, Protected edge modes without symmetry, *Phys. Rev. X* **3**, 021009 (2013).
- [77] M. Barkeshli, Y.-A. Chen, S.-J. Huang, R. Kobayashi, N. Tantivasadakarn, and G. Zhu, Codimension-2 defects and higher symmetries in (3+1)D topological phases, *SciPost Phys.* **14**, 065 (2023).
- [78] M. Barkeshli, Y.-A. Chen, P.-S. Hsin, and R. Kobayashi, Higher-group symmetry in finite gauge theory and stabilizer codes, *SciPost Phys.* **16**, 089 (2024).
- [79] T. Lan, J. C. Wang, and X.-G. Wen, Gapped domain walls, gapped boundaries, and topological degeneracy, *Phys. Rev. Lett.* **114**, 076402 (2015).
- [80] H. Bombin, Clifford gates by code deformation, *New J. Phys.* **13**, 043005 (2011).
- [81] T. J. Yoder and I. H. Kim, The surface code with a twist, *Quantum* **1**, 2 (2017).
- [82] 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).
- [83] T. R. Scruby and D. E. Browne, A hierarchy of anyon models realised by twists in stacked surface codes, *Quantum* **4**, 251 (2020).
- [84] F. Petiziol, Non-Abelian anyons in periodically driven Abelian spin liquids, *Phys. Rev. Lett.* **133**, 036601 (2024).
- [85] R. Kobayashi, Fermionic defects of topological phases and logical gates, *SciPost Phys.* **15**, 028 (2023).
- [86] R. Kobayashi and G. Zhu, Cross-cap defects and fault-tolerant logical gates in the surface code and the honeycomb Floquet code, *PRX Quantum* **5**, 020360 (2024).
- [87] M. A. Levin and X.-G. Wen, String-net condensation: A physical mechanism for topological phases, *Phys. Rev. B* **71**, 045110 (2005).
- [88] A. Krishna and D. Poulin, Topological wormholes: Nonlocal defects on the toric code, *Phys. Rev. Res.* **2**, 023116 (2020).
- [89] G. S. Canright, S. M. Girvin, and A. Brass, Superconductive pairing of fermions and semions in two dimensions, *Phys. Rev. Lett.* **63**, 2295 (1989).
- [90] 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).
- [91] A. Benhemou, J. K. Pachos, and D. E. Browne, Non-Abelian statistics with mixed-boundary punctures on the toric code, *Phys. Rev. A* **105**, 042417 (2022).
- [92] A. J. Landahl and B. C. A. Morrison, Logical fermions for fault-tolerant quantum simulation, *arXiv:2110.10280*.
- [93] M. Iqbal, A. Lyons, C. F. B. Lo, N. Tantivasadakarn, J. Dreiling, C. Foltz, T. M. Gatterman, D. Gresh, N. Hewitt, C. A. Holliman, *et al.*, Qutrit toric code and parafermions in trapped ions, *Nat. Commun.* **16**, 6301 (2025).
- [94] R. Fern, J. Kombe, and S. Simon, How $SU(2)_4$ anyons are Z_3 parafermions, *SciPost Phys.* **3**, 037 (2017).
- [95] A. Hutter and D. Loss, Quantum computing with parafermions, *Phys. Rev. B* **93**, 125105 (2016).
- [96] E. Knill and R. Laflamme, Theory of quantum error-correcting codes, *Phys. Rev. A* **55**, 900 (1997).
- [97] A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane, Quantum error correction and orthogonal geometry, *Phys. Rev. Lett.* **78**, 405 (1997).
- [98] S. Bravyi, D. Poulin, and B. Terhal, Tradeoffs for reliable quantum information storage in 2D systems, *Phys. Rev. Lett.* **104**, 050503 (2010).
- [99] A. R. Calderbank and P. W. Shor, Good quantum error-correcting codes exist, *Phys. Rev. A* **54**, 1098 (1996).
- [100] N. P. Breuckmann, C. Vuillot, E. Campbell, A. Krishna, and B. M. Terhal, Hyperbolic and semi-hyperbolic surface codes for quantum storage, *Quantum Sci. Technol.* **2**, 035007 (2017).
- [101] P. Panteleev and G. Kalachev, Asymptotically good quantum and locally testable classical LDPC codes, *arXiv:2111.03654*.

- [102] S. Bravyi, A. W. Cross, J. M. Gambetta, D. Maslov, P. Rall, and T. J. Yoder, High-threshold and low-overhead fault-tolerant quantum memory, *Nature (London)* **627**, 778 (2024).
- [103] T. Jochym-O'Connor, A. Kubica, and T. J. Yoder, Disjointness of stabilizer codes and limitations on fault-tolerant logical gates, *Phys. Rev. X* **8**, 021047 (2018).
- [104] S. Bravyi and A. Kitaev, Universal quantum computation with ideal Clifford gates and noisy ancillas, *Phys. Rev. A* **71**, 022316 (2005).
- [105] B. Eastin and E. Knill, Restrictions on transversal encoded quantum gate sets, *Phys. Rev. Lett.* **102**, 110502 (2009).
- [106] S. Bravyi and R. König, Classification of topologically protected gates for local stabilizer codes, *Phys. Rev. Lett.* **110**, 170503 (2013).
- [107] D. Litinski, Magic state distillation: Not as costly as you think, *Quantum* **3**, 205 (2019).
- [108] N. Bao, C. Cao, and V. P. Su, Magic state distillation from entangled states, *Phys. Rev. A* **105**, 022602 (2022).
- [109] C. Gidney, N. Shutty, and C. Jones, Magic state cultivation: Growing T states as cheap as CNOT gates, *arXiv:2409.17595*.
- [110] Z.-H. Chen, M.-C. Chen, C.-Y. Lu, and J.-W. Pan, Efficient magic state cultivation on RP2, *PRX Quantum* **7**, 010315 (2026).
- [111] A. M. Steane, Error correcting codes in quantum theory, *Phys. Rev. Lett.* **77**, 793 (1996).
- [112] C. H. Bennett, D. P. DiVincenzo, J. A. Smolin, and W. K. Wootters, Mixed-state entanglement and quantum error correction, *Phys. Rev. A* **54**, 3824 (1996).
- [113] P. W. Shor and J. A. Smolin, Quantum error-correcting codes need not completely reveal the error syndrome, *arXiv:quant-ph/9604006*.
- [114] G. Duclos-Cianci and D. Poulin, Kitaev's Z_d -codes threshold estimates, *Phys. Rev. A* **87**, 062338 (2013).
- [115] R. S. Andrist, J. R. Wootton, and H. G. Katzgraber, Error thresholds for Abelian quantum double models: Increasing the bit-flip stability of topological quantum memory, *Phys. Rev. A* **91**, 042331 (2015).
- [116] F. H. E. Watson, H. Anwar, and D. E. Browne, A fast fault-tolerant decoder for qubit and qudit surface codes, *Phys. Rev. A* **92**, 032309 (2015).
- [117] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th ed. (Cambridge University Press, Cambridge, 2010).
- [118] E. Berlekamp, R. McEliece, and H. van Tilborg, On the inherent intractability of certain coding problems (Corresp.), *IEEE Trans. Inf. Theory* **24**, 384 (1978).
- [119] J. Edmonds, Maximum matching and a polyhedron with 0,1-vertices, *J. Res. Natl. Bur. Stand. B* **69**, 125 (1965).
- [120] J. Edmonds, Paths, trees, and flowers, *Can. J. Math.* **17**, 449 (1965).
- [121] O. Higgott and C. Gidney, Sparse blossom: Correcting a million errors per core second with minimum-weight matching, *Quantum* **9**, 1600 (2025).
- [122] N. Delfosse and N. H. Nickerson, Almost-linear time decoding algorithm for topological codes, *Quantum* **5**, 595 (2021).
- [123] S. Bravyi and J. Haah, Quantum self-correction in the 3D cubic code model, *Phys. Rev. Lett.* **111**, 200501 (2013).
- [124] D. Poulin and Y. Chung, On the iterative decoding of sparse quantum codes, *Quantum Inf. Comput.* **8**, 986 (2008).
- [125] J. Roffe, D. R. White, S. Burton, and E. Campbell, Decoding across the quantum low-density parity-check code landscape, *Phys. Rev. Res.* **2**, 043423 (2020).
- [126] P. Panteleev and G. Kalachev, Degenerate quantum LDPC codes with good finite length performance, *Quantum* **5**, 585 (2021).
- [127] S. Krastanov and L. Jiang, Deep neural network probabilistic decoder for stabilizer codes, *Sci. Rep.* **7**, 11003 (2017).
- [128] G. Torlai and R. G. Melko, Neural decoder for topological codes, *Phys. Rev. Lett.* **119**, 030501 (2017).
- [129] C. Chamberland and P. Ronagh, Deep neural decoders for near term fault-tolerant experiments, *Quantum Sci. Technol.* **3**, 044002 (2018).
- [130] A. W. Senior, T. Edlich, F. J. H. Heras, L. M. Zhang, O. Higgott, J. S. Spencer, T. Applebaum, S. Blackwell, J. Ledford, A. Žemgulytė, *et al.*, A scalable and real-time neural decoder for topological quantum codes, *arXiv:2512.07737*.
- [131] G. Hu, W. Ouyang, C.-Y. Lu, C. Lin, and H.-S. Zhong, Efficient and universal neural-network decoder for stabilizer-based quantum error correction, *arXiv:2502.19971*.
- [132] K. Zhang, S. Wang, L. Kong, F. Zhang, Z. Ji, and J. Chen, Learning neural decoding with parallelism and self-coordination for quantum error correction, *arXiv:2509.03815*.
- [133] L. Caune, L. Skoric, N. S. Blunt, A. Ruban, J. McDaniel, J. A. Valery, A. D. Patterson, A. V. Gramolin, J. Majaniemi, K. M. Barnes, *et al.*, Demonstrating real-time and low-latency quantum error correction with superconducting qubits, *arXiv:2410.05202*.
- [134] N. Liyanage, Y. Wu, S. Tagare, and L. Zhong, FPGA-based distributed union-find decoder for surface codes, *IEEE Trans. Quantum Eng.* **5**, 1 (2024).
- [135] A. B. Ziad, A. Zalawadiya, C. Topal, J. Camps, G. P. Gehér, M. P. Stafford, and M. L. Turner, Local clustering decoder as a fast and adaptive hardware decoder for the surface code, *Nat. Commun.* **16**, 11048 (2025).
- [136] S. Wolanski and B. Barber, Ambiguity clustering: An accurate and efficient decoder for qLDPC codes, *arXiv:2406.14527*.
- [137] B. Barber, K. M. Barnes, T. Bialas, O. Buğdaycı, E. T. Campbell, N. I. Gillespie, K. Johar, R. Rajan, A. W. Richardson, L. Skoric, *et al.*, A real-time, scalable, fast and highly resource efficient decoder for a quantum computer, *Nat. Electron.* **8**, 84 (2025).
- [138] P. Iyer and D. Poulin, Hardness of decoding quantum stabilizer codes, *arXiv:1310.3235*.
- [139] S. Bravyi, M. Suchara, and A. Vargo, Efficient algorithms for maximum likelihood decoding in the surface code, *Phys. Rev. A* **90**, 032326 (2014).
- [140] A. J. Ferris and D. Poulin, Tensor networks and quantum error correction, *Phys. Rev. Lett.* **113**, 030501 (2014).
- [141] C. T. Chubb, General tensor network decoding of 2D Pauli codes, *arXiv:2101.04125*.
- [142] C. Gidney, Stim: A fast stabilizer circuit simulator, *Quantum* **5**, 497 (2021).
- [143] A. Kabir, S. Nguyen, S. Ghosh, T. Kiran, I. H. Kim, and Y. Huang, Sdim: A qudit stabilizer simulator, *arXiv:2511.12777*.
- [144] O. Higgott, Pymatching: A Python package for decoding quantum codes with minimum-weight perfect matching, *ACM Trans. Quantum Comput.* **3**, 1 (2022).

- [145] Y. Wu and L. Zhong, Fusion blossom: Fast MWPM decoders for QEC, [arXiv:2305.08307](https://arxiv.org/abs/2305.08307).
- [146] D. K. Tuckett, Tailoring surface codes: Improvements in quantum error correction with biased noise, Ph.D. thesis, University of Sydney (2020), <https://github.com/qecsim/qecsim>.
- [147] C.-Y. Park and K. Meinerz, Open-source C++ implementation of the union-find decoder (2020), <https://github.com/chaeyeunpark/UnionFind>.
- [148] J. Roffe, LDPC: Python tools for low density parity check codes (2022), <https://pypi.org/project/ldpc/>.
- [149] Qiskit Community, Qiskit QEC: Framework for quantum error correction (2022), <https://github.com/qiskit-community/qiskit-qec>.
- [150] J. Roffe, BPOSD: A Python package for belief propagation with ordered statistic decoding (2021), <https://github.com/quantumcircuits/bposd>.
- [151] A. J. Landahl, J. T. Anderson, and P. R. Rice, Fault-tolerant quantum computing with color codes, [arXiv:1108.5738](https://arxiv.org/abs/1108.5738).
- [152] Y. Takada, Y. Takeuchi, and K. Fujii, Ising model formulation for highly accurate topological color codes decoding, *Phys. Rev. Res.* **6**, 013092 (2024).
- [153] M. Cain, C. Zhao, H. Zhou, N. Meister, J. P. B. Ataiades, A. Jaffe, D. Bluvstein, and M. D. Lukin, Correlated decoding of logical algorithms with transversal gates, [arXiv:2403.03272](https://arxiv.org/abs/2403.03272).
- [154] L. A. Beni, O. Higgott, and N. Shutty, Tesseract: A search-Based decoder for quantum error correction, [arXiv:2503.10988](https://arxiv.org/abs/2503.10988).
- [155] T. Sakashita, Fast and accurate decoder for the XZZX code using simulated annealing, [arXiv:2509.17837](https://arxiv.org/abs/2509.17837).
- [156] International Business Machines Corporation, IBM ILOG, IBM ILOG CPLEX Optimization Studio user's manual, International Business Machines Corporation (2023), <https://www.ibm.com/docs/en/icos>.
- [157] Gurobi Optimization, LLC, Gurobi Optimizer reference manual (2024), <https://www.gurobi.com>.
- [158] J. Feldman, M. Wainwright, and D. Karger, Using linear programming to decode binary linear codes, *IEEE Trans. Inf. Theory* **51**, 954 (2005).
- [159] J. X. Li and P. O. Vontobel, LP decoding of quantum stabilizer codes, in *Proceedings of the 2018 IEEE International Symposium on Information Theory (ISIT)* (IEEE, Piscataway, NJ, 2018), pp. 1306–1310.
- [160] O. Fawzi, L. Groues, and A. Leverrier, Linear programming decoder for hypergraph product quantum codes, in *Proceedings of the 2020 IEEE Information Theory Workshop (ITW)* (IEEE, Piscataway, NJ, 2021), pp. 1–5.
- [161] S. Javed, F. Garcia-Herrero, B. Vasic, and M. F. Flanagan, Low-complexity linear programming based decoding of quantum LDPC codes, [arXiv:2311.18488](https://arxiv.org/abs/2311.18488).
- [162] S. Gu and M. Soleimanifar, Power and limitations of linear programming decoder for quantum LDPC codes, [arXiv:2508.04769](https://arxiv.org/abs/2508.04769).
- [163] D. MacKay, Good error-correcting codes based on very sparse matrices, *IEEE Trans. Inf. Theory* **45**, 399 (1999).
- [164] R. M. Tanner, A recursive approach to low complexity codes, *IEEE Trans. Inf. Theory* **27**, 533 (1981).
- [165] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* (Morgan Kaufmann, San Mateo, CA, 1988).
- [166] F. Kschischang, B. Frey, and H.-A. Loeliger, Factor graphs and the sum-product algorithm, *IEEE Trans. Inf. Theory* **47**, 498 (2001).
- [167] Y. Weiss, Correctness of local probability propagation in graphical models with loops, *Neural Comput.* **12**, 1 (2000).
- [168] T. J. Richardson and R. L. Urbanke, The capacity of low-density parity-check codes under message-passing decoding, *IEEE Trans. Inf. Theory* **47**, 599 (2001).
- [169] J. S. Yedidia, W. T. Freeman, and Y. Weiss, Understanding belief propagation and its generalizations, in *Exploring Artificial Intelligence in the New Millennium* (Morgan Kaufmann, San Mateo, CA, 2003), pp. 239–269.
- [170] B. Criger, and I. Ashraf, Multi-path Summation for Decoding 2D Topological Codes, *Quantum* **2**, 102 (2018).
- [171] M. Fossorier and S. Lin, Soft-decision decoding of linear block codes based on ordered statistics, *IEEE Trans. Inf. Theory* **41**, 1379 (1995).
- [172] G. H. Norton and A. Sălăgean, On the structure of linear and cyclic codes over a finite chain ring, *Appl. Algebra Eng. Commun. Comput.* **10**, 489 (2000).
- [173] D. S. Dummit and R. M. Foote, *Abstract Algebra*, 3rd ed. (John Wiley and Sons, Hoboken, NJ, 2004).
- [174] T. Mulders and A. Storjohann, On mathematical properties of the Howell normal form, *J. Symb. Comput.* **35**, 377 (2003).
- [175] R. Sarkar and T. J. Yoder, The qudit Pauli group: Non-commuting pairs, non-commuting sets, and structure theorems, *Quantum* **8**, 1307 (2024).
- [176] M. Fossorier, Iterative reliability-based decoding of low-density parity check codes, *IEEE J. Sel. Areas Commun.* **19**, 908 (2001).
- [177] S. Varona and M. A. Martin-Delgado, Determination of the semion code threshold using neural decoders, *Phys. Rev. A* **102**, 032411 (2020).
- [178] S.-J. Huang, A. Warman, S. Schafer-Nameki, and Y. Chen, Hybrid lattice surgery: Non-Clifford gates via non-Abelian surface codes, [arXiv:2510.20890](https://arxiv.org/abs/2510.20890).
- [179] N. Manjunath, V. Mattei, A. Tiwari, and T. D. Ellison, Universal quantum computation with group surface codes, [arXiv:2603.05502](https://arxiv.org/abs/2603.05502).
- [180] F. Burnell, Anyon condensation and its applications, *Annu. Rev. Condens. Matter Phys.* **9**, 307 (2018).
- [181] 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).
- [182] M. Iqbal, K. Duivenvoorden, and N. Schuch, Study of anyon condensation and topological phase transitions from a $\mathbb{Z}_4$ topological phase using the projected entangled pair states approach, *Phys. Rev. B* **97**, 195124 (2018).
- [183] In this case, it is optional to remove these qudits or keep them.
- [184] The same logic applies for computing the dimensionality of any excited state sector.
- [185] T. Pandey and E. Dumitrescu, Topological characterization with a twist, condensation, and reflection, [arXiv:2209.11126](https://arxiv.org/abs/2209.11126).
- [186] Note also that if we have two operators multiplying to identity $AB = 1$ with orders m, n i.e., $A^m = B^n = 1$, then $m = n$ and they have the same order. Simply because $(AB)^m = 1 = A^m B^m = 1 B^m$ similarly with n .

- [187] B. Bakalov Jr. and A. Kirillov, On the lego-Teichmüller game, [Transf. Groups](#) **5**, 207 (2000).
- [188] A. Hatcher, Pants decompositions of surfaces (1999), [arXiv:math/9906084](#).
- [189] Q. Zhang and W.-T. Xu, Continuous topological phase transition between $\mathbb{Z}_2$ topologically ordered phases, [Phys. Rev. B](#) **112**, L241107 (2025).
- [190] J. W. Cooley and J. W. Tukey, An algorithm for the machine calculation of complex Fourier series, [Math. Comput.](#) **19**, 297 (1965).
- [191] H. Wymeersch, H. Steendam, and M. Moeneclaey, Log-domain decoding of LDPC codes over GF(q), in *Proceedings of the 2004 IEEE International Conference on Communications* (IEEE, Piscataway, NJ, 2004), Vol. 2, pp. 772–776.
- [192] D. Declercq and M. Fossorier, Decoding algorithms for nonbinary LDPC codes over GF(q), [IEEE Trans. Commun.](#) **55**, 633 (2007).
- [193] P. Hack, L. Menti, F. Lazaro, and A. Paler, Achieving thresholds via standalone belief propagation on surface codes, [arXiv:2603.05381](#).